A method for using a health information exchange system which stores patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising: for each individual EMR within the first plurality of EMRs, performing a computerized context interception process using a processor to intercept context from the individual EMR and to identify therewithin an event whereby a health provider using the individual EMR calls up an individual patient’s record from said individual EMR; and responsive to identification of the event, using a computerized output device for providing patient record data, pertaining to the individual patient, to the health provider.
<table>
<thead>
<tr>
<th>##</th>
<th>Connector</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get Data Requirements</td>
<td>SmartWatch service calls KFW to provide data required for specific Knowledge Module evaluation</td>
</tr>
<tr>
<td>2</td>
<td>Get Data</td>
<td>Based on Data Requirements Smart Watch requests needed data from dbMotion Core Service</td>
</tr>
<tr>
<td>3</td>
<td>Convert Data to KFW format</td>
<td>Data received from dbMotion Core Services is being converted into KFW input data format</td>
</tr>
<tr>
<td>4</td>
<td>Evaluate</td>
<td>SmartWatch requests KFW to evaluate specific Knowledge Module based on the collected Data</td>
</tr>
<tr>
<td>5</td>
<td>Get Terminology Concepts Source -&gt; Destination</td>
<td>KFW requests all the relevant terminology concepts based on Knowledge Module Rule definition from CTS</td>
</tr>
<tr>
<td>6</td>
<td>Evaluate Data</td>
<td>KFW executes Knowledge Module Rule evaluation based on the input data and terminology concepts collected in previous stages</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation Result</td>
<td>KFW returns evaluation result to SmartWatch</td>
</tr>
</tbody>
</table>

FIG. 1B
<table>
<thead>
<tr>
<th><strong>UMS (Unified Medical Schema)</strong></th>
<th>Ontology of medical knowledge that defines the entities and their relationships that represent the medical information to be shared by the dbMotion product.</th>
</tr>
</thead>
<tbody>
<tr>
<td>vocabulary management tool</td>
<td>e.g. dbMotion Vocabulary Management (VM) console. This console provides the Graphical User Interface (GUI) and accompanying features to maintain (i.e., add, remove, update) the repository of both the baseline and the local coding systems in each individual RHIO participant node. The VM is the access point for a user to consume services provided by the global vocabulary services.</td>
</tr>
<tr>
<td>Code System</td>
<td>An institute or a system that defines or produces vocabulary codes, for example LOINC, ICD9, any operational system, etc. Each coding system should have a unique identifier.</td>
</tr>
<tr>
<td>Concept Code</td>
<td>A vocabulary code from a specific Code System which represents a concept. Each concept is defined by a code and a Code System, which is the code namespace.</td>
</tr>
<tr>
<td>Baseline Concept Codes (baseline codes)</td>
<td>Vocabulary codes which are common within a community (e.g. dbMotion network).</td>
</tr>
<tr>
<td>Local Concept Codes (local codes)</td>
<td>Vocabulary codes that are used in the local organization and are not shared within a community. These codes should be mapped to Baseline Concept Codes. These codes should be assigned to at least one vocabulary domain.</td>
</tr>
<tr>
<td>Context (Value set)</td>
<td>A group of concepts. It is used to group together a set of Code System Concepts which have similar semantic relations.</td>
</tr>
<tr>
<td>UMS concept domain</td>
<td>A parent domain that groups all the concepts that belong to a coded attribute in the UMS.</td>
</tr>
<tr>
<td>Sub Domain</td>
<td>A child vocabulary domain within a UMS concept domain.</td>
</tr>
<tr>
<td>Knowledge Module (KM)</td>
<td>encapsulates machine-executable knowledge used to interpret data to reach meaningful conclusions at different levels of abstraction</td>
</tr>
<tr>
<td>Event Monitor</td>
<td>A service designed for SmartWatch whose job is to identify relevant records coming into dbMotion for SmartWatch purposes. This service is the main SmartWatch consumer for CTS services</td>
</tr>
</tbody>
</table>

**FIG. 2**
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Domain Expert</td>
<td>An expert in a particular domain, e.g. diabetes specialist, who participates in the knowledge acquisition process as a contributor.</td>
<td>Represented by CTO, CMOs, UPMCCCL</td>
</tr>
<tr>
<td>2.</td>
<td>Knowledge Engineer</td>
<td>A user familiar with the knowledge representation models used by the KFW to capture domain experts knowledge (not necessarily a software engineer but computer literate), who may assist domain experts during the knowledge acquisition process.</td>
<td>Represented by CTO, VP R&amp;D, CMOs, SWITL, UPMCMN, UPMCCCL</td>
</tr>
<tr>
<td>3.</td>
<td>Knowledge-Base Administrator</td>
<td>A user responsible for the administration and management of the KFW knowledge-base.</td>
<td>Represented by CTO, VP R&amp;D, UPMCMN</td>
</tr>
<tr>
<td>4.</td>
<td>SmartGuard Editor</td>
<td>A user responsible for the creation and maintenance of SmartWatch-based applications.</td>
<td>Represented by CTO, VP R&amp;D, SWITL, UPMCMN</td>
</tr>
<tr>
<td>5.</td>
<td>Consumer</td>
<td>Any user or application acting as a consumer that may benefit from the tools or services provided by the KFW.</td>
<td>Represented by CTO, VP R&amp;D, UPMCCCL</td>
</tr>
</tbody>
</table>

FIG. 4A
<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Domain Expert</td>
<td>The number of domain experts will vary depending on the number of clinical domains requiring KFW support. It is to be expected that a relatively small number of users are often explicitly assigned to this role. Given that a domain expert is not necessarily a computer literate, the user interfaces of the tools used must be highly user friendly and intuitive to use. It also requires integration with a Common Terminology Service (CTS) to look up terms originating from controlled medical vocabularies.</td>
</tr>
<tr>
<td>2</td>
<td>Knowledge Engineer</td>
<td>The number of knowledge engineers will also vary depending on the number of clinical domains that requires KFW support. A knowledge engineer is typically computer literate and therefore the user interface can be more complex although usable too. Similar to the domain expert, it also requires integration with services such as CTS.</td>
</tr>
<tr>
<td>3</td>
<td>Knowledge-Base Administrator</td>
<td>Typically, there will be only a small number of KB administrators. A single KB administrator can manage more than one KB. A KB administrator is an advanced computer user. Therefore, the user interface of the tools he uses can be more complex although usable too.</td>
</tr>
<tr>
<td>4</td>
<td>SmartGuard Editor</td>
<td>The number of SmartGuard editors will vary depending on the number of SmartWatch-based applications to develop and their complexity. A SmartWatch developer is computer literate although not necessarily a software developer. Therefore the user interfaces of the tools can be complex although usable too. A SmartGuard editor will typically consume metadata about possible KFW entities and services during design-time tasks, e.g. query to find relevant knowledge modules.</td>
</tr>
<tr>
<td>5</td>
<td>Consumer</td>
<td>Any user, e.g. clinical researcher, or system, e.g. 3rd party EMR, that uses KFW tools or consumes KFW services directly.</td>
</tr>
<tr>
<td>#</td>
<td>Need</td>
<td>Priority</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>1.</td>
<td>Interpret data to reach meaningful conclusions at different levels of abstraction.</td>
<td>High</td>
</tr>
<tr>
<td>2.</td>
<td>Enable iterative interaction with a consumer with respect to data required to evaluate a request.</td>
<td>High</td>
</tr>
<tr>
<td>3.</td>
<td>Adhere to accepted, promising, or rising standards from recognized healthcare IT organizations.</td>
<td>High</td>
</tr>
<tr>
<td>4.</td>
<td>Express clinical decision rules using full Boolean logic expressions.</td>
<td>High</td>
</tr>
</tbody>
</table>

FIG. 5A
<table>
<thead>
<tr>
<th>#</th>
<th>Need</th>
<th>Priority</th>
<th>Concerns</th>
<th>Proposed Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.</td>
<td>Capture knowledge definitions about patient clinical states, i.e. declarative knowledge. For example, definition of anemia.</td>
<td>High</td>
<td>Often clinical decision support involves the use of declarative knowledge, e.g. inclusion/exclusion criteria.</td>
<td>The KFW will support the acquisition and application of declarative knowledge using specialized inference engines. The KFW will strive to adhere to existing standards, e.g. OpenEHR Archetypes.</td>
</tr>
<tr>
<td>6.</td>
<td>Capture knowledge definitions about clinical procedures, i.e. procedural knowledge. For example, administer a periodic drug regimen.</td>
<td>Medium</td>
<td>Often clinical decision support involves the use of procedural knowledge, e.g. clinical guidelines and protocols.</td>
<td>The KFW will support the acquisition and application of procedural knowledge using specialized inference engines. The KFW will strive to adhere to existing standards, e.g. HL7 GLIF.</td>
</tr>
<tr>
<td>7.</td>
<td>Enable to evaluate dynamic KM queries without necessarily persisting them first.</td>
<td>Medium</td>
<td>A KM may be generated on-the-fly by consumer or persisted within consumer's boundaries.</td>
<td>The KFW will enable to evaluate a KM submitted as dynamic queries.</td>
</tr>
<tr>
<td>8.</td>
<td>Enable to define and evaluate a KM on the fly.</td>
<td>Low</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 5B
<table>
<thead>
<tr>
<th></th>
<th>Need</th>
<th>Priority</th>
<th>Concerns</th>
<th>Proposed Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.</td>
<td>Use terms and concepts originating from Controlled Medical Vocabularies (CMV) in KM definitions, e.g. patient data.</td>
<td>High</td>
<td>Each clinical setting may use various terminologies, proprietary or standard, to represent the same clinical terms. Reach semantic interoperability on the knowledge level.</td>
<td>The KFW will utilize the CTS functionality to embed terms that originate from CMV, e.g. SNOMED-CT, in KM definitions. Note, there is a relation between the CMV used to define knowledge and those CMV used to achieve semantic interoperability on the data level, i.e. local to baseline concepts mapping.</td>
</tr>
<tr>
<td>10.</td>
<td>Include a reference to relevant knowledge management supporting content, e.g. peer-reviewed publications.</td>
<td>High</td>
<td>Provide support to Evidence-Based Medicine (EBM).</td>
<td>The KM representation model will include a placeholder to references of relevant clinical and other literature items.</td>
</tr>
<tr>
<td>11.</td>
<td>Enable to incorporate external inference engines and other decision support services seamlessly.</td>
<td>High</td>
<td>Benefit from proven inference engines, e.g. commercial rule engines, and specialized DSS products, e.g. drug safety controllers.</td>
<td>The KFW will use common design patterns and architectural guidelines to minimize the coupling between different parts of the framework, especially of the inference engines.</td>
</tr>
<tr>
<td>#</td>
<td>Need</td>
<td>Priority</td>
<td>Concerns</td>
<td>Proposed Solutions</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>12.</td>
<td>Provide data monitoring recommendations to timely submit an evaluation request.</td>
<td>High</td>
<td>When is the right time to apply knowledge to data? For example, identifying new population members of a SmartWatch solution can be a difficult task.</td>
<td>The KFW will enable to extract data monitoring recommendations based on KM definition, e.g. required data for evaluation (recommended triggering events).</td>
</tr>
<tr>
<td>13.</td>
<td>Enable to create and maintain KM definitions using a Graphical User interface (GUI).</td>
<td>High</td>
<td>The knowledge acquisition process is a complicated task that requires a highly usable user interface.</td>
<td>The KFW will provide a specialized, graphical and highly usable application for knowledge acquisition that enables to view, create, and update KMs.</td>
</tr>
<tr>
<td>14.</td>
<td>Provide means and ways to simplify, ease, and abstract the task of knowledge acquisition.</td>
<td>High</td>
<td>The knowledge acquisition process is one of the most difficult and time consuming tasks especially when the user is not computer literate, e.g. a clinical domain expert.</td>
<td>The KAT will provide its users with various means, such as templates, macros and wizards, to ease and simplify the process of knowledge acquisition.</td>
</tr>
<tr>
<td>15.</td>
<td>Allow domain experts and knowledge engineers to acquire knowledge in a collaborative and iterative process.</td>
<td>Medium</td>
<td>Need to bridge the gap between a domain expert who is usually not computer literate and a knowledge engineer who usually doesn't have clinical background.</td>
<td>The knowledge acquisition tool will enable knowledge engineers and domain experts to create, and update knowledge modules in a gradual, collaborative and incremental fashion.</td>
</tr>
<tr>
<td>#</td>
<td>Need</td>
<td>Priority</td>
<td>Concerns</td>
<td>Proposed Solutions</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>16</td>
<td>Provide backward compatibility when changing KM definition.</td>
<td>High</td>
<td>Allow applications to stay operational following changes to existing KMs</td>
<td>The KFW will support versioning of KMs and storing the complete version history of each KM.</td>
</tr>
<tr>
<td>17</td>
<td>Control user and consumer access to KFW services and resources to maintain a secured and safe environment (role and rule access control).</td>
<td>High</td>
<td>Prevent unauthorized or malicious access to KFW services and repositories.</td>
<td>The KFW will capitalize on the exiting dbMotion Security layer to achieve a secured and controlled environment both at design-time and runtime.</td>
</tr>
<tr>
<td>18</td>
<td>Provide human-readable and machine-interpretable descriptive information of a KM, i.e. metadata.</td>
<td>High</td>
<td>Comprehension of the purpose and goal of each KM by human and computerized actors alike.</td>
<td>Each knowledge module definition will contain metadata attributes such as creation date, author, coded/non-coded keywords, and also a general free-text description.</td>
</tr>
<tr>
<td>19</td>
<td>Enable to search and retrieve KMs.</td>
<td>High</td>
<td>Need to find a KMs before it can be evaluated.</td>
<td>The KFW will provide a search engine that receives search queries based on KM descriptive information.</td>
</tr>
<tr>
<td>20</td>
<td>Provide an explanation of a conclusion following a KM evaluation request as part of its output.</td>
<td>Medium</td>
<td>Ability to comprehend and follow the process leading to a particular conclusion.</td>
<td>The KFW will include in a KM output a detailed execution trace which lists the inference steps alongside the data that participated in each step.</td>
</tr>
</tbody>
</table>

FIG. 5E
<table>
<thead>
<tr>
<th>#</th>
<th>Need</th>
<th>Priority</th>
<th>Concerns</th>
<th>Proposed Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Support to visually display dependencies and interrelations between KMs.</td>
<td>Medium</td>
<td>Maintenance of complex knowledge definitions.</td>
<td>The KFW will provide means to visualize, explore, and manage KM dependencies and interrelations.</td>
</tr>
<tr>
<td>22</td>
<td>Generate recommendations of a VPO schema while considering clinical needs of both care provider and patient.</td>
<td>Medium</td>
<td>An end user often requires a nonrigid VPO with a flexible data schema to suit user needs efficiently.</td>
<td>The KFW will cater for a KM that returns as output a VPO schema. The inputs to evaluate such a KM may include end user clinical specialty, e.g. endocrinologist, and patient current problems list, e.g. diabetes.</td>
</tr>
<tr>
<td>23</td>
<td>Provide 3rd party applications, e.g. EMR, with an interoperable and standard way to consume KFW services.</td>
<td>Medium</td>
<td>Empower applications with unique services that capitalize on the relative strengths of dbMotion platform and KFW.</td>
<td>The KFW will adhere to standards such as the HSSP DSS. One of the main business purposes of this standard is to promote the use of standardize DSS services by applications such as EMRs.</td>
</tr>
<tr>
<td>24</td>
<td>Provide tools to manage and administrate KFW resources.</td>
<td>High</td>
<td>Assuring reliable, maintainable, and efficient use of the system.</td>
<td>The KFW will provide a management tools suit that enables an administrator to manage and monitor KFW resources.</td>
</tr>
<tr>
<td>25</td>
<td>Track, audit, and log system behavior, system usage, and consumers' access to services and resources of the system.</td>
<td>High</td>
<td>Assist in management, security, and maintenance operations.</td>
<td>The KFW will capitalize on the exiting dbMotion STL (and ADR) for these purposes.</td>
</tr>
</tbody>
</table>

FIG. 5F
<table>
<thead>
<tr>
<th>#</th>
<th>Need</th>
<th>Priority</th>
<th>Concerns</th>
<th>Proposed Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Support probabilistic reasoning capability and reflect confidence level of a conclusion as part of its output.</td>
<td>Medium</td>
<td>It's often not possible to reach deterministic conclusions, e.g. having incomplete data. In addition, it's important to reflect reliability level of data participating in an inference process, e.g. urine deep stick test vs. 24 hrs urine collection test.</td>
<td>The KFW will gradually add support to utilize probabilistic algorithms, e.g. fuzzy logic, where appropriate. Thus, the final output may include a conclusion together with its confidence level.</td>
</tr>
<tr>
<td>27</td>
<td>Support import/export of KMs from one KB to another.</td>
<td>High</td>
<td>Reuse and sharing of KMs.</td>
<td>The KFW will provide a mechanism to import and export KMs between different KFW repositories.</td>
</tr>
<tr>
<td>28</td>
<td>Provide verification capabilities to ensure the validity of KMs definitions.</td>
<td>High</td>
<td>The quality and validity of knowledge acquired by human users, i.e., to err is human.</td>
<td>The KFW will gradually incorporate verification modules that can analyze the KM definition and generate warnings and errors as output accordingly.</td>
</tr>
<tr>
<td>#</td>
<td>Need</td>
<td>Priority</td>
<td>Concerns</td>
<td>Proposed Solutions</td>
</tr>
<tr>
<td>----</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>29.</td>
<td>Support import of different types of knowledge definitions.</td>
<td>Low</td>
<td>Reuse and sharing of KMs.</td>
<td>The KFW will provide a mechanism to import different knowledge definitions formats, e.g. OpenEHR Archetypes, and conversion to a suitable KM format.</td>
</tr>
<tr>
<td>30.</td>
<td>Support export of KMs to different types of knowledge definitions.</td>
<td>Medium</td>
<td>Reuse and sharing of KMs.</td>
<td>The KFW will provide a mechanism to export KMs at different knowledge definitions formats, e.g. OpenEHR Archetypes.</td>
</tr>
<tr>
<td>31.</td>
<td>Enable to test a KM before using it.</td>
<td>Medium</td>
<td>A safety measure before publishing KMs to a</td>
<td>The KFW will gradually provide support to simulate and test the process of evaluating a KM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>production environment.</td>
<td></td>
</tr>
<tr>
<td>Consumer Benefit</td>
<td>Supporting Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Decrease information overload on users by analyzing large amounts of data quickly and efficiently.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Assist in detecting meaningful data abstractions, e.g., patterns and trends, otherwise hidden in the data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data interpretation and inference capability. For example, abstract quantitative data to qualitative data (e.g., platelet count &lt; 150,000 cells/mm³ → low platelet state) or applying temporal reasoning techniques to create time interval abstractions from large series of time-stamped raw data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Consume services in an interoperable and transparent way.</td>
<td>KFW is a SOA-based system. The services, interfaces, and external payload contracts it provides adheres to standards when available and appropriate.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Exchange information using standardized data constructs.</td>
<td>In general, the data requirements needed to evaluate a KM can be provided in an iterative manner. The KFW will provide as part of its service interface methods to retrieve different aspects of a KM including its data requirements. Moreover, the KFW will also provide a method that returns the minimal data requirements to evaluate a KM given inputs such as data currently available to consumer.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Facilitate consumers to optimize the KM evaluation requests with respect to data requirements, e.g., gather data only when absolutely necessary.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Allow to retrieve KM data requirements for evaluation.</td>
<td>KFW provides a robust query language. The query can support the evaluation of dynamic, on-the-fly KM, as well as adding existing KMs. For example, has patient X had KM Value = Low during the last 6 months.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Evaluate requests without having to create a KM first.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Delegate logical processing entirely to the KFW.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consumer Benefit</td>
<td>Supporting Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DSS support that involves Boolean expressions as well as declarative and procedural knowledge.</td>
<td>A robust KM representation model that contains a common section to capture descriptive information and specific section types to capture different knowledge types (e.g. declarative versus procedural knowledge). A KM definition can rely on other KMs using relations such as inheritance and aggregation. Each KM will have a version and history of previous versions definitions will be available to maintain backward compatibility.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Simplify knowledge acquisition process by reusing existing KMs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Enable consumers and users to comprehend KM scope, goal, and function.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Maintain backward compatibility when modifying KM definitions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Facilitate semantic interoperability on the knowledge level.</td>
<td>Embed terms and concepts originating from controlled medical vocabularies in KMs definitions (dbMotion CTS).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Share knowledge across different clinical settings → “define once, use anywhere”</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Assure timely requests to apply knowledge to data.</td>
<td>Automatically extract recommendation to data monitoring that may trigger evaluation of a KM. This will facilitate consumers, for example SmartWatch, to lookout for the specific situations that call for patient re-evaluation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Avoid redundant requests that reveal no new information.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Enables domain experts and knowledge engineers to create and maintain KMs.</td>
<td>Domain experts or knowledge engineers create and maintain KMs in a user-friendly and graphical environment.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Does not necessitate advanced software skills.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Accelerates the knowledge acquisition process through simplification and reuse.</td>
<td>Assist the knowledge editors during the knowledge acquisition process by providing reusable templates, wizards, and macros to create and maintain KMs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Allow users with limited computer skills to have an active role in the process.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Enables to find KMs during design-time, e.g. knowledge engineer.</td>
<td>Search and retrieve KM by submitting search queries based on KM descriptive information traits.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Enables consumers to find KMs during runtime, e.g. SmartGuard.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consumer Benefit</td>
<td>Supporting Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Understand the logic leading to a conclusion.</td>
<td>Attach to each conclusion a human-readable and machine-interpretable explanation of the facts that lead to it. For example, the hematocrit and haptoglobin serum levels of a patient leading to the conclusion of having ongoing hemolysis together with its knowledge definition. When possible, a conclusion will also include a reference that supports the conclusion, e.g., a peer-reviewed literature paper.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Understand the facts leading to a conclusion.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Promote evidence-based medicine.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Maintain inference capability to some extent despite incomplete or insufficient data/knowledge.</td>
<td>Apply inference methods using probabilistic algorithms when appropriate, e.g., Bayesian Networks, Fuzzy Logics.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Reflect reliability levels of facts leading to a conclusion, e.g. urine dipstick test vs. 24 hrs urine collection.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Address information needs based on clinical context.</td>
<td>Generate a VPO schema recommendation based on patient clinical state and other factors such as care provider preferences (e.g., clinical specialty).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Decrease information overload.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Control access to system resources and services.</td>
<td>Provide a secured and safe environment at design-time and at runtime too.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Adhere to privacy and security regulations.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Assist in management and maintenance operations.</td>
<td>Enable to track and log at different level of details input requests, evaluation process, evaluation outputs, and knowledge acquisition activities.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Provide concise information about errors.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Enable usage analysis and users/consumers behavior.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Share knowledge between different clinical settings.</td>
<td>The KFW will support import/export operations that enable to share KMs between different KFW installations and exchange knowledge in different formats.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Exchange clinical knowledge in different formats.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 8C**
class cts

<<Interface>>
CTSRuntime

<<Interface>>
CTSSConceptRT
+ sameConcept
   (CTSSConceptRT): boolean

<<Interface>>
CTSRuntime
+ contains
   (CTSSConceptRT): boolean

FIG. 9

class km

RootEntity

entity::Artifact
+ getData(): byte[]
+ getModificationDate(): Date
+ getType(): String
+ setData(byte[]): void
+ setModificationDate(Date): void
+ setType(String): void

FIG. 10B
FIG. 11B
<EvaluationResponse>
  <FinalKMEvaluationResponses>
    <FinalKMEvaluationResponse>
      <KnowledgeModuleId businessId="isHFPatient">
    scopingEntityId="com.dbMotion.cms.hf" version="1.0.0"/>
      <EvaluationResults>
        <EvaluationResult>
          <EvaluationResultId businessId="isHFPatient">
            <ContainingEntityId
            businessId="isHFPatient">
          scopingEntityId="com.dbMotion.cms.hf" version="1.0.0"/>
          </EvaluationResultId>
          <SemanticPayload>
            <InformationModelSSID businessId="conclusion.booleanConclusion">
          scopingEntityId="com.dbMotion" version="1.0.0"/>
            <BooleanConclusion booleanAnswer="true" effectiveTime="2009-06-15T15:12:33.091" flavor="Computed" title="Is HF Patient">
              <id extension="f69984f7-d7b3-46e0-b311-43382b8776dc" root="2.16.840.1.113883.3.57.1.4.5.1"/>
            </BooleanConclusion>
            </SemanticPayload>
        </EvaluationResult>
      </EvaluationResults>
    </FinalKMEvaluationResponse>
  </FinalKMEvaluationResponses>
</EvaluationResponse>

FIG. 12E
FIG. 15

Clinical Application (complementary application)

SMS

SmartWatch

dbMotion CorePlatform

Either Clinical Views or an external application

Close the loop

Subscribe

Messages

EMail
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCP</td>
<td>Patient’s Primary Care Physician assuming responsibility on his/her clinical condition. A central consumer of SmartWatch, in the context of actions (notifications, reminders, reports) regarding his patients. Communication with this user depends on many factors but first on the PCP’s communication channels (does he have an EMR? Can we push messages there? Is phone the preference?)</td>
<td>Represented by CTO &amp; Clinical team members</td>
</tr>
<tr>
<td>Non-PCP clinician</td>
<td>Clinicians that are not the primary care takers. These could be hospital physicians and nurses, care managers and so on. Are considered consumers for SmartWatch outputs in the context of patients related to them somehow (e.g. in their ward). In this case, the preferred communication channels are thru the user’s standard application workflow.</td>
<td>Represented by CTO &amp; Clinical team</td>
</tr>
<tr>
<td>Patient</td>
<td>Patient may want to consume SmartWatch outputs aimed at him. This can be achieved either in direct channels or indirect channels (PHR-like).</td>
<td>CTO</td>
</tr>
</tbody>
</table>

FIG. 17
CTS Extension Framework Use Cases

- Determine Child BASELINE Concepts for LOCAL ValuesSet
- Determine Parent BASELINE Concepts for BASELINE ValuesSet
- Get Concept Details
- Fill Concept Information in Vocabulary Business Aspect
- + Develop Query Extension

More complicated internal product extended query solutions which are based on extension framework.

CTS Extended Solution Use Cases

- Determine LOCAL Concepts for LOCAL ValuesSet by BASELINE
- Determine LOCAL Concepts for LOCAL ValuesSet by Parent
- BASELINE Concepts in <Context>
- DETERMINE Child LOCAL Concepts for LOCAL ValuesSet by BASELINE
- BASELINE Concepts in <Context>
FIG. 22

uc Get Service Address

FIG. 23

uc Manage Extension

FIG. 24

uc Manage Named Query Metadata
uc CTS Extended Solution Use Cases

1. For local ValuesSet determine baselines
2. For baselines determine all locals

1. For local ValuesSet determine baseline concepts
2. For baseline concepts determine parent baseline concepts in the specific context
3. For parent baseline concepts determine all local concepts

1. For local ValuesSet determine baseline concepts
2. For baseline concepts determine child baseline concepts in the specific context
3. For parent baseline concepts determine all local concepts
pkg CTS First Analysis Use Cases

CTS Metadata Use Cases
- Create Query
  - Get Build Ontology Methodology Recommendations
  - Get Named Predefined Query List
  - Get Named Query Parameters
  - Get Ontology Schema
  - Save Named Predefined Query

CTS Management Use Cases
- Add Local Code
  - Administrate
  - Compile Ontology
  - Export Ontology
  - Import Ontology
  - Manage
  - Manage Extended Ontology
  - Manage Local Terminology
  - Manage Ontology
  - Map Codes
  - Release Ontology
  - Synchronize Ontology in Enterprise
  - Validate Ontology

FIG. 28
act Add Local Code

Start

Add Local Code

Is dbMotion Ontology

dbMotion CTS Expert

Add Local Code to dbMotion Ontology

Map Local Code to dbMotion Baseline

Validation/Test Ontology Process

Export Ontology

Notify to Administrator

CTS Administrator

Import Ontology

Correspond Imported and Released Ontologies

Notify about Release Ontology

Release Ontology

Reload Services Cache

Synchronize Ontology in Enterprise

Finish

FIG. 30
uc CTS Metadata Use Case View

Ontology section
- Get Build Ontology Methodology Recommendations
- Get Ontology Schema

Query section
- Get Named Predefined Query List
- Get Named Query Parameters
- Save Named Predefined Query

Business Use Cases
- Create Query

FIG. 31
class Design Model

The Design Model consists of the Logical Model and the Data Model.

The Logical model includes the System and Framework sections. Both consist of classes and artifacts which define the structure of the code used in the application under development.

The Data Model defines the data structure to be used in the system under development.

---

Logical Model
- [ ] + System
- [ ] + Framework

Data Model
The Logical Model is a model of the software system under construction. It consists of Classes which generally have a direct relationship to source code or other software artifacts that can be grouped together into executable components.

The System package contains the classes and artifacts which are being built or designed as part of the current model. The Framework package generally contains classes and components that have been designed and built earlier, and are being reused as part of the current project.
**class Client**

```
<<interface>>
ObjectConverter
+ open(object...) : void
+ close() : void
+ readNextObject() : ManagementObjectBase
+ writeObject() : void
```

**class Service**

```
<<interface>>
ManagementContentService
+ getScopesList() : String[]
+ getScopeDef(String) : ManagementScope
+ createScope(ManagementScope) : void
+ saveObjects(String, ManagementObjectBase[], boolean) : ManagementObjectReport[]
+ saveObject(String, ManagementObjectBase) : ManagementObjectReport
```

**class Tools**

```
<<interface>>
AdministrationManagementService
```

**FIG. 47**

**FIG. 48A**

**FIG. 48B**
class Shared

Information Model

Query

- + Request Data Contract
  - Query Functions Specification
- + Shared Query Model
- + Request Data Contract
- + Response Data Contract

Metadata

Management

FIG. 51
cts::Concept
- id: String
- code: String
- codeSystemId: String
- type: ConceptType = ConceptType.ONTOLOGY
- classNames: List<String> = new ArrayList<String> {readOnly}
- effectiveTimeStart: Date
- effectiveTimeEnd: Date
- designations: ArrayList<Designation> = new ArrayList<Designation> {readOnly}
- sourceRelations: ConceptRelationsCollection = null
- targetRelations: ConceptRelationsCollection = null
«Framework»
- codeSystem: CodeSystem
«Calculated»
- isObsolete: boolean = false

+ Concept(String, String, String)
+ addDesignations(Designation) : void
- checkPropertiesEquality(Object) : boolean
+ clone() : Concept
- doesDesignationExist(DesignationType) : boolean
+ equals(Object) : boolean
+ equalsLogically(Object) : boolean
+ getDesignationsByType(DesignationType) : Designation[]
+ getRelatedConceptsByType(String) : Concept[]
+ getRelatedSourceConcepts() : Concept[]
+ getRelatedTargetConcepts() : Concept[]
+ getRelations() : ConceptRelation[]
+ getRelationsByType(String) : ConceptRelation[]
+ hashCode() : int
«Framework»
+ Concept()
+ Concept(Concept)
+ Concept(String)
+ Concept(String, String)
+ Concept(String, String, Date)
+ addDesignation(Designation) : void
+ getCodeSystem() : CodeSystem
+ getDesignationsByLocale(String) : Designation[]
+ getObjectType() : ObjectType

FIG. 53
### cts::ObjectsCollection

- objects: List<ObjectBase> = new ArrayList<ObjectBase>(); (readOnly)

+ ObjectsCollection(ObjectsCollection)
+ addAllObjects(List<ObjectBase>): void
+ addAllObjects(ObjectBase): void
+ addObject(ObjectBase): void
+ clone(): ObjectsCollection
+ contains(ObjectBase): boolean
+ containsLogical(ObjectBase): boolean
+ equals(Object): boolean
+ findObject(TObject): TObject
+ getObjects(): List<ObjectBase>
+ getObjectType(): ObjectType
+ getTypedObjects(TObject): List<TObject>
+ setObjectsId(String): void

### cts::CodeSystem

- id: String
- localName: String
- fullName: String
- description: String
- scopeName: String
- copyright: String
- editableCodesIndication: boolean = false
- lastModified: Date
- lastPublished: Date
- version: String

```java
«Framework»
```
- concepts: ConceptsCollection

- checkPropertiesEquality(Object): boolean
+ clone(): CodeSystem
+ equals(Object): boolean
+ equalsLogically(Object): boolean
+ hashCode(): int

```java
«Framework»
```
+ CodeSystem()
+ CodeSystem(CodeSystem)
+ CodeSystem(String)
+ getConceptByCode(String): Concept
+ getConcepts(): List<Concept>
+ getObjectType(): ObjectType

**FIG. 54**
**cts::ObjectBase**

- parameters: ValuesId: String
- properties: HashMap<String, String> = null

```java
# ObjectBase(String)
+ clone : () ObjectBase
+ equals(Object) : boolean
+ equalsLogically(Object) : boolean
+ hashCode() : int

<Framework>
+ getObject() : ObjectBase
+ getObjectType() : ObjectType
```

**cts::ConceptRelation**

- relationType: ConceptRelationType
- sourceConceptId: String
- targetConceptId: String
- description: String

```java
<Framework>
- sourceConcept: Concept
- targetConcept: Concept

+ ConceptRelation(ConceptRelationType)
+ clone() : ConceptRelation
+ equals(Object) : boolean
+ getRelationType() : ConceptRelationType
+ hashCode() : int

<Framework>
+ ConceptRelation
+ ConceptRelation(ConceptRelation)
+ getObjectType() : ObjectType
```

**cts::Designation**

- type: DesignationType
- id: String

```java
+ Designation(Designation)
+ Designation(DesignationType)
+ clone() : Designation
+ equals(Object) : boolean
+ hashCode() : int

<Framework>
+ Designation
```

**cts::TextValueItem**

- locale: String = ""
- text: String

```java
+ TextValueItem()
+ TextValueItem(TextValueItem)
+ clone() : TextValueItem
+ equals(Object) : boolean
+ hashCode() : int
```

**FIG. 55**

**FIG. 56**

**FIG. 57**
management::ManagementScope

- name: String
- targetScopeName: String
- description: String
- objectSettings: ManagementObjectSettings
- assignments: ManagementAssignmentsCollection = null

ObjectBase

- parametersValuesId: String
- properties: HashMap<String, String> = null

Framework

- targetScope: ManagementScope

+ ManagementScope(ManagementScope)
+ clone() : ManagementScope
+ equals(Object) : boolean
+ getTargetScopeName() : String
+ hashCode() : int
+ setTargetScopeName(String) : void

Framework

+ ManagementScope()
+ ManagementScope(String)
+ getObjectType() : ObjectType
+ getTargetScope() : ManagementScope
+ setTargetScope(ManagementScope) : void

FIG. 59
<table>
<thead>
<tr>
<th>management::ManagementCodeSystem</th>
<th>CodeSystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>- assignmentId: String</td>
<td></td>
</tr>
<tr>
<td>- originalCodeSystemId: String</td>
<td></td>
</tr>
<tr>
<td>- conceptPropertyTypes: List&lt;ManagementPropertyType&gt; =</td>
<td></td>
</tr>
<tr>
<td>new ArrayList&lt;ManagementPropertyType&gt; = null</td>
<td></td>
</tr>
<tr>
<td>::CodeSystem</td>
<td></td>
</tr>
<tr>
<td>- id: String</td>
<td></td>
</tr>
<tr>
<td>- localizedName: String</td>
<td></td>
</tr>
<tr>
<td>- fullName: String</td>
<td></td>
</tr>
<tr>
<td>- description: String</td>
<td></td>
</tr>
<tr>
<td>- scopeName: String</td>
<td></td>
</tr>
<tr>
<td>- copyright: String</td>
<td></td>
</tr>
<tr>
<td>- editableCodesIndication: boolean = false</td>
<td></td>
</tr>
<tr>
<td>- lastModified: Date</td>
<td></td>
</tr>
<tr>
<td>- lastPublished: Date</td>
<td></td>
</tr>
<tr>
<td>- version: String</td>
<td></td>
</tr>
<tr>
<td>::ObjectBase</td>
<td></td>
</tr>
<tr>
<td>- parametersValuesId: String</td>
<td></td>
</tr>
<tr>
<td>- properties: HashMap&lt;String, String&gt; = null</td>
<td></td>
</tr>
</tbody>
</table>

```
<Framework>
- scope: ManagementScope
- assignment: ManagementAssignment
::CodeSystem
- concepts: ConceptsCollection

+ clone(): CodeSystem
+ equals(Object): boolean
```

FIG. 60
management::ManagementAssignment

- objectSettings: ManagementObjectSettings
- id: String
- name: String
- creatorUser: String
- assignedUser: String
- managerUser: String
- deadline: Date
- initiator: String
- lastModified: Date
- timeLocked: Date = null
- isStarted: boolean = false
- type: ManagementAssignmentType
- scopeName: String

::ObjectBase
- parametersValuesId: String
- properties: HashMap<String, String> = null
 «Calculated»
- isLocked: boolean = false
 «Framework»
- scope: ManagementScope
- codeSystems: ManagementCodeSystemsCollection = null
- concepts: ManagementConceptsCollection = null
- conceptRelations: ManagementConceptRelationsCollection = null

+ ManagementAssignment(ManagementAssignment)
+ clone() : ManagementAssignment
+ equals(Object) : boolean
+ getConceptRelations() : List<ManagementConceptRelation>
+ getConcepts() : List<ManagementConcept>
+ hashCode() : int
 «Framework»
+ ManagementAssignment()
+ ManagementAssignment(String)
+ getObjectType() : ObjectType

FIG. 61
<table>
<thead>
<tr>
<th>management::ManagementConcept</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>- objectSettings: ManagementObjectSettings</td>
<td></td>
</tr>
<tr>
<td>- assignmentId: String</td>
<td></td>
</tr>
<tr>
<td>- intentionChangedIndication: boolean</td>
<td></td>
</tr>
<tr>
<td>- version: String</td>
<td></td>
</tr>
<tr>
<td>- activationDate: Date</td>
<td></td>
</tr>
<tr>
<td>- deactivationDate: Date</td>
<td></td>
</tr>
</tbody>
</table>

::Concept
- id: String
- code: String
- codeSystemId: String
- type: ConceptType = ConceptType.ONTOMETRY
- classNames: List<String> = new ArrayList<String> [readOnly]
- effectiveTimeStart: Date
- effectiveTimeEnd: Date
- designations: ArrayList<Designation> = new ArrayList<Designation> [readOnly]
- sourceRelations: ConceptRelationsCollection = null
- targetRelations: ConceptRelationsCollection = null

::ObjectBase
- parametersValuesId: String
- properties: HashMap<String, String> = null

«Calculated»
- isActive: boolean = true

::Concept
- isObsoleten: boolean = false

«Framework»
- assignment: ManagementAssignment

::Concept
- codeSystem: CodeSystem

+ ManagementConcept(String, String, String)
+ clone() : ManagementConcept
+ equals(Object) : boolean

«Framework»
+ ManagementConcept()
+ ManagementCodeSystem(ManagementCodeSystem)
+ ManagementCodeSystem(String)
management::ManagementConcept

- objectSettings: ManagementObjectSettings
- assignmentId: String
- intentionChangedIndication: boolean
- version: String
- activationDate: Date
- deactivationDate: Date

::Concept
- id: String
- code: String
- codeSystemId: String
- type: ConceptType = ConceptType.ONTObLOGY
- classNames: List<String> = new ArrayList<String>... {readOnly}
- effectiveTimeStart: Date
- effectiveTimeEnd: Date
- designations: ArrayList<Designation> = new ArrayList<Date>... {readOnly}
- sourceRelations: ConceptRelationsCollection = null
- targetRelations: ConceptRelationsCollection = null

::ObjectBase
- parametersValuesId: String
- properties: HashMap<String, String> = null

«Calculated»
- isActive: boolean = true

::Concept
- isObsolete: boolean = false

«Framework»
- assignment: ManagementAssignment

::Concept
- codeSystem: CodeSystem

+ ManagementConcept(String, String, String)
+ clone() : ManagementConcept
+ equals(Object) : boolean

«Framework»
+ ManagementConcept()
+ ManagementConcept(ManagementConcept)
+ ManagementConcept(String, String)
class Management Data Contracts

Management : Management Query Data Contract

management::ManagementContentObjectsCollection

- list: List<ManagementContentObject> = new ArrayList<ManagementContentObject> {readOnly}

«interface»
management::ManagementObject

+ getObject() : ObjectBase
+ getObjectSettings() : ManagementObjectSettings
+ setObjectSettings(ManagementObjectSettings) : void

ValidationContent
management::ManagementContentObject

- orderid: int = -1
- values: Map<String, Object> = null
- operation: ManagementOperation = null

FIG. 64
FIG. 74

```
<table>
<thead>
<tr>
<th>FunctionBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>function:: FormattingFunction</td>
</tr>
<tr>
<td>{leaf}</td>
</tr>
<tr>
<td>- formattingParameter: FormattingParameter</td>
</tr>
<tr>
<td>+ FormattingFunction()</td>
</tr>
<tr>
<td>+ FormattingFunction(FormattingParameter)</td>
</tr>
<tr>
<td>+ getFormattingParameter(): FormattingParameter</td>
</tr>
<tr>
<td>+ setFormattingParameter(FormattingParameter): void</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionParameterBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter:: FormattingParameter</td>
</tr>
<tr>
<td>- formattingType: FormattingParameterType</td>
</tr>
<tr>
<td>+ FormattingParameter()</td>
</tr>
<tr>
<td>+ FormattingParameter(FormattingParameterType)</td>
</tr>
<tr>
<td>+ getFormattingType(): FormattingParameterType</td>
</tr>
<tr>
<td>+ getFunction(): FunctionBase</td>
</tr>
<tr>
<td>+ hashCode(): int</td>
</tr>
<tr>
<td>+ merge(FunctionParameterBase): void</td>
</tr>
<tr>
<td>+ setFormatting(FormattingParameterType): void</td>
</tr>
</tbody>
</table>

<enumeration>

<table>
<thead>
<tr>
<th>parameter:: FormattingParameterType</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAT_LIST</td>
</tr>
<tr>
<td>BY_PARAMETER_VALUE_SET_ID</td>
</tr>
<tr>
<td>DISTINCT_FLAT_LISTWITHOUT_PARAMETER_VALUE_SET_ID</td>
</tr>
</tbody>
</table>
```
**FunctionParameterBase**

- `isAllAspectsRequired`: boolean = false

  # `containsAspect(Object : )boolean`
  + `getFunction()`: FunctionBase
  + `isAllAspectsRequired()`: boolean
  + `isAspectRequired(Object)`: boolean
  + `merge(FunctionParameterBase)`: void

  # onEnableAllAspects(): void

  # `onMergeRestrictionParameter(RestrictionParameterBase : )void`
  + `setAllAspectsRequired(boolean)`: void

**LocaleRestrictionParameter**

- `locales`: List<String> = new ArrayList<String... {readOnly}

  + `LocaleRestrictionParameter()`
  + `LocaleRestrictionParameter(boolean)`
  + `LocaleRestrictionParameter(String)`
  + `addLocale(String)`: void

  # `containsAspect(Object)`: boolean

  - `doesLocalesContain(String)`: boolean
  + `getLocales()`: List<String>
  + `hashCode()`: int

  # `onEnableAllAspects()`: void

  # `onMergeRestrictionParameter(RestrictionParameterBase)`: void

*FIG. 77*

From FIG. 76

+ restrictions

From FIG. 78
class Client

Query
  + Service Access

Management
  + Service Access

class Management

Service Access

FIG. 79B

FIG. 79C

class Service Access Total Diagrams

See FIG. 81

See FIG. 82

See FIG. 83

FIG. 80
management::ManagementServiceAdapter

+ DEFAULT_MANAGEMENT_SERVICE_EXT_ENDPOINT_NAME: String = "ManagementService" {readOnly}
- instance: ManagementServiceAdapter = new ManagementS...

+ get Instance() : ManagementServiceAdapter
+ set Instance(ManagementServiceAdapter) : void
+ ManagementServiceAdapter() 
+ ManagementServiceAdapter(EndPoint)
+ executeView(String) : ManagementDataSet
+ executeView(String, FunctionParameterBase[]) : ManagementDataSet
+ executeView(String, ObjectBase[]) : ManagementDataSet
+ executeView(String, ObjectBase[], FunctionParameterBase[]) : ManagementDataSet
+ fill(ManagementDataSet, String) : void
+ fill(ManagementDataSet, String, ObjectBase[]) : void
+ fill(ManagementDataSet, String, ObjectBase[], FunctionParameterBase[]) : void
+ fill(ManagementDataSet, String, RestrictionParameterBase[]) : void
+ get ManagementServiceProxy() : ManagementService
- get ManagementServiceProxy(EndPoint) : ManagementService
- get ManagementServiceProxy(String) : ManagementService

FIG. 85
class Service Access

- Client Basic Scenarios

### MetadataServiceAdapter

- **framework**

```java
query::QueryServiceAdapter
```

+ DEFAULT_QUERY_PROCESSOR_EXT_ENDPOINT_NAME: String = "QueryProcessor" {readOnly}
+ DEFAULT_QUERY_SERVICE_EXT_ENDPOINT_NAME: String = "QueryService" {readOnly}
- instance: QueryServiceAdapter = new QueryServiceAdapter

+ getInstance() : QueryServiceAdapter
+ setInstance(QueryServiceAdapter) : void
+ QueryServiceAdapter()
+ QueryServiceAdapter(EndPoint)
+ execute(CommandBase) : QueryDataSet
+ execute(CommandBase, String) : QueryDataSet
+ execute(String) : QueryDataSet
+ execute(String, String) : QueryDataSet
+ execute(String, String[]) : QueryDataSet
+ execute(String, String[], String) : QueryDataSet
+ fill(QueryDataSet, CommandBase) : void
+ fill(QueryDataSet, CommandBase, String) : void
+ fill(QueryDataSet, String) : void
+ fill(QueryDataSet, String, String[]) : void
+ getQueryProcessorProxy() : QueryProcessor
+ getQueryProcessorProxy(String) : QueryProcessor
+ getQueryServiceProxy() : QueryService
+ getQueryServiceProxy(EndPoint) : QueryService
+ getQueryServiceProxy(String) : QueryService

↓ «use»

See FIG. 89A

See FIG. 89B
<framework>
query::QueryDataSet
- otherObjects: Map<Integer, ObjectBase> = new HashMap<Integer... {readOnly}
- objectsCollections: Map<Integer, ObjectsCollection> =
  new HashMap<Integer... {readOnly}
- codeSystems: Map<Integer, CodeSystem> = new HashMap<Integer... {readOnly}
- concepts: Map<Integer, Concept> = new HashMap<Integer... {readOnly}

+ containsCodeSystems() : boolean
+ containsConcepts() : boolean
+ containsObjectCollections() : boolean
+ containsOtherObjects() : boolean
+ getCodeSystem(String) : CodeSystem
+ getCodeSystemsByParametersValuesId(String) : Collection<CodeSystem>
+ getConcept(String) : Concept
+ getConcept(String, String) : Concept
+ getConcepts(String, String) : Collection<Concept>
+ getConceptsByParametersValuesId(String) : Collection<Concept>
+ getObjectsCollectionByParametersValuesId(String) : ObjectsCollection
+ getCodeSystem(int) : CodeSystem
+ getConcepts() : Collection<Concept>
+ getConcept(int) : Concept
+ getCodeSystems() : Collection<CodeSystem>
# getObject(int, Collection<TObject>) : TObject
+ getObjectsCollection(int) : ObjectsCollection
+ getObjectsCollections() : Collection<ObjectsCollection>
+ getOtherObject(int) : ObjectBase
+ getOtherObjects() : Collection<ObjectBase>
# indexCodeSystem(CodeSystem) : void
# indexConcept(Concept) : void
# indexConceptRelation(ConceptRelation) : void
+ putObject(ObjectBase) : ObjectBase
+ putObjects(Collection<? extends ObjectBase>) : void
+ reindex() : void
# registerObject(TObject, Map<Integer, TObject>) : TObject
+ putObjects(ObjectBase) : void

FIG. 89A
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  <bean id="service.model" class="com.dbmotion.common.service.configuration.ServiceModel">
    <property name="endpoints" ref="client.endpoints" />
  </bean>

  <bean id="client.endpoints" class="com.dbmotion.common.service.configuration.EndPointsList">
    <constructor-arg>
      <list>
        <property name="name" value="CTS.QueryService" />
        <bean class="com.dbmotion.common.service.configuration.EndPoint">
          <property name="binding" value="FILE" />
          <property name="address" value="\StubRepository" />
          <property name="description" value="DON'T REMOVE: testing of default CTS query service endpoint and default service provider" />
        </bean>
        <property name="provider" ref="cts.query.provider" />
      </list>
    </constructor-arg>
  </bean>

  <bean id="cts.query.provider" class="com.dbmotion.cts.client.impl.query.QueryServiceProviderImpl">
    <property name="description" value="CTS query service provider" />
    <property name="path" value="??" />
  </bean>

</beans>

FIG. 89B
Current cache-optimization should include the following steps:
1. For every bulk parameters attempt to find cached object
2. In case if exists -
   - put to QueryDataSet
   - remove bulk from existing ParametersSet
...
N. In analysis of result cache objects in the Cache by bulk hash-code and endpointname
class Query Processor

- processing::QueryProcessor
  + createExecutionPlan(Query, ParametersValuesSet) : QueryExecutionPlan
  + process(QueryExecutionPlan) : Output

- «create»
  - processing::QueryExecutionPlan

- «interface»
  - task::FilterTask
    + createFilterExpression() : Object
    + executeFilter(Object, Object) : void
    + getFilter() : FilterBase
    + setFilter(FilterBase) : void

- «interface»
  - task::Task
    + execute(Object) : void
    + getObjects() : ObjectsCollection
    + getParametersValuesSet() : ParametersValuesSet
    + setModelManager(ModelManager) : void
    + setObjects(ObjectsCollection) : void
    + setParametersValuesSet(ParametersValuesSet) : void
class Query Processor - Abstract Model

See FIG. 95

«interface»
processing::QueryExecutionPlan

See FIG. 96

«interface»
task::Task

+ AbstractFilterTask(ModelManager)
+ createFilterExpression : ()Object
+ execute(Object) : void
+ executeFilter(Object, Object) : void
+ getFilter() : FilterBase
+ setFilter(FilterBase) : void

FIG. 94
**processing::AbstractQueryExecutionPlan**

- tasksChain: List<Task> = new ArrayList<T>...
- query: Query = null
- parametersValuesSet: ParametersValuesSet = null

+ AbstractQueryExecutionPlan()
+ AbstractQueryExecutionPlan(Query, ParametersValuesSet)
+ getTasksChain(): Task[]
+ addTask(Task): void
+ getQuery(): Query
+ setQuery(Query): void
+ getParametersValuesSet(): ParametersValuesSet
+ setParametersValuesSet(ParametersValuesSet): void

**FIG. 95**

**task::AbstractTask**

- parametersValuesSet: ParametersValuesSet
- objects: ObjectsCollection
- functionsChain: FunctionsChain
- modelManager: ModelManager

+ AbstractTask(ModelManager)
+ execute(Object): void
+ getFunctionsChain(): FunctionsChain
- getLocaleRestrictions(): String[]
+ getModelManager(): ModelManager
+ getObjects(): ObjectsCollection
+ getParametersValuesSet(): ParametersValuesSet
+ setAction(FunctionsChain): void
+ setModelManager(ModelManager): void
+ setObjects(ObjectsCollection): void
+ setParametersValuesSet(ParametersValuesSet): void

**FIG. 96**
class Shared

```
class SessionManagement

+ SessionContainer
+ SessionFactory
- SessionManagerImpl
< O + ISessionManager

FIG. 97
```

class SessionManagement

```
<<interface>>

ISessionManager

+ createSession(string) : SessionContainer
+ removeSession(string) : void
+ getSession(string) : SessionContainer

SessionFactory

+ GetSessionManager() : ISessionManager

SessionManagerImpl

Create

FIG. 98A
```

SessionContainer

```
- SessionId: string
- AbsoluteTime: date
- ExpiresTime: int
- LastAccess: date
- Content: java.util.Map<string,object>

+ SessionContainer(java.util.Date, int) : void
+ getExpiredState() : boolean

<<property get>>

+ getSessionId() : string
+ getExpiresTime() : int
+ getAbsoluteTime() : date
+ getContent() : java.util.Map<string,object>
```

FIG. 98A
The diagram shows a class hierarchy and annotated text:

**Class System Services**
- **Query Processor**
  - + QueryProcessorImpl

**Class Metadata Service**
- MetadataServiceImpl
  - + MetadataServiceImpl

**Class Query Processor**
- QueryProcessorImpl
  - Analyser - parsing and logical analysing of query text. Factory query object should work through Analyser

*FIG. 98B* *FIG. 98C* *FIG. 98D*
class Service Contracts

```
«interface»
QueryService
+ executeCommand(Command) : OutputBase
+ getNextResponse(string) : OutputBase
+ disconnectFromRequest(string) : void
```

FIG. 101

class Data Model

This model describes the data which must be stored and retrieved as part of the overall system design.

Typically this will mean relational database models which describe the tables and data in detail and allow generation of DDL scripts to create and setup databases.

- Read about Data Modeling
- View Further Examples
- How to import an existing schema
- How to generate DDL scripts

A schema package contains a logical grouping of tables.

FIG. 102
The Implementation Model defines how classes, artifacts and other low level elements are collected into high level components and the interfaces and connections between them.

Components are compiled software artifacts that work together to provide the required behavior within the operating constraints defined in the requirements model. Components will generally be deployed to varying hardware platforms described in the Deployment Model.

The Components package contains modeled components and their structural constituents. These include additional exposed interfaces, ports and other gateways or internal structural components. The connectivity and internal structure of these are further modeled in the Internal Structures and Connections packages.

Internal Structures provide a detailed view of the internal workings and dependencies of a component. Using a Composite Structure diagram, they illustrate how the component fulfills its behavioral contracts and provides interface behavior to other components within the system.

The Connections package models the dependencies and connectivity between the various components, and how each is used as part of a co-operative system to accomplish required tasks. Typically, Components expose interfaces and API's which are used by other Components.

Read about Component Modeling

View Further Example

~~

cts-api

+ com

~~
<table>
<thead>
<tr>
<th>Operation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Evaluation</td>
<td>Data Event Monitor triggers SmartWatch evaluation based on the new data that arrived to dbMotion CDR</td>
</tr>
<tr>
<td>Trigger Evaluation</td>
<td>Temporal Monitor triggers SmartWatch evaluation based on time schedule</td>
</tr>
<tr>
<td>Get Patient Identifiers</td>
<td>SmartWatch retrieves all existing patient identifiers for the patient provided in the evaluation trigger</td>
</tr>
<tr>
<td>Evaluate</td>
<td>SmartWatch evaluates the patient based on the information in evaluation trigger, patient identifiers and other medical information that can be retrieved from dbMotion core System across the entire federation.</td>
</tr>
<tr>
<td>Perform Action</td>
<td>SmartWatch delegated action performing to Action Manager</td>
</tr>
</tbody>
</table>

FIG. 106B
ActivateSmartGuard - Basic Flow Fig. 107 - (Sequence diagram)

1: ChangeGuardRuntimeStatus()
   1.1: ChangeGuardRuntimeStatus()

   1.1.1: GetGuardDetails()
   1.1.2: GetGuardDetails()

GetSmartGuard

[SmartGuard has been modified]

1: ReconcileGuard()
2: ReconcileGuard()

[SmartGuard existing and hasn't been modified]

1: GetMembersByGuardId()
2: GetMembersByGuardId()
3: ActivateMemberSubscriptions()
   3.1: ActivateSubscriptions()
   3.2: ActivateSubscriptions()
4: ActivateMemberSubscriptions()

Activation handlers are built from scheduling contexts and sent to the Temporal Monitor's scheduling service.

Loop 1*

1: Activate()
   1.1: ActivateSubscriptions()
   1.2: ActivateSubscriptions()

2: Activate()
   1.1: ActivateSubscriptions()
   1.2: ActivateSubscriptions()

1.1.5: GetSchedulingContexts()
1.1.6: GetSchedulingContexts()

2: AddScheduling()
   ActivationHandler
   ActivationHandler

3: AddScheduling()

1.1.7: SaveGuardRuntimeState()
1.1.8: SaveGuardRuntimeState()

2: ChangeRuntimeStatus()
ProcessTask Fig. 112 - (Sequence diagram)

1: Process()

ref GetSmartGuard

1.1: GetEvaluationTaskById()

1.2: GetEvaluationTaskById()

loop [1,*]

1: GetRuleExecutor()

2: GetRuleExecutor()

1: from RuleInvocationSettings()

2: Evaluate()

loop [0,*]

1: Evaluate()

2: Evaluate()

alt [Rule failed evaluation]

1: ThrowRuleEvaluationException()

2: throwRuleEvaluationException()

alt [Rule Evaluated]

ref ProcessActivity

2: Process()}
Fig. 113

sd Activate Tasks

ST: SmartGuardTimer

1: OnTimeElapsed()

1.1: From registerDetails() => ac: ActivationContext

ref

HandleActivation

1.2: OnTimeElapsed()
Harmonization may include:
- Terminology (local to baseline)
- Unit of measurement

Patient exclusion (population discovery - don't tell me of people I know) would probably need to be evaluated separately, while inclusion (monitoring - tell only of people I know) can be implicit with subscription matching.
Update Subscription Fig.116 - (Activity diagram)
Fig. 118A
The design package will be split in two assemblies:

* Interfaces - part of product
* Default implementations for Initialize - application block to be customized at project level
GetClusterWithChanges - Basic Flow Fig. 128A - (Sequence diagram)

1: GetClusterWithChanges()
   1.1: create()
     Result: ClusterStateObject

1.2: SetCurrentCluster()
   1.3: SetCurrentCluster()

1.4: GetCluster/Subscriptions()
   1.5: GetCluster/Subscriptions()

[loop]
[0-1]
1: AddChangeLogEntry()
2: AddChangeLogEntry()

(from SWService)
RefreshPersonIndices - Basic Flow Fig.129 - (Sequence diagram)

This may include preferences added during subscriptions which indicate the desired frequency based in subscribers sensitivity to delays.

The call to GetCluster will update the new cluster and handle the rules. Note - force skipping the check in DB whether the cluster is fresh enough.
UpdateSubscription - Basic Flow Fig.132 - (Sequence diagram)

subService: SubscriptionService
subMgr: SubscriptionManager
mapping: PatternRuleMapping
dal: SubscriptionStorage
runtimeManager: RuntimeManager

1: UpdateSubscription()

loop
[0,]

1: LookupRuleMapping()
2: LookupRuleMapping()
3: construct according to mapping()

4: ValidateSubscriptionArguments()
5: ValidateSubscriptionArguments()
6: PrepareSubscriptionObject()
7: PrepareSubscriptionObject()

alt
not all subscription arguments are valid

1: throw exception()
2: throw exception()

all
all valid

1: StoreSubscriptions()
2: StoreSubscriptions()
3: OnSubscriptionChange()
4: OnSubscriptionChange()
SmartGuard Processing Use Cases

System Management
- Define & configure SmartGuard
- Manage SmartGuard runtime

Event Monitor
- Event Monitor
- Maintain Subscriptions, Patients and Indices
- Manage triggering rules subscriptions

Process General Element
- Discover new population element
- Activate Task(s)
- Process Known Population elements

Event Monitor
- Event Monitor
- SmartWatch Actions
- Execute action

PERIS
- Temporal Monitor
- Notify Consumer of new results
- Deliver results per consumer request

Person Identity Service

FIG. 139
<table>
<thead>
<tr>
<th>Full name</th>
<th>Count</th>
<th>Relevant?</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is a (attribute)</td>
<td>513378</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Finding site (attribute)</td>
<td>84118</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Episodicity (attribute)</td>
<td>68805</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Clinical course (attribute)</td>
<td>68688</td>
<td>Yes</td>
<td>Acute, Chronic</td>
</tr>
<tr>
<td>Severity (attribute)</td>
<td>68578</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Method (attribute)</td>
<td>58059</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Associated morphology (attribute)</td>
<td>57716</td>
<td>Yes</td>
<td>Associates finding to morphology</td>
</tr>
<tr>
<td>Priority (attribute)</td>
<td>55188</td>
<td>No</td>
<td>Emergency, routine qualifier</td>
</tr>
<tr>
<td>Part of (attribute)</td>
<td>47411</td>
<td>No</td>
<td>Body structure - why don’t we need it?</td>
</tr>
<tr>
<td>SAME AS (attribute)</td>
<td>41087</td>
<td>Import</td>
<td>Historical relationships</td>
</tr>
<tr>
<td>Access (attribute)</td>
<td>35772</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Procedure site - Direct (attribute)</td>
<td>33005</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>MAY BE A (attribute)</td>
<td>29078</td>
<td>Import</td>
<td>Historical relationships</td>
</tr>
<tr>
<td>Interprets (attribute)</td>
<td>25744</td>
<td>Yes</td>
<td>Associates Finding to a relevant measurement</td>
</tr>
<tr>
<td>Causative agent (attribute)</td>
<td>21359</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Has active ingredient (attribute)</td>
<td>17873</td>
<td>No</td>
<td>Meds</td>
</tr>
<tr>
<td>Laterality (attribute)</td>
<td>16412</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Has dose form (attribute)</td>
<td>10722</td>
<td>No</td>
<td>Meds</td>
</tr>
<tr>
<td>Component (attribute)</td>
<td>9636</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
<tr>
<td>Occurrence (attribute)</td>
<td>8888</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Finding method (attribute)</td>
<td>8471</td>
<td>No</td>
<td>Epistemology</td>
</tr>
<tr>
<td>Procedure site - indirect (attribute)</td>
<td>8217</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Direct morphology (attribute)</td>
<td>7620</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Has interpretation (attribute)</td>
<td>6087</td>
<td>Yes</td>
<td>Value</td>
</tr>
<tr>
<td>Has definitional manifestation (attribute)</td>
<td>5966</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Full name</td>
<td>Count</td>
<td>Relevant?</td>
<td>Comment</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------</td>
<td>-----------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Procedure site (attribute)</td>
<td>5833</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Using device (attribute)</td>
<td>5321</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Direct substance (attribute)</td>
<td>4828</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>REPLACED BY (attribute)</td>
<td>4517</td>
<td>import</td>
<td>Historical relationships</td>
</tr>
<tr>
<td>Has intent (attribute)</td>
<td>4357</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Subject relationship context (attribute)</td>
<td>3885</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Temporal context (attribute)</td>
<td>3884</td>
<td>Yes</td>
<td>The time of the event</td>
</tr>
<tr>
<td>Direct device (attribute)</td>
<td>3863</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Associated finding (attribute)</td>
<td>3202</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Has focus (attribute)</td>
<td>3060</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Surgical approach (attribute)</td>
<td>2776</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Finding informer (attribute)</td>
<td>2493</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Finding context (attribute)</td>
<td>2486</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Associated with (attribute)</td>
<td>2194</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Due to (attribute)</td>
<td>2191</td>
<td>Yes</td>
<td>Disorder</td>
</tr>
<tr>
<td>Using substance (attribute)</td>
<td>2064</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>After (attribute)</td>
<td>1957</td>
<td>?</td>
<td>Sub type of associated with</td>
</tr>
<tr>
<td>Has specimen (attribute)</td>
<td>1956</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
<tr>
<td>Associated procedure (attribute)</td>
<td>1781</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>MOVED TO (attribute)</td>
<td>1501</td>
<td>import</td>
<td>Historical relationships</td>
</tr>
<tr>
<td>Procedure context (attribute)</td>
<td>1390</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Revision status (attribute)</td>
<td>1247</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
</tbody>
</table>

FIG. 141B
<table>
<thead>
<tr>
<th>Full name</th>
<th>Count</th>
<th>Relevant?</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using access device (attribute)</td>
<td>1215</td>
<td>No</td>
<td>Procedure related – not during phase 1</td>
</tr>
<tr>
<td>Specimen source topography (attribute)</td>
<td>986</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Specimen substance (attribute)</td>
<td>619</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Specimen procedure (attribute)</td>
<td>575</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Indirect morphology (attribute)</td>
<td>518</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Pathological process (attribute)</td>
<td>345</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Procedure device (attribute)</td>
<td>294</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Procedure morphology (attribute)</td>
<td>294</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Using energy (attribute)</td>
<td>291</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Recipient category (attribute)</td>
<td>112</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Route of administration (attribute)</td>
<td>105</td>
<td>No</td>
<td>Procedure related</td>
</tr>
<tr>
<td>Property (attribute)</td>
<td>95</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
<tr>
<td>Indirect device (attribute)</td>
<td>74</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Specimen source morphology (attribute)</td>
<td>69</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Scale type (attribute)</td>
<td>49</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
<tr>
<td>Specimen source identity (attribute)</td>
<td>40</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Measurement method (attribute)</td>
<td>5</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
<tr>
<td>Time aspect (attribute)</td>
<td>1</td>
<td>Yes</td>
<td>For LOINC concepts</td>
</tr>
</tbody>
</table>

FIG. 141C
<SubscriptionSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xml="http://www.w3.org/2001/XMLSchema">
    <Subscriber>
        <Root>dbMotion.CareEvents.DataProviders</Root>
        <Extension>EncounterProvider</Extension>
    </Subscriber>
    <PatientInclusionDirective>Exclude</PatientInclusionDirective>
    <Patients />
    <RuleSubscriptions>
        <RuleSubscription>
            <SubscriptionContext>
                <ID>AdmissionState</ID>
                <Qualifier>Admitted</Qualifier>
            </SubscriptionContext>
            <TriggerringRule>
                <PatternRule MonitorType="DEM" ID="encounter" Version="1.0" />
                <SubscriptionArguments>
                    <Parameter id="code">
                        <CodeParam Operation="MemberIn">
                            <Values>
                                <Value CodeSystem="2.16.840.1.113883.5.4" Code="IMP" />
                                <Value CodeSystem="2.16.840.1.113883.5.4" Code="NONAC" />
                                <Value CodeSystem="2.16.840.1.113883.5.4" Code="ACUT" />
                                <Value CodeSystem="2.16.840.1.113883.5.4" Code="ACUTE" />
                                <Value CodeSystem="2.16.840.1.113883.5.4" Code="EMER" />
                            </Values>
                        </CodeParam>
                    </Parameter>
                    <Parameter id="statusCode">
                        <CodeParam Operation="Equals">
                            <Values>
                                <Value CodeSystem="2.16.840.1.113883.5.14" Code="active" />
                            </Values>
                        </CodeParam>
                    </Parameter>
                    <Parameter id="dischargeDate">
                        <DateParam Operation="Empty" />
                    </Parameter>
                </SubscriptionArguments>
                <IncludeExtendedInfo>false</IncludeExtendedInfo>
            </TriggerringRule>
        </RuleSubscription>
    </RuleSubscriptions>
</SubscriptionSet>
<PatternRule Id="encounter" Version="1.0" Name="Encounter Pattern Rule">
  <Description>Use this rule to subscribe to an encounter</Description>
  <AppendRule Id='demography' Version='1.0'/>
  <Parameters>
    <Parameter Id="code" Name="Encounter code" IsMandatory="false" Type="UMSConcept">
      <Description>Type of encounter (i.e. encounter of a given code)</Description>
    </Parameter>
    <Parameter Id="statusCode" Name="Encounter StatusCode" IsMandatory="false" Type="UMSConcept">
      <Description>Status of encounter (i.e. encounter of a given StatusCode)</Description>
    </Parameter>
    <Parameter Id="admissionDate" Name="Encounter AdmissionDate" IsMandatory="false" Type="DateTime">
      <Description>Admission date of encounter i.e. effectiveTime_start (i.e. encounter where admission date is)</Description>
    </Parameter>
    <Parameter Id="dischargeDate" Name="Encounter dischargeDate" IsMandatory="false" Type="DateTime">
      <Description>Discharge date of encounter i.e. effectiveTime_end (i.e. encounter where discharge date is)</Description>
    </Parameter>
    ...
  </Parameters>
</PatternRule>

FIG. 143
<table>
<thead>
<tr>
<th>#</th>
<th>DRI Semantic Signifier</th>
<th>DRI Query</th>
<th>Business Method</th>
<th>Command Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Condition</td>
<td>Filter condition code (HF conditions) - list of codes e.g. as per description herein of heart failure patient classification's entry task's CTS's KNEO.</td>
<td>GetProblemList</td>
<td>Problems::Execute GetProblemList FilterInstruction='Calc_Local Code IN AND Calc_LocalCodeSystem IN'</td>
</tr>
<tr>
<td>2</td>
<td>Patient Administration</td>
<td>None</td>
<td>Get Demography</td>
<td>Demography::Execute GetDemography</td>
</tr>
</tbody>
</table>

FIG. 144A
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectiveTime</td>
<td>The time of the evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>title</td>
<td>Is HF Patient</td>
<td>In rule</td>
</tr>
<tr>
<td>text</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>infoButton</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>dbmAvailabilityTime</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>derivationExpr</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>ActClassId</td>
<td>OBS</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>id</td>
<td>root = KFW OID, extension=GUID</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>statusCode</td>
<td>code= 'active'; codeSystem=2.16.840.1.113883.5.14'</td>
<td>Default auto-generated</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The disclosure of information about this evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>InformationModelSSID</td>
<td>The identifier of the Boolean Conclusion Semantic Signifier (triplet): [scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>KMEvaluationResultid</td>
<td>The identifier of the KM Evaluation Result that concluded the result (quartet): [itemBusinessId, scopingEntityId, businessId, version]</td>
<td>Note that the id above is per instance and this id is per KM. Note that the first attribute is the ER id and the last three attributes represent the KM id.</td>
</tr>
<tr>
<td>SeverityCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>UrgencyCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

FIG. 144B

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitHFPatientsAction</td>
<td>Action Invocation Activity</td>
</tr>
<tr>
<td>ExitHFPatientsClassification</td>
<td>Classification Decision Activity</td>
</tr>
<tr>
<td>#</td>
<td>DRI Semantic Signifier</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>Encounter</td>
</tr>
<tr>
<td>2</td>
<td>PatientAdministration</td>
</tr>
</tbody>
</table>

FIG. 145A
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectiveTime</td>
<td>The time of the evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>title</td>
<td>Is Eligible HF Encounter</td>
<td>In Rule</td>
</tr>
<tr>
<td>text</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>infoButton</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>dbmAvailabilityTime</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>derivationExpr</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>ActClassId</td>
<td>OBS</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>id</td>
<td>root = KFW OID, extension=GUID</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>statusCode</td>
<td>code= 'active'; codeSystem='2.16.840.1.113883.5.14'</td>
<td>Default auto-generated</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The disclosure of information about this evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>InformationModelSSID</td>
<td>The Identifier of the Boolean Conclusion Semantic Signifier (triplet): [scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>KMEvaluationResultId</td>
<td>The Identifier of the KM Evaluation Result that concluded the result (quartet): [itemBusinessId, scopingEntityId, businessId, version] Note that the id above is per instance and this id is per KM. Note that the first attribute is the ER id and the last three attributes represent the KM id.</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>SeverityCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>UrgencyCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Reasons</td>
<td>The patient record data that lead to the recommendation</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>ChainedConclusions</td>
<td>The patient record data that lead to the recommendation</td>
<td>Auto-generated</td>
</tr>
</tbody>
</table>
### Table 145C

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryAdmittedHFPatientsAction</td>
<td>Action Invocation Activity</td>
</tr>
<tr>
<td>EntryAdmittedHFPatientsClassification</td>
<td>Classification Decision Activity</td>
</tr>
</tbody>
</table>

### Table 145D

<table>
<thead>
<tr>
<th>#</th>
<th>DRI Semantic Signifier</th>
<th>DRI Query</th>
<th>Business Method</th>
<th>Command Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clinical Document</td>
<td>(may be xml-implemented) Retrieve Patient Documents Types which include &quot;LVS Function Evaluation&quot; - list of codes e.g. as per KNEO for CTS for LVS function evaluation's monitoring task</td>
<td>GetClinicalDocumentsList</td>
<td>ClinicalDocuments::Execute GetClinicalDocumentsList FilterInstruction=&quot;DocumentTypeCode IN AND DocumentTypeCodeSystem IN&quot;</td>
</tr>
</tbody>
</table>

### Table 146A

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvaluationOfLVSFunction</td>
<td>Quality Conclusion ER</td>
</tr>
<tr>
<td>HF-2_IsPerformed</td>
<td>Boolean Conclusion ER</td>
</tr>
<tr>
<td>HF-2_IsRequired</td>
<td>Boolean Conclusion ER</td>
</tr>
</tbody>
</table>

### Table 146B

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectiveTime</td>
<td>The time of the evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Value</td>
<td>Notes</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>title</td>
<td>HF-2: Is Performed; HF-2: Is Required; HF-2: Evaluation of LVS Function</td>
<td>3 titles for 3 ERs (Boolean, Boolean and Quality)</td>
</tr>
<tr>
<td>text</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>infoButton</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>dbmAvailabilityTime</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>derivationExpr</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>ActClassId</td>
<td>OBS</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>id</td>
<td>root = KFW OID, extension=GUID</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>statusCode</td>
<td>code='active'; codeSystem='2.16.840.1.113883.5.14'</td>
<td>Default auto-generated</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The disclosure of information about this evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>InformationModelSSId</td>
<td>The identifier of the Boolean/Quality Conclusion Semantic Signifier (triplet): [scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>KMEvaluationResultId</td>
<td>The identifier of the KM Evaluation Result that concluded the result (quartet): [itemBusinessId, scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td></td>
<td>Note that the id above is per instance and this id is per KM.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note that the first attribute is the ER id and the last 3 attributes represent the KM id.</td>
<td></td>
</tr>
<tr>
<td>SeverityCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>UrgencyCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Reasons</td>
<td>The patient record data that lead to the recommendation</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>ChainedConclusions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 146C
<table>
<thead>
<tr>
<th>#</th>
<th>DRI Semantic Signifier</th>
<th>DRI Query</th>
<th>Business Method</th>
<th>Command Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Substance Administration</td>
<td>Retrieve ACEI Medications for current Encounter (CPQP)</td>
<td>GetEncounters GetMedications List</td>
<td>Encounter::Execute GetEncountersList FilterInstruction=&quot;Id_Root=&quot;$EncRoot&quot; AND id_Extension=&quot;$EncExtension&quot; Retrieve from the result all id's from ActRelationship where RelatedActType=128 (Substance Administration) Medication::Execute GetMedicationsList FilterInstruction=&quot;Id_Root IN (&quot; + retrieved roots + &quot;) AND Id_Extension IN (&quot; + retrieved extensions + &quot;) AND Calc_LocalMedicineCode IN (&quot; + row(CTS list) + &quot;) AND Calc_LocalMedicineCodeSystem IN (&quot; + row(CTS list) + &quot;)&quot;</td>
</tr>
<tr>
<td>2</td>
<td>Substance Administration</td>
<td>Retrieve ARB Medications for current Encounter (CPQP)</td>
<td>Same as #1</td>
<td>Same as #1</td>
</tr>
<tr>
<td>3</td>
<td>Condition</td>
<td>Retrieve diastolic dysfunction conditions (subset of HF conditions)</td>
<td>GetProblemList</td>
<td>Problems::Execute GetProblemList FilterInstruction=&quot;Calc_LocalCode IN (&quot; + CTS list + &quot;) AND Calc_LocalCodeSystem IN (&quot; + CTS list + &quot;)&quot;</td>
</tr>
<tr>
<td>4</td>
<td>Condition</td>
<td>Retrieve ACEI and ARB Contraindications Conditions</td>
<td>GetProblemList</td>
<td>Problems::Execute GetProblemList FilterInstruction=&quot;Calc_LocalCode IN (&quot; + CTS list + &quot;) AND Calc_LocalCodeSystem IN (&quot; + CTS list + &quot;)&quot;</td>
</tr>
<tr>
<td>5</td>
<td>Allergy Intolerance</td>
<td>Retrieve ACEI Allergies</td>
<td>GetAllergies IntoleranceList</td>
<td>AllergyIntolerance::Execute GetAllergiesIntoleranceList FilterInstruction=&quot;Calc_LocalIntoleranceValueCode IN (&quot; + CTS list + &quot;) AND Calc_LocalIntoleranceValueCodeSystem IN (&quot; + CTS list + &quot;)&quot;</td>
</tr>
<tr>
<td>6</td>
<td>Allergy Intolerance</td>
<td>Retrieve ARB Allergies</td>
<td>Same as #5</td>
<td>Same as #5</td>
</tr>
</tbody>
</table>

FIG. 147
<table>
<thead>
<tr>
<th>Output</th>
<th>Required</th>
<th>Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HF3-T6</td>
<td></td>
</tr>
<tr>
<td>HF3-T7</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contraindications</th>
<th>ACEI or ARB Conditions?</th>
<th>ACEI Allergy?</th>
<th>ARB Allergy?</th>
<th>Medication Prescribed During Encounter?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Not LVSD</th>
<th>Diastolic Dysfunction?</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TRUE</td>
</tr>
<tr>
<td>2</td>
<td>FALSE</td>
</tr>
<tr>
<td>3</td>
<td>FALSE</td>
</tr>
<tr>
<td>4</td>
<td>FALSE</td>
</tr>
<tr>
<td>5</td>
<td>FALSE</td>
</tr>
<tr>
<td>6</td>
<td>FALSE</td>
</tr>
<tr>
<td>7</td>
<td>FALSE</td>
</tr>
<tr>
<td>8</td>
<td>FALSE</td>
</tr>
<tr>
<td>9</td>
<td>FALSE</td>
</tr>
<tr>
<td>10</td>
<td>FALSE</td>
</tr>
<tr>
<td>11</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

FIG. 148
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACElorARBforLVSD</td>
<td>Quality Conclusion ER</td>
</tr>
<tr>
<td>HF-3_IsPerformed</td>
<td>Boolean Conclusion ER</td>
</tr>
<tr>
<td>HF-3_IsRequired</td>
<td>Boolean Conclusion ER</td>
</tr>
</tbody>
</table>

FIG. 149
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectiveTime</td>
<td>The time of the evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>title</td>
<td>HF-3:1s Performed; HF-3:1s Required; HF-3:ACEI or ARB for LVSD</td>
<td>3 titles for 3 ERs (Boolean, Boolean and Quality) In Rule</td>
</tr>
<tr>
<td>text</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>infoButton</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>dbmAvailabilityTime</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>derivationExpr</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>ActClassId</td>
<td>OBS</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>id</td>
<td>root = KFW OID, extension=GUID</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>statusCode</td>
<td>code= 'active'; codeSystem= '2.18.840.1.113883.5.14'</td>
<td>Default auto-generated</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The disclosure of information about this evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>InformationModelSSID</td>
<td>The identifier of the Boolean/Quality Conclusion Semantic Signifier (triplet): [scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>KMEvaluationResultId</td>
<td>The identifier of the KM Evaluation Result that concluded the result (quartet): [itemBusinessId, scopingEntityId, businessId, version] Note that the id above is per instance and this id is per KM. Note that the first attribute is the ER id and the last three attributes represent the KM id.</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>SeverityCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>UrgencyCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Reasons</td>
<td>The patient record data that lead to the recommendation</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>ChainedConclusions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 150
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>effectiveTime</td>
<td>The time of the evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>title</td>
<td>Is Discharged HF Encounter</td>
<td>In Rule</td>
</tr>
<tr>
<td>text</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>infoButton</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>dbmAvailabilityTime</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>derivationExpr</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>ActClassId</td>
<td>OBS</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>id</td>
<td>root = KFW OID, extension=GUID</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>statusCode</td>
<td>code='active'; codeSystem='2.16.840.1.113883.5.14'</td>
<td>Default auto-generated</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The disclosure of information about this evaluation result</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>InformationModelSSID</td>
<td>The Identifier of the Boolean Conclusion Semantic Signifier (triplet): [scopingEntityId, businessId, version]</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>KMEvaluationResultId</td>
<td>The identifier of the KM Evaluation Result that concluded the result (quartet): [itemBusinessId, scopingEntityId, businessId, version] Note that the id above is per instance and this id is per KM. Note that the first attribute is the ER id and the last three attributes represent the KM id.</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>SeverityCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>UrgencyCode</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Reasons</td>
<td>The patient record data that lead to the recommendation</td>
<td>Auto-generated</td>
</tr>
<tr>
<td>ChainedConclusions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 151
### FIG. 152A

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryTempDischargedHFPatientsAction</td>
<td>Action Invocation Activity</td>
</tr>
<tr>
<td>EntryTempDischargedHFPatientsClassification</td>
<td>Classification Decision Activity</td>
</tr>
</tbody>
</table>

### FIG. 152B

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExitTempDischargedHFPatientsAction</td>
<td>Action Invocation Activity</td>
</tr>
<tr>
<td>ExitTempDischargedHFPatientsClassification</td>
<td>Classification Decision Activity</td>
</tr>
<tr>
<td>CodeSystem</td>
<td>Code</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T1</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T2</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T3</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T4</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T5</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T6</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF3-T7</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF2-T1</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.1</td>
<td>HF2-T2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CodeSystem</th>
<th>Code</th>
<th>Designation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.2</td>
<td>CategoryB</td>
<td>Not in the measure population</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.2</td>
<td>CategoryD</td>
<td>In measure population</td>
</tr>
<tr>
<td>2.16.840.1.113883.3.57.1.4.6.2</td>
<td>CategoryE</td>
<td>In numerator population</td>
</tr>
</tbody>
</table>

**FIG. 153**
## Example of Applicant Specific Requirements

<table>
<thead>
<tr>
<th>Content/VPO Analyzer</th>
<th>Pr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved data typically should comply with the standard HIES security policy (such as patient context, user context and role, privacy and security)</td>
<td>P0</td>
</tr>
<tr>
<td>&quot;Exclude mine&quot; rule – Present Clinical data that is not available in the EHR in the user works with. This rule can be implemented in several level: 1. By Source – Basic - All Acts which received from my EHR source. 2. By Source – Advanced – All Acts which coming from my EHR as well as sources of my EHR (e.g. Labs which flows to both Allscripts and HIES, typically should not be presented when the EHR is Allscripts)</td>
<td>P0</td>
</tr>
<tr>
<td>In continue to previous 3. Semantically – in addition to #1 and #2, exclude acts which are semantically equivalent to information available in my EHR.</td>
<td>P2</td>
</tr>
<tr>
<td>&quot;new since last seen&quot; rule – get all clinical acts (new or updated) since last time I accessed HIES Clinical Viewer. E.g. if user access Problems page in clinical viewer or in SmartAgent on 1/1/2010. Then when rule is applied, the results will include all problems new in HIES since 1/1/2010.</td>
<td>P1</td>
</tr>
<tr>
<td>Extensibility - It typically should be possible to add new rules in extensible way. Preferably - without need for new product version.</td>
<td>P0</td>
</tr>
<tr>
<td>The handled context typically should include: User Patient Application Type (Allscripts EEHR, MyWay, Cerner, EPIC, Sunrise, etc.) EHR Application Identifier (Allscripts EEHR in JUP, Cerner H1 and H2 in UPMC, etc.) Workflow Context (Optional. E.g. which Tab the user selected in the EHR)</td>
<td>P0</td>
</tr>
</tbody>
</table>

**FIG. 154B**
| Example of Applicant Specific Requirements - Pr. |
|-------------------------------------------------|---|
| **Context Capturing and Sharing**               | **P0** |
| To facilitate integration with EHR which voluntarily shares context with HIES, there typically should be a simple, standard (preferred) way to enable the EHRs written in different technologies (such as Java, web, .Net, C++, etc.) to share the context (user, patient, application and workflow). | |
| In case EHR cannot share context, it typically should be possible to proactively capture the following context elements: Application ID, Patient (MRN), User (Username, User Details), from the EHR screen (without any development from the EHR side) | **P0** |
| It typically should be possible to capture workflow operations (e.g. access to Labs/Meds pages) | **P2** |
| Ability to support different versions of an her | **P2** |
| In case NO MRN is available, it typically should be possible to capture demographics for EMPI search | **P2** |
| Extendibility - It typically should be possible to extend capturing support for different EHRs. Preferably as an extension points configured by eth professional service. | **P1** |

**FIG. 154C**
<table>
<thead>
<tr>
<th><strong>Semantic Search</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable semantic search over codified patient data.</td>
<td>P2</td>
</tr>
<tr>
<td>e.g. Search for Beta Blockers or Hemoglobin HbA1c typically should find results for codes that are semantically related to those codes – either directly or through concepts connections.</td>
<td></td>
</tr>
<tr>
<td>Enable semantic search over unstructured patient data (e.g. clinical documents, acts notes, descriptions, etc.).</td>
<td>P2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Configurations</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>It typically should be possible to configure which rules will presentation and their order.</td>
<td>P1</td>
</tr>
<tr>
<td>It typically should be possible to personalize configurations.</td>
<td>P1</td>
</tr>
<tr>
<td>Personalized configuration typically should be user driven and NOT machine driven.</td>
<td>P1</td>
</tr>
<tr>
<td>Configurations typically should be kept on the centralized server side as much as possible, and prevent as much as possible local machine wise configurations.</td>
<td>P1</td>
</tr>
</tbody>
</table>

FIG. 154D
<table>
<thead>
<tr>
<th>Example of Applicant Specific Requirements - Floating Application</th>
<th>Pr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a HIES Client Application (AKA SmartAgent) that presents information from HIES within the EMR context and workflow.</td>
<td>P0</td>
</tr>
<tr>
<td>The HIES SmartAgent typically should NOT be dependent on development for integration by the EMR side. Instead it should have its own capability to support different EMRs written in different technologies.</td>
<td>P0</td>
</tr>
<tr>
<td>The floating application should be bound to a specific active instance of an EMR (as the user might work with several EMR applications in parallel).</td>
<td>P0</td>
</tr>
<tr>
<td>The user experience should fit the EMR UI as much as possible (position, application binding, size, etc).</td>
<td>P0</td>
</tr>
<tr>
<td>The solution’s UI typically should NOT cause any interruptions to the user’s regular workflow. Rather, provide clear indications.</td>
<td>P0</td>
</tr>
<tr>
<td>To meet different EMR’s look and feel and user preferences, there typically should be an ability to apply skins/configure appearance of elements such as UI color, icons/buttons and fonts</td>
<td>P1</td>
</tr>
<tr>
<td>The SmartAgent typically should present the data in a summarized way taking into consideration the Application limited space.</td>
<td>P0</td>
</tr>
<tr>
<td>Serve as a user-friendly gateway to other HIES applications, such as Clinical Viewer, Collaborate and other future HIES Application.</td>
<td>P0</td>
</tr>
<tr>
<td>At any point, it typically should be easy for the user to return to the regular workflow – i.e. easy hiding/closing of the UI.</td>
<td>P0</td>
</tr>
<tr>
<td>Ability to View Results/Clinical in printable format and print it.</td>
<td>P1</td>
</tr>
<tr>
<td>Ability to View HIES Collaborate Recent Events and do actions such as File, Reminders, Referrals, Etc.</td>
<td>P1</td>
</tr>
<tr>
<td>Ability to perform HIES collaborate Actions such as “File”, “Print”, “Referral” etc.</td>
<td>P1</td>
</tr>
</tbody>
</table>

FIG. 154E
### Example of Applicant Specific Requirements -

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Pr.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1.1 Auditing</strong></td>
<td></td>
</tr>
<tr>
<td>SmartAgent and VPO Analyzer – Relevant navigation operations and failures in the SmartAgent typically should be audited in the HIES Auditing System (STL).</td>
<td>P1</td>
</tr>
<tr>
<td>SmartAgent – it typically should be possible to extract from STL what Clinical Aspects (Problems, Labs etc.) the user accessed. In the same way the clinical viewer auditing behaves.</td>
<td>P1</td>
</tr>
<tr>
<td><strong>1.1.2 Security</strong></td>
<td></td>
</tr>
<tr>
<td>Authentication – there typically should be manual log-in option</td>
<td>P1</td>
</tr>
<tr>
<td>Authentication – There typically should be a windows SSO authentication support</td>
<td>P2</td>
</tr>
<tr>
<td>Authentication – The SmartAgent typically should be able to generate SAML token based on the user identity captures or provided by the EHR.</td>
<td>P1</td>
</tr>
<tr>
<td>SmartAgent and VPO Analyzer – security authorization operations typically should be applied according to the standard HIES security policy.</td>
<td>P1</td>
</tr>
<tr>
<td><strong>1.1.3 Localization</strong></td>
<td></td>
</tr>
<tr>
<td>There typically should be an internationalization support including presentation in different languages, ability to translate captions and text , capturing elements from screens in different languages.</td>
<td>P2</td>
</tr>
<tr>
<td>There typically should be an ability to present right-to-left text and open panels to both right and left directions.</td>
<td>P2</td>
</tr>
</tbody>
</table>

**FIG. 155A**
### Topology and pre-requisites

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The solution typically should support deployment of the SmartAgent either on Local machine and on Citrix (or Citrix alike products)</td>
<td>P0</td>
</tr>
<tr>
<td>SmartAgent typically should support Windows XP and Windows 7 Operations Systems</td>
<td>P0</td>
</tr>
<tr>
<td>It is expected to have the SmartAgent User interface be able to reused as part of solution for Non windows platforms such as iPhone, iPad, blackberry Android etc.</td>
<td>P3</td>
</tr>
<tr>
<td>It is NOT expected to have the windows based solution be used as is (screen capturing, deployment, etc.) on non windows platforms</td>
<td></td>
</tr>
<tr>
<td>It typically should be possible to easily apply ongoing updates for the SmartAgent without requiring the user initiating the process. Including presentation updates, capturing updates, more EHR supports.</td>
<td>P1</td>
</tr>
<tr>
<td>It typically should be possible to deploy the SmartAgent as a download from a website (e.g. ClickOnce installation).</td>
<td>P1</td>
</tr>
<tr>
<td>It typically should be possible to deploy the SmartAgent in distribution mode (in case of corporate control machines).</td>
<td>P1</td>
</tr>
<tr>
<td>The SmartAgent installation typically should include ALL required components including 3rd party .NET frameworks etc. We typically should NOT assume any pre-requisites beside OS.</td>
<td>P1</td>
</tr>
</tbody>
</table>

**FIG. 155B**
### 1.1.4 Performance

| Expected response time from getting into patient record in the EHR until presenting data from HIES in HIES within the SmartAgent typically should be - 1 Second. Worst case – 2 Seconds. | P0 |
| User experience for any action within the SmartAgent typically should look seamless to the user (under 0.5 Seconds) | P0 |

**FIG. 155C**

### 1.1.5 Reusability and integrability

| The SmartAgent typically should use as much as possible existing pages from the clinical Viewer and CareBoard in embedded panels of the SmartAgent. | P0 |
| The Clinical Viewer and CareBoard must take into consideration in the design the ability to the other applications such as the SmartAgent to be able to use internal modules (e.g. use module of Laboratory page" or “My Recent Events” as a component within the SmartAgent Panel. | P1 |
| The SmartAgent typically should take into consideration the ability to reuse the modules in other HIES Applications such as Clinical Viewer and CareBoard. | P2 |

**FIG. 155D**
<table>
<thead>
<tr>
<th>Footnote</th>
<th>Label</th>
<th>Interactions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HIESLaunchButton</td>
<td>OnClick: Launch HIES Clinical Viewer</td>
<td>This Button opens the HIES Clinical Viewer with the Patient and User context.</td>
</tr>
<tr>
<td>2</td>
<td>SearchButton</td>
<td>OnClick: Enable Search: Set Floating Application Small Panel state to FloatingSearchOpen</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ToolsButton</td>
<td>OnClick: Enable Configurations: Set Floating Application Small Panel state to FloatingToolOpen</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Patient Info</td>
<td>OnClick: OpenMainPanel:</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Collaborate Button</td>
<td>OnClick: Open Collaborate Recent Events:</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Recent Events Button</td>
<td>OnClick: Open Recent Events:</td>
<td></td>
</tr>
</tbody>
</table>
There are no items to show in this view.

FIG. 157A
<table>
<thead>
<tr>
<th>Footnote</th>
<th>Label</th>
<th>Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>File All</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>File CareEvent:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Show GrayBackground - Below Agent, Preview Panel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Preview Panel state to File - All Acts</td>
</tr>
<tr>
<td>5</td>
<td>CRerate Refferal</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create Refferal:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Show GrayBackground - Below Agent, Preview Panel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Preview Panel state to New Refferal</td>
</tr>
<tr>
<td>6</td>
<td>Expand All</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expand All:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Clinical Info Pane state to All Expand</td>
</tr>
<tr>
<td>7</td>
<td>Collapse All</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Collapse All:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Clinical Info Pane state to ClinicalInfoTabCollapsed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Evaluation Panel state to Population Health Tab Collapsed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Evaluation Panel state to Population Health Tab Collapsed</td>
</tr>
<tr>
<td>8</td>
<td>Print</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Print Preview:</td>
</tr>
<tr>
<td>9</td>
<td>Filter</td>
<td>OnClick:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case 1 (If value of variable Filter equals &quot;1&quot;):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set value of variable Filter equal to &quot;0&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Filter to Default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Case 2 (Else if value of variable Filter equals &quot;0&quot;):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set value of variable Filter equal to &quot;1&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set Filter to Selected</td>
</tr>
<tr>
<td>Interactions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Problems CV:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Medications CV:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Launch CV</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Allergies CV:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Launch Documents CV:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand Problems:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand Medications:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collapsed Relevant Clinical Info:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand Labs:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Clinical Info Pane state to Lab Expanded</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand Documents:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Clinical Info Pane state to Documents Expanded</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OnClick:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand Allergies:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Clinical Info Pane state to Allergy Expanded</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 157D
| **Not in my EHR** | My EHR is not the source of the data  
My EHR is not a consumer of data from this source |
|-------------------|-----------------------------------------------------------------------------------|
| **New Since last Seen** | The Data was Modified or New in the CDR since I last accessed the Clinical Viewer or the Open the SmartAgent.  
The idea of this rule is to use a timestamp (ADR/PLV?) that indicate when is the last time the user could have viewed the data in one of the HIES Applications.  
Last seen filter will be calculated per patient session. |
<p>| <strong>Time filter</strong> | The Data meet timeframe criteria |</p>
<table>
<thead>
<tr>
<th>Clinical Data</th>
<th>Eventually those rules typically should be flexible enough to be applied differently for each clinical Act type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a. Not my EHR</td>
</tr>
<tr>
<td></td>
<td>b. Not my EHR + Only last X month</td>
</tr>
<tr>
<td></td>
<td>c. Not my EHR + Only last X month + “New since Last Time Seen”</td>
</tr>
<tr>
<td></td>
<td>d. Not in My EHR + “New since Last Time Seen”</td>
</tr>
<tr>
<td></td>
<td>e. Only last X month</td>
</tr>
<tr>
<td></td>
<td>f. Only last X month + “New since Last Time Seen”</td>
</tr>
<tr>
<td>Recent Events (Working List)</td>
<td>a. Recent Events</td>
</tr>
<tr>
<td></td>
<td>b. Referrals</td>
</tr>
<tr>
<td>Populations Health</td>
<td>a. New Measure, Updated Measure (e.g. Diabetic Patient have new Measure)</td>
</tr>
<tr>
<td></td>
<td>b. Alerts (an Alert available)</td>
</tr>
<tr>
<td></td>
<td>c. Reminders (Due date is soon)</td>
</tr>
<tr>
<td>Clinical Research</td>
<td>a. Candidate for Research</td>
</tr>
<tr>
<td>User Profile</td>
<td>a. Profile of the User is X</td>
</tr>
</tbody>
</table>

FIG. 159C
<table>
<thead>
<tr>
<th>Functional Area</th>
<th>Use Case</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation</td>
<td>Present Patient Data</td>
<td>In this UC the system presents patient data to the user. The content and nature of the presentation is well defined in the mockup and requirements.</td>
</tr>
<tr>
<td></td>
<td>Present Clinician Data</td>
<td>In this UC the system presents data from HIES that is not related to the patient viewed in the EHR. For example - &quot;my recent events&quot; from HIES Collaborate, &quot;my admitted patients&quot; and so on. Low priority</td>
</tr>
<tr>
<td></td>
<td>Launch HIES Viewer</td>
<td>In this UC the system acts just as a entry point to the existing viewer (Clinical Viewer, Collaborate). The launch may or may not include context (user, patient, app). For example - navigation menu for medications (even though no medication record is currently shown).</td>
</tr>
<tr>
<td></td>
<td>Send To My EHR</td>
<td>In this UC the user selects act to be sent to his EHR and the system delivers them to the EHR to be presented there.</td>
</tr>
<tr>
<td></td>
<td>Perform Search</td>
<td>In this UC the user searches for records within the patient record. The search can either be on the codified data, or free search over text (notes within acts and the content of clinical documents).</td>
</tr>
<tr>
<td></td>
<td>Control floating application state</td>
<td>e.g. Minimize, expand, size, position, dock, close etc.</td>
</tr>
<tr>
<td>Context</td>
<td>Identify Application</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Interception</td>
<td>In this UC the system intercepts the application (EHR) in which the user is using.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example - is the user working with Allscripts MyWay, Cerner etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This context may be captured from the EHR screen or provided by the EHR.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note - same application can be used for multiple instances, for example Cerner app</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is used in different &quot;regions&quot; in UPMC - Cerner H1, H2, H3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The instance must be used when applying content rules (&quot;exclude mine&quot;, therefore,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>must solve this issue.</td>
<td></td>
</tr>
</tbody>
</table>

| Identify User           | In this UC the system intercepts the user that is using the EHR in order for HIES   |
|                         | to present data to that user, according to his security privileges etc. EHR side.    |
|                         | This context may be captured from the EHR screen or provided by the EHR.             |
|                         | There are several options here.                                                     |

| Identify Patient        | In this UC the system intercepts the patient that the user is currently looking at |
|                         | in the EHR. This context may be captured from the EHR screen or provided by the    |
|                         | EHR.                                                                                |

| Identify Workflow       | In this UC the system intercepts the workflow in which the user is at on the EHR    |
| Context                 | side. For example - is the user on "labs" tab or on "medication" tab. This context |
|                         | may be captured from the EHR screen or provided by the EHR.                          |
|                         | An advanced scenario here is responding to "free" selection on the screen, e.g. -   |
|                         | user selects "Hgb" on the screen and we need to show history graph.                  |
|                         | Priority is low                                                                       |

**FIG. 160B**
| System Health | Monitor Service Health | In this UC the system need to monitor the server side of the applications. This may include standard monitoring services, performance counters etc. Priority is medium |
| Monitor Agents Health | In this UC the system monitors the scattered agents out there. For example - keep track of when each deployed agent has contacted, enable reporting (?). The approach typically should be based on using calls made anyway by the agent rather than opening a dedicated channel for such monitoring. Another option is "report error" feature - either when the system fails, or when timeouts occur for example. Priority is medium |
| Data Preparation | Prepare Patient Data for presentation | In this UC the system prepares data to be presented to the user. This includes fetching the data as well as filter out irrelevant records, based on rules. In essence this is the area where VPO analyzer comes into play |
| Apply Rules | Acting on patient data, the system decides which records are to be presented using rules. It typically should be able to add more rules and adjust existing rules. The rules currently known are: - "exclude my EHR data" - "exclude what I saw |

FIG. 160C
<table>
<thead>
<tr>
<th>Configuration &amp; Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install Agent on Citrix box</td>
</tr>
<tr>
<td>Install Agent on corporate box</td>
</tr>
<tr>
<td>Install Agent on Non-Corporate box</td>
</tr>
<tr>
<td>Deploy updates to Agent</td>
</tr>
<tr>
<td>Configure &amp; Personalize presentation options</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extension Developments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop &amp; Configure Presentation Rules</td>
</tr>
</tbody>
</table>

FIG. 160D
### Use Case Model Overview

**Versatile Actors**
- Care Management
- Clinician
- dbMotion Developer
- dbMotion Services
- Developer
- EHR
- GetVPO Service
- Implementation Eng

Actors that participate in use cases from multiple functional areas.

**Presentation**
- Control floating application state
- Launch dbMotion Viewer
- Perform Search
- Present Clinician Data
- Present Patient Data
- Send To My EHR

**Context Interception**
- Identify Patient Interaction
- Identify User - managed users Interaction
- Intercept User Details
- Identify Application
- Identify Patient
- Identify User
- Identify Workflow Context

**Configuration & Deployment**
- IT admin
- Configure & Personalize presentation options
- Deploy updates to Agent
- Install Agent on Citrix box
- Install Agent on corporate box
- Install Agent on Non-Corporate box

### class "perspective" Overviews

- "perspective" Overviews:
  - Actors Overview

- Context Diagram

All actors in the use case model, depicted as an inheritance tree.

Key use cases of the system, shown in system context, with performing Actors shown outside the system.

### Data Preparation
- Apply Rules
- Apply Rules
- Prepare Patient Data for presentation

### System Health
- Monitor Agents Health
- Monitor Service Health

### Extension Developments
- Develop & Configure EHR support
- Develop & Configure Presentation Rules

**FIG. 162**
The user is working within the EHR, the system needs to intercept different elements that are changing its context. This can be either in passive mode - (we capture from the screen) or active (the EHR throws an event etc).

**uc Context Interception**

EHR

*from Versatile Actors*

Clinician

*from Versatile Actors*

Identify Application

Identify Workflow Context

Identify Patient

Identify User

*∞*

FIG. 163C
<table>
<thead>
<tr>
<th>User Identifier</th>
<th>Patient Identifier</th>
<th>Last seen Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>Patient 1</td>
<td>Datetime 1</td>
</tr>
<tr>
<td>User 1</td>
<td>Patient 2</td>
<td>Datetime 2</td>
</tr>
<tr>
<td>User 1</td>
<td>Patient 3</td>
<td>Datetime 3</td>
</tr>
<tr>
<td>User 2</td>
<td>Patient 4</td>
<td>Datetime 4</td>
</tr>
<tr>
<td>User 3</td>
<td>Patient 5</td>
<td>Datetime 5</td>
</tr>
</tbody>
</table>

FIG. 165
Identifying User-managed User Interaction

Clinician

EHR

SmartAgent

dbMotion Security

PPOL

OpenLogin() - [Active]

User Change (EHR user context - userName, role, SAML etc)

[Passive]

Capture EHR user details()

Clear User from context()

Got EHR user

Resolve EHR user details to dbMotion User()

Create SAML token()

Authenticate()

Got EHR user

Use Windows current principal as user()

Authentication failed

Manual Login page()

Authenticate()

Authentication Succeeded

Set User to context()

Order of preference (EHR user first, Windows user, login) may change and need to be configured.

Resolving should be based on solid identifiers, such as username in EHR, or identifier in EHR. Another option (risky) is to use user full name and match it against medical staff.

FIG. 167
<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
<th>Parameters</th>
</tr>
</thead>
</table>
| BeginInterception() | Called before the first field is intercepted | char\* [in] scenario  
 | void Public         |                                            | int [in] scenarioLen                            |
| InterceptField()    | Called as a relevant field has been intercepted | char\* [in] scenario  
 | void Public         |                                            | int [in] scenarioLen  
 |                     |                                            | char\* [in] fieldId  
 |                     |                                            | int [in] fieldIdLen                             |
|                     |                                            | char\* [in] interceptedValue  
 |                     |                                            | int [in] interceptedValueLen                   |
| EndInterception()   | Called after the last field in the scenario was intercepted | char\* [in] scenario  
 | void Public         |                                            | int [in] scenarioLen                            |

FIG. 173
FIG. 174
<table>
<thead>
<tr>
<th>Method</th>
<th>Notes</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateContextInterceptor()</td>
<td>To be called when a new EHR window is discovered and so needs to be assigned with a context interceptor</td>
<td>int [in] applicationHandle</td>
</tr>
<tr>
<td>IEHRContextInterceptor Public</td>
<td></td>
<td>int [in] applicationType</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreatePositionInterceptor()</td>
<td>To be called when a new EHR window is discovered and so needs to be assigned with a context interceptor</td>
<td>int [in] applicationHandle</td>
</tr>
<tr>
<td>IEHRWindowStateInterceptor Public</td>
<td></td>
<td>int [in] applicationType</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreateLaunchInterceptors()</td>
<td>to be called when the SmartAgent boots in order to &quot;plant&quot; the different launch interceptors for the different supported EHRs</td>
<td></td>
</tr>
<tr>
<td>IEHLaunchInterceptor[] Public</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 175**
This is a logical diagram, the actual implementation will probably use PPO-Ls, where relations are distinct by a code (similar to provider/provider links, which can hold office manager relationships, or "specialist referral" relationships).
This sequence assumes as a precondition that:
1. EHR is open
2. EHR is identified
3. A ContextInterceptor and a ContextManager had been assigned and initialized to the EHR
Need to look in search results for identity which:
1. Has index from the requested assigning authority
2. the index demography (not the golden demography) matches the demographics from the context exactly.

**FIG. 182**
For each individual EMR within a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, performing a computerized context interception process using a processor to intercept context from the individual EMR and to identify therewithin an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR.

Responsive to identification of said event, using a computerized output device for providing, to said health provider, patient record data pertaining to said individual patient and stored by a health information exchange system which stores patient record data regarding a multiplicity of patients.

Using a processor to apply at least one predetermined rule involving intercepted EMR context and using a computerized output device to provide an alerting indication drawing the health provider’s attention to at least some of said patient record data according to said predetermined rule.

FIG. 183
<table>
<thead>
<tr>
<th>Clinical Domain</th>
<th>Terminology</th>
<th>Subset Size (# of concepts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions</td>
<td>SNOMED-CT</td>
<td>5135</td>
</tr>
<tr>
<td>Medications</td>
<td>RxNorm</td>
<td>5248</td>
</tr>
<tr>
<td>Medications (Hierarchy)</td>
<td>NDF-RT</td>
<td>266</td>
</tr>
<tr>
<td>Labs</td>
<td>LOINC</td>
<td>704</td>
</tr>
<tr>
<td>Immunization</td>
<td>CVX</td>
<td>137</td>
</tr>
<tr>
<td>Allergies (Non Drug allergies)</td>
<td>UNII</td>
<td>284</td>
</tr>
<tr>
<td>Allergies (Drug Allergies)</td>
<td>NDF-RT</td>
<td>1016</td>
</tr>
<tr>
<td>Administrative Codes</td>
<td>HL7</td>
<td>3473</td>
</tr>
</tbody>
</table>

FIG. 185A

<table>
<thead>
<tr>
<th>Clinical Domain</th>
<th>dbMotion Baseline Terminology Code Set</th>
<th>Allscripts Local Terminology Code Set</th>
<th>Mapping Size (# of concepts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>SNOMED-CT</td>
<td>Medcin</td>
<td>1086</td>
</tr>
<tr>
<td>Condition</td>
<td>SNOMED-CT</td>
<td>ICD9</td>
<td>1092</td>
</tr>
<tr>
<td>Medications</td>
<td>RxNorm</td>
<td>NDC</td>
<td>1574</td>
</tr>
<tr>
<td>Medications</td>
<td>RxNorm</td>
<td>Medispan</td>
<td>1574</td>
</tr>
<tr>
<td>Medications</td>
<td>RxNorm</td>
<td>Multum</td>
<td>477</td>
</tr>
</tbody>
</table>

FIG. 185B
SYSTEM AND METHODS FOR FACILITATING COMPUTERIZED INTERACTIONS WITH EMRS

REFERENCE TO COPENDING APPLICATIONS

[0001] Priority is claimed from:

FIELD OF THE INVENTION

[0004] The present invention relates generally to systems for processing medical information and more particularly to computerized interactions with EMRs.

BACKGROUND OF THE INVENTION

[0005] Conventionally, information is transferred between an HIE and EMR via messages exchanged between the two systems. For example, an EMR may know how to import lab results from an HIES. An EMR may provide a tab in its main application, sometimes known as the “community tab” which enables a user to view, but not manipulate or import, HIE-provided information about a patient. Varying medical technology between the HIE and EMR, combined with a lack of ability to semantically resolve the variation, may impose limitations to these modes of information transfer.

[0006] According to Wikipedia, an Enterprise Master Patient Index (EMPI) is “a form of customer data integration (CDI) specific to the healthcare industry. Healthcare organizations or groups of them will implement EMPI to identify, match, merge, de-duplicate, and cleanse patient records to create a master index that may be used to obtain a complete and single view of a patient. The EMPI will create a unique identifier for each patient and maintain a mapping to the identifiers used in each record’s respective system.” It has been claimed that by using an EMPI for “correctly matching patient records from disparate systems and different organizations”, it is possible to obtain “a complete view of a patient”.

[0007] Known technologies relevant to the field of the invention include context management, single sign-on, CCOW or Screen capturing method for context interception.

[0008] Other state of the art health information exchange and integration systems, and conventional technology pertaining to certain embodiments of the present invention, are described in the following publications inter alia:

[0009] 1. US20070118540
[0010] 2. US20090125555
[0012] 4. WO2007010485
[0013] 5. JP20023152
[0014] 6. DE10163469
[0015] 7. US20040141661
[0016] 8. US20090080408
[0017] 9. US20040122709
[0019] 11. US20040122787
[0020] 12. US20040122707

[0022] 14. Published US Application US20050144043; and

Non-Patent Literature describing health information exchange through the use of semantic technology includes:

- Comput Methods Programs Biomed., 2009, 93 (3), 297-312
- XML technologies for the Omaha System: a data model, a Java tool and several case studies supporting home healthcare
- Vittorino Pierpaulo; Tassinio Antonutti; di Orso Ferdinando
- Digital Society, 2009. ICDS’09. Third International Conference,
- Semantic Exchange of Medicinal Data: A Way Towards Open Healthcare Systems
- Puustjärvi, J and Puustjarvi, L
- Annual International Conference of the IEEE, 1726-1729
- Interoperability of personal health records
- Lähteenmäki, Jaakko; Leppänen, Juha and Kajianmaranta, Hannu
- International Conference; 308-313
- Healthcare Applications Interoperability through Implementation of HL7 Web Service Basic Profile
- Hussain, M; Afzal, M; Ahmad, H; F; Khalid, N and Ali, A
- Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE
- International Symposium; 1-6
- Ontology-based approach to achieve semantic interoperability on exchanging and integrating information about the patient clinical evolution
- Miyoshi, N; Ferreira, A and Felipe, J. C
- Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE
- International Symposium; 1-6
- Semantic biomedical image management and analysis
- Chunh, C; Inagaki, Y; Cootan, C; Curnumis, B and Sheu, P. C


[0028] The disclosures of all publications and patent documents mentioned in the specification, and of the publications and patent documents cited therein directly or indirectly, are hereby incorporated by reference.

SUMMARY OF THE INVENTION

[0029] Certain embodiments of the present invention seek to provide a technical solution for the problem of allowing medical data to be effectively retrieved, stored, and presented to medical service providing users where the medical data exists in digital form within a plethora of non-compatible, partially overlapping software systems which are constantly being updated.
Certain embodiments of the present invention seek to provide an interoperability solution for medical databases, providing a health information exchange system typically storing complete, single and harmonized patient records, and easing access thereto by bringing relevant information to a user at points in time in which that information is useful and as part of her or his workflows.

Certain embodiments of the present invention seek to bring relevant context-based information from inside the health information exchange system to a user, e.g., physician’s working environment and workflows (typically EMR), rather having the physician leave his working environment and search for the information he needs in a Clinical Viewer as an external application.

Certain embodiments of the present invention seek to enhance effective software compatibility including reducing dependency on EMR vendors to customize their software products in order to integrate with the health information exchange system platform (Button, Smart Access, and other services).

Certain embodiments of the present invention seek to provide easy and efficient access to specific context-based patient information eliminating the need to navigate through many patient’s views in an external application.

Health Information Exchange (HIE) is defined as the mobilization of healthcare information electronically across organizations within a region, community or hospital system. HIE provides the capability to electronically move clinical information among disparate health care information systems while maintaining the meaning of the information being exchanged. The goal of HIE is to facilitate access to and retrieval of clinical data to provide safer, more timely, efficient, effective, equitable, patient-centered care. To meet this goal, HIE providers develop computerized infrastructures and applications that enable the information exchange and viewing of exchanged information. As HIE solutions complement the EMR applications, the EMR and HIE vendors are looking for ways to integrate with each other in order to enable:

1. Data exchange from the EMR the HIE and vice versa
2. Integrate the information hold in hold in the HIE within the EMR application and user workflow
3. Enrich EMR capabilities with the HIE solutions and services.

When an HIE solution is integrated with the EMR, both accessibility and User Context may be taken into account.

Certain embodiments of the present invention seek to provide an SOA-based platform that enables healthcare organizations and health information exchanges (HIEs) to integrate their information assets, through the creation of a virtual patient record by logically connecting a group of care providers and organizations without requiring the replacement of existing information systems. By providing ubiquitous access to integrated patient information, the solution virtually bridges gaps that often exist between inpatient/acute care and community care.

Typically, a single, virtual patient record contains complete and harmonized patient data by logically connecting a group of care providers and organizations without requiring the replacement of existing information systems. Smooth and easy access to the care-critical information stored in the HIE should be facilitated, by providing a user with relevant patient information at the point in time it is needed, as part of the clinical workflow.

The system shown and described herein may perform any or all of the above functionalities:

Provide important, relevant, context-based information from the HIE within a physicians’ work environment and workflows (typically their EMRs)—as opposed to having the physicians leave their work environments and enter the HIE’s Clinical Viewer functionality as an external application to search for the information needed.

Reduce the HIE’s dependency on EMR vendors to customize their products in order to integrate with the HIE’s platform (e.g., launch button, SSO, services).

Provide efficient access to specific context-based patient information, thereby eliminating the need to navigate through multiple clinical views in an external application such as a dbMotion Viewer.

The SmartAgent is a client application that is designed to meet the EMR users’ need to get comprehensive and relevant clinical information on patients from sources of information which are not in their EMR, and in addition to serve as a gateway to HIE applications and solutions.

The client application, typically installed on the user’s machine, is termed herein a SmartAgent client.

Examples of use scenarios include but are not limited to the following:

1. Smart Button within User, Patient and System Context: User opens a patient record in his EMR. SmartAgent, which is installed in his client machine, “captures” the patient identifier (MRN), the User Context (Username/Role) and the System context (SystemId) and calls a VPO Analyzer web service or Virtual Patient Object Clinical Data Web Service) that identifies the System, user and the patient. The user is authorized and the patient is found in the health information exchange system. The Client SmartAgent gets the response and presents a Floating Button. The Floating button includes Link to Launch Viewer with user and patient context. The User presses the button and seamlessly accesses the health information exchange system’s Clinical Viewer.

2. VPO Analyzer attention rules: In order to bring more relevant information to the user, smart evaluations are typically provided on the VPO in the context of the user, patient and system. One of the Analyzer’s attention rules may be “Exclude System Data” which excludes from the VPO Data that exists in the physician’s own system. The response is “Clean” data excluding what a user can see in his EMR, which may be presented within the Results or Viewer Panes. The rule is typically constructed and operative to analyze the patient’s clinical data and to alert the user in the SmartAgent client application that information that meets the rule exists and is available for viewing.

3. Semantic Search: A user may for example be looking for data on Diabetes in the health information exchange system. To do that he enables a search option in the floating toolbar and type the phrase “Dia”. Search suggestions are presented and user selects the “Diabetes” Suggestion. As a result a “Results and Navigation” pane opens and presents the results for Diabetes from the Patient’s VPO organized by Clinical Aspects (Medications, Problems, Population Membership, etc.). User presses “Diabetes” population and the Diabetes View is opened in the View pane.

4. Data Presentation and Launch Viewer: Any information found may be presented in a Data and Navigation Panel. The
information is organized according to the different clinical aspects (Laboratory, Medications, etc.) and evaluation aspects (Population membership, Metrics, Notifications, Alerts etc.). The clinical aspects and actual presented data may constitute a link to a relevant page in the health information exchange system’s Clinical Viewer. The user can see under a Laboratory Results menu, a result for hba1c from, say, a previous week. Aside from the result, 2 buttons may be provided, one to open the Laboratory Clinical View and another to open the Lab Result Page with the hba1c history.

The present invention also typically includes at least the following embodiments:

a. A computerized system for supplying a human user with relevant, context-based patient information within the user’s work environment and workflows.

b. A system according to embodiment a wherein the user’s work environment includes at least one EMR.

c. A system according to embodiment b which does not require customization of the EMR.

d. A system according to embodiment ‘a’ which supplies information without requiring the user to navigate through multiple clinical views in an external application.

e. A system according to embodiment ‘a’ wherein the system includes a proactive apparatus which operates proactively, responsive to user context operations, to present relevant clinical information.

f. A system according to embodiment ‘a’ wherein the system includes a processor operative to select relevant information including performing a computerized analysis of a computerized patient record and deriving, from the analysis, relevant clinical information which is presented to the user, whereas other clinical information is not presented to the user.

g. A system according to embodiment ‘f’ wherein differentiation of relevant clinical information from other clinical information is based on at least one of the following: user context, profile, patient illness, ward context, EMR Workflow Context.

h. A system according to embodiment ‘a’ and wherein the system is operative to provide information, within the workflow, on overall patient events and evaluations for each individual physician or user.

i. A system according to embodiment ‘a’ and also comprising at least some aspects of a skin application shown and described herein.

j. A computerized method for supplying a human user with relevant, context-based patient information within the user’s work environment and workflows.

k. A method according to embodiment T wherein the user’s work environment includes at least one EMR.

l. A method according to embodiment ‘k’ which does not require customization of the EMR.

m. A method according to embodiment T which supplies information without requiring the user to navigate through multiple clinical views in an external application.

n. A method according to embodiment T wherein the method includes a proactive apparatus which operates proactively, responsive to user context operations, to present relevant clinical information.

o. A method according to embodiment T wherein the method includes a processor operative to select relevant information including performing a computerized analysis of a computerized patient record and deriving, from the analysis, relevant clinical information which is presented to the user, whereas other clinical information is not presented to the user.

p. A method according to embodiment ‘o’ wherein differentiation of relevant clinical information from other clinical information is based on at least one of the following: user context, profile, patient illness, ward context, EMR Workflow Context.

q. A method according to embodiment ‘a’ and wherein the system is operative to provide information, within the workflow, on overall patient events and evaluations for each individual physician or user.

r. A method according to embodiment T and also comprising at least some aspects of a skin application shown and described herein.

s. A computer program product, comprising a computer usable medium having a computer readable program code embodied therein, the computer readable program code adapted to be executed to implement any of the methods shown and described herein.

[0046] Certain embodiments of the present invention seek to provide a decision making system including a system of logic including hierarchical semantic relationships, a plurality of systems of medical information which are provided in a plurality of local terminologies respectively, and a decision making apparatus for transforming the medical information in the local terminologies to transformed information usable by the system of logic and for using the system of logic to make at least one decision based on the transformed information, without translating the system of logic into the plurality of local terminologies. The term “terminology” is intended to include any scheme for representing medical information. The following terms and other terms defined herein may be construed either in accordance with any definition thereof appearing in the prior art literature or in accordance with the specification, or as follows:

Classification Type—A base set of classifications which all others derive from. The existing classifications are:

[0047] Candidate—represents a new population element entering the system.

[0048] ActiveMember—represents a member of the population currently being monitored.

[0049] DormantMember—represents a member who is “sleeping” or currently active but not being monitored (in a dormant state).

[0050] Evaluation Task—an evaluation task combines a set of executable rules, an evaluation goal, activation, and a set of triggering rule subscriptions. When a member is associated with a task (by having a specific classification) the triggering rule subscriptions are sent to the Data Event Monitor for that member. When task processing is activated, if that member has had any matching triggering rules fire, the task is sent to be processed (along with the member details).

[0051] Member—The population element of a specific Guard, each member is tagged with its population source and contains a list of classifications.

[0052] Member Classification—Guard evaluation tasks are grouped by classifications. If a member belongs to a specific classification, that member has certain tasks associated with him or her.

[0053] Population Source—the source of members for the Guard, could be an external list, an enrollment service, or a data event monitor.

[0054] Triggering Rule Subscription—subscription for the Abstract Rule Monitor, contains a Pattern Rule Identifier and a set of subscription arguments.
[0056] Schedule—an alarm (scheduled or event based) used to activate processing for a particular evaluation task (or set of evaluation tasks).

[0057] DEM—data event monitor e.g. as described herein

[0058] EMPI—Conventional Enterprise Master Patient Index service

[0059] Principal Index—aka (also termed herein) Leading Index

[0060] VIA—a commercial Virtual Identity Aggregation service provided by DBMotion Inc., Israel

[0061] ACEI—angiotensin-converting enzyme inhibitors

[0062] LV—Left Ventricular Systolic Dysfunction

[0063] LVSD—Left Ventricular Systolic Dysfunction

[0064] DBMotion—refers to a functionality which is either commercially available from DBMotion Inc., Israel and/or is shown and described herein. Other definitions, acronyms, and abbreviations useful in understanding certain embodiments of the present invention, are provided in the table of FIG. 2.

[0065] In accordance with an aspect of the invention, there is provided a health information exchange system comprising an apparatus for archiving health information using a health information encoding procedure only if the health information fulfills a criterion of frequent use; and an apparatus for using a first procedure to respond to queries pertaining to the health information which fulfills the criterion of frequent use and using a second procedure to respond to queries not pertaining to the health information which fulfills the criterion of frequent use.

[0066] In accordance with an aspect of the invention, there is further provided a health information exchange system comprising an ontological apparatus for defining and storing ontological link elements ontologically linking between individual health care information items within a first population of health care information items; an apparatus for receiving a second population of health care information items and for associating at least some individual items within the second population, with corresponding individual items within the first population of health care information items; and an apparatus for responding to queries regarding particular information items in the second population including translating the particular information items into items in the first population corresponding to the particular information items and using link elements linking the items in the first population corresponding to the particular information items to generate data pertaining to the particular information items in the second population.

[0067] In accordance with an embodiment of the invention, there is provided a system comprising an apparatus for making at least one health decision based on the queries.

[0068] In accordance with an embodiment of the invention, there is further provided a system also comprising apparatus for implementing the at least one health decision.

[0069] In accordance with an embodiment of the invention, there is further provided a system also comprising apparatus for making at least one health decision based on the queries.

[0070] In accordance with an embodiment of the invention, there is further provided a system also comprising apparatus for implementing the at least one health decision.

[0071] In accordance with an aspect of the invention, there is provided a health information exchange method comprising archiving health information using a health information encoding procedure only if the health information fulfills a criterion of frequent use; and using a first procedure to respond to queries pertaining to the health information which fulfills the criterion of frequent use and using a second procedure to respond to queries not pertaining to the health information which fulfills the criterion of frequent use. In accordance with an aspect of the invention, there is provided a health information exchange method comprising defining and storing link elements linking between individual health care information items within a first population of health care information items; receiving a second population of health care information items and associating at least some individual items in the second population, with corresponding individual items within the first population of health care information items; and responding to queries regarding particular information items in the second population including translating the particular information items into items in the first population corresponding to the particular information items and using link elements linking the items in the first population corresponding to the particular information items to generate data pertaining to the particular information items in the second population.
[0078] In accordance with an embodiment of the invention, there is yet further provided a computer program product wherein the method also comprises making at least one health decision based on the queries.

[0079] In accordance with an embodiment of the invention, there is yet further provided a computer program product wherein the method also comprises implementing the at least one health decision.

[0080] In accordance with an embodiment of the invention, there is yet further provided a computer program product wherein the method also comprises making at least one health decision based on the queries.

[0081] In accordance with an embodiment of the invention, there is yet further provided a computer program product wherein the method also comprises implementing the at least one health decision.

[0082] In accordance with an embodiment of the invention, there is yet further provided a method wherein the second population of health care information items are expressed in a local terminology and are mapped to a baseline terminology in which the first population of health care information items are expressed, to enable terminology interoperability at least when responding to queries.

[0083] In accordance with an embodiment of the invention, there is yet further provided a method wherein the baseline terminology is semantically enriched by associating semantic information therewith, the method also comprising generating conclusions about health information expressed in at least one local terminology by using the semantic information rather than by defining semantic relations for the local terminology.

[0084] In accordance with an embodiment of the invention, there is yet further provided a system in which only a subset of a universe of health information is archived.

[0085] In accordance with an embodiment of the invention, there is yet further provided a system wherein the apparatus for responding to queries uses a first procedure to respond to queries pertaining to the subset and uses a second procedure to respond to queries not pertaining to the universe of health information but not pertaining to the subset.

[0086] In accordance with an embodiment of the invention, there is yet further provided a system also including an end user interface allowing end users to define rules; and a decision support subsystem (DSS) interacting with the end user interface and using semantic capabilities of a baseline terminology in which the first population of health information items is encoded, to simplify definition of rules by the end users.

[0087] In accordance with an embodiment of the invention, there is yet further provided a system wherein the decision support subsystem comprises an Enterprise DSS which has a process cycle and which uses DSS rules to define all phases in the process cycle.

[0088] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using is applied to a use case involving processing of Smart Guard Adapters, the processing including at least one of developing, defining and configuring.

[0089] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using is applied to a use case involving a Smart Watch System, the use case including at least one of processing and monitoring health of the system.

[0090] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using are applied to a use case involving Managing Guard runtime.

[0091] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using are applied to a use case involving applying Guard changes.

[0092] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using are applied to a use case involving task activation based upon a schedule.

[0093] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using are applied to a use case involving identifying patients to be added to a defined population of patients.

[0094] In accordance with an embodiment of the invention, there is yet further provided a method wherein the translating and the using are applied to a use case involving monitoring a population of patients including determining if they need to be evaluated, evaluating them thereby to generate at least one evaluation result, and responding to the evaluation result.

[0095] In accordance with an embodiment of the invention, there is yet further provided a method wherein the health information encoding procedure includes mapping health information expressed in at least one local terminology to a baseline terminology to enable terminology interoperability and storing ontological information interrelating health information items expressed in the baseline terminology.

[0096] In accordance with an embodiment of the invention, there is yet further provided a system wherein the ontological apparatus includes interrelationships between clinical-level information items.

[0097] In accordance with an embodiment of the invention, there is yet further provided a system wherein the clinical-level information item comprises at least one health care information item specifying at least one of a disease, rather than only a class thereof, and a medication, rather than only a class thereof, such as "Left Ventricular Heart Failure", rather than "Cardio-vascular disorder", and "Amoxicillin 250 MG Oral Capsule [Amoxymed]", rather than "Antibiotic", respectively.

[0098] In accordance with an embodiment of the invention, there is yet further provided a system wherein the ontological apparatus maps at least one legacy concept expressed in local terminology to at least one ontology concept expressed in a baseline terminology thereby allowing queries on the level of a single legacy concept to be responded to, for example, the following legacy concept: (System: ICD9, Code: 428.9, Designation: HEART FAILURE NOS) may be mapped to the following Ontology concept: (System: SNOMED-CT; Code: 84114007; Designation: Heart failure (disorder)). Very generic examples of classifications are "Diagnosis", "Medicine", "Procedure"; more specific classifications are "Cardio-vascular disorder", "Antibiotics" etc. Classifications do not identify a patient's clinical status; for example, it is not enough to say in a clinical record that the patient has "Cardiovascular disorder" as there are many types of such disorders, and it is typically useful to know which disorder the patient suffers from, to decide how to treat it. Examples of clinical-level information items are "Left Ventricular Heart Failure", "Amoxicillin 250 MG Oral Capsule [Amoxymed]", these information items are sufficiently detailed to describe aspects
of an individual patient's clinical status and/or treatment rather than mere classifications thereof.

Also provided is a computer program product, comprising a typically non-transitory computer usable medium or computer readable storage medium, typically tangible, having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement any or all of the methods shown and described herein. It is appreciated that any or all of the computational steps shown and described herein may be computer-implemented. The operations in accordance with the teachings herein may be performed by a computer specially constructed for the desired purposes or by a general purpose computer specially configured for the desired purpose by a computer program stored in a typically non-transitory computer readable storage medium.

Any suitable processor, display and input means may be used to process, display e.g. on a computer screen or other computer output device, store, and accept information such as information used by or generated by any of the methods and apparatus shown and described herein; the above processor, display and input means including computer programs, in accordance with some or all of the embodiments of the present invention. Any or all functionalities of the invention shown and described herein may be performed by a conventional personal computer processor, workstation or another programmable device or computer or electronic computing device, either general-purpose or specifically constructed, used for processing; a computer display screen and/or printer and/or speaker for displaying; machine-readable memory such as optical disks, CDROMs, magnetic-optical discs or other discs; RAMs, ROMs, EPROMs, EEPROMs, magnetic or optical or other cards, for storing, and keyboard or mouse for accepting. The term "processor" as used above is intended to include any type of memory or manipulation or transformation of data represented as physical, e.g. electronic, phenomena which may occur or reside e.g. within registers and/or memories of a computer. The term processor includes a single processing unit or a plurality of distributed or remote such units.

The above devices may communicate via any conventional wired or wireless digital communication means, e.g. via a wired or cellular telephone network or a computer network such as the Internet.

The apparatus of the present invention may include, according to certain embodiments of the invention, machine readable memory containing or otherwise storing a program of instructions which, when executed by the machine, implements some or all of the apparatus, methods, features and functionalities of the invention shown and described herein. Alternatively or in addition, the apparatus of the present invention may include, according to certain embodiments of the invention, a program as above which may be written in any conventional programming language, and optionally a machine for executing the program such as but not limited to a general purpose computer which may optionally be configured or activated in accordance with the teachings of the present invention. Any of the teachings incorporated herein may however suitable operate on signals representative of physical objects or substances.

The embodiments referred to above, and other embodiments, are described in detail in the next section.

Any trademark occurring in the text or drawings is the property of its owner and occurs herein merely to explain or illustrate one example of how an embodiment of the invention may be implemented.

Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions, utilizing terms such as, "processing", "computing", "estimating", "selecting", "ranking", "grading", "calculating", "determining", "generating", "reassessing", "classifying", "generating", "producing", "stereo-matching", "registering", "detecting", "associating", "superimposing", "obtaining" or the like, refer to the action and/or processes of a computer or computing system, or processor or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within the computing system's registers and/or memories, into other data similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices. The term "computer" should be broadly construed to cover any kind of electronic device with data processing capabilities, including, by way of non-limiting example, personal computers, servers, computing system, communication devices, processors (e.g. digital signal processor (DSP), microcontrollers, field programmable gate array (FPGA), application specific integrated circuit (ASIC), etc.) and other electronic computing devices.

The present invention may be described, merely for clarity, in terms of terminology specific to particular programming languages, operating systems, browsers, system versions, individual products, and the like. It will be appreciated that this terminology is intended to convey general principles of operation clearly and briefly, by way of example, and is not intended to limit the scope of the invention to any particular programming language, operating system, browser, system version, or individual product.

Elements separately listed herein need not be distinct components and alternatively may be the same structure.

Any suitable input device, such as but not limited to a sensor, may be used to generate or otherwise provide information received by the apparatus and methods shown and described herein. Any suitable output device or display may be used to display or output information generated by the apparatus and methods shown and described herein. Any suitable processor may be employed to compute or generate information as described herein e.g. by providing one or more modules in the processor to perform functionalities described herein. Any suitable computerized data storage e.g. computer memory may be used to store information received by or generated by the systems shown and described herein. Functionalities described and herein may be divided between a server computer and a plurality of client computers. These or any other computerized components shown and described herein may communicate between themselves via a suitable computer network.

BRIEF DESCRIPTION OF THE DRAWINGS

Certain embodiments of the present invention are illustrated in the following drawings:

FIG. 1A is a simplified functional block diagram illustration of a health information exchange and integration system constructed and operative in accordance with certain embodiments of the present invention.
FIG. 1B is a table of sequence interaction descriptions useful in understanding the operation of the system of FIG. 1A according to certain embodiments of the present invention.

FIGS. 2 and 3 are a table and a diagram useful in understanding an embodiment of the CTS sub-system of FIG. 1A, a knowledge framework sub-system and a Smart-watch sub-system, which interact with a core platform for processing medical information, such as the core platform product available from DBMotion Inc., Israel.

FIGS. 4A-8C are diagrams and other illustrations useful in understanding an embodiment of the knowledge framework sub-system of FIG. 1A.

FIGS. 9-13C are diagrams and other illustrations useful in understanding an example detailed implementation of the knowledge framework sub-unit of FIG. 1A.

FIGS. 14-17 are diagrams and other illustrations useful in understanding an embodiment of the Smart-watch sub-system of FIG. 1A.

FIGS. 18A-137 are diagrams and other illustrations which, particularly in conjunction with FIGS. 9-13, are useful in understanding an example health information exchange and integration system constructed and operative in accordance with certain embodiments of the present invention. Specifically:

FIGS. 18A-105 are diagrams and other illustrations useful in understanding an example implementation of the CTS sub-unit of FIG. 1A.

FIGS. 106A-137 are diagrams and other illustrations useful in understanding an example implementation of the smart-watch sub-unit of FIG. 1A. Specifically:

FIG. 106a is a simplified functional block diagram illustrating the smart-watch sub-unit constructed and operative to certain embodiments of the present invention.

FIG. 106b is a table describing operations, some or all of which may be performed by the system of FIG. 106a.

FIGS. 106C-114 are diagrams and other illustrations useful in understanding an example implementation of the smart-watch service unit of FIG. 106a.

FIGS. 115-123 are diagrams and other illustrations useful in understanding an example implementation of the data event monitor of FIG. 106a.

FIGS. 124-130 are diagrams and other illustrations useful in understanding an example implementation of the person identity service of FIG. 106a.

FIGS. 131-134 are diagrams and other illustrations useful in understanding an example implementation of the temporal monitor of FIG. 106a.

FIGS. 135-137 are diagrams and other illustrations useful in understanding an example implementation of the action manager of FIG. 106a.

FIG. 138 is a SmartWatch-wide high level use case diagram of SmartWatchService use cases and their associated actors provided in accordance with certain embodiments of the present invention.

FIG. 139 is a use-case focused diagram of the smart watch service illustrating SmartWatchService use cases provided in accordance with certain embodiments of the present invention.

FIG. 140 is a diagram of an example Knowledge-Module Entities Model from which a knowledge base repository for the KFW sub-system of FIG. 1A may be derived.

FIG. 141 is a table of typically conventional ontological relationships serving as an example of semantic relationships which may be used by the CTS subsystem of FIG. 1A.

FIG. 142 is an example of a pattern rule comprising a subscription snippet, in XML format.

FIG. 143 is an example of a pattern rule comprising a definition snippet, in XML format.

FIGS. 144A-153 are diagrams and other illustrations useful in understanding an example Enterprise Clinical Decision Support System which may utilize the SmartWatch sub-unit of FIG. 1A. In particular:

FIGS. 144A-144C illustrate an example HF (heart failure) patients classification including triggering rules and other characteristics for entry and exit tasks.

FIGS. 145A-145D illustrate example admitted HF patient sub-classifications, including triggering rules and other characteristics for entry and exit tasks.

FIGS. 146A-146B illustrate an example of an LVS function evaluation method, including triggering rules and other characteristics for a monitoring task.

FIGS. 147-150 illustrate an example ACEI or ARB for LVSD.

FIGS. 151-152B illustrate an example sub-classification for temp discharged HF patients including triggering rules and other characteristics for entry and exit tasks.

FIG. 153 is a table of Example vocabulary codes useful in the system of FIGS. 144A-152A.

FIG. 154a is a simplified functional block diagram of a high level architecture of a smart agent system constructed and operative in accordance with certain embodiments of the present invention.

FIG. 154b is a table summarizing an example set of functional requirements of a content/VPO analyzer included in the apparatus shown and described herein.

FIG. 154c is a table summarizing an example set of functional requirements of a content capturing and sharing functionality included in the apparatus shown and described herein.

FIG. 154d is a table summarizing an example set of functional requirements of a semantic search functionality included in the apparatus shown and described herein.

FIG. 154e is a table summarizing an example set of functional requirements of a floating application included in the apparatus shown and described herein.

FIG. 155a is a table summarizing an example set of non-functional auditing, security and localization requirements of apparatus shown and described herein.

FIG. 155b is a table summarizing an example set of non-functional topological and pre-requisite requirements of the apparatus shown and described herein.

FIG. 155c is a table summarizing an example set of non-functional requirements of apparatus shown and described herein.

FIG. 155d is a table summarizing an example set of non-functional reusability and integrability requirements of the apparatus shown and described herein.

FIG. 156a is an example user interface useful in entering a patient file and launching the SmartAgent system provided according to certain embodiments of the present invention, including an example of how a smart agent may position itself vis a vis an E.H.R. according to certain embodiments.
FIG. 156b is a simplified pictorial illustration of an example user interface for a FloatingClosed functionality.

FIG. 156c is an example object table useful in understanding the functionality of the user interface of FIG. 156b.

FIG. 156d is a simplified pictorial illustration of an example user interface for a FloatingSearchOpen functionality.

FIG. 157a illustrates the SmartAgent application, according to certain embodiments of the present invention, hovering on top of an example EMR (Allscripts Sunrise—commercially available EMR) in an expanded mode, e.g. by presenting the smart agent’s clinical data on top of the EMR e.g. as described herein with reference to FIGS. 156a-156c.

FIG. 157b is an enlarged view of the expanded SmartAgent panel, according to certain embodiments of the present invention.

FIG. 157c is an example object table useful in understanding the functionality of the user interface of FIG. 157b.

FIG. 157d is an example Object Table.

FIG. 158 is an example user interface for a laboratory screenshot of a preview panel useful in accordance with certain embodiments of the present invention.

FIG. 159a is a simplified pictorial illustration of an example method for highlighting of the background of a patient’s name field.

FIG. 159b is a table presenting an example set of basic rules.

FIG. 159c is a table presenting an example set of rule combinations.

FIG. 160a is a table of an example set of presentation use cases.

FIG. 160b is a table of an example set of context gesture use cases.

FIG. 160c is a table of an example set of system health use cases.

FIG. 160d is a table of an example set of data preparation, configuration & deployment, and extension development use cases.

FIG. 161 illustrates an Interaction example—User context interception sequence, in Enterprise Architect UML format.

FIGS. 162, 163a-163c, 164-166 present an example use case model useful in conjunction with the system of FIG. 154a.

FIG. 165 is an example table suitable for storing a timestamp per user and per patient according to certain embodiments.

FIG. 168 is a diagram of an example Design Model of a smart agent constructed and operative in accordance with certain embodiments of the present invention.

FIG. 169 is a logical diagram of the smart agent of FIG. 168 according to certain embodiments.

FIG. 170 is a logical diagram of the context entities of FIG. 169 according to certain embodiments.

FIG. 171 is a logical diagram of the ScreenCapturing functionality of FIG. 169 according to certain embodiments.

FIG. 172 is a pictorial diagram of a medical domain comprising an interconnected network of entities according to certain embodiments.

FIG. 173 is a table of the InterceptionEventDispatcher’s operations according to certain embodiments.

FIG. 174 is a logical diagram of controllers according to certain embodiments.

FIG. 175 is a table of Operations, some or all of which may be performed by the InterceptorsFactory functionality of FIG. 174.

FIG. 176 is a logical diagram of Context Interception functionality provided according to certain embodiments.

FIG. 177 is a diagram of example Framework and Contracts pertaining to the VPOAnalyzer of FIG. 154a.

FIG. 178 is an example Logical diagram for the SystemIdentity functionality of FIG. 177.

FIG. 179 is a logical diagram of Service & business layers provided according to certain embodiments.

FIG. 180 is an example logical diagram of the rules functionality of FIG. 179.

FIG. 181 is an Identify Patient UC (use case) realization sequence diagram illustrating an example Use Case (UC) Realization process.

FIG. 182 is a sequence diagram of an example server operative to identify patient UC realization.

FIG. 183 is a top-level simplified flowchart illustration of a method of operation for the smart agent system of FIG. 154a, according to an embodiment of the present invention.

FIG. 184 is a pictorial diagram showing parameters useful for defining EMR Agent shell window position relative to EMR application window according to certain embodiments.

FIG. 185a is a table of example origins of Baseline Ontology Terminology Components in an example HIE according to certain embodiments.

FIG. 185b is a table of Terminology Maps that map between Local Terminologies according to certain embodiments.

FIG. 186 is a simplified functional block diagram illustrating of an embodiment of the invention showing a smart agent which may be constructed and operative in accordance with any of the embodiments shown and described herein.

FIG. 187 is a diagram illustration of an example CCOW context Interceptor according to certain embodiments.

FIG. 188 is a diagram illustrating filtering of clinical information whose source of information is a given EMR according to certain embodiments.

Computational components described and illustrated herein can be implemented in various forms, for example, as hardware circuits such as but not limited to custom VLSI circuits or gate arrays or programmable hardware devices such as but not limited to FPGAs, or as software program code stored on at least one intangible computer readable medium and executable by at least one processor, or any suitable combination thereof. A specific functional component may be formed by one particular sequence of software code, or by a plurality of such, which collectively act or behave or act as described herein with reference to the functional component in question. For example, the component may be distributed over several code sequences such as but not limited to objects, procedures, functions, routines and programs and may originate from several computer files which typically operate synergistically.

Data can be stored on one or more intangible computer readable media stored at one or more different locations, different network nodes or different storage devices at a single node or location.
[0191] It is appreciated that any computer data storage technology, including any type of storage or memory and any type of computer components and recording media that retain digital data used for computing for an interval of time, and any time of information retention technology, may be used to store the various data provided and employed herein. Suitable computer data storage or information retention apparatus may include apparatus which is primary, secondary, tertiary or off-line; which is of any type or level or amount or category of volatility, differentiation, mutability, accessibility, addressability, capacity, performance and energy use; and which is based on any suitable technologies such as semiconductor, magnetic, optical, paper and others.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

[0192] The following terms may be construed either in accordance with any definition thereof appearing in the prior art literature or in accordance with the specification, or as follows:

[0193] E.H.R., EMR: In the present specification and claims, these are used generally interchangeably in that any reference to one might also refer to the other, mutatis mutandis. A repository of computerized health/medical data including a multiplicity of patient records and an interface allowing a user e.g. physician to access such data and, typically, manipulate such data.

[0194] HIE, HIE: health information exchange (HIE) system such as DHM Health’s HIES, typically including computerized apparatus accepting and storing information from, and providing information to, a plurality of EMRs.

[0195] Context: as in CCOW terminology, this term includes information which may be shared by or derived from a computerized system such as an EMR. This information may include who is operating (e.g. EMR user context), who is the patient whose data is currently being processed/displayed by the EMR (patient context) and what operation (e.g. prescription, referral, other) the user is currently performing. “context interception” — obtaining, for a first computerized system such as a HIES, context from a second computerized system such as an EMR, e.g. by sharing or deriving, typically including a determination of how the information from the second computerized system is to be used or related to by the first computerized system. Example of context interception provided according to certain embodiments of the present invention: When an EMR enters a medical record, the smart agent may retrieve the patient identifier e.g. by first recognizing the EMR page through which the patient user accesses the medical records and then, e.g. using prior knowledge of the location of the patient identifier on that page, capturing the user identifier.

[0196] Business rule or business layer: As defined in Wikipedia, this term of the art simply refers to a computerized processing rule or layer of computerized processing respectively and does not have any financial connotation whatever.

[0197] Mm: patient ID e.g. Medical record number.


[0199] There is thus provided, at least the following embodiments:

Embodiment 1

[0200] A method for using a health information exchange system which stores patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

[0201] for each individual EMR within said first plurality of EMRs, performing a computerized context interception process using a processor to intercept context from the individual EMR and to identify therewithin an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and

[0202] responsive to identification of said event, using a computerized output device for providing patient record data, pertaining to said individual patient, to said health provider.

[0203] Context sharing used by Healthcare systems such as EMR and HIE solution’s Context Interception processes are used commonly in the healthcare industry in order to insure synchronization for the clinical data presented for a given provider. In the healthcare industry, the context is commonly referred to a healthcare system (such as HIE or EMR), patient and user identifiers are intercepted by the HIE or EMR and are used to authenticate and authorize to user by given user context, to resolve the patient by the given patient identifiers, and to enable an automation of opening the patient file in the EMR. An example of this embodiment is described below with reference to FIG. 186 which is a diagram of an example component model, as well as in FIGS. 161, 166-167, 170, 174, 176.

Embodiment 2

[0204] A method according to embodiment 1 wherein said patient record data includes at least one information item which is tagged to indicate a source EMR which provided said information item to said health information exchange system and wherein said patient record data provided to said health provider is filtered so as to filter out said at least one information item if said information item’s source EMR equals said individual EMR.

[0205] An HIE system may be built to aggregate a patient’s clinical information from various sources (such as EMRs, lab systems, radiology systems, etc.), in order to enable a provider to view a complete set of clinical information on a given patient at the point of care. The clinical data aggregated into the HIE may additionally include Metadata that enables unique identification of the source system of each specific clinical data element (e.g. specific lab test done for a given patient, or prescribed medication).

[0206] In order to enable filtering of clinical information based on the source of the data, it may be possible to enable the system to request information to define the relevant filter condition. For example, if a provider would like to view given patient clinical information that does not exist in the provider’s EMR system, and under the assumption the given EMR Data is aggregated into the HIE as any other clinical data source, the HIE may enable the clinical data web services to get as an input the EMR identifier which can be retrieved by the computerized context interception process described above. The response of the web service then typically excludes the EMR data. Typically, as described above, the HIE has the metadata of the given EMR which includes the source system identifier, in which case the clinical data service is able to exclude this EMR information, e.g. as described below with reference to FIGS. 154a and 177.
Embodiment 3

[0207] A method according to embodiment 1 wherein said interpreting includes using a context management protocol.

Embodiment 4

[0208] A method according to embodiment 3 wherein said context management protocol comprises CCOW.

[0209] The Common Context Management protocol in the healthcare industry refers to HL7 CCOW. A SmartAgent CCOW Context Interceptor is typically provided which is operative to register to a given CCOW Context Manager (e.g. Sentillion) and get synchronized with the context the Context Manager receives from other EMR or clinical systems register to the Context Manager. The methods for context from a context manager may include some or all of the following HL7 CCOW standard operations: Join context, Leave context, Follow context, Handle context termination and Handle “ping” request.

Embodiment 5

[0210] An example architecture of the Smart Agent CCOW context Interceptor is illustrated in FIG. 176. A diagram illustration of an example CCOW context Interceptor is provided in FIG. 187. The Endpoints in the mid area of FIG. 187 are optional SmartAgent Context Interceptors, which may be an integral part of the SmartAgent Client of FIGS. 154a and 176.

Embodiment 6

[0211] A method according to embodiment 1 wherein said health provider is served by a display screen when using said individual EMR and wherein said interpreting includes capturing information from said display screen.

[0212] A method according to embodiment 1 wherein said interpreting includes:

[0213] pre-storing a patient identifier location for each EMR within said first plurality of EMRs; and

[0214] capturing said individual patient’s patient identifier from said patient identifier location.

[0215] Suitable methods for capturing information from said display screen are known e.g. in Windows operating systems which provide native API for elements displayed in the screen such as Windows, UI controls, and text within these controls. Additional available computerized solutions such as ScreenScraping Studio, described online at the following http location: screen-scraping.deskexperience.com, capture elements from the screen in various different ways. Screen Scarping tools such as Screen Scraping studio, enable stating of a location within a screen or in correlation to an application’s position on a given screen, and interception of the data within it.

[0216] The SmartAgent context interceptors infrastructure, e.g. as described in FIGS. 154a, 169 and 170, may enable a Screen scraping context interceptor designed to intercept relevant context elements such as user and patient context, to apply suitable rules to refine and to validate intercepted information and to pass the intercepted information to the SmartAgent for application, user and patient resolving. The SmartAgent ScreenScraping context interceptor typically intercepts both user and patient context.

Embodiment 7

[0217] A method according to embodiment 1 wherein said providing patient record data includes subscribing to events exchanged between the individual EMR and an operating system serving the EMR, ascertaining in real time at least one current EMR display operation and mimicking the EMR display operation when providing the patient record data.

[0218] The SmartAgent windows state interceptor of FIG. 176 typically uses operating system events to identify a relevant EMR now displayed. This identification is used by the SmartAgent to position itself on top of the EMR, e.g. by applying suitable position algorithms such as but not limited to providing a SmartAgent User Interface Shell container for a desktop application which uses some or all of the following parameters which are defined pictorially in FIG. 184:

[0219] Horizontal offset—horizontal offset of the specified Shell container window angle from the specified EMR window angle

[0220] Vertical offset—vertical offset of the specified Shell container window angle from the specified EMR window angle

[0221] EMR angle—one of the 4 angles of EMR window to anchor Shell container to

[0222] Shell angle—one of the 4 angles of Shell container window to anchor.

[0223] FIG. 156a is an example of a smart agent positioning itself vis a vis an EHR. In the example of FIG. 156a, the EMR is Allscripts Sunrise and the SmartAgent position in the top right, tuned by number of pixels from the top right corner of the EMR Application angles.

[0224] Interception configuration may be loaded at runtime from InterceptionConfig.xml file. The file may contain a list of configurations for known EMR applications. Every configuration may specify some or all of: Launch interceptor class type, Window state interceptor class type, Context interceptor class type and Shell container class type. Custom configuration XML blocks may be specified for each interceptor.

[0225] When a SmartAgent runtime manager instantiates an interceptor, the manager typically gets the interceptor’s configuration class type and then deserializes custom XML block into instance of this configuration class type.

[0226] The SmartAgent runtime manager may process the configuration by performing some or all of the following steps:

[0227] a. Read all configurations

[0228] b. For each configuration gets Launch interceptor block

[0229] c. Instantiate Launch interceptor and assign unique identifier thereto

[0230] d. Save mapping between Launch interceptor identifier and whole configuration block

[0231] e. When a particular Launch interceptor fires an EMR opened event, the event gets a configuration block corresponding to this Launch interceptor

[0232] f. Context Interceptor, Window state interceptor and Shell blocks specified in the configuration block are read

[0233] g. 3 classes are instantiated accordingly

[0234] h. Read and instantiate custom configuration block for each of the 3 items

[0235] i. Configure each item with an instantiated custom configuration object
Embodiment 8

[0236] A method according to embodiment 7 wherein said EMR display operation comprises at least one of a translation along a display screen, a minimization of size and a maximization of size.

[0237] The position of the SmartAgent on top of the EMR is typically determined relative to the EMR’s position and/or state on the screen e.g. as shown herein in FIGS. 156a (minimized) and 157a (maximized). A windows state interceptor functionality “listens” for any change in the state of the EMR window which may have occurred and changes the state of the SmartAgent accordingly. For example:

  a. If an Operation system event of minimizing the EMR occurred, the SmartAgent may be minimized as well.
  b. If the user moved the window of the EMR 10 pixels to the left, Operation systems events occur and are intercepted, and the SmartAgent may move 10 pixels to the left as well.

Embodiment 9

[0238] A method according to embodiment 1 wherein said intercepting and identifying includes screen-capturing EMR context.

Embodiment 10

[0239] A method according to embodiment 5 wherein said patient record data includes at least one information item which is tagged to indicate a category of information to which said information item pertains.

[0240] and wherein said capturing includes capturing screen information indicative of a category of information currently being viewed by the health provider;

[0241] and wherein said patient record data presented to said health provider is filtered so as to filter out said at least one information item pertaining to a category of information other than the category of information currently being viewed by the health provider.

[0242] Context based viewing is typically based on information collected during the screen capturing. Presentation of information in a system based on context in a different system may be achieved by using screen capturing e.g. according to the following example workflow:

[0243] a. Health Provider views Patient chart in his EMR.

[0244] b. Health provider selects to view a tab such as the “Allergies” tab where the patient’s allergies are displayed.

[0245] c. SmartAgent Screen scraping configured to identify the Allergy (say) tab is selected and intercepts this workflow context. The context is passed to the workflow context interceptor e.g. as described herein in “Identify Workflow Context” FIGS. 160b, 163c, 169-170 and 181.

[0246] d. Workflow context interceptor performs “Allergy Selected” (say) action where the SmartAgent request dispatcher performs an action in the “Expand SmartAgent and open Allergy” category.

Embodiment 11

[0247] A method according to embodiment 10 wherein said category of information includes at least one of the following group of categories: laboratory information, medications, allergies, procedures, vital signs, pathologies, imaging results, clinical documents, immunizations, problems, and diagnosis.

Embodiment 12

[0248] A method for using a health information exchange system, storing patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

  For each individual EMR within said first plurality of EMRs, using a processor to automatically identify an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and
  using a processor to apply at least one predetermined rule involving intercepted EMR context and using a computerized output device to provide an alerting indication drawing the health provider’s attention to at least some of said patient record data according to said predetermined rule.

[0250] HIE systems consolidate typically large amounts of information per patient, from a wide variety of information systems such as EMRs, lab systems, imaging systems, etc. SmartAgent rules are typically applied on patient information arriving from the HIE to filter out irrelevant clinical information or information that already exits in the system the health provider is working on, e.g. a given EMR. These rules may apply various combinations of filters on the Patient Clinical data. Filters may for example include:

[0251] a. Time filters filtering out information that is outdated in the sense of healthcare; for example, 5 year old lab results are almost always irrelevant hence typically have a display priority which is lower than 5 year old allergy records which are often relevant; and/or

[0252] b. Source filters filter out information whose source system is the EMR itself.

[0253] Rules may be applied by the VPO Analyzer component e.g. as shown and described herein with reference to FIGS. 154u-154d, 159c, 177, 180 and 188.

Embodiment 13

[0254] A method according to embodiment 12 wherein said generating an alerting indication comprises presenting patient record data, pertaining to said individual patient, to said health provider.

Embodiment 14

[0255] A method according to embodiment 12 wherein said predetermined rule comprises alerting said health provider’s attention if said patient record data includes at least one item of information not included in said individual EMR.

[0256] Attention rules are components which may be provided, typically on the EMR Agent server side which typically runs rules on clinical acts in each VPO and marks clinical acts which meet the rule’s conditions. The SmartAgent client application typically checks for pre-selected clinical acts and performs an action such as a basic “blink” action or “Expand the SmartAgent” or “Make Clinical acts Bold” if certain clinical acts meet certain rules. Example: the patient record includes 3 acts. An attention rule (“Rule 1”) states: “Mark (i.e. select) clinical acts NOT from my EMR.” Assume only Act 1 meets this rule. Act 1 is then marked in the VPO a Rule 1 annotation. The SmartAgent may then be configured to perform an action of flashing if it finds at least one clinical act that meet a certain rule e.g. rule 1 in the present example.
Embodiment 15

[0257] A method according to embodiment 12 wherein a said rule involves health-provider specific context and wherein said applying includes identifying said health provider's identifier.

[0258] Such provider identification may be performed by the User Context interceptor of FIG. 181.

Embodiment 16

[0259] A method according to embodiment 14 wherein items of information included in said patient record data are stored in computerized storage, in association with respective information source tags indicating an information source supplying each said item, and wherein a determination is made, based on said information source tags, of whether or not said patient record data includes at least one item of information not included in said individual EMR.

[0260] As described above, data stored in the HIE can be identified as having arrived from a specific EMR source. The VPO Analyzer which is part of the SmartAgent service may use Clinical Data provided by the Clinical Data Services as described above, and metadata of each clinical data element (e.g. Specific Allergy) in order to filter clinical information whose source of information is a given EMR e.g. as shown in FIG. 188. For example, the VPO Analyzer may apply "Exclude my EMR" data and filter out clinical information whose source system, according to its metadata, is identical to the EMR identifier provided by the EMR Interceptor.

Embodiment 17

[0261] A method according to embodiment 16 wherein a list of information sources supplying an individual EMR is stored in computerized storage and wherein said determination includes determining whether all items in said patient record data are supplied by one of: said individual EMR; and an information source appearing in the list of information sources supplying the individual EMR.

[0262] In many cases, EMR Systems which serve as data providers or sources for an HIE, also receive clinical data from other systems. A common example is Lab Results clinical data that may be recorded in dedicated (non EMR) lab information systems.

[0263] The lab results are sent by the lab system and may be received by EMRs and also by the HIE. In this case, it is common to find information that already exists in the EMR, which was not sent by the EMR and also appears in the HIE. For such cases, a configuration may be provided which determines which EMR served as a source, which received information from other sources (e.g. a lab system). In order to apply a rule or filter such as "Exclude my EMR data", this configuration may be used, so in addition to filtering clinical information from specific source, a filter is applied on sources that act as a source of the given EMR. In the present example, clinical information which arrives from the lab system which is acting as a source of the given EMR, may be filtered.

Embodiment 18

[0264] A method according to embodiment 15 wherein said predetermined rule comprises alerting said health provider's attention if said patient record data includes at least one item of information which was not seen by the health provider identified by said identifier when he was last served by said health information exchange system.

[0265] Examples of a rule is "Show me only clinical data I (as provider) haven't seen on this patient".

[0266] In order to filter out clinical information that a given provider has not seen for a particular patient, some or all of the following operations may be performed, suitably ordered e.g. as follows:

a. SmartAgent’s user context interceptor intercepts user identifier e.g. as described herein with reference to FIG. 181.

b. SmartAgent’s patient context interceptor intercepts patient information and identifies the patient.

c. For the given user and patient, systems get the Timestamp on which the user last accessed this patient

d. SmartAgent service runs service clinical data service and gets patient’s clinical data.

e. SmartAgent service runs the "new acts since last seen" VPO Analyzer rule. If at least 1 clinical act date is later than the timestamp, the rule is applied, the correlated clinical record is marked with the relevant annotation of the rule, and the SmartAgent identifies those acts which are later than the timestamp and enables the alert.

f. Provider views the patient’s clinical data in SmartAgent. In order for the workflow to be applied the next time the provider enters the patient file, the SmartAgent saves the timestamp of the last time the provider saw the patient’s data in the SmartAgent.

g. Save last viewed timestamp for the given user and given patient. The timestamp per user and per patient can for example be stored in the table of FIG. 165.

Embodiment 19

[0266] A method according to embodiment 15 wherein said identifying said health provider’s identifier employs single-sign-on authentication of said health provider.

[0267] CCOW and screen capturing are suitable methods to intercept user context as described above. User context intercepted thereby may include a user identifier which is used by information systems to automatically authenticate and authorize the user, including passing the user context to the information system security system for user resolution and authentication. User authentication may proceed as described herein with reference to FIGS. 154a, 163b, and 166-167. A single sign-on process is advantageous because it facilitates a health provider's workflow by not requiring log-in credentials to be provided separately for each health system being used e.g. in the SmartAgent.

Embodiment 20

[0268] A method according to embodiment 15 wherein said identifying said health provider’s user name includes capturing said health provider’s identifier from a screen display employed by said health provider.

Embodiment 21

[0269] A method according to embodiment 1 wherein said intercepting and identifying includes providing an architecture which interacts with at least one context interception adaptor and, when operating in conjunction with a specific EMR having a specific context sharing capability, using an adaptor adapted to utilize said specific context sharing capability as said context interception adaptor. The Context interceptor architecture typically enables creation of a separate
custom interceptor for each EMR, in the event that no out-of-the-box context interceptor meets the interception requirements of a given EMR. FIG. 171 is an example of an architecture of a custom context interceptor, termed a “Corner Context interceptor” that inherits interception capabilities from a parent class of EMR context interceptors. This custom context interceptor may be a plug-in into the SmartAgent Client service.

**Embodiment 22**

[0270] A method according to embodiment 1 wherein said context intercepted includes an indication of at least one display characteristic characterizing a graphic user interface being generated by the EMR and wherein said providing patient record data includes generating a graphic user interface for displaying said patient record data which shares said display characteristic.

[0271] The ability, also termed herein “skins”, to modify the look and feel of GUI based applications is often part of Development frameworks such as .Net and Java. The term “skins” refers to different colors, fonts, or other appearance preferences in which the application can be configured to be presented. In order to enable health providers to use their EMR with the SmartAgent, and experience the SmartAgent appearance to be similar to the EMR’s, the SmartAgent enables different appearance configurations to be applied for each EMR, e.g. as described in FIG. 160/ (“Configure and personalize presentation options”), since different EMRs more often than not have different appearance characteristics.

**Embodiment 23**

[0272] A method according to embodiment 22 wherein said at least one display characteristic includes at least one of: a color characteristic, a texture characteristic, a text font, a text size, and a characteristic of an icon such as a button.

**Embodiment 24**

[0273] A system for using a health information exchange system, storing patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the system comprising:

[0274] a record-call up identifying processor operative, for each individual EMR within said first plurality of EMRs, to automatically identify an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and

[0275] an intercepted context rule applying processor applying at least one predetermined rule involving intercepted EMR context and controlling a computerized output device to provide an alerting indication drawing the health provider’s attention to at least some of said patient record data according to said predetermined rule.

**Embodiment 25**

[0276] A system for using a health information exchange system which stores patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

[0277] a record-call up identifying processor operative, for each individual EMR within said first plurality of EMRs, for performing a computerized context interception process using a processor to intercept context from the individual EMR to identify therewithin an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and

[0278] a record call-up identifying processor-driven output device controller operative, responsive to identification of said event, to control a computerized output device for providing patient record data, pertaining to said individual patient, to said health provider.

[0279] FIG. 1A is a simplified functional block diagram illustration of a health information exchange and integration system constructed and operative in accordance with certain embodiments of the present invention. FIG. 1B is a table of sequence interaction descriptions useful in understanding the operation of the system of FIG. 1A according to certain embodiments of the present invention. As shown, the health information exchange and integration system of FIG. 1A includes 3 typically symbiotic sub-systems namely a CTS sub-system, a knowledge framework sub-system and a smart-watch sub-system, which interacts with a core platform for processing medical information, such as the core platform product available from DBMotion Inc., Israel. Embodiments of these subsystems are now described with reference to FIGS. 2-3, FIGS. 4A-8C and FIGS. 14-17 respectively. Detailed descriptions of example implementations of these three subsystems are described further on with reference to FIGS. 18A-18B, 9-13C and 106A-137 respectively.

[0280] First, an embodiment of the CTS sub-system is now described generally with reference to FIGS. 2-3. The CTS sub-system is also termed herein the SeNS or “Semantic Network Services” and may also be termed “semantic framework” and may include some or all of the following units, each preceded by their reference numeral in FIG. 3: 302 Query Façade; 304 Extensible specific query interface framework; 306 Metadata interface; 308 Generic query interface; 310 Get Concept; 312 Get Baseline; 314 Get SameAs; 316 Get ValueSet; 318 Custom Interfaces; 320 CTS Access Control; 322 Query processor; 324 Reasoner; 326 Local terminology; 328 Persistence layer; 330 Global Ontology; 332 Value Set; 334 Metadata service; 336 Notification engine; 338 Ontology extension; 340 Management Application; 342 Management Façade; 344 Navigation tool; 346 Mapping tool; 348 Approval process; 350 Add local concept interface; 352 CRUD interface for ValueSets; and 354 CTS Kernel. Typically, the chief functional breakdown of the CTS subsystem’s function is Ontology (content), Querying and Management.

[0281] Possible considerations for designing the CTS subsystem of FIG. 1A are now described in detail.

[0282] The representation of medical information is a key issue in the use of computerized systems in healthcare. In everyday life, medical concepts are represented by words and phrases. This form of information representation allows a high degree of freedom but is difficult to use in computer systems. Phrases might be too long and are often inconsistent: “pain in lower abdomen” could also be described as “complaint of recurring pain in the lower part of the abdomen”. In order to avoid these problems, medical information saved in computer systems is codified. The process of codifying medical information involves the representation of medical concepts, such as observations or procedures, using a fixed coded instead of free text.

[0283] A platform is described which addresses this by providing some or all of: robust information model, a reposi-
tory of standard vocabularies, tools for searching for concepts, mapping tools to help map proprietary vocabularies to standard ones, tools for defining semantic neighborhoods and relation patterns between concepts, routines updates to vocabularies when released by the standards-development organization, and a set of semantic services such as some or all of: Managing local and Control Terminologies, Semantic Maps (mapping) between local and baseline (standard) concepts, Semantic Business Services, VPO Enrichment, Semantic Navigation (in Semantic Networks).

Semantic Interoperability includes ensuring that information exchanged between systems is understandable in the manner in which it was intended by the original creator of that information. It may enable systems to aggregate information from different sources and process it in a meaningful manner.

Semantic Interoperability between heterogeneous healthcare information systems is provided in accordance with certain embodiments. One of the primary obstacles to interoperability is the use of independent sets of terms and codes by the participating systems. When consumers of data (e.g., physicians, EMRs, workflows, Decision Support Systems, BI tools and more) refer to clinical concepts that originated from various sources, they cannot easily interpret, analyze, compare or rationalize this data for visualization. In other words, semantic information is lacking.

The problem of unified medical representation and clinical code mapping has become more and more crucial in the medical world. The effort of creating a global vocabulary and mapping local codes to baseline codes requires time, experience, and knowledge of clinical code standards and clinical terms. Therefore, it is desired to provide an ability to create such vocabulary artifacts easily, and to make the project of mapping flexible and controllable.

System goals may include some or all of:

A. Retrieve patient information: Typically, it is desired to bring together data from disparate clinical information systems in a uniform structure and semantics to support patient care, clinical decision support, and various secondary uses of clinical data such as research.

B. Semantic interoperability: Typically by creating solutions and services for a clinical decision support that makes use of the integrated, uniform data. The system described herein may comprise the integration point for data, and is privy to data that the various source systems are not. Thus, the system of the invention typically can provide enhanced decision support on top of the decision support that the source EMRs provide. This is done by creating solutions for the knowledge-representation and inference capabilities of existing commercial CDSSs.

An example of what may be achieved is the population of an electronic medical record or EMR A with an allergy—documented in EMR B—that patient Y has to medication X, the result of which is the ability of the decision support component of EMR A to fire a drug-allergy alert when the physician writes a prescription for patient Y for a medication in the same allergy class as medication X. Unless EMR A understands the code used for medication X and has knowledge about the relationships between drugs and their allergy class, it cannot fire such an alert.

In order to accomplish the sharing of patient information and clinical decision support solutions, a semantic interoperability paradigm may be based on some or all of the following harmonized elements:

a. Unified Medical Schema—Ontology of medical knowledge that describes the information and semantics of a medical domain and maintains relations between different domains.

b. Vocabulary Ontology—A natural continuation of the UMS, which describes the medical terminology ontology. This ontology utilizes and adheres to well-adopted and advanced standards and methodologies (such as SNOMED, LOINC, HL7 v3, UMLs etc.).

c. Common Terminology services (CTS)—Services that maintain cross-terms of terminologies enabling interpretation and translation of information encoded in different terminology systems (such as but not limited to industry known systems e.g. SNOMED, LOINC, ICD and proprietary coding systems). CTS typically includes and incorporates Semantic Network Services (SeNS): services based on medical terminology ontology, providing semantic navigation capabilities that allow interpretation of medical information. Therefore, CTS and SeNS are used herein generally interchangeably.

The platform described herein typically enables clinicians to draw conclusions based on a meaningful and unambiguous presentation of information, and serves as the foundation for clinical rules and alerts based on the aggregated information. The system is typically vocabulary agnostic, working with a broad range of standards, coding methods, and vocabularies.

The first step in accomplishing the sharing of patient information and clinical decision support solutions may include Unified Medical Schema such as the UMS of dbMotion as a uniform structure for data. The UMS is, in the terminology of data modeling, a logical data model. The UMS may be created by using the HL7 version 3 Reference Information Model as a starting point.

As part of the implementation process of the above solution, human designers together study the structure of the data from the client’s EMRs (and other systems to be integrated) and map or translate those structures to the unique structure of the UMS. For example, this process might involve looking at various HL7 v2.x messages, finding various data elements such as drug code and medical record number and mapping the fields in the messages that contain these data elements to the appropriate data elements in the UMS. In this way, all the information in the network has one unique representation and thus can be shared among different users within different systems and organizations.

The Vocabulary Ontology describes the medical terminology network. The vocabulary ontology is based on a data model that represents concepts, concept relations, coding systems and contexts. The content of the ontology typically includes some or all of:

a. UMS, such as dbMotion UMS’s, domains and related concepts together with coding systems and other attributes associated with them

b. relations between concepts to represent all logic employed by a particular medical application
c. Grouping of concepts for creating contexts.

d. This ontology may serve as a foundation for the CTS subsystem.

e. Accomplishing a uniform structure and semantics for data also typically relies upon a solution to the “vocabulary” problem, where different systems use different codes (or “words”) to refer to the same entities. For example, when two different EMRs (Cerner and EpicCare) use different
codes for the class of medications known as atorvastatin 80 mg oral tablet; Ceremé—7247, EpicCare—045772. Typically, the CTS subsystem includes some or all of the following functionalities:

[0304] a. Managing of local Terminologies for each operational system within all nodes defined.
[0305] b. Maintaining cross-terminology mappings
[0306] c. Enabling interpretation and translation of information encoded in different terminology systems (like industry known SNOMED, LOINC, ICD etc and proprietary coding systems)
[0307] d. Enabling definitions of categories and contexts for groups of concept codes.
[0308] This enables the system to store mappings of code systems in the CDR. For example, the following mappings may be created to the RxNorm code for atorvastatin 80 mg oral tablet (259255) to translate data between EMRs at UPMC: 7427 to 259255 and 045772 to 259255.
[0309] The CTS subsystem typically provides some or all of the following functionalities pertaining to relations between concepts and searching capabilities:

[0311] b. Providing relation patterns between different concepts.
[0312] c. Enabling search capabilities on the concepts tree.
[0313] Definitions, acronyms and abbreviations useful in understanding certain emblems of the present invention e.g. as described in FIG. 3 and henceforth, are provided in the table of FIG. 2. An example business layer suitable for CTS users is now described, including medical information retrieval and filtering functionalities thereof.

[0314] According to certain embodiments, when retrieving medical information there is a significant role to retrieving and interpreting the vocabulary codes bounded to the retrieved information most times in the context of a UMS domain. The business layer may decide what is the codes' information to be listed in the business output schema e.g. local code, baseline code, both, calculated typically according to different business rules, code's neighborhood, and hierarchy. In order to do so, the business typically online queries the CTS requesting the relevant information.

[0315] Another CTS use of the business layer may be for advanced filtering—when there is a need to retrieve information according to one concept code e.g. "give me all WBC lab results") or according to a context (e.g. "give me all beta blocker medications"). Another possible use is to retrieve all information that is related according to relation patterns (e.g. "give me all relevant information for the contra indication pattern"—which would result in all medications, conditions, procedures, etc. that have contra indication to a certain drug).

[0316] A Data Integration layer may operate to recognize concept codes during loading and to load new concept code. When a code arrives through a message to the DIL, a query to the CTS typically results in information regarding this code if such exists or indication that this code is new. When a new code arrives through a message to the DIL, the new code is sent to the Vocabulary Ontology for further processing such as but not limited to being mapped to the right concept or adding new contexts.

[0317] When building a knowledge module, the knowledge expert may query the CTS™ and SenST™ for various terminology operations (e.g., translation of a code between vocabularies, identification of semantic relationships between codes).

Example

[0318] KM decision logic—If the patient is treated with Coumadin Medication AND has a Diagnosis of Bioprosthetic Mitral valve AND the duration of the treatment is more than 3 months, then send message to stop Coumadin.
[0319] CTS interaction: Search medication—the knowledge engineer provides CTS with the name “Coumadin” and requests all relevant codes. The CTS may retrieve 34 different codes from which the user chooses the appropriate one, say Coumadin Tab 5 mg—RxNorm code—209081.
[0320] Relations—the knowledge engineer retrieves from the CTS all relations associated with the selected medication and selects the appropriate one: “May Prevent by” relation.
[0321] Diagnosis—the knowledge engineer navigates from Coumadin on the “May Prevent by” relation and may retrieve a list of Diagnosis. The user chooses, say, the Bioprosthetic Mitral valve diagnosis.
[0322] The knowledge engineer enters a suitable duration for the selected diagnosis (Bioprosthetic Mitral valve): 3 months. Then Action—the knowledge engineer writes the “Then” clause: send message to stop Coumadin.

[0323] Consumers of CTS services in SmartWatch, other than KWF, typically include the Event Monitor described herein. The Event Monitor typically receives triggering rules for which it needs to look for matching records for some reason. These triggering events come from clinical knowledge. Triggering rules are envisioned to be predicated over the data. Every such rule can be thought of as composed of two elements: a pattern rule such as "lab result whose code is x from code system y and whose value is greater than z" and subscription parameters—actual values to be filled into the pattern rule. In the above case the values may include x=SNOED, y=11111 and z=15. Event Monitor may support a limited number of such pattern rules, and may monitor a large number of specific triggering rules based on parameterized subscriptions to these pattern rules. Event Monitor typically does not make final decisions, but rather triggers the evaluation of the patient using KWF. For that reason, when at risk of ambiguity—it is usually better to have false positives (false alarms) over false negatives (missing a relevant record).

[0324] The action manager component is typically operative to communicate SmartWatch findings to the external world, to care providers and external systems. Thus, the Event Monitor gets the message across to the receiving end in a way that is as easy as possible, the Action Manager typically expects the CTS to convert in runtime from baseline terminology in which information was provided, into the terminology the external system uses, probably a local code system.

[0325] A third SmartWatch consumer for CTS may be a SmartGuard manager tool described herein, in which SmartGuards are defined. Within this tool rules are defined on which SmartWatch makes decisions and captures relevant pieces of information. Application-specific needs here may be similar to the application-specific KWF needs.

[0326] Vocabulary administrators may use the vocabulary management tool in order to manage all vocabulary aspects. A user, by using the mapping tool, may perform the actual mapping of the local concept codes to the dbMotion Vocabulary Ontology.

[0327] Vocabulary human experts typically include a specialist e.g. MD familiar in a clinical area serving as an authority for clinical terms and deciding whether mapping is correct. The Vocabulary experts use the vocabulary management tool to approve the mappers’ suggested mapping using the
vocabulary management tool and also may provide suggestions for improvement of the dbMotion Vocabulary Ontology. The following processes I-IV, some of which may be provided in dbMotion core, and solutions that employ vocabulary services, are now described.

1. Loading Local vocabulary information

In order to represent the local data of each system and in order to be able to translate it to a common terminology, the local vocabulary information is typically located into commercially available dbMotion (e.g.) Vocabulary Ontology. The following steps may be followed, completely or in part:

a. Initial stage: In the initial stage of pre-deployment, each organization provides dbMotion a list of all local coding systems with all their concept codes (manual process). All the local codes&rsquo; concept codes from the predefined file/s are imported to the dbMotion Vocabulary Ontology and each code is assigned to a vocabulary domain.

b. Production stage including an offline process and an online process, typically including loading of vocabulary codes on a fluent basis. The vocabulary elements can be loaded to the system both in offline process e.g. using a vocabulary management tool and/or in online process (automatic during messages loading using the Data Integration Layer).

Offline process: By using a vocabulary management tool typically with appropriate permissions, updates can be made to the vocabulary content both by using an import service or by manually influencing the vocabulary elements on the management tool.

As in the initial stage of pre-deployment, when there is an update in any local coding system, the updates or the whole list of codes can be transformed to a predefined file format in a manual process and may be imported to the Vocabulary Ontology using a vocabulary manager tool. Alternatively, when only a few changes occur, changes may be effected manually in the vocabulary management tool.

Deleted vocabulary elements: Obsolete local vocabulary elements that are not acting in any record in the CDR can be removed manually from the Vocabulary Ontology using a vocabulary manager tool.

Online process: When a new Local code enters a CDR through a message, it is loaded to the CDR and assigned to a predefined concept domain according to the message type and related field. The CDR contains only the Local Codes as received from the local systems. No mapping process may take place in the Data Integration process. The new local code is marked as new in the Vocabulary Ontology and an event is sent to the Event Log for the Vocabulary administrator to monitor it on a regular basis. The administrator can relate the concept to a UMS sub domain and map the code to the Vocabulary Ontology.

A code arriving in a message is always associated with a codeSystem and may also be associated with an effectiveDate for the cases when codes in the coding system are editable i.e. can change their meaning over time, such as ICD9 and CPT4 codes. The Vocabulary Ontology takes this into account.

II. Mapping local codes to the ontology

Mapping allows a user to map between Local Concept codes and the Vocabulary Ontology (e.g. baseline concept code) in the context of UMS Domains. The process of mapping is executed using the vocabulary manager tool. For each UMS concept domain, the local code can be mapped to one entry point in the dbMotion Vocabulary Ontology. The mapper chooses local code or bulk of local codes for mapping. The mapper can apply different filters/parameters for the mapping. A mapping process that occurs in the Mapping Tool provides mapping suggestions to each of the selected codes with scoring. These suggestions can be approved/ rejected by the Mapper. The mapping approval includes also the decision of what vocabulary domain to assign. Mapping to a post-coordinated baseline concept typically refers to mapping in which a combination of more than one baseline codes together give the exact mapping—and relation between a collection of baseline codes when mapped to a local code.

III. Vocabulary management

Suitable vocabulary management functionalities a-h, some or all of which may be provided, are now described.

a. Manage versions: When there is new version of a coding system, the version's validation start date is updated for any vocabulary object that was updated according to the new version updates. Until the new version is activated, the Vocabulary Ontology maintains both old and new information. When retrieving changed code information, the system may check if the retrieved code's information is up-to-date. Typically, only if it is, the information is retrieved.

b. Maintaining of concept codes: Concept codes are maintained on a regular basis. If changes occur in a coding system, this change may be analyzed and imported to the Vocabulary Ontology in a predefined format. The vocabulary manager tool may manage the different versions. When the change is of a baseline code, it is the responsibility of the commercially available dbMotion product to provide those updates when they occur. If it is a change in a local coding system, then it is the responsibility of each node separately.

c. Maintain concept relations to other concepts: The vocabulary manager tool may allow the change of relations between concepts—creating new relations and deleting relations no longer existing. This change is reflected in the level of the Vocabulary Ontology and is typically approved before being activated.

d. Maintain mapping of a local code to Ontology: The vocabulary manager tool may allow the changing of mapping of local codes to the Vocabulary Ontology. This change typically must be approved before being activated and versioned.

e. Maintain contexts/value sets: The vocabulary manager tool may allow the creation and update of contexts and value sets. Each change typically must be approved before being activated and versioned.

f. Maintaining UMS domain information: The vocabulary manager tool may allow the change of UMS domain information, when sub-domains are being updated. When updating a sub-domain, its related concept codes may no longer be relevant and other concept codes may now be relevant. When deleting sub-domain, the vocabulary administrator typically first decides what to do with all its related concept codes. All changes typically must be approved and versioned before being activated.

g. Publish changes to subscribers: Any change in a subscribed concept may be published to the subscribers in a predefined format.

h. Approval process: Each change in the Vocabulary Ontology is typically approved by authorized users. The vocabulary manager tool may allow and implement the approval process for each changed element.
IV. Semantic navigation: Querying procedures A-F, some or all of which may be employed by dbMotion core product and/or by the CTS subsystem, are now described:

A. Retrieve codes according to a given context: In each node, if the given context exists in its CDR, a process that generates the list of all valid concept codes that are assigned to this domain or to one of its sub-domains, is executed. For each of the generated codes, a process that retrieves the relevant data from Vocabulary Ontology is executed.

B. Retrieve Codes according to a specific code within a specific context: In the initiator node, if the given code exists in the CDR’s vocabulary domain, and is also assigned to the given domain or to one of its sub-domains, a process that retrieves the valid code’s baselines is executed. This baseline code is then passed to all the network’s nodes. In each node, a process that generates the list of the baseline’s mapped local codes is executed. The mapped codes are all the local codes that are mapped to the baseline code. For each of the generated codes, typically including the baseline, a process that retrieves the relevant data from the CDR is executed. A relevant data is the information in the CDR that has the generated code in one of its predefined attributes.

C. Search for a concept: Locate and retrieve a certain vocabulary concept using a search service. This can be a string. Input Parameters may include some or all of the following:

1. Search string: a string (not case sensitive) that is the motivation of the search. Search typically finds items related to it.

2. Domains/sub domains context: limit the search to pre-defined domains and sub-domains. If null, search the entire database. Use the internal mapping/created list of domains.

3. Context string: limit the search to the provided context string.

4. Related codes relationship types: The output result may include the related codes/concepts as well. In the output, indicate those as “related” and provide the relation type. If null, no related concepts are retrieved, “All” may retrieve all related concepts.

5. Relation pattern: the searching is done according to the given relation pattern which defines the semantic navigation pattern.

6. Input language: the language in which the search string is provided.

7. Output language(s): The code/concept designation language(s).

8. Coding system(s) or abstract concept for result

9. Scoring cut off

10. Max results per search

D. Dynamic querying: A user might build a specific query. dbMotion SeNSSTM may allow building of such a query, save it and enable to reuse it when appropriate.

E. Query control tool: In order to be able to build a query according to changing needs, dbMotion SeNSSTM may include a query control tool. This tool may address all application-specific requirements for searching and querying using a dedicated GUI for this purpose. In this tool, the user may be able to effect some or all of the following functionalities: Build a dynamic query, and save it for later use; Use existing queries; Use a template for creating a query; Update existing queries; Navigate graphically in the dbMotion Vocabulary Ontology, in order to find, and select concepts; and Define Value set of concepts from a query result.

F. Mapping Tool: The CTS subsystem typically includes a mapping tool which allows a user to match between string, codes and phrases to the right place within the Vocabulary Ontology by using a dedicated GUI. In this tool, the user may be able to perform some or all of the following functionalities: Mapping local codes to standards (dbMotion Vocabulary Ontology) with high performance and quality; match the best suggested mapping according to a map ranking for each suggested mapping; and apply different search methods or algorithms as appropriate.

The term “Concept” as used hereinabove refers to or points to a terminology element that describes one item of classification knowledge such as a specific drug or disease, like congestive heart failure. A “classification concept” points to a higher level, typically more general, knowledge element, such as a cardiovascular disorder.

A “Node” as used hereinabove refers to a software deployment component that provides some or all of the system functionality described herein for an individual medical organization. A Node may contain one or more physical servers and is usually but not necessarily installed in each hospital within a medical enterprise. Each Node typically has its own Clinical Data Repository (CDR) that stores information collected from this specific facility.

An embodiment of the knowledge framework subsystem is now described generally with reference to FIGS. 4A-8C. Specifically, an embodiment of the Knowledge Framework (KFW) of FIG. 1A is now described which includes capabilities used by stakeholders and target users. The KFW system may enable knowledge-based interpretation of clinical data to facilitate automatic decision support capabilities. The KFW may comprise a tool set, services, and repositories that enable its users and consumers to acquire, maintain, store, and apply knowledge to data in a meaningful way.

The system of the present invention, even without the knowledge framework, typically enables healthcare organizations to securely share and exchange medical information, creating a Virtual Patient Object (VPO) by logically connecting a group of care providers and organizations without enforcing data centralization or replacement of existing information systems. By sharing real-time medical information, medical staff can make clear critical decisions, thereby providing safer and more efficient healthcare.

The information flowing from the network shown and described herein can leverage development of services that go beyond sharing and exchange of information for purposes such as presentation. Moreover, to avoid overloading care providers with increasing amounts of raw data, it is desired to have services that can interpret raw data, e.g., hemoglobin lab test measurements, into more meaningful concepts at different levels of abstraction, e.g., anemia state, which is derived from subsequent hemoglobin lab test results.

SmartWatch (SMW) enables continuous monitoring of a population, e.g., patients, within a defined medical information space. The SMW framework is utilized by application-specific and/or customer-specific Enterprise Clinical Decision Support Systems (ECDSs). These SMW-based ECDS systems, also known as SmartGuards, are outside the scope of certain embodiments of the present invention and suitably employ core platformibilities of certain embodiments of the present invention and services as well as consuming new services which may be provided by the KFW. For example, the core platform of certain embodiments of the
present invention provides a unified holistic view of the patient clinical state manifested by the Virtual Patient Object (VPO) while the KFW provides the ability to interpret the data contained within the VPO to generate healthcare delivery recommendations.

0374] An example of an Enterprise Clinical Decision Support System which may utilize the SmartWatch subunit shown and described herein is described herein in detail with reference to FIGS. 144A-153.

0375] Thus, the KFW may utilize the holistic view of a patient as captured in the VPO provided by certain embodiments of the present invention to enable knowledgeable abstraction of data to meaningful concepts. This data abstraction ability serves as a key vehicle to support the creation of clinical decision support services. For example, improve the quality of medical care by creating Enterprise Clinical Decision Support Systems (ECDSS) that consume data interpretation results to generate recommendations to care providers and administrators. Another example might be to generate a VPO schema recommendation that takes into consideration preferences such as clinician specialty, e.g., cardiologist, and patient clinical state, e.g., congestive heart failure (CHF), to address specific requests for information.

0376] Another goal of the KFW according to some embodiments is to facilitate the dissemination and utilization of clinical expertise to improve the quality of care. Clinical expertise is accumulated by clinical experts at different medical domains over years of experience. The KFW may enable clinical experts to play an active role during the knowledge acquisition process while collaborating with other actors, e.g., knowledge engineers.

The KFW is typically operative to provide computerized support e.g., to one or more medical domain experts, researchers, clinicians, nurses, social workers, administrators, knowledge engineers and other, non-human consumers, such as EMR, workflow, CPOE process, another DSS, to interpret raw data in order to reach meaningful conclusions at different levels of abstractions otherwise hidden in the data. The KFW typically facilitates application of knowledge to data to reach meaningful concepts at different levels of abstractions which are useful in computerized clinical decision support.

0377] Applications for this system extend throughout the healthcare community at large. It may be domain experts playing the role of knowledge editors to disseminate their valuable knowledge accumulated over years of practicing medicine, care providers benefiting from knowledge-based interpretation of data to improve the medical care they deliver, or healthcare applications developers using KFW services to create ECDSS applications. The common denominator for these diverse scenarios is an objective of acquisition and application of knowledge to data to reach meaningful concepts at different levels of abstraction, thereby improving the quality and safety of medical care, while potentially reducing overall clinical costs.

0378] The ability to achieve this task capitalizes on the unique offering of the platform provided in accordance with certain embodiments of the present invention in the healthcare IT market as described hereinabove.

0379] An example list of users is provided in the table of FIG. 4A. An example User environment is described in the table of FIG. 4B. Typical Key Stakeholder/User-defined Needs are set out in the table presented in FIGS. 5A-5H, taken together.

0380] A high level description of the Knowledge Framework (KFW) capabilities, interfaces to consumers and the system configuration, now follows. Part of the KFW may rely on 3rd party components. For example, one may incorporate an existing commercial rule engine instead of developing one ‘from scratch’.

0381] The KFW typically utilizes services provided by the core platform shown and described herein. The KFW is responsible to provide services to acquire, maintain, store, and apply knowledge to data to reach meaningful conclusions at different levels of abstraction. The KFW may provide its services according to the Service Oriented Architecture (SOA) principles. These services may be available to systems shown and described herein and applications internally, e.g., business domains developed using Business Layer infrastructure shown and described herein, as well as to other healthcare IT applications externally, such as 3rd party Electronic Medical Records (EMR).

0382] SMW is a framework that enables continuous monitoring of a population, e.g., patients, within a defined medical information space. The SMW framework allows the development of Enterprise Clinical Decision Support Systems (ECDSS). These SMW-based ECDSS, also known as SmartGuards, capitalize on core platform abilities shown and described herein as well as consuming new services which may be provided by the KFW. For example, the core platform shown and described herein provides a holistic view of the patient clinical state manifested by the Virtual Patient Object (VPO) while the KFW provides the ability to interpret the data contained within the VPO to generate healthcare delivery recommendation.

0383] The new system may interface with the existing platform core shown and described herein to utilize several of its existing services. These services may include but are not limited to:

0384] The Security Layer—for services such as role/rule access control.

0385] The Support and Testability Layer—for services such as auditing and logging.

0386] The Common Terminology Service (CTS)—for services such as looking up terms originating from controlled medical vocabularies (e.g., SNOMED-CT).

0387] The KFW may provide SOA-based public interfaces to consume the services it provides, such as creating of a new Knowledge Module (KM) or evaluating a Knowledge Module giving specific patient data, to internal and external applications and authorized users of the system shown and described herein as shown in FIG. 6 which is a context diagram for the knowledge framework unit. The KFW System may comprise a Knowledge Acquisition Tool (KAT) and a suite of Management tools client components, a KFW Access Service, Knowledge-Base repository and one or more inference Engines server components as illustrated in FIG. 7 which is a system overview of the knowledge framework.
according to certain embodiments. KFW main functional units may include some or all of: a management functional unit also termed herein “KAI” and “PCT management tools”, access service (SOA), knowledge base repository, PCT management tools suite, and inference engines.

[0388] Typically, a KM or Knowledge Module comprises a knowledge evaluation rule, an input data model, an output data mode and terminology/ontology elements that are used by the KM. The server component resides on a variety of servers including an application server that hosts the KFW Access Service, an application server that hosts one or more Inference Engines, such as but not limited to RatfLah’s JBoss Drools engine, and a data server that hosts the Knowledge-Base repository. The server component may interface with existing components shown and described herein e.g. as detailed previously. This interface is supported by the existing SDK shown and described herein, although it may be extended or even enriched with new functionality if application-specific requirements arise.

[0389] The management tools suite is an optional functionality. KFW’s knowledge base repository may be derived from a suitable Knowledge-Module Entities Model such as, for example, that of FIG. 140 and a suitable programming technique, such as Hibernate, an open source Java persistence framework project, may be employed to convert business objects into DB structures, such that no DB modeling need be done explicitly. The client components reside on personal computers either as web-based thin clients or as full-blown desktop applications using technologies such as ASP.NET and SmartClient technologies respectively. Once the client component is installed on the PC, the user may access the KFW System from the PC through the network and described herein or via a secure Internet access following a successful authentication and authorization process.

[0390] The table presented in FIGS. 8A-8C, taken together, identifies capabilities, all of which are typically but not necessarily provided, of the KFW System in terms of benefits and features. The features are further described hereinbelow.

[0391] Some or all of the following assumptions and dependencies typically characterize capabilities of certain embodiments of the KFW System:

[0392] The KFW consumer is solely responsible to collect the data which may be used to apply a Knowledge Module.

[0393] The KFW typically stores neither information being used for evaluations nor the results of (e.g. conclusions derived from) the evaluations. Generally, the term “persist” is used herein to refer to storing in service storage such as a database.

[0394] The KFW information model may be inspired from the UMS shown and described herein, but does not necessarily rely on it.

[0395] The Common Terminology Service (CTS) shown and described herein provides functionality which may be used to embed terms originating from controlled vocabularies, clinical (e.g., LOINC) or administrative (HL7 Ethnicity), during the knowledge acquisition process. The functionality may include one or more of:

[0396] Find concepts using different search criteria (e.g., name, type, and domain).

[0397] Navigate through concept graph using semantic links, e.g. is_a or part_of associations.

[0398] Obtain metadata of a particular concept, e.g. value range type (numeric or qualitative), unit of measurement family.

[0399] The UMS information model and its derivations, such as baseline terminologies, are accessible electronically via the Reference Model shown and described herein or similar resource.

[0400] The knowledge acquisition process is performed by collaboration between domain experts and knowledge engineers.

[0401] Some of the KFW functionality may involve embedding commercially available software components e.g. commercial business rule engines such as Drools which is part of the commercial product JBoss distributed by RatfLah. Some of the KFW functionality relies on existing components of the platform shown and described herein, e.g. STI and Security layers.

[0402] Features of the KFW System, according to certain embodiments, some or all of which may be provided, are now described:

[0403] FEAT8 Data interpretation and inference capability: The KFW may receive as input (1) an identifier of a Knowledge Module to evaluate, e.g. definition of chronic hypertension, and relevant patient data, e.g. blood pressure measurements. The KFW may process the data and knowledge using an inference engine of some kind, e.g. a temporal reasoning engine. As output, the KFW may generate a conclusion e.g. patient had chronic hypertension during last year.

[0404] FEAT8.1 Iterative Knowledge Module Evaluation Interaction: The KFW enables its consumers to provide it with the data it uses to perform a Knowledge Module evaluation in an iterative manner. Thus, a possible intermediate evaluation result is the fact that additional data is to be provided to complete the current evaluation request processing. Then, after the consumer collected the missing data, it sends another evaluation request to the KFW. The intermediate result may be marked as not final.

FEAT8.2 Inference Evaluation Explorer: Each conclusion result may optionally contain an explanation of the knowledge and facts that lead to it. This explanation may be in a human-readable and machine-interpretable format. One of the evaluation request arguments may explicitly direct the KFW whether to generate such an explanation or not, as part of its output. For example, the hematocrit and haptoglobin serum levels of a patient, leading to the conclusion of having ongoing hemolysis, may be presented together with the definition of what an ongoing hemolysis is.

FEAT8.3 Temporal Reasoning Capability: A special type of inference capability that the KFW may support is temporal reasoning. A common feature of many healthcare information types is having a time stamp. The valid time of a WBC lab test result, the request time to perform a MRI imaging study and gestation condition start time are only some examples. The temporal reasoning method enables to interpret large sets of time-stamped raw data into meaningful concepts at different levels of abstraction. This may include abstraction of individual time points to longitudinal time intervals, computation of trends and gradients from series of consequent measurements, and detection of different types of patterns, which may be otherwise hidden in the raw data.

FEAT8.4 Probabilistic Reasoning Capability: Another type of inference is undeterministic or probabilistic reasoning. Optionally, insufficient, incomplete, or less reliable data is input to an inference process. Or, non-determin-
istic data abstraction methods such as fuzzy logic may optionally be applied including reflecting the data quality implications on the final output in terms of confidence level. For these purposes exactly probabilistic algorithms, such as Bayesian Networks or Belief Networks, exist which reflect the data quality implications on the final output in terms of confidence level. This optional capability typically utilizes a specialized knowledge representation model that captures the probabilistic nature of the knowledge and corresponding inference mechanisms.

FEAT9.5 Context-Based VPO Schema Generator: A special type of evaluation result is a recommendation to an information schema of a Virtual Patient Object (VPO). A VPO schema can be generated when given as inputs the clinical state of a patient, e.g., current protein list, and possibly the preferences of a care provider, e.g., clinical specialty. This custom-tailored VPO may benefit the care provider by decreasing the amount of information to review during a care delivery session. FEAT8.6 Knowledge Evaluation Simulator: A helpful feature to a knowledge editor is to simulate the evaluation process of a Knowledge Module. The simulator typically supports debug-like mode enabling to find problems in the Knowledge Module definition or inference engine. In addition, the simulator may have a capability to generate a representative data set that covers possible inference execution branches.

FEAT9 Robust Knowledge Module Representation Model

[0406] FEAT9.1 Descriptive Information Traits: A section common to all types of Knowledge Modules is the one that contains descriptive information about the Knowledge Module. The Knowledge Module metadata describes its scope, goal, function, etc. This may include its creation date, modification dates, authors, version details, and classification keywords, whether codified or not. This information serves as the basis to search and retrieve Knowledge Modules by submitting search queries that comprise the descriptive information traits.

[0407] FEAT9.2 Standard Boolean Expressions: Simple rule-based clinical knowledge is often expressed using straightforward Boolean expressions, i.e., and/or clauses. Thus, the KFW may support the representations of such expressions.

FEAT9.3 Declarative and Procedural Knowledge: Clinical knowledge can be divided into two main types, declarative and procedural knowledge. The declarative knowledge type defines meaningful clinical concepts related to the patient’s clinical state. For example, a state Knowledge Module definition (i.e., low, medium, or high) comprises declaring possible qualitative ranges of diastolic and systolic blood pressure to each qualitative value. The procedural knowledge type usually defines clinical workflows and procedures. This may be for instance a recommendation of treatment for patients suffering from high blood pressure (e.g., routine blood pressure monitoring). It usually involves the explicit specification of control structure such as do-in-parallel, decision steps, or mandatory actions. The KFW may provide specific knowledge representation (KR) models to capture these two distinct types of knowledge to support a broad range of Clinical Decision Support Systems (CDSS). Moreover, the KR models used to capture these knowledge definitions may be based, or at least adhere, to existing healthcare standards such as HL7 GLIF for declarative and procedural knowledge respectively.

FEAT9.4 Terminologies and Vocabularies: A Knowledge Module definition often comprises terms and concepts originating from controlled medical vocabularies. These concepts are typically found both in the metadata section of a Knowledge Module, e.g., an applicable clinical state such as diabetes mellitus using coded keywords from SNOMED, and in its knowledge definition section, e.g., referring to a certain type of blood lab test such as WBC count using LOINC codes. Thus, the KR model of a Knowledge Module provides special placeholders enabling to embed terms and concepts during the knowledge acquisition process. Moreover, embedding terms and concepts originating from well-accepted and commonly-used terminologies facilitates sharing and reuse of Knowledge Modules at different clinical settings.

[0408] FEAT9.5 Interrelation between Knowledge Modules: A Knowledge Module definition often relates to knowledge definitions of other Knowledge Modules. To ease and simplify the knowledge acquisition process an ability to interrelate Knowledge Modules may be provided. Common Knowledge Module interrelations include inheritance and composition. For example, the Proteinuria state Knowledge Module (i.e., protein levels in urine) can be derived to capture gestational proteinuria without repeating the common knowledge parts. As another example, the HELLP syndrome Knowledge Module comprises hemolysis state, liver enzymes state, and platelet state Knowledge Modules, each being valuable on its own too. Thus, the KR may cater for this by enabling to base the definition of one Knowledge Module on other Knowledge Modules via common interrelations such as specialization (i.e., is-a) and composition (i.e., part-of).

[0409] FEAT9.6 Automatic Triggering Events Extraction: The evaluation process of a Knowledge Module may impose on the requesting consumer to perform preliminary operations, for example gathering data according to application-specific Knowledge Module data requirements, which are costly in terms of time and computational resources (e.g., network traffic). In addition, in some cases it is critical to evaluate a Knowledge Module as soon as one of its comprising data elements changes in the CDR. Therefore, from a Knowledge Module definition, a consumer may be able to automatically extract events, in terms of data changes in a CDR (e.g., arrival of a new WBC lab test message), which trigger the necessity to evaluate the Knowledge Module.

[0410] FEAT10 Knowledge Base Repository

[0411] FEAT10.1 Knowledge Module Search Engine: There may be a mechanism that enables to search and retrieve Knowledge Modules. During runtime, before requesting to evaluate a Knowledge Module, it first may be retrieved and explored to provide application-specific data requirements as input. During design-time, to maintain the Knowledge Module definition, a knowledge editor may be able to locate it before editing. The Knowledge Module search engine allows searching and retrieving of Knowledge Modules by submitting search queries. These search queries capitalize on the descriptive information section of a Knowledge Module. Two possible search queries are free-text search query or a context-based search query. The free text search query contains a single search string to be matched against any text-based section of a Knowledge Module. The context-based search query comprises a structured query, for example XML, which defines for each searchable attribute or trait of a Knowledge Module its expected value. For example, a search for Knowledge Modules of John Foster may use the Author descriptive
information trait. Another example is a search for Knowledge Modules that expect to receive a certain LOINC-coded lab test typically using the data requirements’ Knowledge Module attribute.

FEAT10.2 Knowledge Module Definition Import and Export Mechanism: A mechanism that enables to import and export Knowledge Module definitions in or out of a KB repository. An XML schema defines the Knowledge Module file format and allows a Knowledge Module to be imported from one KB and exported into another KB.

FEAT11 Knowledge Acquisition Tool: One of the KFW client components is the Knowledge Acquisition Tool (KAT) that facilitates domain experts and knowledge engineers, acting as knowledge editors, to collaboratively acquire, edit, and maintain the knowledge definitions and descriptive information of Knowledge Modules stored in the KFW knowledge-base repository. Besides knowledge editing capabilities, the KAT may contain several other features, as described hereinbelow. FEAT11.2 UMS Explorer: The underlying clinical information model of the KFW is the UMS shown and described herein. Hence, the definition of any Knowledge Module typically comprises UMS entities such as LabResult, Diagnosis (type of Condition entity), and Medication. The UMS information model is relatively large in scope and contains diverse attributes, complex data types, and various relations between its entities. Thus, the KAT may contain a specialized module, named UMS Explorer, to explore the UMS information model and assist the knowledge editor to navigate and find UMS entities and attributes that comprise a Knowledge Module definition. The functionality expected of the UMS explorer includes: free-text search and context-based search of UMS entity and attribute (e.g., entities that use a certain vocabulary domain), descriptive information, and navigation according to relations (e.g., related-act).

FEAT11.3 Knowledge Module Explorer: A Knowledge Module definition may relate to another Knowledge Module. Different types of relations can exist between Knowledge Modules, including specialization (i.e., is-a) and aggregation (i.e., part-of). For example, the Hypertension Knowledge Module, which defines the clinical state of having elevated blood pressure, can be derived by another Knowledge Module to define gestational hypertension. Thus, the KAT may contain a specialized module, named Knowledge Module Explorer, to explore the KB repository and assist the knowledge editor to navigate and find Knowledge Modules during the knowledge acquisition process. The functionality expected of the Knowledge Module explorer includes: free-text search and context-based search of Knowledge Modules (e.g., all Knowledge Modules that use a certain SNOMED code), descriptive information, and navigation according to relations (e.g., is-a).

FEAT11.4 Concept Explorer: A Knowledge Module definition may contain terms and concepts originating from controlled medical vocabularies (e.g., LOINC code for WBC count). Different types of terminology relations exist serving different coding habits. For example, the SNOMED terminology is usually used to code diagnosis terms, signs and symptoms. Thus, the KAT may contain a specialized module, named Concept Explorer, to explore the variety of terminologies and assist the knowledge editor to navigate and find concepts to embed in a Knowledge Module definition. The functionality expected of the Concept explorer includes: free-text search and context-based search of concepts (e.g., diabetes code in SNOMED), descriptive information, and navigation according to relations (e.g., is-a), if such exist, between concepts.

FEAT12 Role and Rule Access Control: By capitalizing on the Security Layer shown and described herein, the KFW provides a secured and controlled environment at design-time and runtime. Users and consumers may be authenticated and receive authorization before accessing KFW services and knowledge. Special roles and rules may be defined for the KFW to cater for its unique services and protect its special resources. For example, a knowledge editor role may be defined as well as a rule determining who can edit a certain Knowledge Module.

FEAT13 Auditing and Logging: By capitalizing on the STI Layer shown and described herein, the KFW provides a robust and flexible audit and log capability. The level of auditing and logging can be changed according to different scenarios, for example testing vs. production. Special auditing and logging policies can be applied to different services and resources of the KFW, for instance applying a meticulous auditing and logging policy to knowledge creation and maintenance operations.

Applicable Standards: The KFW functionality is largely based on the functionality specified in the HSSP DSS SFM as described herein above to facilitate interoperability with KFW services.

An embodiment of the smart-watch sub-system is now described generally with reference to FIGS. 14-17. By their nature, clinical decision support systems are very complex. The challenge of creating such systems is even greater in a heterogeneous and distributed environment where the data—the cornerstone of any decision support system—exists in different locations (often hard to access) with different structures, standards and terminologies, often lacking any common semantics. Many of the existing EMR systems provide a rich set of decision support functionalities but these rely on each system’s limited data. Typically, an enterprise-wide decision support framework co-exists with the EMR CDS modules.

There are substantial market opportunities created as a result of the above challenge in the context of increasing pressures from government, payers, employers and patients to improve and substantiate patient safety and treatment outcome while reducing costs. Focused efforts to improve care of patients with chronic diseases (e.g., CMS targets, pay for performance) seek to provide orchestration of care and related information across venues of care both within IDN’s and across facilities not within the same organization. Realization of the promised benefits of electronic medical record (EMR) systems, ensuring completeness of the data used by the EMR’s in assisting with clinical decision support and bridging the gap between care locations and facilities across the continuum of care with different EMR’s employ interoperability solutions, including enterprise clinical decision support capabilities and services.

To enable automated decision support services at the enterprise level the dBMotion Platform product may be employed, thereby achieving interoperability, and presenting medical information from disparate sources as if it originated from one normalized source, using common representation, terminology and semantics. This enables the creation of an
advanced infrastructure for an enterprise-wide clinical decision support framework termed herein the SmartWatch Subsystem.

[0422] The Enterprise Clinical Decision Support System (ECDDSS), also termed herein the SmartWatch subsystem, may work in conjunction with the commercially available dbMotion platform. The SmartWatch subsystem addresses the challenge in such a way that no centralized database is mandatory, thus enabling enterprise decision support services both in a fully distributed as well as centralized model (and any hybrid in between). The SmartWatch subsystem is not intended to replace existing clinical decision support systems currently running within any of the existing EMRs, or any other systems in the organization. Instead, it aims to address problems that cannot be tackled by any operational/legacy system (e.g. EMR) that often has access only to a partial subset of the patient’s clinical information.

[0423] The SmartWatch subsystem may facilitate both the design time and runtime activities involved in the creation and use of ECDDSS. These activities may include some or all of: knowledge elicitation, eligibility determination, continuous patient monitoring providing timely alerts, notifications and more. Care providers are then able to carry out the appropriate actions, in a timely manner, thereby increasing the quality of care and lowering costs that might arise from possible deterioration in a patient’s condition.

[0424] SmartWatch does not attempt to replace the physicians’ role in complex decision making instead it is targeted at giving help and support with decision making—performing time-consuming, data-driven or time-driven tasks, and bringing the human factor into the picture only when human involvement is inevitable.

[0425] The SmartWatch subsystem may rely on a built-in knowledge base, maintained by medical professionals. This knowledge base allows the SmartWatch subsystem definitions (matching rules, etc.) to be set in an abstract, high level language, rather than having to code rules in a computer language.

Definitions, Acronyms and Abbreviations used herein include:

[0426] SmartGuard: —A SmartWatch solution that is built to watch over a specific population. A live SmartWatch system may include many SmartGuards. SmartGuards are logically (and probably technically) separated from each other. One may envision the SmartGuard to be a set of configuration settings defining a SmartWatch virtual instance, the population involved, the information to be employed, the rules and actions that may be performed.

[0427] Population: —A set of patients grouped by some criteria. These criteria could be medical (people diagnosed as . . . ) encounter based (Currently hospitalized in ward . . . ) demography-based (people over 90) or any other UMS-based criteria. A population may also include other elements derived from dbMotion’s UMS such as organizational units, physicians and so forth.

[0428] Information: —The set of data elements to be collected for a population member (a patient). Typically each SmartGuard would define different application-specific information needs.

[0429] EMPI: —Enterprise Master Person Index: Software that attempts to cluster patient records from separate applications (AKA operational systems) usually using probabilistic methods. An EMPI cluster contains a set of indexes, each representing a demographic record in an operational system.

[0430] Subscriber: —is an entity that signs up (to some SmartGuard) to be notified when a certain condition matches a population member. The subscriber may be a person, or another system.

[0431] HIPAA: —Health Insurance Portability and Accountability Act, which is the leading regulation in the United States for the use and disclosure of patient health information.

[0432] Population criteria: —the criteria that defines population membership. This may include, for example, inclusion/exclusion over UMS entities, or more complex rules which use other considerations such as time, age and so on.

The SmartWatch subunit may have some or all of the following properties and functionalities:

[0433] Population—The problems focus on populations of patients (or other entities) that share some characteristics such that monitoring them is beneficial.

[0434] Information—The relevant population is monitored for some limited set of clinical (or other) data elements—giving SmartWatch an advantage where information is scattered over an enterprise, in different systems and at disparate locations, possibly coded using different terminologies.

[0435] Rules—The population being watched is expected to be “ok” most of the time for most examined subjects (patients), the objective being to find the abnormal elements—to identify situations where some action is to be taken for a population member (patient). The problems that fall into the SmartWatch profile would usually involve clinical knowledge, introducing two complex issues:

[0436] The ability to apply clinical knowledge to the raw data and derive decisions from it involves intelligent inference mechanisms & knowledge representation.

[0437] Clinical knowledge changes constantly and therefore requires constant knowledge maintenance by knowledge experts.

[0438] Actions—the functionality of communicating monitoring results to the outside world. Many communication mechanisms can be envisioned to serve this, allowing much flexibility in implementation. Also, a possible need for workflow capabilities is recognized in order to allow escalation reminders and so on, in order to close the loop.

[0439] Three examples of tasks the SmartWatch subsystem may perform are now described:

[0440] 1. Active monitoring of a patient’s clinical condition in a distributed, semantically heterogenous environment. The importance of monitoring a patient’s condition can be viewed for example in the context of chronic patients. In a typical chronic patient scenario, the target audience is identifiable, and so are the indicators to monitor. In such a scenario, monitoring is a long term process, and identifying an abnormality that leads to a corrective action can be of great value. However, this monitoring is greatly facilitated by computerized assistance: the number of patients can be very large. Patient data volume can be overwhelming. Events are not detectable locally—patients visit different hospitals,
clinics, meet different caregivers. Lack of semantic interoperability, different data structures and languages exist. A typical multi-hospital environment introduces more difficulties such as different systems, usually distributed; different APIs; different semantics are used. No unique patient identifier exists.

Example: Monitor all diabetic patients—checking that their clinical diabetes indicators such as Lipid profile (LDL), Glucose, HbA1C, Creatinine, urine microalbumin and so forth are within range, and are performed promptly: making sure Eye exam (DRE) and Foot exam are carried out promptly and so forth. When a certain test is not done within the timeframe, or results demand provider attention, SmartWatch may notify the local EMR, with a mail to the primary physician. This affects quality of healthcare. Poor monitoring may lead to complications, deterioration of patient condition, and potentially further treatment costs. Non-automatic monitoring is time-consuming and inefficient. Certain embodiments seek to provide a preemptive action leading to a care provider response to the patient’s condition.

Gathering medical research data: When Collecting clinical data for research, such data sometimes needs to be de-identified. This affects researchers who may need to gather clinical data records to base their research on. The impact includes high costs of assembling research data from a variety of sources, cherry-picking patients that match the criteria, inconsistency in patient records due to diverse source semantics, patient privacy issues, and wrong patient identification due to de-identification methods. Certain embodiments seek to eliminate or reduce any need to collect such data manually, improve data integrity by supplying an integrated patient record, and enhance patient privacy with automated patient records gathering, so patient identity is not disclosed to researchers in any way.

The task of public health surveillance may include looking out for abnormal healthcare patterns on a large scale. This can be done by monitoring information sources such as ER visits, medication consumption, diagnoses and so on.

For example, an alert may be issued when, in one week, 5 patients from the same area (zip code) are suspected to be suffering from bacterial meningitis. This task typically affects disease control centers officials, ER managers and public health. Poor epidemic detection does not allow authorities to respond on a timely basis, resulting in the potential infection of more people, with higher treatment costs. The impact of the epidemic outbreak may be drastically reduced, following corrective action, such as containment, before the epidemic gets out of hand.

The SmartWatch subsystem is a platform for developing services that facilitate an enterprise-wide clinical decision support system (ECDS) that utilizes the Service Oriented Architecture (SOA) environment provided by the dbMotion platform. The subsystem is geared to identify a target population according to eligibility rules, and monitor the state of the target population members over time, according to another set of monitoring rules, and eventually trigger proper actions such as alerts or reminders when a rule is matched.

The subsystem may facilitate both the design time and runtime activities involved in the creation and use of ECDS. These activities include knowledge elicitation, eligibility determination, monitoring patient state continuously, providing timely alerts, notifications and more. Consequently, care providers can carry out suitable actions when appropriate, thereby increasing the quality of healthcare and lowering costs arising from deterioration in the patients’ condition.

The SmartWatch subsystem is intended to rely on a built-in knowledge base, maintained by medical professionals. This knowledge base allows the SmartWatch subsystem definitions (matching rules, etc.) to be set in an abstract, high level language, rather than having to code rules in computer language.

The SmartWatch subsystem can serve as a complementary framework that may benefit other systems. Currently, cutting edge EMR systems provide CDSS capabilities usually during data entry (i.e. in real time) although they are dependent on the data available to the EMR itself. In contrast, SmartWatch’s design enables retrospective CDSS capabilities offering a near real-time response. Thus, a preferred model for The SmartWatch subsystem is one of synergy in which the SmartWatch subsystem provides a continual monitoring of clinical information and notification services to other applications.

SmartWatch does not aim to replace the physicians’ role in complex decision making, but to provide help and support in decision making by performing time-consuming, data-driven or time-driven tasks, and bringing the human factor into the picture only when needed.

The SmartWatch model may include 4 logical layers as follows:

1. Population: The Population layer defines and manages the target population to be monitored. This is accomplished either with eligibility criteria that define the population boundaries and detect new population members automatically, or by letting users enroll members directly.

2. Information: The Information layer controls information to be gathered and processed for the population members, once their membership has been established. This information can be used later by other (higher) layers, for example for rule evaluation or within an action’s output message. Required information, if any, is not necessarily given in accordance with the member state, as different stages of analysis may employ different data elements.

3. Rules: The Rules layer defines the conditions for which any action may be taken for population members. It defines the abnormal behavior that the SmartWatch subsystem monitors for its population members.

4. Actions: The Actions layer handles actions that may be taken when a population member (patient) matches a rule, usually when some abnormal condition has been identified. Actions typically comprise a workflow detailing reminders, escalation procedures, and close-the-loop functionalities and may include services to deliver messages using standard channels to humans or other IT systems.

A high level conceptual view of SmartWatch, based on the 4 layers mentioned above is illustrated in FIG. 14. As illustrated, there is communication between the 4 logical SmartWatch layers. Each layer depends on preceding layer artifacts. In addition, SmartWatch communicates with dbMotion platform services such as security services, EMPI, and
the management layer. Also illustrated is the dependency of SmartWatch on the Knowledge Framework in supporting clinical decision making.

SmartWatch may employ several storage elements that include SmartWatch Metadata, in which the SmartGuard (the SmartWatch Solution) properties are retained, population and possibly VPO storage, in which the population and information are stored. The Knowledge Framework may add storage—a knowledge base—of its own. The VPO storage system is used to store ePHI, and for that reason careful thought may be taken on how to store them without compromising patient and business security and privacy. SmartWatch provides services to external parties, allowing complementary applications to fit in and form a solution.

The knowledge-framework role is to allow SmartGuards to be defined in terms of clinical concepts and abstractions, rather than referring directly to raw UMS data. The knowledge framework may be able to connect a concept, or a knowledge module, to data and to a suitable computation mechanism.

The SmartGuard can be thought of as a single SmartWatch subsystem solution, monitoring a specific population. The SmartWatch subsystem may host several SmartGuards running simultaneously. Each such SmartGuard may contain all the details it uses in order to operate: the population criteria, information to be employed, the rules to be matched, and the actions to take when rules fire.

SmartGuards are tools for retrieving information between hospitals. To make sure that no unauthorized query is performed in a hospital repository, SmartGuards may require hospital approval to operate.

SmartWatch is a framework, which provides services, and therefore employs additional components to form a comprehensive solution. FIG. 15 presents such a solution that includes a dbMotion platform that provides the application-specific required data for the SmartGuards, SmartWatch itself, and a clinical application that users interact with in their daily routine, such as a CPOE or EMR. This clinical application interacts with SmartWatch, and provides the user with a subscription mechanism, a message viewer (e.g., view SmartWatch patient messages within the patient file; action items for the physician after login to the system and so on), and an ability to close the loop (e.g., by marking a checkbox verifying that the message was addressed, by simply viewing the message, or by initiating a process tracking the patient condition). None of these interfaces is mandatory, but failing to provide one may result in the creation of a partial solution only.

A clinical application as shown in FIG. 15 may be referred to here as a complementary application. This document assumes the existence of such an application, provided by dbMotion or by an external provider, to form a complete solution. Also shown in FIG. 15 are messages sent to users directly, via for example SMS or Email protocols. Messages sent to human users in this form, would usually be accompanied by a corresponding message to the complementary application, while providing elementary information and directing the user to view the full message details in the complementary application.

FIG. 16 presents a domain model for SmartWatch, implying both its static service-oriented structure, as well as the dynamic processes involved. It shows SmartWatch containing several SmartGuards, maintaining their properties through services, as well as the dynamic messages coming into the SmartGuard process from dbMotion network, and out of the process to users and systems. Also shown is a clinical knowledge base, and its editor, CDSS that may be called by SmartWatch during patient records analysis.

Some or all of the following functionalities, and associated supporting features, may be provided:

- Patient condition monitoring: Physicians gain a monitoring tool for their patients that relieves them of certain tedious tasks, allowing them to concentrate on more complex cases. In a pay-for-performance environment, physicians/clinics gain actual income when SmartWatch helps them to meet suitable quality measures. For this functionality, associated supporting features may include some or all of the following:
  - User direct notifications
  - Alerts to external, complementary applications
  - Reminders & Escalation notifications
  - Mandatory message properties
  - SmartGuard definition tools
  - Additional pluggable workflows
  - Subscription service
  - Population enrollment service
  - Close the loop capabilities

Researchers may greatly benefit from a tool that gathers comprehensive, semantically coherent, research-oriented patient records from distributed systems. This eliminates the effort of piling up research data from various inconsistent sources, and the need to reconstruct de-identified patients’ records into virtual patients. For this functionality, associated supporting features may include some or all of:

- Alerts to external, complementary applications
- SmartGuard definition tools
- SmartGuards rely on high level medical abstract concepts
- Subscription service
- Research purposes output service
- Public health surveillance

A SmartGuard monitors admission records to ER, identifying abnormal patterns in a patient’s condition, alerting a possible epidemic. A benefit of this is public health, but hospitals and regional/local disease control centers also benefit from task automation. For this functionality, associated supporting features may include some or all of:

- User direct notifications
- Alerts to external, complementary applications
- Reminders & Escalation notifications
- Mandatory message properties
- SmartGuard definition tools
- Additional pluggable workflows
- Subscription service
- Close the loop services

According to certain embodiments of the invention, SmartWatch relies on the dbMotion infrastructure as its main data source and relies on some or all of the following dbMotion Services: Security, Auditing and Logging (STL, CEB).

This may enhance re-use of any needed functionality which already exists in a dbMotion product. SmartWatch relies on its decision logic in the Knowledge Framework component for modeling clinical knowledge, and in applying that knowledge to patient data. SmartWatch is a service-based product, that assumes the existence of complementary applications (not necessarily part of the basic SmartWatch) from which users can actually subscribe to SmartGuards, view the
artifacts—resulting messages, and let SmartWatch know when the issue is addressed (close the loop). SmartWatch analysis assumes the existence of an EMPI service, with a one-to-one relationship between the EMPI cluster and the virtual patient SmartWatch refers to. Even if there are more than one (several) clusters returned from the EMPI, SmartWatch may use only one of them. If the EMPI clustering quality is poor, SmartWatch’s decision quality may be affected, as it may miss patient records or be confused by records actually related to another patient.

[0492] According to certain embodiments of the invention, SmartWatch assumes each EMPI vendor supplies the “member-of” functionality, whose input is an index and its output is the cluster the input belongs to, along with all cluster indexes. This converts an operational system patient index into a virtual patient which contains all indexes in the cluster. SmartWatch assumes that EMPI vendors do not supply a change log service, detailing clusters whose state has changed. However, if such a service is available, keeping up with patient EMPI state may be made much easier and may simplify the process.

[0493] The system need not provide an application (GUI) for managing the actions and states related to population members. Instead it may only provide SOA based infrastructure that may enable developing such capabilities.

[0494] Possible Consumers for the SmartWatch subunit are described in the table of FIG. 17.

User environment: In general, the expert, Super User category is envisioned using networked, personal computers. This is a small, predefined set of users:

[0495] The SmartGuard editor user role may use a designated application to configure the SmartGuard’s he/she is authoring. This application may be the SmartWatch management tool or a similar tool.

[0496] The SmartGuard/SmartWatch administrator role is envisioned using the SmartWatch management tool to perform administrative, maintenance, monitoring and configuration of a SmartWatch product implementation. This includes tasks like review & approval of SmartGuards before they actually run, viewing status & operation statistics (e.g. population statistics, number of actions taken, exceptional conditions etc.). This user type may exist in any dbMotion node.

[0497] As far as the Consumer category is concerned, the situation is far less clear. This is envisioned as an area that includes many professional services with customization & configuration, aimed at getting to end-users, in the most effective way, to meet their needs with working modes in the specific project rather than the “one suit fits all” product approach. This statement implies that users may be working in any kind of electronic environment, from mobile phones, thru PDAs, networked personal computers, mainframe terminals to fax machines. The following is a classification of delivery methods that may be useful:

[0498] Direct, standard delivery channels (e-Mails, SMS, fax and so on).

[0499] Complementary application that may be the users’ working environment, to present the SmartGuard’s output. Such an application may be a commercial EMR, dbMotion Clinical Views™, or some other proprietary clinical application.

[0500] In addition, the following factors (among others) may be considered in deciding which channels are best in a given situation:

[0501] Urgency level—Life-threatening messages may be accompanied by a text message to the relevant physicians’ mobile device, while less urgent messages may be pushed to an application queue, to be presented to the next caregiver.

[0502] Enterprise preferences—Enterprise policy may prefer to send notifications to its physicians’ mobile devices, or to a reliable follow up mechanism/EMR system. Each implementation may include a configuration stage during which it may be decided how and where the notifications are addressed.

[0503] Personal preferences—Consumer-type users in general and PCPs in particular, may have different levels of automation in their offices. Also, the nature of their activities may dictate out-of-office hours during which a physician would prefer to be notified through a mobile device. These typically simplified constraints may require a diversity of abilities to receive and respond to notifications.

[0504] Time factors—An SMS message at night time may either be rescheduled to a reasonable time, or sent to the on-shift physician who is currently responsible for the patient.

[0505] Medical records may be gathered for specified groups of patients (population), providing proactive messages based on the records collected. SmartWatch may provide an enterprise-wide view of patients and an ability to analyze the information, apply rules to them, and as a result perform actions in which messages are sent to external actors to respond to SmartWatch findings. The response time (measured starting from an event happening in real life, and ending as user views the resulting message) expectation is at most near real-time.

[0506] SmartWatch may be supplied with an existing content that covers common medical cases.

[0507] This content is to be supplied in the form of existing definitions of populations, related knowledge and actions, packaged as a set of SmartGuards. This means that each SmartGuard may be a deployable unit, packed and deployed together.

[0508] Some or all of the following functionalities I-VI may be provided to facilitate identification and management of populations:

[0509] 1. SmartWatch may support population discovery by several parallel methods, as detailed below. The fact that population members are discovered by different mechanisms may not impact the rest of the layers, i.e. in a given SmartGuard there may be population members (patients or other) that were discovered automatically and others that were enrolled by their physician, but the rest of the SmartGuard flow may work the same for all cases. Therefore, some or all of the following population discovery mechanisms I-5 may be provided:

1. Automatic population discovery by eligibility criteria: SmartWatch is typically able to discover patients (or other UMS entities) as matching the SmartGuard criteria by using knowledge from the Knowledge Framework (triggering events).

2. Population based on external list: In some cases the population could be known to an external system that can provide a prepared list of patients to work with. SmartWatch may provide an interface that enables project specific implemen-
3. Patient enrolled in population by care provider: optionally, SmartWatch may allow consumers (e.g., care providers) to enroll patients into some SmartGuard population. This requirement arises from two scenarios: scenario 1) — let physicians register their patients to an existing population, for which they are not qualified by the eligibility criteria—imagine a SmartGuard watching over CHF male patients over 65, and a physician who wants to add his patient to this population even though he is just 62-enabling human decision-making to override the computer logic; scenario 2) — a SmartGuard that is voluntary, includes no eligibility criteria, and is based on patients/physicians deliberate enrollment—for example, monitoring the influence of a new drug, a diet and so forth. This requirement, if provided, probably involves much customization to supply users with tools to perform such an enrollment (for example search and choose leading record; enroll button from patient file). SmartWatch may provide an interface for adding patients to a population (high priority), together with tool (GUI). This tool may also be useful for testing.

4. Patient enrolment in a population by patients themselves—SmartWatch may enable patients to enroll themselves in a SmartGuard solution.

5. Patients pushed between SmartGuards: it may be useful to move patients between SmartGuards when they fit some pattern. SmartWatch may therefore enable a SmartGuard action to add patients to another SG population. For example, a population that monitors elderly patients (over 65) may push patients to another SmartGuard that are related to age criteria.

II. Partially matched population: In defining SmartGuard’s population eligibility criteria, it is very likely that even though certain patients do not qualify for the population, they may not be excluded from it and may be added in the future. Some examples: a lab result is missing, but might be provided shortly; age—the patient is too young and may qualify in x days; patient identity may change to add an index that carries with it the data which may be used to qualify for the SmartGuard. The requirement is typically not to lose such patients, and to somehow put the system into a “sleeping mode” (for these patients), waiting for the missing condition to occur.

III. Online connection as a SmartWatch population information source: SmartWatch typically utilizes the dbMotion core product, using it as its information source. However, it cannot assume the dbMotion CDR as the sole information source. It also addresses information sources that are available through dbMotion online connections as well. A list of application-specific requirements from an information source may be assembled as well as a list of scenarios that could not be supported in such a case. Examples for online information sources are medications thru RxHub, claims data (P4P BUC), and Pan-Canadian standard.

IV. STRQ55 Run as background process and on demand: SmartWatch may be able to run as a background process, expected to perform most of its data fetching activities outside working hours whenever possible, in order to avoid network overload. The schedule in which SmartWatch collects data depends on the SmartGuard it is working for. These schedules are typically adjustable per SmartGuard. However, it may also be possible to make SmartWatch run a specific SmartGuard on demand (immediately) regardless of the defined schedule.

V. SmartWatch observations do not have to be patient-centric: dbMotion is also capable of collecting data on medical elements that are not actual patients; SmartWatch may exploit this to monitor entities that are not patients as well. Although most populations described in this document relate to patients, it may be understood in the broader sense of UMS-entity centric. For example SmartWatch may also observe the patient capacity of a ward, where the observation is ward-centric; or the usage of some set of expensive medication, where the population is medication-centric.

VI. Find mass Trends: SmartWatch may be able to identify events resulting from an accumulation in some population, rather than focusing on the individual patient. Example: disease control scenario, in which the event is — X people of the same zip code area are hospitalized within N days and diagnosed with diagnosis D.

SmartWatch may depend on the knowledge framework in formulating & executing its clinical rules and decisions. Functionalities of the knowledge framework which may interact with SmartWatch or otherwise be used thereby, may include some or all of the following functionalities A-G:

1. Separation of the knowledge (Knowledge Framework) from the execution & delivery engines (SmartWatch). Separation may be desirable for some or all of the following reasons 1-3:

   a. Allow clinical knowledge & software engines to be separately maintained and released.

   b. The clinical knowledge that would be modeled for SmartWatch has a different timeline to the software updates. The knowledge changes are driven by clinical innovations and discoveries whose timeline cannot always be planned (such as taking a drug off the shelves). The engines on the other hand, are managed in software lifecycles, driven by new application-specific requirements, performance issues and so on. For this reason it is highly desirable to have the ability to deliver knowledge updates on a parallel track to software updates.

   c. Clinical professionals may actively participate in knowledge modeling & elicitation.

   d. One problem with incorporating clinical knowledge into code written by programmers is the poor communication that arises, resulting in modeling errors. Knowledge Framework application-specific requirements in the long run typically incorporate the ability of clinical subject matter experts to model and validate the content. Therefore, Knowledge Framework may provide a tool for knowledge management/acquisition.

   e. Managing and changing the knowledge at project level: Knowledge customization at the customer level may facilitate SmartWatch solutions in areas where clinical consensus is lacking. This ability may assist in overcoming out-of-consensus situations where the knowledge involved in the out-of-the-box solution is not accepted by customers’ subject matter experts, by allowing local experts to alter the knowledge as they see fit. Therefore, SmartWatch and Knowledge Framework may enable managing and changing the knowledge at project level (rather than by product level). Furthermore,
such changes may not affect the original out-of-the-box components, but may be done on the “Save as . . .” version of the implementation project, in order to minimize compatibility issues in the future.

[0522] B. Probabilistic decision logic: The clinical decision support may involve, where appropriate, probabilistic capabilities. Such probabilistic capabilities may be used where either the decision model dictates directly, or in cases where there are grey areas, for example, age weighed against other clinical indicators (patient is not 65, she is 64.95 but has strong indicators). In such cases, the decision (e.g., whether to invoke an action, to include or exclude some patient in the population) can be made using a threshold which is a property of the SmartGuard. In a way, the threshold value determines the balance between possible false positives (threshold set too low, which may result in high rate actions to be taken based on false interpretations) and false negatives (threshold is set too high, missing true actions whose score was not high enough). Setting such a threshold depends on the nature of the SmartGuard—the risks involved in each kind of error, the impact of each and more.

[0523] C. Knowledge may be stated using normalized concepts represented in controlled medical vocabularies.

[0524] D. Rules may include full Boolean logic operators & expressions.

[0525] E. Time-dependent rules: Many situations may include time aspects in which SmartWatch may initiate a revisit to patient information driven by time related conditions, for example, verifying that a certain test is performed every 3 months. Simply waiting for the test to show up may be inappropriate.

[0526] F. The decision support logic may be backed up with references & explanations based on individual data.

[0527] In order to persuade consumers of the correctness of SmartWatch recommendations, an explanation documenting the decision support logic may be employed. This may include a reference to information on which the knowledge was modeled, as well as an explanation about how this knowledge was applied to the patient data—resulting in the recommendation. Such information may be sent to consumers in the message or viewed in the audit report. The purpose is to show who the knowledge authority is (society, expert, guideline, CMS, P4P) and to show how the system reached the conclusion.

[0528] G. User Defined Logic: In contrast to enterprise-defined rules to which an end-user can only subscribe, without the ability to change the rule logic, SmartWatch may also allow a user (physician) the ability to define his own rules, or refine an existing rule.

[0529] During a SmartGuard operation, actions are invoked, typically including various means to get a message across, making sure SmartWatch’s recommendations reach the user in the most suitable way for him/her, within the specific context and, if necessary, making sure it is properly handled. This calls for a very powerful, flexible set of services for taking advanced actions. This may be a combination of out-of-the-box solutions and ‘application blocks’ as well as an environment to code a per-case/need workflow/process. An example set of requirements suitable for implementing this includes some or all of the following requirements 1-20:

1. SmartWatch message typically contains sufficient information for the recipient to understand it: SmartWatch actions’ output could be a readable message, sent to human actors via e-mail, fax, SMS and so on; or a message to other electronic systems, such as EMR, ClinicalViews, Database and so on, perhaps serving as a trigger to some process (integrated into the EMR workflows, integrated into the provider/patient portal/PHE etc.). In either case, the message typically includes sufficient semantic information and should conform to the relevant standards to make it understandable for the receiving actor (human or application).

2. SmartWatch outcomes as part of the VPO: SmartWatch’s recommendations may remain, in the long run, as part of the VPO, accessible to dbMotion consumers in the same way as any other patient related information. One possibility for implementation might be for any recommendation generated by SmartWatch to be documented in the UMS. A new UMS domain may be built to accommodate the new information, following the HL7 RIM principles. The information may include some of SmartWatch metadata as well. For example, link the action (alert/notification) class linked (many-to-one) to a SmartGuard class linked with creator class and so on.

3. Close the loop capabilities: SmartWatch may raise issues so important to the patient’s well-being that an Email or SMS to the primary physician are not appropriate. Such a message is typically accompanied by some mechanism that verifies that proper actions have been taken, or at least that the issue was dismissed by a proper authority. SmartWatch keeps track of the events it sends, and provides a service that allows the receiving application to notify SmartWatch that the issue has been addressed (loop is closed). This service may also be able to supply a list of non-closed issues for a consumer. The details of when to remind, how escalation is handled and so on, can be part of the SmartGuard properties, although actual interaction with the user is made by the receiving application.

“Remind Me” mode: In cases where the action channels include acknowledgment from a human user, a desired ability is for the user to ask to be reminded at a later time. This is helpful when the SmartWatch outcomes encourage a care provider to do something that he/she prefers to postpone. In this case, a ‘remind me’ service can be very helpful. SmartWatch may provide a service that enables the receiving application to utilize such a “remind me” functionality.

3. Escalation workflow: An optional escalation process may be provided within the actions workflow to make sure issues raised by SmartWatch are addressed, by directing the message to another recipient after a predefined timeframe in which no response is received. Such a process may verify that the previous message was acknowledged and action has been taken in a timely manner. A system-defined need for such a step is introduced from communication reliability limitation (it is possible to ensure a fax is sent, but how can one verify that it was read?) and from physician availability (sent a message to a physician’s EMR but she is on vacation)—for example, if the physician has not taken any action on an urgent case, the case may be escalated to her manager 24 hours later. SmartWatch may provide a service that may enable the receiving application to utilize such functionality.
4. Narrative (free text) simple messages: In cases where an output message from SmartWatch actions is aimed at a human actor, the message may be expressed as a narrative text that describes the recommendation. For example “Patient belongs to XXXX population. It is highly recommended to perform the following blood tests . . .”

5. Narrative (free text) complex messages: When an output message from SmartWatch actions is aimed at a human actor, the message may be expressed as a narrative text that describes the recommendation and the logic behind it, and uses dynamic data to formulate the message. For example “Patient belongs to XXXX population. Based on <yy parameter> and <xx parameter> it is highly recommended to perform the following blood tests: <recommendation ZZZ>. . .”

6. EMR Empowerment: Most typically, the preferred way for care providers to consume SmartWatch outcome is within the application they use from day-to-day—mostly the EMR they use for standard activities (workflow, CPOE, etc.). This is why interaction with the EMR, allowing SmartWatch to use the EMR to get the message across, is useful.

7. Structured messages: When an output message from SmartWatch actions is intended for another system, the message may be formatted in a structured format, so as to enable other systems to consume it. The message may contain semantics as appropriate, such that the information delivered to the consumer is understandable by the SmartGuard definition.

8. Use Healthcare IT standards as SmartWatch output format: Wherever possible, in interacting with other systems, adhere to/comply with relevant recognized healthcare IT standards, such as CCD, CDA, HL7 V2/3 etc.

9. Built in mechanism to standard delivery channels: The out-of-the-box offering may include the ability to send secured messages to standard devices and protocols such as SMS, email, fax. A lower priority channel, which may be useful for communication with patients, can be paper-based mail.

10. Out-of-the-box actions workflow: SmartWatch may come with a set of out-of-the-box action solutions, and with ‘application blocks’, SmartWatch may allow the configuration/customization of the out-of-box action solutions to meet customer needs.

11. Provider identity resolving/PCP service: When SmartWatch sends a message to a patient’s care provider (in most cases the PCP), it is typically operative to correctly identify the relation between the patient and his provider. This PCP service is envisioned as part of dbMotion Core product, but it is crucial to SmartWatch.

12. Action workflows and messages which are context-sensitive: Message format (text, color) and delivery method may depend on the context in which the message is sent, including, for example, the recipient role, urgency, time of day, and delivery method. The goal is to get to the consumers, mostly care providers, in a way most convenient to them, non-disruptive to their daily tasks. For example, when sending the output of a SmartGuard (e.g. alert/notification) as SMS and/or email, the choices may be smart and have some logic/workflow rule based capabilities (e.g. not sending an SMS to physician X on Saturdays and Sundays between 09:00 to 17:00). SmartWatch therefore may have some basic business rule capabilities for this non-clinical logic (clinical knowledge is managed by Knowledge Framework).

13. Paper/Fax messages: Some recipients of messages and reports may not be connected to the dbMotion network or only be loosely connected. Such a mode may only permit simple, one-way communication by fax or paper-based reports sent by mail. For example, transfer of care use cases introduces nursing homes as an actor that is a) not affiliated with the dbMotion network; b) not computerized in almost any way. A fax may be a good means of communication for such cases. Note may be taken as to how to obtain these contact details (e.g. recipient registry or Subscription) and how to tie them to the patient records. Another example relates to agencies to which reports may be delivered. An alternative simpler scenario could be when the paper report is electronically transmitted to a person, who is then responsible to fax/mail it to the target facility.

14. SmartReports capabilities: SmartWatch is typically able to generate a report with the SmartGuard result/outcome. This may include all relevant information for the consumer/user. This report may be sent (as an option) automatically to a given printer/fax or be saved (as PDF for example). The report may be either patient specific or contain a batch of patients. Batching patients is useful for all the Use Cases that are classified as Reporting Use Cases in many scenarios. Example 1—preparation of a report for a physician of all his patients whose Coumadin-related evaluation points are due this week (with all information which may be used to make the call) allows him to get the job done much more efficiently than referring to them individually. Example 2—report to the authorities of quality performance/measurements.

SmartWatch is typically operative to: 1. enable report generation, using standard reporting tools (e.g. MS Excel). 2. Provide basic/simple out of the box report generation capabilities.

15. Patient de-identification and re-identification: When sending messages to some systems or external agents there is a recognized need for patient de-identification, in order to maintain patient privacy and comply with HIPAA regulations. This de-identification may be performed in such a way that allows SmartWatch to reconstruct the patient identity (re-identify). De-identification omits any identifying patient details (names, addresses, phone numbers and so on), or replaces these with gibberish details. The de-identification procedure is consistent in that it yields the same result when applied again on the same input info. The de-identification method (in which fields may be scrambled/omitted from the message) may be customizable per project. Usage example 1: A SmartGuard for research purposes, either to identify eligible people or to gather data; Usage example 2—P4P measures computation may assume aggregation in some BI tool, where patient identity is irrelevant—however the ability to drill down back to the patient is desirable (assuming user’s security privileges are adequate) which typically requires de-identification. There is great demand for such a service in many other domains of the dbMotion Core product. It may therefore be designed as a service that can be consumed by other layers/modules.

16. Action’s result urgency level: Actions taken and resulting messages may include an urgency level property indicating the urgency with which the message may be treated. An outgoing message should be marked with a proper urgency level indicating the impact of the delivery of that message. SmartWatch findings may be life-saving, but may also be less urgent, and may be treated accordingly. Urgency level may also be a subscription parameter (wants to be notified for very important items) or influence the delivery method (e.g. sometimes must send an SMS in a life-threatening issue). Such
“lookup values” management functionality can be designed to use dbMotion Core Vocabulary Management and/or CTS.

17. Confidence measure: As a decision support tool in a world of different semantics, units, possible missing information, and patient identification issues—a confidence measure may be assigned for each message. Such measures can also serve (similarly to urgency level) in determining the appropriate message delivery.

18. Sending Messages to remote node recipients/subscribers: SmartWatch may support subscribers coming from different nodes. Securing subscription of users not from the local SmartWatch Node is usually a provided functionality; if there are anticipated issues with securing subscribers that the node, managing a SmartGuard, cannot authenticate & authorize.

19. SmartWatch may have a subscription mechanism which allows consumers (e.g., physicians, health plans, external firms & agencies) to subscribe to get notifications from a given SmartGuard. Subscription information may include some filtering logic (a specific patient, all my patients, urgency level conditions, confidence measure threshold) contact information (fax number, email, ... ) as well as preferences (e.g., preferred method of communication, time-of-day/day of week constraints)

A possible business model is a customer paying per notification he receives. To accomplish this, SmartWatch keeps track of how many notifications each recipient/system receives, and stops sending notifications when the contract expires in a manner which may depend on urgency level. Another option (less realistic) is to limit a consumer subscription to a population whose size is limited by the consumer contract. A possible solution can control notifications based on dbMotion core Event Log counters.

An enterprise may define mandatory notifications that a physician must receive, without voluntary registration. At the same time, it may be possible to define notifications that a physician actively subscribes/un-subscribes to. SmartWatch typically supports all 3 of the following options: a: mandatory, b: opt-in and c: opt-out. Note that opt-in/opt-out options rely on having a subscription/un-subscription mechanism.

20. Activate EMR Workflows: In some cases, it might be beneficial for SmartWatch to initiate the ordering of new tests (e.g., laboratory tests). For example when confidence level is low, some new test may clarify the picture. This can be achieved in many ways, including notifying the PCP or integrating into EMR workflows. Optionally, SmartWatch is able to integrate directly with the EMR or Clinical Systems. Alternatively, SmartWatch is able to trigger EMR/Clinical System Workflows.

The importance and difficulty of achieving high adoption rates is known. Some or all of the following requirements A-C may be provided in this context, including software requirements from the infrastructure, as well as guidelines or best practice for the solutions to be built on that infrastructure:

A. Enable feedback from users: In an interaction with users, SmartWatch may allow a user to provide feedback that can be considered for future product refinement. This applies to SmartWatch and Knowledge Framework management tools (SmartGuard editor, Knowledge acquisition tool, control room), as well as to interaction through messages, and interaction integrated into the care-provider workflow (EMR empowerment). Feedback may include user response (free text) as well as internal information (application info, SmartGuard info . . . ). The feedback may be sent to dbMotion and the relevant IT owner. One simple example for such interaction is to provide the user who receives the message an email address to send his feedback. A more advanced solution is a web form to submit his feedback which may also include the context of that specific message. A possible solution could utilize dbMotion Core ADR.

B. Fit into clinician workflow: A decision support tool typically has its recommendations integrated into the clinicians’ application in a way that is not disruptive and is context-sensitive. This is mostly relevant for SmartWatch solutions built on top of the framework—but the framework may facilitate it, for example, by providing the infrastructure for EMR empowerment as described herein.

C. Look for knowledge consensus and knowledge authority: When building a SmartGuard (SmartWatch vertical solution) the question of knowledge consensus may arise. Many clinical scenarios are difficult in this regard and it may be hard to achieve users’ acceptance of decision support recommendations based on knowledge resources they doubt. An independent knowledge authority such as medical associations and societies that publish their best practice care plan may be helpful. SmartWatch and SmartGuard System Management Requirements may include one or more of the following requirements

1. SmartGuard activation: SmartWatch typically has the ability to start and stop SmartGuard independently of one another. When stopping a SmartGuard and restarting it at a later time special caution should be taken in making sure that no input events are lost in the process (all relevant records that were inserted during the time the SmartGuard was down are revisited when the SmartGuard is activated).

2. SmartGuard Expiration Date: One of the SmartGuard attributes may be Expiration Date. If such a date is defined for the SmartGuard, it may stop automatically when the date is reached.

3. Monitoring tools—Auditing, Logging: SmartWatch typically employs a set of monitoring tools and a consistent auditing and logging methodology with tools to assist in tracking activities over time—for example, alerts which the system sent over time, to whom, sees that the loops were properly closed, and so on. This would greatly help in evaluating the ROI of a given SmartGuard SmartWatch and leverage the existing tools provided by the dbMotion product.

4. SmartWatch Control Room: The monitoring and system management tools may be part of the larger concept, the Care Center. Someone sitting in the SW (virtual/physical) control room and in front of one/a few screens is able to manage the whole operation: track log, activate/de-activate SmartGuard, see/correct errors etc. Such an application may track all activities in the different SmartGuard and monitor their operation—track ALL the activities in the SmartWatch network, enable deactivate/activate SmartGuard, audit the various types of errors (and act on them), filter all the active SmartGuard, see the status and all relevant information for each one.

5. Document and manage all changes made to the system after deployment: SmartWatch may follow common practice in the content management field in order to make sure that the SmartGuard metadata and related knowledge always have the most updated information, versioning, and so on. Several examples: approval date, owner (company, person), update dates etc (Content Management). SmartWatch can utilize the existing dbMotion core product mechanisms for management change (mainly the Reference Model).
6. Maintain SmartGuard integrity: SmartWatch relies on a suitable source such as the dbMotion core product as an information source, and is typically made aware of changes in dbMotion. Such changes, low down, could be for unrelated reasons, where the people making a change are unaware of the effect it may have on SmartGuards that hover above and consume the data and the chance that a change may affect some SmartGuard function in the wrong way. The outcome can be a false negative, which may be potentially dangerous. Semantic changes—for example, changing business rules and/or data modeling decisions without changing the structure—are handled with caution. Examples of dbMotion changes that can affect SmartWatch: changes in the UMS, business methods, mapping in terminologies, archetype, KW, lab values (and many more). These changes can easily give rise to a malfunctioning SmartGuard. A possible solution utilizes the dbMotion Reference Model system to maintain SmartGuard integrity. The reference model manages the overall dbMotion system configuration and the relations between the dbMotion components. The importance of the Reference Model in this case goes way beyond the implementation domain (project V's product).

7. Search capabilities in SmartWatch metadata: SmartWatch may provide the ability to query the SmartWatch metadata repository for SmartGuards based on some topic, keyword or clinical term (using some vocabulary). For example, a list of all SmartGuards related to hypertension.

8. SmartGuard feasibility testing: Once a SmartWatch is defined, estimation may be made of population size, verifying that the SmartGuard is feasible. This information may be considered for SmartGuard approval (in each node). For business security, this information is not exposed across nodes.

9. SmartGuard full description: A description of the SmartGuard in free text: —e.g., what is the population?, what information is gathered? and why?—the decision logic process, the actions that are taken. This may be a generated report, but may include meta data text pieces that were captured along the way.

Security & privacy requirements may include some or all of the following requirements 1-12:

1. General:

[0531] Users are authenticated prior to accessing the system. Users are authorized for the operation they perform. Access to any storage system is secured. Secured Communication may be established for ePHI. In any case where patient data is used for research purposes, all patient identifying properties may be removed from it (de-identification). Every significant action in the system that relates to a covered entity (patient) may be recorded.

2. Actions may protect patient privacy. Since SmartWatch has the potential to distribute patients’ data outside the dbMotion world and to non-authorized users, it may provide means to protect patient confidentiality with regard to information that is included in notifications and the actors who receive them. SmartWatch should have authorizing tools that verify who receives patient-related notifications, and a security methodology for SmartGuard development, configuration and deployment. A methodology is useful to prevent overlooking patient privacy. The authorizing tools are employed to facilitate verification that outgoing messages are authorized. The SmartGuard may always send the minimal data set that is relevant to the context of the specific SmartGuard.

3. Subscription security authorization: SmartWatch enforces authorization for subscribing to SmartWatch actions to get messages, to avoid unauthorized access to confidential patient information as a result of notification.

4. SmartWatch management tools security and auditing: SmartWatch controls access to its management tools and audit user activities for several reasons: a) sensitive information may be accessible to users of these tools, for example through auditing messages sent b) changing security-related settings can cause HIPAA violation (for example if patient consent policy setting is altered) and c) changing SmartWatch settings definitions may jeopardize the proper functioning of the system. As to the question of who may stop and start SmartGuards, this is typically done only by an authorized administrator and is audited. Deactivating a SmartGuard can have severe implications if clinicians rely on its actions. SmartWatch may therefore provide and manage secure access (authentication and authorization) to SmartWatch management tools.

5. Access data across the enterprise (using dbMotion) in a secure manner: SmartWatch is typically operative to access data across the enterprise (using dbMotion) in a secure manner, utilizing a secure channel, authentication and authorization—for example, a need to prevent SmartWatch from accessing unauthorized data domains.

6. Enterprise business security: SmartWatch introduces an amplification of all the barriers found in the dbMotion core with regard to ownership. While the "per-patient and only for-patient-at-the-point-of-care" argument usually works in the patient centric viewer model of dbMotion core, this will not always be the case with SmartWatch. The customers concerned (hospitals) may have doubt about querying their CDRs/operational systems in a "batch-mode" (in contrast to treatment-based context implied when a clinician views patient ePHI) which may be mitigated by a federated model and a mechanism that guarantees to some degree that the data is used for the purposes agreed upon only. This concern is greater in a RHIO setting where SmartWatch has the potential of revealing too much business-sensitive information between hospitals/organizations that may be business competitors. SmartWatch may therefore enable the customer to manage and control access to his data repositories.

7. Securing SmartWatch storage: The SmartWatch process may involve short-term or long-term storage of ePHI. The storage devices are secured. Furthermore, customers may require enforcing regulations that prevent duplication or centralized storage of identifiable ePHIs.

8. STRQ43.9 Auditing: SmartWatch may enable auditing of all operations that relate in any way to accessing patient information (ePHI). An audit entry may include details of the user, the time the operation took place, and the ability to generate audit reports—with the option to audit any changes in SmartGuard settings, any operation in SmartGuard or SW management, and any message that was sent. In addition, audit of SmartWatch Actions may include ALL the information about a recommendation/notification/alert, including its content and narrative explanation (mainly for medico-legal reasons).

9. STRQ43.10 Notifications over unsecured delivery channels: SmartWatch may be designed to deliver messages to both secured systems and to people over potentially unsecured channels such as Email, Fax or SMS. Provisions are typically made both for protecting patient privacy in the message content and for technologically securing the channel.
For example, Email could be signed and encrypted, an SMS message may include partial, non-confidential information, directing the user to view the full notification in a secured system, to which the message was sent in parallel. The recommended practice is using secure channels only.

10. Patient consent to be incorporated into the SmartWatch process: SmartWatch may find a way to take into account patient consent considerations when deciding on including a patient in a SmartGuard. Policies can be established at enterprise or SmartGuard level that may include (but are not limited to) opt-in, opt-out, no consent required. SmartWatch may utilize the dbMotion core PAS.

11. SmartGuard can use ePHI from using ePHI retrieved under a different role/contract, or ePHI retrieved under different authorization settings (i.e. role/contract): SmartGuard may use information it was not authorized to view. In other words, data sharing between SmartGuards can put patients’ privacy at risk unless they share the same authorization level (for example, the same role/contract which means they have exactly the same permission).

12. Use existing dbMotion Core Security System and Auditing System: SmartWatch may use dbMotion Product Security and Auditing, as much as possible, for meeting security and privacy needs. This may reduce costs in the SmartWatch Product Development, and simplify the work of the customer security admin—using the same tools and settings both for dbMotion and SmartWatch. This means that SmartWatch may not have its own Security Layer, but rather utilize dbMotion core security. Since there are many cases where there is no cross-enterprise unique patient identifier, patient identity is not deterministic. This may impose some or all of the following constraints 1-3 on SmartWatch:

1. SmartWatch cannot assume patient clustering is stable. When using records collected at different points in time, SmartWatch cannot assume that patient identity has not changed, i.e. that the patient’s cluster as determined by the EMPI system is the same as before. Special note is taken for the join/add operation (when a person’s identity is added to the EMPI cluster), which is hard to spot. Therefore, SmartWatch typically employs cluster consistency procedures to verify that the patient cluster has not changed along the way.

2. Patient enrollment scenarios:
   When a physician selects a patient cluster/index retrieved from the EMPI system, the selection may be stored in SmartWatch. Proposal: save a leading index whose cluster may be used as the virtual patient.

3. SmartWatch may be able to work in an environment that has a unique patient identifier for all systems, without EMPI.

   This applies to some European countries and Israel.

[0532] Other design considerations may include the following considerations 1-7:

[0533] 1. Any audit entry may include the relevant unique SmartGuard identifier as well as the date and time. All the SW’s auditing and tracking may utilize the dbMotion core STD & ADR (CEB).

[0534] 2. Population size limits constraint. Since population primary criteria are a matter of configuration, it is potentially possible to define very large populations that include all patients (i.e. age >40). Such a population might have a significant effect on the system performance—both dbMotion and SmartWatch, and practically hang the system while collecting data for it. For this reason, constraints may be set on either population size (data may not be collected for a population with over 10k members), or on the rate of collecting the data (allow collecting data for 1k members per day). Similar constraints may be set to interaction with external systems, especially EMPI providers. Another solution could be a testing methodology and tools to identify such large populations.

[0535] 3. Knowledge-level adjustability: The clinical knowledge SmartWatch uses ought to be adjustable at the customer site by knowledge engineers with a clinical background, using a knowledge acquisition tool/archetype designer, e.g. as described herein with reference to Figs. 4A-8C.

[0536] 4. System-integrator-level adjustability: SmartWatch may allow much space for the Project Implementation Team (system integrator level) customization & configuration with regard to the following aspects:

[0537] 5. It is envisioned that much customization may be carried out in the action layer, since the external systems and preferred communication methods may vary greatly between customers. The Action Manager is therefore typically flexible and adaptive and offers ready-made communication tools, and best practices with regard to actions’ workflow SDK’s. The Action Manager may be generally prepared for customization by the Project Implementation team.

[0538] 6. The ability to create and modify SmartGuards may be available at least at the system integrator level. Preferably, most activities should be performed by the customer’s administrator level. The functionality includes a set of tools that allows the system integrator to configure the criteria defining the patients to collect data for, the data to collect, calculations on that data, the rules applied on them, actions to be taken when patient data matches the rules’ logic, and a ‘close the loop’ workflow.

Some guidelines that may help in this regard are the use of templates; allowing their creation based on an existing SmartGuard (save as, e.g.); following the 4 layer path which makes sense to clinician; SDK and Application Blocks.

[0539] 7. Taking into account the fact that the infrastructure cannot be developed to support any future need ahead of time, all SmartWatch components may be extensible. Some examples are the ability to communicate to proprietary delivery channels, a need to monitor triggering events that were not supported in the original design and supporting implementation with no CDR. Testability requirements may include some or all of the following requirements 1-5:

[0540] 1. SmartGuard test mode: SmartWatch may provide a testing mode to facilitate the construction of new SmartGuards or changes to existing ones. This may be beneficial both in the development environment as well as on the customer site. When working in such a mode, the SmartGuard may be limited in some ways, for example in its outside communication capabilities. The test mode may be applied at the SmartGuard level, allowing a tested SmartGuard to run side-by-side with production SmartGuards.

[0541] 2. Enable Automatic Testing thru SOA: Some GUI capabilities may also be provided through services, in order to allow these server-side functionalities to be tested using automatic testing implemented by program tools.
3. Test-dedicated tracking events (STL) to promote testing processes: SmartWatch may add entries to STL for test purposes, at each major step along the way, to allow testers to track process progress. Preferably, STL may be triggered on entry and exit of each major function, stating status and duration. However, logging level configuration may be enabled (so in production there may be less system events than in debug mode.). A similar methodology to the system events tracking in the dbMotion product may be used.

4. Log messages verbose mode: SmartWatch may supply a verbose mode in which state variables, input parameters, intermediate computation results and so on can be written to log files. This may help in testing and debugging (during the development phase) and may also be helpful in analyzing problems at an operational site.

5. Allow offline testing using test data emulator: Because during SmartGuard development, a dbMotion network and other components (DSS, EMP) might not be available, and since it may be hard to provide meaningful data for a variety of SmartGuards, it may be possible to replace dbMotion, VIA, and KFW with test data simulators—preferably existing dbMotion Product simulators.

An example of health information exchange and integration system constructed and operative in accordance with certain embodiments of the present invention is now described with reference to FIGS. 18A-137 which may operate in conjunction with healthcare information integration software commercially available from dbMotion Inc., US Steel Tower, 600 Grant Street, Suite 22017, Pittsburgh, Pa. 15219, such as dbMotion’s service oriented architecture (SOA) based dbMotion™ Solution.

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering used in generating the above drawings. The standard is managed, and was created by, the Object Management Group. UML includes a set of graphic notation techniques to create visual models of systems with software components.

The CTS sub-unit of the example health information exchange and integration system is first described with reference to FIGS. 18A-1051. Specifically, FIGS. 18A-18B, taken together, illustrate a Use Case Model—(Use Case diagram). FIG. 19 illustrates an Actor View—(Use Case diagram). FIG. 20 illustrates a CTS Runtime Use Case View—(Use Case diagram). FIG. 21 illustrates CTS Metadata Use Cases—(Use Case diagram). FIG. 22 illustrates a Get Service Address functionality—(Use Case diagram). FIG. 23 illustrates a Manage Extension functionality—(Use Case diagram). FIG. 24 illustrates a Manage Named Query Metadata functionality—(Use Case diagram). FIG. 25 illustrates CTS Extension Use Cases—(Use Case diagram). FIG. 26 illustrates a Fill Concept Information in Vocabulary Business Aspect—(Activity diagram). FIG. 27 illustrates CTS Extended Solution Use Cases—(Use Case diagram). FIG. 28 illustrates CTS First Analysis Use Cases—(Package diagram). FIG. 29 illustrates a CTS Management Use Case View—(Use Case diagram).


FIG. 100 illustrates a Common functionality—(Logical diagram). FIG. 101 illustrates Service Contracts—(Logical diagram). FIG. 102 illustrates a Data Model—
(Logical diagram). FIG. 103 illustrates an Implementation Model—(Component diagram). FIG. 104 illustrates a CTS (Logical diagram). FIG. 105A illustrates a management functionality—(Logical diagram). FIG. 105B illustrates a parameter—(Logical diagram). FIG. 105C illustrates an exception—(Logical diagram). FIG. 105D illustrates a model—(Logical diagram).

[0553] In FIG. 18A, the CTS query use case Responsibility may include that it provides information about concepts in dbMotion ontology including Concept details and/or Search mechanism. The CTS metadata service use case Responsibility may include that it provides metadata information about CTS service for consumer-services for integration in the design and runtime. An Extension framework may provide the following Use Cases: Creation and developing of CTS extensions; and/or Internal product simple basic query extensions.

[0554] In FIG. 27, the first determining step may include some or all of the following operations:

[0555] 1. For local ValueSet determine baseline concepts
[0556] 2. For baseline concepts determine parent baseline concepts in the specific context
[0557] 3. For parent baseline concepts determine all local concepts.

[0558] The second determining step may include some or all of the following operations:

[0559] 1. For local ValueSet determine baseline concepts
[0560] 2. For baseline concepts determine child baseline concepts in the specific context
[0561] 3. For previous resulting baseline concepts determine local concepts.

[0562] In FIG. 32, the Design Model may include at least the Logical Model and the Data Model. The Logical model included the System and Framework sections. Both may include classes and artifacts which define the structure of the code used in the application under development.

[0563] In FIG. 33, the Logical Model is a model of the software system under construction. It may include classes which generally have a direct relationship to source code or other software artifacts that can be grouped together into executable components. The system package contains the classes and artifacts which are being built or designed as part of the current model. The Frameworks package generally contains classes and components that have been designed and built earlier and are being reused as part of the current project.

[0564] In FIG. 92, current cache-optimization may include the following steps:

[0565] 1. For every bulk parameter, attempt to find cached object
[0566] 2. In case it exists—put to QueryDataSet and remove bulk from existing ParametersSet.

[0567] In analysis of results, it may cache objects in the cache by bulk hash-code and endpointname. In FIG. 98D, an Analyser may be imperative for parsing and logical analyzing of query text. Factory query object may work through Analyser.

[0568] In FIG. 102, typically a schema package contains a logical grouping of tablets. This model describes the data which must be stored and retrieved as part of the overall system design. Typically this will mean relational database models which describe the tables and data in detail and allow generation of DDL scripts to create and setup databases.

[0569] In FIG. 103, typically, the implementation Model defines how classes, artifacts and other low level elements are collected into high level components and the interfaces and connections between them. Components are compiled software artifacts that work together to provide the required behavior within the operating constraints defined in the requirements model. Components may be deployed to varying hardware platforms e.g. as described in the Deployment Model. The Components package contains modeled components and their structural constituents. These include additional exposed interfaces, ports and other gateways or internal structural components. The connectivity and internal structure of these are further modeled in the internal Structures and Connections packages.

[0570] Internal structures provide a detailed view of possible internal workings and dependencies of a component. Using a Composite Structure diagram, they illustrate how the component fulfills its behavioral contracts and provides interface behavior to other components within the system. The Connections package models the dependencies and connectivity between the various components, and how each is used as part of a co-operative system to accomplish required tasks. Typically, Components expose interfaces and API’s which are used by other Components.

[0571] The knowledge framework sub-unit of the example health information exchange and integration system is described, referring back to FIGS. 9-13C. The illustrated embodiment is an example implementation of the KFW sub-unit of FIG. 1A which is not intended to be limiting.

[0572] Specifically, the implementation’s CTS Runtime Interfaces are described with reference to FIG. 9. A Knowledge-Framework and Knowledge-Module Static Model are described, including business identifiers described with reference to FIG. 10A, an artifact described with reference to FIG. 10B, and a knowledge-Module Entities Model described with reference to FIGS. 10C-10F. A KFW Business-Object-Model (BOM) is described with reference to FIGS. 11A and 11B. A KFW Evaluation Interface is described generally with reference to FIG. 12A. An Evaluation Request for the KFW Evaluation Interface of FIG. 12A is described with reference to FIGS. 12B-12C. An Evaluation Response for the KFW Evaluation Interface of FIG. 12A is described with reference to FIGS. 12D-12E: A KFW-Evaluation Process described with reference to FIG. 13A. An Evaluation Plug-in Mechanism is described with reference to FIGS. 13B-13C.

[0573] Generally, in the example implementation, Knowledge-Framework (KFW) is a decision-support service. KFW manages and evaluates Knowledge-Module (KM) entities. A KM contains a piece of clinical business-logic. A KM has a well defined contract: a set of specified input data and a set of results. KMs are modular and can be re-used, i.e., the output of one KM, can be used as input for another one. The body of a KM is its decision-logic. The decision-logic is based on input data and set the KM evaluation-results: The decision-logic is created and managed by knowledge-experts (e.g., clinicians). The decision-logic may use CTS ontology concepts. These concepts are defined implicitly as queries (commands) over dbMotion’s CTS ontology only.

[0574] At runtime, KFW consumers send requests to evaluate actual patient data. The raw patient data is stored in dbMotion database using its local terminology. Before the data (e.g., a lab result) reach KFW service, the data is pre-processed and enriched with a mapping to dbMotion’s CTS
ontology. As a computation of the requested KM starts, KFW asks CTS to evaluate the pre-defined CTS queries. CTS returns a list of CTS ontology concepts. During the KM's decision-logic evaluation, KFW may check whether the patient actual data corresponds to one of the concepts in the result of the expected query.

Example CTS Runtime Interfaces are now described with reference to FIG. 9. KFW defines the following interfaces and allowed operations on CTS objects:

- **a. CTSServerRuntime**: a marker interface.
- **b. CTSConceptRT**: represents a single CTS concept identifier. sameConcept(CTSConcepRT other): returns true if the other concept is equivalent to this concept.
- **c. CTSCollectionRT**: a list of CTS concepts. Contains(CTSConcepRT item): returns true if this list contains the given item.
- **d. KnowledgeFramework and Knowledge-Module Static Model is now described. Business Identifiers**: FW identifies its entities using identifiers, e.g., two types of identifiers, (a) and (b), as shown in FIG. 10A:
  - **a. EntityId**: identifies main entities and may include:
    - **Scoping/EntityId**: a scope of ids (similar to namespace)
    - **BusinessId**: business id of the entity within the scoping entity
  - **b. Version**: version of the entity b. **ItemIdentifier**: for entities that are composed in main entities. May include:
    - **Version**: version of the entity
    - **Containing/EntityId**: the id of the containing entity
    - **ItemId**: unique id of this item within the containing entity
- **Artifact**, e.g., as shown in FIG. 10B, is used to store not-structured data or data that KFW is agnostic to its structure.

An example Knowledge-Module Entities Model is illustrated in the class KM diagram formed by FIGS. 10D. Attributes, KFW internally operates on 'Entities'. KFW-Entities is an object-model mapped to a relational database. The main entity in KFW is the 'KnowledgeEntities' (KM). A KM extends from 'Scoping/Entity' and hence is identified by an 'EntityIdentifier' (see above). A KM may include some or all of the following members:

- **DecisionLogic**: determines the type of the decision-logic used in this KM. These are possible values: 'decision-table', 'rules', 'java code' etc.
- **Status**: determines the status of the KM. Possible values are: 'draft', 'in_production', 'obsolete' etc.
- **DataRequirements**: a set of DataRequirementItems (DRIs) employed in a particular application, when evaluating a KM. A DRI is a KMItem, and hence is identified by an 'ItemIdentifier' (see above). Each DRI specifies one named data-input of the KM, e.g., 'last year encounters'. The DRI is specified in one or more alternate DataModels. Each model format of the expected DRI data is defined by referencing a particular XML type in an XML schema (e.g., 'encounters' XML type).

In addition, the DataModel can also constrain the returned data by specifying a query (the internal structure of the query is not in the scope of KFW). Typically, a Data model may be defined in one of dbMotion's schemas, using dbMotion's query language.

- **AdditionalDataRequirements**: an optional set of DRIs that may be required. For example, if certain patient data is rarely needed by the KM and is expensive to collect, it may be defined as a DRI in AdditionalDataRequirement. Example: an initial call to evaluate a KM may contain only InitialDataRequirements (those that are easy to collect). Only if KFW failed to reach a conclusion using this data, the KFW response may contain a requirement for these AdditionalDataRequirements.

- **EvaluationResults**: a set of evaluation-results. Each evaluation result declares the format of the returned data. The set is the output signature of a KM. For example, if a KM returns a yes/no answer it may have one evaluation-result of type Boolean. An evaluation-result can be computed in this KM, or propagated from the evaluation of another dependent KM.

**DecisionLogicItem**: a set of abstract source artifacts that defines the KM decision-logic. For example, it can be an Excel file that defines a decision-table. Or it can be a Java program that parses the input data and computes an evaluation-result. The actual type of the decision-logic artifacts depends on the value of DecisionLogicType attribute on the KM level.

- **CTSItem**: a set of abstract items specifying one or more CTS ontology concepts. As shown in FIG. 10E. CTSExtensionItem uses an Artifact e.g., as described above to specify a CTS command.

**CTSCommand** is the main object to query CTS, e.g., as illustrated in FIG. 10F. In FIG. 10F, typically, the command state is created at design-time by browsing CTS metadata service and selecting (or creating) a command. The command is stored in an artifact. When calling contains (CTSConcepRT item) method, the CTSItem calls CTS Query Service passing the command. CTS returns a flat list of CTS ontology concepts that satisfy the command. Finally, the method checks if the given item is included in this result list.

**CTSExtensionItem** typically comprises a hard-coded list of CTS ontology concepts and implements the contains() operation. CTSExtensionItemSingleItem typically comprises a hard-coded single CTS ontology concept and implements the sameConcept() operation.

A suitable KFW Business-Object-Model (BOM) is now described. As described above, at runtime KFW evaluates patient data. The data reaches KFW service as XML documents (in dbMotion's format) and is transformed internally into (Java) objects. The model of this data is called the 'Business-Object-Model', or BOM in short. For example, Condition is a class in the BOM. The BOM may refer to CTS concepts. For example, Condition.Code is coded using CS class. CS class represents a CTS concept. At its root level it contains code and codeSystem attributes to identify a concept in a local terminology. The CDTDevelopments composition contains translations of this concept to other terminologies. In particular, this collection contains the translation to dbMotion's CTS ontology.

**CS class also implements the CTSCollectionRT interface. Hence, operations sameConcept() and contains() that were defined on KM CTSItem can be applied on BOM attribute values. The sameConcept() method in CS class is implemented by comparing the other.code and codeSystem attributes to CS root level attributes, or by finding an element
in CDTTranslators that satisfies sameConcept(other). FIG. 11 is a diagram of an example class condition.

[0599] An example KFW Evaluation Interface is now described. Typically, KFW exposes several interfaces. The evaluation interface of FIG. 12A is an example of a suitable interaction between KFW service and CTS. The interface of FIG. 12A defines five evaluation methods. The flow in all methods may be similar, so for simplicity only the flow of the first method—evaluate()—is described hereinbelow. The method receives an EvaluationRequest and returns an EvaluationResponse. An example EvaluationRequest is now described with reference to FIG. 12B. In FIG. 12B, KM evaluationRequests includes a list of KM ids to evaluate in this request. DRIIData includes a list of patient data to use in the evaluation. Each DRII in the request object refers to DRII of an existing KM (using DRIIID). The data itself is enveloped in a SemanticPayload which includes a payload of data in xml element format. The infomationModelSSId specifies the actual format of the data.

[0600] FIG. 12C is an example of an EvaluationRequest in xml format. This is a request to evaluate a single KM, with a single DataRequestItem(Data (DRII)Data) of type ‘Conditions’ which contains (in this example) a single ‘Condition’. Condition.value is a CS element that carries a local (i.e., from legacy system) concept code. In addition, CDTranslator element specifies a mapping to a dbMotion’s ontology concept. This translated code may correspond to the CTS codes expected by the KM.CTSItems.

[0601] An example EvaluationResponse is now described with reference to FIG. 12D. In FIG. 12B:

[0602] FinalKMEvaluationResponses: a collection of responses for KMs that KFW service has successfully reached a final result.

[0603] KnowledgeModuleId: the id of the KM that this response refers to.

[0604] EvaluationResultsData: a collection of evaluation result data. One item per an EvaluationResultItem in the referenced Km.

[0605] EvaluationResultId: the id of the referenced EvaluationResultItem.

[0606] SemanticPayload: the data itself, in the format declared by the EvaluationResultItem.

[0607] IntermediateKMEvaluationResponses: a collection of responses for KMs that KFW service could not reach a final result, probably because of missing data.

[0608] RequiredDRIID: a collection of KM DataRequirementItem Ids that are missing for reaching a final result of this KM.

[0609] FIG. 12E illustrates an example of a KM evaluation result.

[0610] An example of a KFW-Evaluation Process is now described. As described above, one runtime use-case of the KFW service is ‘evaluate’. The evaluate method receives an EvaluationRequest and returns an EvaluationResponse (both were described above). FIG. 13A is a top level simplified flow diagram of an example evaluation process. EvaluationLongBean is the class that implements the Evaluation interface. Typically:

[0611] 1. The consumer sends EvaluationRequest messages to the bean.

[0612] 2. The request contains pointers to the KMs to evaluate. Hence, the first step is to retrieve the KM entities from the database.

[0613] 3. A DataContext is created. The DataContext initially contains all the DRIIData provided by the consumer. During the process, computed evaluation-result data are accumulated on this DataContext.

[0614] 4. An EvaluationPlan is created. The problem is that an EvaluationRequest may comprise one or more KMs that can have inter-dependencies. KFW has to determine a sequence of KMs to evaluate. A plan serves as a strategy of the KM evaluation sequence.

[0615] For example, one may assume this knowledge base: [a, b, c]->[a, b] (i.e., km c depends on the outputs of kms a and b.), then on runtime KFW receives a request to evaluate km c. One option is to evaluate the KMs in this sequence: [a, b, c]. Another strategy might be [a, b, c] (i.e., to compute a and b in parallel and to compute c).

[0616] 5. The plan is executed with the dataContext. During the plan execution evaluationResult data are accumulated in the DataContext.

[0617] 6. An EvaluationResponse is assembled by looking at which KMs were originally requested and the actual data is collected from the DataContext.

[0618] An Evaluation Plug-in Mechanism is now described which is useful in the process of evaluating a single KM. As described herein with reference to the KM static model, KM decision-logic is stored in artifacts. The KFW supports a mechanism to plug-in new decision-logic types. For example, the decision-logic can be declared in rules, or using a decision-table. The plug-in mechanism that evaluates a particular KM is determined by its KMDecisionLogicType attribute value. Hence, KFW holds a map where the keys are decisionLogicTypes and the values are DecisionLogicPluginProvider objects. For performance reasons, the KM decision-logic artifacts are compiled into a plug-in proprietary form. This compiled form is also stored in the KM static model in a special artifact. FIG. 13B is a diagram of an example class manager. In the embodiment of FIG. 13B:

[0619] DecisionLogicPluginProvider: a factory object that provides appropriate compilation and evaluation plug-ins.

[0620] getCompiler( ) returns a DecisionLogicCompiler object.

[0621] newEvaluator( ) receives a KMSignature and a compiled artifact and returns a new Evaluator.

[0622] DecisionLogicCompiler: an object that knows to compile source artifacts in a specific decision-logic format.

[0623] Compiled( ): returns a new artifact that contains the compiled form of the source artifacts.

[0624] getErrors( ): returns a list of errors in the source artifacts.

[0625] Evaluator: an object that evaluates requests and returns responses.

[0626] A typical KFW-plug-in evaluation process is illustrated in FIG. 13C in which an AtomicEvaluationPlan is the implementation class of EvaluationPlan interface for evaluating a single KM. A KM entity is passed in the constructor call. If the KM does not contain a compiled artifact, the sources have to be compiled. In practice, a compiled artifact is usually already set in production systems (it may be null only during the KM authoring stage). EvaluationPluginManagerImpl is a singleton object that contains a map of decisionLogicType strings to DecisionLogicPluginProvider objects. This map is initialized from configuration. The plan
as the provider to get a compiler object. The compiler gets the KM entity and returns a compiled artifact. This compiled artifact is stored.

When the plan is executed, the plan asks the provider to provide an Evaluator object. The Evaluator object is constructed from KM entity (expecting a not null compiled artifact). The Evaluator.evaluate() method is called passing the deData (extracted from the request). The evaluator is state-less and the same evaluator can be used several times to evaluate different requests of the same KM.

The SmartWatch sub-unit of the example health information exchange and integration system of FIG. 1A is now described with reference to FIGS. 106A-137. FIG. 106A is a simplified functional block diagram illustration of the SmartWatch sub-unit constructed and operable to certain embodiments of the present invention. FIG. 106B is a table describing operations, some or all of which may be performed by the system of FIG. 106A.

As shown in FIG. 106A, the SmartWatch sub-unit of FIG. 1A typically includes some or all of a SmartWatch-service unit, a data event monitor, a person identity service, a temporal monitor and an action manager. These functional units are now described in detail herein, with reference to FIGS. 106C-114, 115-123, 124-130, 131-134, 135-137 respectively. Various architectural views are provided, such as one or more, as appropriate, of the following UML-defined view: use case view, a logical view, a process view, a deployment view, an implementation view and a data view, where:

1. Use-Case Model: Describes the actors and use cases for a system which may be software-implemented
2. Design Model: Describes the design elements of the system, how they cooperate to realize system use-cases; how they trace to requirements if any; user experience and key analysis elements; and how they map into implementation elements
3. Implementation Models: Describe the implementation elements of the system
4. Deployment Model: Describes the deployment structure of the system.

The architecture of the SmartWatch service unit is typically designed to be a focal point of lifetime management/ storage of all Guard instances and to concurrently process multiple Guard evaluation tasks for an assortment of member task combinations, all as described in detail below.

Use-Case View: Certain SmartWatchService use cases illustrating functions of the service based upon its population processing activities, some or all of which functions may be provided in any given embodiment, are now described. These use cases, referred to herein as use cases (i)-(x), and their associated actors are shown in the SmartWatch-wide high level use case diagram of FIG. 138 and in the use-case focused diagram of the SmartWatch-service of FIG. 139. The following use-cases employ multiple applications of the SmartWatch system, which may include SmartWatch Management GUI, Person Identity Service, and other components of the total system.

Develop and Deploy Smartguard (Also Termed Herein SG) Adapters

ii. Define & Configure Guard

Register and validate Guard configuration
Check-out Guard configuration
Supplying metadata for SmartWatch Management GUI

iii. Configure SmartWatch System

iv. Manage Guard Runtime

A Guard administrator may be able to activate and deactivate a Guard, in addition to viewing the Guards current status (i.e. Active or Inactive). Activities included in this use-case would be: Activating and Deactivating a Guard and Viewing Guard current execution status and log.

v. Monitor SmartWatch System Health

Activities included in this use-case would be maintaining and reporting Guard runtime performance count.

vi. Apply Guard Changes

In this use case SmartWatch reconciles changes to existing population members, such as splits/joins as defined in conventional EMPI systems, which may affect previous decisions (activity executions). Activities included in this use case would be, given a new version of a Guard, reconciling changes to recognized members.

vii. Discover New Population Element

SmartWatch looks for candidates to be added to the population and joins them into the population if they fit. Smart Guard definition determines what may be required for a patient to fit the population; the decision could be based on eligibility criteria from KFW, an external db view or some other source. The following activities are typically included in this use case:

- a. Retrieving Candidates from the Data Event Monitor (or other population source)
- b. Retrieving patient indexes using a conventional EMPI system and using identity monitor
- c. Filtering for duplicate indexes on fired subscriptions
- d. Verifying candidates are not already a part of the population
- e. Pushing a member into a default classification

viii. Process Known Population Element

SmartWatchService monitors its current population members by determining if they need to be evaluated, evaluating them, and responding to an evaluation result. Activities included in this use case may be: Retrieving applicable members from monitors (based upon triggering rules which have fired); and Filtering for duplicate members.

ix. Process General Population Element

This represents a primary processing use case for this service. The SmartWatchService evaluates a patient within the context of a specific Guard. The goal includes determining classification eligibility (e.g. entry criteria). Activities included in this use case may include some or all of:

- a. Retrieving (aggregating) rules to evaluate based upon the patient and classification(s) and evaluating them
- b. Stepping a member into a classification and accumulating monitoring activities
- c. Stepping a member out of a classification and removing associated monitoring activities
- d. Updating triggering rule subscriptions
- e. Handling monitoring activities (e.g. Action Invocation Request, Triggering Rules Subscriptions)
- f. Remove a patient from the population
x. Activate Task

[0658] This use case encompasses SmartWatchService’s activation needs based upon a schedule; however according to an alternative embodiment, triggering is by the event monitor when new results are waiting. Still another alternative embodiment includes activating from an external application (GUI tool or command line). Activities included in this use case would be:

[0659] a. Wake-up according to flow
[0660] b. Find the appropriate task
[0661] c. Begin processing member or population discovery

[0662] Architectural Design Elements are now described (logical view), e.g. as shown in overview in FIG. 106C. Typically, two layers, a business logic layer and a data access layer, share a common set of entities.

[0663] The business logic layer (BL) of the SmartWatchService contains a group of managers that perform at least certain functionalities of SmartWatchService. The MonitorBrokerManager orchestrates requests from the other managers within the business logic layer to the external monitors for triggering rule matching and subscribing services. The ServiceBrokerManager orchestrates requests to other external services. The CallbackManager serves as a point for external services to send information to the SmartWatchService. Services for the business logic layer may include some or all of:

[0664] a. GuardExecutionManager—The guard execution manager coordinates both discovery of a new member for a guard, and retrieving triggered rules for monitoring a guard’s population.
b. GuardManager—Manages all of the Guards and their runtime/lifecycle information. This manager can activate or deactivate a guard, or set it to another intermediate runtime status. This manager also performs lifetime management operations on an individual Guard, such as loading, activating, and deleting Guard instances.
c. GuardMemberManager—The guard member manager performs operations on individual guard members. It coordinates member classification changes, in addition to registering and activating subscriptions for member monitoring through the MonitorBrokerManager.d.

[0665] MemberProcessingManager—The processing manager receives items which contain both a member and a list of tasks that are to be processed for them. Each task contains a list of rules which are to be evaluated. The MemberProcessingManager communicates with a RuleEvaluator to complete this, and returns a set of activities to execute for each member. An example of an activity could be to change a member’s classification, or the send an action to the ActionManager. The Data Access Layer typically handles storage for various entities. Entities stored here may include some or all of:

[0667] b. Bookmarks (operation history)
[0668] c. Metadata information for SmartWatchService and the Guards
[0669] d. Guard processing itemsExample use-case implementations or realizations for the above SmartWatch Wide Use-Cases (1)-(x) are now described: i. Develop and Deploy Smart Guard Adapters may include:

[0670] a. Development of new population providers for a Guard population source

[0671] b. Development of new Identity Change Strategy for Guard population source
[0672] c. Development of new types of activities
[0673] d. Changes in Business Object Model (i.e. modifying an existing action, creating new activities)

ii. Define & Configure Guard: User registers a new Guard configuration. Guard configuration classes are validated for conformity to basic rules e.g. whether valid activations are present and whether a population source exists. A guard runtime structure is built from the configuration classes. A second implicit validation takes place during this phase, in which the service verifies that the configuration alasses are valid, that no cyclic sources exist and that no member classifications contain invalid tasks. A guard’s detail structure is also built from this configuration. The details include the current lifecycle and runtime state, in addition to the default guard properties taken from the configuration.

Supplying metadata for SmartWatch management GUI is a scenario which may be supported by the SmartWatchMetadata service as this service evolves to a concrete representation. The metadata service may expose metadata from the service or configuration file.iii. Configure SmartWatch System—this use case is self-explanatory.

iv. Manage Guard runtime: Typically, the User activates a Guard, e.g. as shown herein in FIG. 107. Guard activation may comprise some or all of the following steps, suitably ordered e.g. as follows:

[0674] a. The Guard’s details are pulled from the existing list
[0675] b. If the Guard has been modified, some reconciliation takes place (including verifying that the existing members, classification, configurations are still valid)
[0676] c. If the Guard is currently in ‘inactive’, or “halted” state—any current members have their triggering rules re-activated. This is done through the GuardMemberManager.
[0677] d. The triggering rules for the population source are re-activated. In case this is the first activation, the provider may register them first and then activate.
[0678] e. The Guard’s runtime status is changed to active: A user de-activates a Guard, Guard de-activation may comprise some or all of the following steps, suitably ordered e.g. as follows:

[0679] a. The Guard’s details are retrieved based upon the provided guard identifier.
[0680] b. The Guard’s activation schedules are removed from the Temporal Monitor through a call to ServiceBrokerManager
[0681] c. The triggering rules for a Guard’s members are deactivated
[0682] d. The triggering rules for each of the Guard’s population providers are deactivated
[0683] e. The Guard’s runtime state is changed to inactive
[0684] f. The system may View Guard execution status and log.

v. Monitor SmartWatch system health e.g. Maintain and report Guard runtime performance count—this use case is self-explanatory.

vi. Apply Guard Changes—this use case is self-explanatory. Guard (or SmartGuard) is a term used herein to identify the instance. SmartWatch handles many instances of evaluations.
Each such instance typically includes various evaluations such as, for example, population definition, monitoring definition, actions definition.

vii. Discover New Population Element, e.g. as shown herein in FIG. 108. Discovery of a new population element may comprise some or all of the following steps, suitably ordered e.g. as follows:

[0085] a. Discover new population element is called directly by the user or by a callback from activation.

[0086] b. For each population source configured under the Guard, the service retrieves the new patients.

[0087] c. The new patients are added to an exclusion list for each population source.

[0088] d. For each potential new member the cluster is retrieved from the Person Identity Service.

[0089] e. Duplicate patient clusters are joined (filtered).

[0090] f. Patients existing within the Guard’s population source are filtered.

[0091] g. The patient is added to the Guard’s collection of members.

[0092] h. The new member is given the initial classification targeted by the population source.

[0093] i. The new member is pushed into the processing queue with the list of applicable tasks from the classification just set.

viii. Process Known Population Element, e.g. as shown herein in FIGS. 109 and 110.

[0094] Processing of a known population element may comprise some or all of the following steps, suitably ordered e.g. as follows:

[0095] a. Process is triggered by user or activation (timer or monitor).

[0096] b. Check for temporal triggering rules which fired for subscribed events.

[0097] c. Check for data event triggering rules which fired for subscribed events.

[0098] d. Check for identity triggering rules that fired.

[0099] e. For each rule—check for identity changes and handle as appropriate. Unclassify the current member and add to initial classification (re-evaluation of all classifications). If appropriate—add new members for split cluster fragments.

[0100] f. Check for data changes (updates/deletes) which might change past recommendations.

[0101] g. The member is pushed into the processing queue with the list of applicable tasks.

ix. Process General Population Element, e.g. as shown herein in FIGS. 111-112.

[0102] Processing of a general population element may comprise some or all of the following steps, suitably ordered e.g. as follows:

[0103] a. Get applicable tasks based upon Guard and member classification.

[0104] b. Execute Task Rule (Knowledge Module or hard-coded rule) e.g.:

[0105] b-1. Invoke KNEO to evaluation knowledge module.

[0106] b-2. KNEO invokes the KFW to evaluation the knowledge module.


[0108] KNEO (Knowledge Evaluation Orchestration) is a functionality within SmartWatch which simplifies SMW communication with KFW since in order to execute KM in KFW, SmartWatch typically collects patient data, which may optionally, as a design choice, be effected by a separate service, namely KNEO.e. Receive evaluation result and handle activities, activity could be of any of the following types:

[0109] c-1: Registering a specialized triggering rule.


[0111] c-3: Step in or out of a classification (see Reclassification Activity).

[0112] d. Log all activities.

Re-Classification Activity may include any or all of the following:

[0113] a. Add or remove a member to the classification based upon the evaluation result.

[0114] b. Analyze member’s current classification status. If the member is not a part of any existing classification, Unsubscribe from Person Identity Service and Unsubscribe from applicable Monitors. If the member is a part of existing classification, check whether the classifications have changed.

[0115] c. Collect applicable classification triggering rules including Update triggering rule subscriptions.

[0116] d. Update patient process history and Guard run history.

x. Activate Task: Cause a specific task to be activated by schedule or by callback e.g. as shown herein in FIGS. 113 and 114.

[0117] The data event monitor is now described with reference to FIGS. 115-123. An architectural overview of certain embodiments of the data event monitor, using different architectural views to depict different aspects of the system, is now provided. The data event monitor (DEM) and Temporal Monitors typically provide a way to trigger patient evaluations in SMW. Any suitable implementation of such triggers may be employed, which may differ from those specifically described herein e.g. due to specifics of the interlaying system such as how the CDR is built and what other capabilities are. For example, triggering may occur when the incoming message is being processed rather than as described herein.

[0118] The following use cases may be provided:

[0119] i. Manage triggering rules subscriptions (add/remove subscriptions to pattern rules): Accept subscription changes from consumers, and reflect them in the subscription database. This may include new subscriptions, along with the patient indices (for either inclusion or exclusion). This also handles updates such as removing triggering rules from patients and changes to the triggering rules for patients. The subscription information typically includes enough information to allow the consumer to pull results later on with a different level of granularity (SmartGuard, classification, Task, member).

[0120] This information is typically reflected in the results as well.

[0121] ii. Maintain Subscription’s patients and indices: When there are EMPI (Enterprise Master Patient Index) changes of a known member, “fix” all of the subscriptions to ensure that the correct patients continue to be monitored. In terms of population management, when a member is added to or removed from the Smart Guard, this use case changes the patient list while retaining the actual subscription. Use cases iii and iv are both “Collect events” use cases which map CDR data to an event object model e.g. that of FIG. 140 which is a diagram of an example of one suitable event object model. Typically, a suitable event object model is generated for each medical domain. Each domain typically comprises a
realm of medicine such as Encounters, Labs, Medications, Documents, Imaging, Immunization, Allergies, and Pathology. Encounters typically include any type of physician visit, either to a PCP (Primary Care Physician) or as admission into hospital.

iii. Collect new events: This use case gets “latest” events from a data source such as the CDR for processing, either in push or pull mode. A typical event would arrive from the CDR, but there is the option to receive events from other sources.

iv. Collect old events (also termed herein “bootstrap scenario”): Get events from the data provider from any timeframe for processing (“Bootstrap” scenario). These events are to be tagged in order to be matched only against the set of subscriber/pattern rules that “missed” them (either the new Smart Guard or a freshly reactivated one as described herein with reference to the Control rule execution use case. This operation is performed at the subscriber level, for all of the subscriber’s subscriptions.

Use cases v-vii are all “Deliver results” use cases.
v. Deliver results per consumer request (consumer pulls results): When a consumer requests results from the data event monitor, a use case may provide such results. It is the consumer’s responsibility to ask for the start and end points of the results.

vi. Notify Consumer of new results: “Pings” the consumer when there are results waiting for it. This may be configured to only ping any given consumer at a certain interval (no more than once every 30 minutes, for example).

vii. Process Events: As shown in FIG. 115, this use case covers the “event processing chain” in the data event monitor. When data events enter this chain, this use case may harmonize the data (terminology & unit of measure), determine what subscriptions are fired by the data, and store the results of the matching operation.
viii. Maintain pattern rules: General maintenance of the pattern rules that the data event monitor can support. These may include creating new stored procedures for matching, defining pattern rule parameters, and mapping the two together. Three components are ultimately used by the Data Event Manager; also termed herein “data event monitor”, when matching events to subscriptions: Rules Processors, Event Working Tables, and Subscription Tables. The Rules Processors make the various Event Working Tables/Subscription Tables pairings meaningful.

ix. Control rule execution (stop/start at the subscriber level)

[0723] In this use case, the Data Event Monitor allows to stop or start the rule execution. When starting over, there is a need to collect all events that were missed during the time in which the rules were deactivated (similar to the bootstrap scenario).

[0724] Example implementations or realizations of the above data event monitor use-cases (i)-(ix), other than where self-explanatory, are now described.
i. Manage triggering rules subscriptions (add/remove subsections to pattern rules)

[0725] When monitoring data for new subscriptions, historical events may or may not be looked at, going as far back as is predetermined to be appropriate, according to the application. Subscriber accesses the subscription service. Subscriber may provide:

[0726] a. Pattern rule

[0727] b. Patient Indices, including the instructions for inclusion (population monitoring) or exclusion (population discovery).

[0728] c. Subscription Information (parameters for the pattern rules).

[0729] d. Ping Interval—What is the minimum time allowed between pings from the event monitor when triggering rules fire. This may be the anti-flood mechanism.

[0730] Processing may include the following operations:

[0731] a. Look up pattern rule.

[0732] FIG. 142 is an example of a pattern rule comprising a subscription snippet, in XML format. FIG. 143 is an example of a pattern rule comprising a definition snippet, in XML format. It is appreciated that the above are merely examples of pattern rules which allow other suitable application-specific pattern rules to be developed analogously, and are not intended to be limiting, but are rather merely illustrative.

[0733] b. Validate subscription against pattern rules.

[0734] c. Detect triggering rule changes to existing patient—subscriber associations


[0736] e. Persist the actual triggering rules: Include information that may allow subscriber to later pull results with desired levels of granularity, such as SmartGuard, classification, task, member. The event monitor may mirror this type of information in the results.

[0737] f. Attach patient ids to the triggering rules for inclusion or exclusion.

[0738] g. Begin monitoring data for these rules.

[0739] Updates may be performed, e.g. as illustrated herein the diagrams of FIGS. 116-117A.

[0740] Deprecate triggering rules from patients (cease using these, without actually deleting them), e.g.:

[0741] a. Receive subscriber id, member id, triggering rule, task, classification code, pattern rule parameters.

[0742] b. Get subscription handler for that rule.

[0743] c. Deprecate the rules in the database.

ii. Maintain Subscription’s patients and indices e.g. as illustrated in the diagram of FIG. 117B. Typically, the following occurs:

[0744] a. Consumer notifies Subscription Service that there are EMPI (Enterprise Master Patient Index) changes for a member. Update the patient indices in the patient subscription tables.

[0745] b. Consumer notifies Subscription Service that a member may be added-to, or removed-from a subscription. Add/Remove the member association to the subscription.

[0746] Historical events may or may not be looked at for these scenarios, going as far back as is predetermined depending on the application. Reference is made in this connection to FIGS. 117A-117B which are diagrams illustrating a basic flow of UpdateSubscription and UpdateSubscriptionIdentities functionalities respectively.

iii. Collect new events e.g. as per the diagrams of FIGS. 118A-118B. Event collection service manages all of the events that it gets from the providers. Providers work uncoupled from the Event Collection Service by knowing how to collect data from whatever source they are in charge of Modes may include a Pull mode which is operative to receive data from source in real-time (ala CEĐEVENT.Listerner); and a Pull mode which is operative to retrieve proactively from source (ala CDR).

[0747] At a desired frequency such as once per half hour, new events are collected from the source using a suitable conventional event collecting functionality such as the
TableLookupEventProvider or OnlineCommandEventLoader Functionalities are mentioned in FIG. 118A. Record retrieval may be based on a sort of handle, which can be: timestamps, record number, or other identifier of the information source. Activation is time-based. OnlineCommandEventLoader adheres to the Online Source's policies. For example, proof may be demanded that the patient is actually being treated by a user of SmartWatch. The Event Provider maps the new event data into the object model, and passes the object(s) to the Event Collection Service. Events are stored in an event table, based on a suitable event object model such as the example illustrated in FIG. 140. Events receive a batch number from the batch manager. The Process Events use case is initiated. FIG. 118A is a diagram of a basic flow of a new event collection functionality. FIG. 118B is a diagram of a basic flow of a by-criteria event collection functionality.

iv. Collect old events (also termed herein "bootstrap scenario"), e.g., as per the diagram of FIG. 119 which illustrates a basic flow for an old event collection process. Typically, consumer instructs Data Event Monitor to collect and process all data events based on a timeframe (issues a "Bootstrap" command). Note that the subscription may already exist separately before a Bootstrap is requested. All subscriptions for a subscriber are bootstrapped. Query the Data Provider for the events, typically by performing some or all of the following steps, suitably ordered e.g., as shown:

[0748] a. Use the DataEventCriteria object to define the event set.

[0749] b. Insert these into a suitably constructed event working table(s) tagged so as to indicate that the operation is for BootstrapSubscription ID X. Subscription ID links back to the Subscriber.

[0750] c. Get Batch ID from batch manager.

[0751] d. Harmonization and Matching proceed as normal, and when the "bootstrap" exclusion phase later runs within the matching processor, it may strip out any results for events that were triggered by any other subscription ID.

v. Deliver results per consumer request (consumer pulls results), e.g., as shown herein in the diagram of FIG. 120. Results, when delivered, may be sent at the subscriber level, or at a more fine-grained level, such as results for a specific task for a subscriber. Consumer requests Data Event Monitor Service for results; the operation may include the following steps:

[0752] a. Consumer provides “bookmark.”

[0753] b. In the SmartWatch system, the bookmark is the batch number of the last batch that was closed during the last results retrieval. When the Data Event Monitor is called for more results with this batch number, it increments the number, and provides results from batch+1 onward, to the new most-recently-closed batch.

[0754] c. Query Results table for tasks that fit the consumer's timeframe.

[0755] d. Only results in “Closed” batches are eligible.

[0756] e. Provide the requested results.

vi. Notify Consumer of new results, typically including:

[0757] a. Continuously monitor the results database for undelivered, closed events.

Pong subscriber if the minimum interval has passed since the last ping

[0758] b. When the subscriber receives a ping, raise event that Batch X is closed.

[0759] c. Event Handler looks at every result, and identifies the subscription id's for subscriptions and the ping time; if any. Attempt to add subscriber to “ping list” along with when the ping may happen. Set a timer at the subscriber level so that the ping will sound when it is supposed to. If the subscriber is already on the “ping list”, then if the ping interval for this subscription is supposed to happen sooner than the “time left” on the ping list for that subscriber, overwrite the entry on the ping list. Otherwise, do nothing.

[0760] d. When a timer pops up from the pinging list, ping the subscriber.

vii. Process Events e.g. as shown herein in FIG. 121.

[0761] This Use Case is realized by the processor chain. As new data events enter the queue, a Harmonizer picks them up and harmonizes events, including some or all of:

[0762] a. Terminology Mapping

[0763] b. Unit of Measure (convert to baseline)

[0764] c. Split pertinent events into multiple events as appropriate.

[0765] d. Notify batch manager

[0766] e. Drop events from processing chain as appropriate.

[0767] f. Notify batch manager

[0768] g. Mark event as “harmonized” in the working event table(s).

[0769] A Matcher then performs some or all of the following operations:

[0770] a. Monitors its “working tables”, namely various tables where events are exchanged to be stored later “matched” against subscriptions in subscription tables.

[0771] b. Waits until the current batch fills up with X records (configurable), or until Y seconds (configurable) pass, whichever comes first.

[0772] c. Runs all pattern rule processors against the events in a table, and the subscriptions that work with those events. An example of a pattern rule processor might be a stored procedure in the working table.

[0773] d. Rule Processors are “keyed” in such a way as to say what working tables they support, and what subscription tables they support.

[0774] e. For discovery triggering rules, the matcher may compare the results against the members that the subscribers already have in the populations, and may remove those results from the final result set.

[0775] f. Another removal step is the “bootstrap removal phase,” where the matching processor removes results that were generated by subscriptions other than the subscription for which the event is keyed.

[0776] g. Another removal step is performed for deactivated subscriptions. This is typically performed as a separate step because patient monitoring rules abstract the individual patient when doing the matching, and only join them after the rule fires.

[0777] h. Results are delivered to the results table.

[0778] FIG. 121 is a diagram showing a basic flow for an event provision process operative in accordance with one embodiment of the present invention.

viii. Maintain pattern rules e.g. as shown herein in the diagram of FIG. 122.

[0779] a. A New Pattern Rule is designed, typically by a human operator and translated into a Rules Processor (for example, a stored procedure).

[0780] b. Determine what Event Working tables and Subscription tables the new pattern rule “works” with.

[0781] c. Create new Event Working tables and Subscription tables if appropriate.
d. Define the compatible data types for the operands in the triggering rules within any Rules Processor.

ix. Control rule execution (stop/start at the subscriber level)

a. Subscriber orders a “stop” for its subscriptions. This may affect all subscriptions that the subscriber has. Persist the “IsActive” field for each subscription object to false.

b. Consumer orders a “start” for a subscription. Persist the “IsActive” field for each subscription object to true. Operation may be similar to the bootstrap scenario described above.

FIGS. 123A and 123B are subscription activation and de-activation basic flows, useful for implementing control rule execution use case (ix) according to certain embodiments of the present invention.

The SmartWatch Person Identity Service of FIG. 1A is now described, according to certain embodiments of the present invention, with reference to FIGS. 14-17. As described herein, SmartWatch is operative to achieve automated decision support focused around population with common characteristics. SmartWatch relies on the functionality of gathering clinical data from diverse systems. SmartWatch is designed to operate under a significant design constraint—the reality in which a patient does not have a unique identifier across the enterprise. To meet this constraint SmartWatch communicates with conventional EMPI (Enterprise Master Patient Index) systems that know how to cluster patient identifiers (indexes) based mostly on probabilistic algorithms. The SmartWatch Person Identity Service serves as an arbitrator between SmartWatch and conventional EMPI systems.

The Person Identity Service is typically different to VIA, which is a commercial Virtual Identity Aggregation service provided by DMotion, Israel. While VIA is an abstraction layer from concrete EMPI vendor implementation allowing dmotion components to make an EMPI call using a unified API, Person Identity Service (PERIS), on the other hand, implements at least one additional functionality that is not part of the services proposed by EMPI vendors, namely a subscription capability to effect changes in person identity.

The Person Identity Service is typically operative to:

a. receive EMPI (Enterprise Master Patient Index) clustering information about specific sets of patients (e.g., up to date cluster state information (on demand) and clustering changes information (publish subscribe pattern)).

b. Mediate the above capabilities of an EMPI provider such as initiate.

c. Minimize calls to EMPI provider. Since a central consumer of the service is SmartWatch—an automated system—the load may be significantly more difficult than human users making requests.

EMPI (Enterprise Master Patient Index) clustering changes over time is a constraint stemming from the nature of the distributed healthcare environment without a unique patient identifier. Since EMPI cluster changes, the service is operative to help its consumer(s), such as SmartWatch manage the information it keeps for the long run on patients, in various points in time, as such changes may change decisions being made. A Use-Case View of the Person Identity Service is shown in FIG. 124. Some or all of the following use cases may be provided:

i. Retrieve Person Identity (Cluster): When the system is requested by its consumer to retrieve a person current cluster—the set of indexes that are currently clustered together according to the EMPI (Enterprise Master Patient Index) provider. The entry point is an index, related to the principal index, and the outcome is the set of indexes currently associated with it by the EMPI provider. Unlike search operations in VIA (a commercial Virtual Identity Aggregation service provided by DMotion, Israel)—there is no human user making decisions—the EMPI provider is the sole authority according to which later decisions are made.

ii. Manage identity change rules subscriptions (ARM): Accept subscription changes from consumers, and reflect them in the subscription database. This may include new subscriptions and modifications to existing subscriptions. The subscriptions are meant to allow service consumers to be aware of changes to identity changes of persons (patients) that the consumer is interested in. For example, the consumer (SmartWatch Service) would like to be notified whenever EMPI (Enterprise Master Patient Index) system has made any change to a given patient of his (a member) such as join or split. The subscription includes a person index that identifies the person for the consumer and is also termed herein the “principal index”.

iii. Use cases iii and iv are Deliver results use cases (ARM).

iii. Deliver results per consumer request (consumer pulls results): When a consumer requests for results from the service, this use case may provide those results. It is typically the consumer’s responsibility to ask for the start and end points of the results.

iv. Notify Consumer of new results: “Pings” the consumer when there are results waiting. This may be configured to only ping any given consumer at a certain interval (no more than once every 30 minutes, for example).

v. Maintain Patient Identities: synchronizes system’s knowledge of person cluster state with the EMPI (Enterprise Master Patient Index) provider. Several workflows may be employed here, depending on the capabilities of the EMPI provider, on performance requirements and allowed latency. Workflows may include receiving regular updates on all person identity changes from EMPI provider (push updates); receive a snapshot of the cluster database from EMPI provider (push snapshot); proactively request cluster state for selected person(s)—whose state was not checked for a relatively long period of time (pull proactively); or some hybrid of the above.

The diagram of FIG. 125 illustrates the significant design packages and subsystems of the service. The design packages of FIG. 125 typically include:

a. Business Logic Layer: implements the business logic and provides the services described above.

b. Cluster Resolving Service: allows consumers to retrieve a person cluster which comprises a set of indexes that currently represent the same person according to an EMPI (Enterprise Master Patient Index) provider. This call may yield a call to EMPI provider (EMPI-Get), but may also be resolved within the service itself, in case the cluster is already known and reasonably up-to-date. Consumers may specify a time interval, which defines what “reasonably up-to-date” means from their perspective.
c. ClusterResolvingService: a service which allows EMPI (Enterprise Master Patient Index) providers, or some mediator process on their behalf, to communicate clustering changes to the system.
d. SubscriptionService: implements publish subscribe pattern, allowing consumers to subscribe to identity change rules for selected people that they are interested in. The subscription includes a principal patient index as the identifier of the person of interest.
e. Data Layer: A layer which handles the storage of the various entities in order for the system to function. Entities to be stored may include Subscriptions & subscribers (people of interest to subscribers); Indexes and their current clusters; and Results (i.e.  indexes which changed and the subscriber relations.)
f. EMPI (Enterprise Master Patient Index).Adaptors: A package (typically a shared dll), overviewed in FIG. 126, which contains interface as well as an out-of-the-box adapter operative to retrieve a cluster from the EMPI provider. The EMPI adaptors package abstracts away from the system the exact details of how a “get” operation that retrieves a cluster, given that an index is performed in the specific project. The implementing classes may be customized at the project level to accommodate the different EMPI configurations at different customer sites.

Example implementations or realizations of the above use-cases are now described, except where self-explanatory.

Retrieval Person Identity (Cluster): a call is made to ClusterResolvingService. Within such a call the service may first check whether the cluster can be found at the DAL, and whether it is “fresh” enough for consumer requirements if any. If not, the cluster is retrieved from EMPI (Enterprise Master Patient Index) provider, stored and returned to consumer. In parallel, this starts an analysis process of cluster changes by the changes manager, one embodiment of which is illustrated in FIGS. 127A-127B.

An alternative flow, as shown in FIGS. 128A-128B, is returning the cluster together with changes made that the consumer (also a subscriber) may be aware of. In this case the consumer supplies also subscriber id and a handle (timestamp or record number) to retrieve results by, the handle typically representing the latest knowledge of consumer for the given person.

Manage identity change rules subscriptions (ARM): An UpdateSubscriptions call is made to Person Identity Service’s Subscription service. The service “persists” the subscriber, with its personal interest in its subscription storage; typically this involves the service saving the subscription, comprising a subscriber or patient to be monitored, in storage.

Maintain Patient Identities: An example proactive flow is illustrated in FIG. 29. The proactive flow of FIG. 29 is significant when the EMPI (Enterprise Master Patient Index) vendor is unable to push updates to the system. It is realized by a call from a scheduling system to the ClusterUpdatesService typically under a suitable specialized interface termed herein ProactiveClusterUpdatesService.

The passive flows (pushes from EMPI (Enterprise Master Patient Index) provider or a mediator on their behalf) are realized by calls to the ClusterUpdatesService, as illustrated in FIGS. 130A-130B. The proactive flow (main flow) is realized by an internal process which looks for persons whose identity is outdated and queries them with EMPI vendor to update the system knowledge of the cluster state.

According to one centralized embodiment of the present invention, PersonIdentityService is deployed in one node and SmartWatchService(s) calls it remotely. According to another distributed embodiment of the present invention, wherever an individual SmartWatchService is deployed, PatientIdentityService is deployed as well, serving only that individual “local” SmartWatch system. The two embodiments may be suitably combined. The SmartWatch Temporal Monitor is now described, with reference to FIGS. 131-134. Typically, the monitor creates a component to facilitate SmartWatch execution of event driven tasks. The monitor typically provides mechanisms for subscribing, publishing, and delivering the results using a pull and push model, all without losing results and without pushing results to wrong consumers.

A Use-Case View of the SmartWatch temporal monitor is provided in FIG. 131A. An overview thereof is provided in FIG. 131B. Use cases of the temporal monitor may include some or all of the following:

i. Manage triggering rules subscriptions: The use case starts as a result of the “Process Population Element” use case, usually when the monitoring criteria is fired for a population member. According to SmartGuard definitions and business logic, SmartGuard Service requests the Temporal Monitor system to subscribe for one or more triggering rules.

ii. Schedule Tasks: The use case starts with the “Manage Guard runtime” use case, when a guard is activated. At that time, typically, the system ensures guard tasks are being run according to schedule defined within the guard. The system would subscribe these requests similar to i. Manage triggering rules subscriptions use case.

iii. Calculate Rules: This use case is triggered by a timer when the time calculated based on a subscription triggering rule elapses. The elapsed time event triggers a workflow typically including some or all of the following steps:

a. Populates the results repository with new data
b. Using the subscription’s triggering rule calculates the event’s next occurrence
c. Updates the prospective tasks repository with the new calculated event occurrence
d. If the subscription delivery preference is to use a push method the results are pushed to the URI specified by the consumer

If the rule’s calculated number of occurrences or the subscription’s end date are reached, the subscription is removed

iv. Deliver results per consumer request: This use case starts as a result of receiving a consumer request for getting results delivered after the execution of prospective tasks scheduled based on the submitted by the consumer triggering rules. The system searches the results repository using the criteria specified by the consumer. The use case ends as the system delivers the results.

v. Notify consumers of new results: The use case starts when the calculated subscription triggering rule fires and the subscription delivery preference is to use a push method. The results are pushed to the URI specified by the consumer. The use case ends as the system delivers the results.

FIG. 131B is a logical view of the temporal monitor according to certain embodiments of the present invention. As shown, the temporal monitor typically comprises a service...
layer which exposes services and/or packages to its consumer e.g. SmartWatch, such as one or more of the following services and packages:

a. SubscriptionService: service which provides functionality for managing the subscriptions and delivering the results to SmartWatch as a consumer.

b. SchedulingService: façade that allows consumers (SmartWatch service) to store and remove subscriptions. Receives a scheduling request, validates it, and stores it into the subscription repository.

c. PatternRuleManagementService: A service providing functionality for managing the Temporal Monitor triggering rules used by the system to create the subscription’s prospective tasks executed by business layer at runtime.

A Business Layer encapsulates the Temporal Monitor business logic exposed as subsystems. The subsystems include internal service providing functionality for managing subscriptions, monitoring execution of prospective tasks based on the subscription’s triggering rules, and delivering the results to the consumer. Functionalities which may be provided may include one, some or all of the following:

a. SubscriptionManager: responsible for managing the subscription’s lifecycle. Also the subsystem notifies the Runtime Manager subsystem of new subscriptions that need an initial prospective task to be stored in the prospective tasks repository and are ready for execution.

b. ResultManager (not shown): delivers results per user request or pushes the results to a consumer defined location (URI).

c. RuntimeManager: Subscribes for notification events coming from the Subscription Manager subsystem or a prospective task elapsed time event. A business logic is applied to each event and a set of actions is executed. The next RR occurrence is computed and results are pushed to consumers.

[0820] A Data Access Layer (DAL) manages database access for the business layer.

[0821] Example Implementations or realizations for the temporal monitor’s use cases are now described, except where self-explanatory:

i. Develop SmartWatch Triggering Rules Subscription Workflow: This may be a non-automated process effort e.g. as shown.

ii. FIG. 132 is a diagram showing the basic flow of the Update Subscription functionality also termed herein the “schedule tasks” functionality. The workflow is operative to manage triggering rules subscriptions. The system receives a set of triggering rules subscriptions including the results delivery preferences (pull or push method). The subscriptions are validated based on predefined business logic and stored in the subscriptions database. The use case ends after the subscriptions are saved and a new Subscription Event is fired for all new subscriptions entering the system. The event triggers the execution of a workflow that calculates the triggering rules e.g. as described herein with reference to the iii. Calculate Rules use case iii of the temporal monitor. This results in creation of a new record in the prospective tasks repository and if the time of the new prospective task is closer than the “closest event” time held so far, the time is reset to the new prospective task’s time.

[0822] This core use case typically employs the service, business, and data access layers. The steps involved in this use case are outlined in the sequence diagram of FIG. 132. After receiving a subscription request, the Subscription Service calls the subscription manager which performs subscription rule mapping, after successful subscription arguments validation the Subscription Manager constructs a subscription object which is passed to the Temporal Monitor data access layer. In case of a validation error, an exception is thrown describing the type of error.

iii. Calculate Rules: This use case is triggered by a timer when the time calculated based on a subscription triggering rule elapses or when a triggered subscription change event is fired by the Subscription Manager. Steps which may be included in this use case are illustrated in the sequence diagrams of FIGS. 133-134 and may include some or all of the following steps:

[0823] a. The Runtime Manager gets the prospective task associated with the triggering rule.

[0824] b. Based on the subscriber’s delivery preferences, the system stores the results into the Results database ready to be retrieved by the subscriber, or pushes the results to the subscriber ResultAwaitCallback service.

[0825] c. The Calculate Rule workflow calculates the next occurrence and saves it in the Prospective Tasks database.

[0826] d. After rule computation, in case of no future occurrence, the task is removed.

[0827] e. Runtime Manager retrieves the earliest task from a prospective tasks database and sets the next time on the timer.

iv. Deliver results per consumer request: In this use case the service would retrieve results from its result data base, based on the parameters supplied to it in the request (ARM compliant). The bookmark in this case is a timestamp.

v. Notify consumers of new results: This use case is based on a callback to the subscriber, to a URI specified at subscription time. Based on subscription time preference, the results may be stored in the DB for the subscriber to pull them (ping mode) or just sent directly (push). The push mode is expected only for scheduling service subscription (in which the subscriber wants a “wake-up call” but would not need to pull any information past that).

vi. Maintain pattern rules: This may be a manual process, performed by a developer who defines new triggering rule parameters, writes a handler that would know how to construct the proper subscription object from the input parameters, and registers the new rule.

[0828] Referring now to FIGS. 135-137 inter alia, the Action Manager carries out SmartWatch recommendations to the outside world by notifying human users or other systems about SmartWatch findings. The Action Manager provides flexibility to PS developers in creating mechanisms to fit the specific client’s policies and needs, including message structure, format and look & feel, channels, schedule, workflows. This lowers the cost of project implementation by promoting quick development cycle of (sometime complex) actions to meet a specific client’s need (project versus product). Typically, integration with the customer’s current clinical workflow is seamless and the end consumer typically receives the messages within his day-to-day operational systems. Implemented actions follow up on sent notifications to “close the loop” and ensure notifications are being appropriately used, including a notification consumers’ feedback mechanism that would allow continuous system improvement and system tune up. Typically, messages and notifications are not lost, notifications cannot be sent to wrong recipients, and message replays are allowed in the event of system error.

[0829] Action manager use cases may include some or all of the following:

[0830] i. Develop SmartWatch Action Workflow: developer develops a new action workflow to support client needs. ii. Execute action: starts as a result of the “Process Population Element” use case described herein, usually
when the monitoring criteria is fired for a population member. According to SmartGuard definitions and business logic, SmartGuard Service requests the Action Manager system to execute one or more Action Workflows. The Action Workflow collects appropriate information, creates a notification message, decides on the Delivery Channel, formats the message to fit the Delivery Channel, and sends the message through the channel. The end result includes sending a notification message to an Action Recipient, which can be either a Care Provider or an Action Receiving/Responding System. This Use Case describes the available activities that can be used to create an Action Workflow. For example, Action Workflows can be developed in order to send e-mail or SMS, to add a record to a database/CDR, to invoke a workflow in an EMR/CPOE system, to send a report through fax and to activate another Action Workflow. The Action Workflow includes the logic to determine what data to be included in the notification, appropriate formatting, delivery channel, metadata properties. The Action Workflow can also include custom logic to interact with other systems and resources, e.g. dbMotion Security, Subscription System, and any other system that Action Manager supports. The use case ends when the requested Action Workflow is executed.

[0831] iii. Get Action Workflow Metadata. This use case typically defines data elements used for Action Workflow, defines Security settings, and/or uses delivery channels.

[0832] iv. Recall action: deals with situations in which actions SmartWatch invoked in the past are overridden and the new decision may be communicated to consumers who already received messages during original action invocation, for example, data changes in the CDR such as final lab results, or deleted records as a result of a DLI receiving update message.


[0834] vi. Replay notifications

[0835] vii. Send notification (broken down from an executed action).

[0836] FIG. 135 is a logical overview of the Action Manager according to certain embodiments of the present invention. As shown, the action manager may comprise:

[0837] a. Business Logic Layer: exposes services to its consumer—SmartWatch. Typically includes:

[0838] a-1: ActionMetadataService: Provides consumer with metadata about the action manager. Used mostly by design time components such as SmartWatch management tool during SmartGuard editing.

[0839] a-2: ActionInvocationService: A façade that allows consumers (SmartWatch service) to invoke actions supported by Action Manager, including receiving an invocation request, validating it and submitting the action for execution.

[0840] a-3: ActionDirectory: An internal service which holds mapping between action definitions and concrete action implementations (usually a workflow but maybe a simple class). Used by ActionMetadataService in order to list actions and describe them. Used by ActionInvocationService in order to map action request parameters implementing workflow and validation rules.

[0841] a-4: ActionRuntimeManager: An internal service which manages the running action workflow’s lifetime. Starts off with an execution being submitted by ActionInvocationService and is operative to monitor, report, manage (e.g. lookup hanged workflows and stop them). Also includes set of tools & services to facilitate rapid development of workflows, to meet developer productivity goals.


[0843] b. Communication Layers: responsible for handling any communication the workflow employs. It may include some or all of the following types of communications: Input—which provide data to be included in notifications and to be used in workflow decision making; Output—delivering notification to recipients in the outside world (humans or systems); and Response—receiving messages from recipients (humans or systems) that include either acknowledgments or answers (from recipients) to specific questions included in notification.

[0845] The communication layer typically includes:

[0846] b-1: dbMotionServices & AdditionalServices are packages which supply workflow developer with a set of input communication to services it consumes. The package contains service client proxies. dbMotionServices are product-based (supplying access to services such as security, KNEO, dbMotion Business layer, PCP and so on), and AdditionalServices supplies customer-specific information sources.

[0847] b-2: NotificationDeliveryService: delivers messages through delivery channels. It starts off with a notification request from a workflow, maps it to one of its delivery channels and assures reliable delivery, properly logged and so on. A commercially available product, such as a communication enabled business process (CEBP), may be used.

[0848] b-3: ActionResponseService: allows recipients to influence the workflow by responding to it. Responses may trigger events in the corresponding workflow. Based on pluggable listeners that know to communicate responses back to the workflow. This subsystem typically includes a Response Listeners package in which customized listeners reside.

[0849] A Data Access Layer (DAL) manages database access for business and communication layers.

[0850] Example implementations or realizations of the above-described use cases for the action manager are now described:

[0851] i. Develop SmartWatch Action Workflow: Typically non-automated. The workflow may be in accordance with the diagram of FIG. 136.

[0852] ii. Execute action: This use case employs several services from the business and communication layers and also typically employs the DAL in recording any operation being executed and artifact being produced (notifications, recipients, . . .).

[0853] The following functionalities, action validation and notification delivery, typically form part of the execute action process above and typically comprise implementations of the execute action use case:

[0854] Validate Action: The diagram of FIG. 137A illustrates how actions are invoked by SmartWatchService, how the action request is validated and how the action request is submitted by the ActionInvocationService to ActionRuntimeManager. Next, ActionRuntimeManager initializes and starts the workflow e.g. as shown in the diagram of FIG. 137B which, for simplicity, does not include details of the workflow being run. A typical action workflow may include decisions about who and what to communicate, including getting data to support decisions and to be included in the message, pre-
paring message content (formatting), deciding on recipients, sending notification(s), waiting for responses and reacting to them.

Deliver Notification: FIG. 137C is a diagram of a notification delivery process. The workflow has followed
in a properly formatted message and prepared a notification message (content formatted). It calls NotificationDeliveryService to manage the delivery of the notification to its recipient(s).

Get Action Workflow Metadata: This use case is realized by a call from SmartWatch Management tool to ActionMetaDataService. Management tool which is a GUI component in which SmartGuard is edited. The metadata service would call ActionDirectory to locate the metadata such as list of actions, parameters, workflow associated with an action and so on.

Get Action Response: This use case starts when one of the Action Response Listeners receives an Action Response Message. The message is forwarded to ActionResponseService for processing the response. This in turn would trigger the action workflow (assuming the workflow is waiting for that response) or the ActionResponseService would trigger a new workflow to process the Action Response e.g. as shown in FIG. 137E.

FIG. 141 is a table of ontological relationships, defined in conventional, public SNOMED terminology. The set of relationships in FIG. 141 is an example of semantic relationships, some or all of which may be used by the CTS subsystem of FIG. 1A. It is appreciated that this example is provided merely for purposes of clarification by way of example and, like many other examples provided herewith, is not intended to be limiting.

The system shown and described herein typically but not necessarily uses conventional cache implementations such as Cache Application Block from Microsoft Enterprise library to provide client side caching functionalities in the CTS client subunit, which may be used by various of the SMW subsystems described herein.

An example HF (heart failure) patients classifications is described below with reference to FIGS. 144A-144C, including triggering rules and other characteristics for entry and exit tasks. Example admitted HF patient sub-classifications are described below with reference to FIGS. 145A-145D, including triggering rules and other characteristics for entry and exit tasks. An example of an LVFS function evaluation method, including triggering rules and other characteristics for a monitoring task, is described below with reference to FIGS. 146A-146C. An example ACEI or ARB for LVFS is described below with reference to FIGS. 147-150. An example classification is described below with reference to FIGS. 151-152B, including triggering rules and other characteristics for entry and exit tasks. Example vocabulary codes are described herein with reference to the table of FIG. 153.

An example HF PATIENTS CLASSIFICATION is now described with reference to FIGS. 144A-144C, including triggering rules and other characteristics for entry and exit tasks.

An example Entry Task is now described. Triggering Rules may include one or both of: Identify new HF problems—an example list of codes is provided below with CTS triggering rules; and

Example Decision logic is now described in words and in pseudo-code.

| In words | When Patient is alive AND Patient has a HF Condition AND The latest HF Condition is not negated Then Return isHFPatient = True Else Return isHFPatient = False |

In pseudo code

When PatientAdministration/DeceasedInd <=> 1 //Defined as bit data type in the CDR
AND HF Conditions DRI is not empty Conditions/Condition/value/(code,codeSystem) Member In CTS list e.g. as described below (CTS list for KFW).
AND Latest (effectiveTime_Start) HF Conditions negationInd <=> 1 //No use of Condition.negationInd in UPMC.
OR Latest (effectiveTime_Start) HF Conditions statusCode notMemberIn 
[(2,16,840,1.113883,5,14, aborted), [2,16,840,1.113883,5,14, cancelled],
[2,16,840,1.113883,5,14, suspended]]
Condition status concept may be retrieved from the CTS (design-time)
Then Set Evaluation Result to True
Outputs may include isHFPatient, whose type is Boolean Conclusion ER.

Attribute names, values and other characteristics of outputs are set out in the table of FIG. 144B.

SmartWatch KM is now described. Inputs may include:

“Is HF Patient” KM evaluation result (Boolean Conclusion ER)

Decision logic, in words and in pseudo-code, may be as follows:

In words:
When “Is HF Patient” KM evaluation result is True Then Add member to population “HF Patients”, Step in classification Remove member from population “Candidates”, Step out classification Else

In pseudo code:

When “Is HF Patient” KM evaluation result = True Then Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/activityId = “HFPatients” /move = “In” AND Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/activityId = “Candidates” /move = “Out”
Outputs may include EntryHFPatientsClassification, whose type is Classification Decision Activity.

CTS: Retrieve HF Condition codes

**Triggering Rule may be:**

```java
[0883]

CTS list=
    named query="GetLocalCodesForConceptHierarchy"
    Parameters="
        @code="84114007"
        @codeSystemId="2.16.840.1.113883.6.96"
    
    ]

KNEO may be:

[0864]

```java

CTS list=
    named query="GetLocalCodesForConceptHierarchy"
    Parameters="
        @code="84114007"
        @codeSystemId="2.16.840.1.113883.6.96"
    
    ]

KFW

[0865]

```java

CTS list=
    named query="GetConceptHierarchy"
    Parameters="
        @code="84114007"
        @codeSystemId="2.16.840.1.113883.6.96"
    
    ]

An Exit Task for the HF patients classification may be as follows:

**Triggering Rules**, each of which may be implemented in XML, may include some or all of:
- Identify cancelled HF Problems for specific patient—list of codes may be as above (CTS list for triggering rule). (No use of Condition, negationInlined in UPMC); and
- Identify expiration of specific patient
- Is HF Patient KM: may be same as for the entry task.

**SmartWatch KM:**

[0866] Inputs may include:
- "Is HF Patient KM evaluation result (Boolean Conclusion ER)"

**Decision logic**, in words and in pseudocode, may be as follows:

In words:
When "Is HF Patient KM" evaluation result is False
Then
Remove member from population "HF Patients", Step out classification
Exit criteria type: "not relevant" — patient was removed from the population (patient is deceased) i.e. there is no need to monitor the

-continued

The result stored in the EDR is the "Is HF Patient" (clinical) KM evaluation result as-is.

In pseudo-code:

```
When "Is HF Patient" KM evaluation result = False
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classificationId = "HFPatients"
/move = "Out"

AND
Create Evaluation Result - Action Invocation Activity:
ActionInvocationActivity/ActionInvocation/actionId = "StoreToEDR"
/ActionInvocationParameter/alias = "Is HF Patient"
/receiId = null
/extension = null
/InformationModelSSID = $IsHFPatientER.SSId
/value = $IsHFPatientER

Outputs are described in the table of FIG. 144C. CTS may be as for the entry task.

Admitted

[0867] An example sub-classification for admitted HF patients is now described with reference to FIGS. 145A-145D, including triggering rules and other characteristics for entry and exit tasks.

**Entry Task** may be as follows:

**Triggering Rules** may be implemented in XML and may include some or all of the following:
- a. Identify Active (status=active AND discharge date=null) Inpatient Encounters in specific Facility for specific Patient
- b. Identify (status=null AND discharge date=null) Inpatient Encounters in specific Facility for specific Patient
- c. Identify Completed (status=completed AND discharge date < > null) Inpatient Encounters in specific Facility for specific Patient
- d. Identify Completed (status=null AND discharge date < > null) Inpatient Encounters in specific Facility for specific Patient

Is Eligible HF Encounter KM may be as follows:

**Inputs/Query Parameters** are described in the table of FIG. 145A.

**Decision logic**, in words and in pseudocode, may be as follows:

In words:
When
Encounter is active OR Encounter discharge date is from the past 24 hours
AND
Encounter from a given facility
AND
Encounter deals with hospitalized patients (inpatient OR emergency)
AND
Encounter length of stay is less than 120 days
AND
Patient age exceeds 18 years (at the admission date)
The SW KM (smart watch knowledge module) e.g. as described below may save the encounter id (CPQP) in the classification. The encounter id may be extracted from the IsEligibleHFEncounterER.Reasons. In general, only one encounter may be retrieved, but in exceptional cases more than one may be retrieved. In order to guarantee that the SW KM may only find one encounter id, enforce also a latest operation (on top of all other conditions).

Then Return is EligibleHFEncounter=True
Else Return is EligibleHFEncounter=False

In pseudo code:

When

((Encounters/Encounter/statusCode/code = "active" AND Encounters/Encounter/statusCode/codeSystem = "2.16.840.1.113883.5.1.4") OR
Encounters/Encounter/statusCode/code = null)
AND
Encounters/Encounter/effectiveTime__end = null)
OR
((Encounters/Encounter/statusCode/code = "completed" AND Encounters/Encounter/statusCode/codeSystem = "2.16.840.1.113883.5.1.4") OR
Encounters/Encounter/statusCode/code = null)
AND
Encounters/Encounter/effectiveTime__end = 24 hours > Now)
Encounter status concept may be retrieved from the CTS (design-time) AND
(Encounters/Encounter/assignedOrganization/id/root = "2.16.840.1.113883.5.3.11.1.1.1.6.5") AND
Encounters/Encounter/assignedOrganization/id(extension = "RE") OR
(Encounters/Encounter/assignedOrganization/id/root = "2.16.840.1.113883.5.3.11.1.1.1.6.5") AND
Encounters/Encounter/assignedOrganization/id(extension = "RI") AND
(Encounters/Encounter/code = "EMER") AND
Encounters/Encounter/code.system = "2.16.840.1.113883.5.4") OR
Encounters/Encounter/Code = "IMP") AND
Encounters/Encounter/code.system = "2.16.840.1.113883.5.4") OR
Encounters/Encounter/code = "ACUTE") AND
Encounters/Encounter/code.system = "2.16.840.1.113883.5.4") OR
Encounters/Encounter/Code = "NONAC") AND
Encounters/Encounter/code.system = "2.16.840.1.113883.5.4")
Encounter type concept may be retrieved from the CTS (design-time) AND
When Encounters/Encounter/effectiveTime__end = null Then Now () -
Encounters/Encounter/effectiveTime__start <= 120 days Else
Encounters/Encounter/effectiveTime__end -
Encounters/Encounter/effectiveTime__start <= 120 days
AND
Encounters/Encounter/effectiveTime__start -
Patient/Administration/BirthTime >= 18 years

The SW KM e.g. as described below may save the encounter id (CPQP) in the classification. The encounter id may be extracted from the IsEligibleHFEncounterER.Reasons. In general, only one encounter may be retrieved, but in exceptional cases more than one may be retrieved. In order to guarantee that the SW KM may only find one encounter id, enforce also a latest operation (in addition to all other conditions).

Then Set Evaluation Result to True

In pseudo code:

When

("Is Eligible HF Encounter" KM evaluation result = True)
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classification.id = "Admitted HF Patients"
/move = "In"
/ClassificationDecisionParameter/key = "SW_ADMITTED_HF_ENC.Root"
/value = $sEligibleHFEncounterER.Reasons...Act_ref/Root
/ClassificationDecisionParameter/key = "SW_ADMITTED_HF_ENC_EXT"
/value = $sEligibleHFEncounterER.Reasons...Act_ref/Extension
AND
Create Evaluation Result - Action Invocation Activity:
ActionInvocationActivity/ActionInvocation/actionId = "StoreToEDR"
/ActionInvocationParameter/alias = "Is Eligible HF Encounter"
/Id = null
/extension = null
/InformationModelSSID = $sEligibleHFEncounterER.KSSId
/value = $sEligibleHFEncounterER

Outputs are described in the table of FIG. 145C.
An example Exit Task for the sub-classification of the admitted HF patients is now described.

In words:
When "Is Eligible HF Encounter" KM evaluation result is False
Then
Remove member from population "Admitted HF Patients", Step out classification
Exit criteria type: "to be completed" — patient was removed from the population due to some other event (not death) hence completion of all his recommendations is still to be monitored.
Action: Store Result to EDR

Inputs may include:
"Is Eligible HF Encounter" KM evaluation result (Boolean Conclusion ER)
Decision logic, in words and in pseudocode, may be as follows:

In words:
When "Is Eligible HF Encounter" KM evaluation result is True
Then
Add member to population "Admitted HF Patients", Step in classification and save encounter id (CPQP) when classification

Action: Store Result to EDR

The result stored in the EDR is the “Is Eligible HF Encounter” (clinical) KM evaluation result as-is.

In pseudo code:

When

("Is Eligible HF Encounter" KM evaluation result = True)
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classification.id = "Admitted HF Patients"
/move = "In"
/ClassificationDecisionParameter/key = "SW_ADMITTED_HF_ENC.Root"
/value = $sEligibleHFEncounterER.Reasons...Act_ref/Root
/ClassificationDecisionParameter/key = "SW_ADMITTED_HF_ENC_EXT"
/value = $sEligibleHFEncounterER.Reasons...Act_ref/Extension
AND
Create Evaluation Result - Action Invocation Activity:
ActionInvocationActivity/ActionInvocation/actionId = "StoreToEDR"
/ActionInvocationParameter/alias = "Is Eligible HF Encounter"
/Id = null
/extension = null
/InformationModelSSID = $sEligibleHFEncounterER.KSSId
/value = $sEligibleHFEncounterER

An example SmartWatch KM is now described.
The result stored in the EDR is the “Is Eligible HF Encounter” (clinical) KM evaluation result as-is.
In pseudo code:

```
When "Is Eligible HF Encounter" KM evaluation result = False
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classificationId = "AdmittedHFPatients"
moves = "Or"
AND
Create Evaluation Result - Action Invocation Activity:
ActionInvocationActivity/ActionInvocation/actionId = "StoreToEDR"
/ActionInvocationParameter/alias = "Is Eligible HF Encounter"
/сотrend = null
/extension = null
/InformationModelSSID = $IsEligibleHFEncounterER.SSId
/value = $IsEligibleHFEncounterER
Outs is described in the table of Fig. 145D.
HF-2: Evaluation of
```

An example, also termed herein HF-2, of an LVS function evaluation method, including triggering rules and other characteristics for a monitoring task, is now described with reference to FIGS. 146A-146B.

An example Monitoring Task for the above LVS function evaluation method is now described.

Triggering Rules may include:

- Identify changes in the existence of LVS Function Evaluation Document for specific patient—list of codes; e.g. as for the triggering rule for the CTS described below, e.g. using a suitable XML implementation.
- An example Evaluation of LVS Function KM may be as follows:

- Decision logic, in words and in pseudocode, may be as follows:

In words:

When At least one LVS Function Evaluation Document exist in Patient Record
Then
Return Code HF2-T1
AND
Return isPerformed = True
AND
Return isRequired = False
Else

Return Code HF2-T2
AND
Return isPerformed = False
AND
Return isRequired = True
```

Code translations may be e.g. as shown in FIG. 153.
In pseudo code:

```
//codeSystem = "2.16.840.1.113883.3.57.1.4.6.1"
/displayName = null
/effectiveTime = null
/domainCode = null
Set Evaluation Result (IsPerformed) to True
Set Evaluation Result (IsRequired) to False
```

Rule #1:

When
LVS Documents DRI is empty
Documents/ClinicalDocument/code/(code,codeSystem) Member In CTS list described below.
Then
Create Evaluation Result - Quality Conclusion:
QualityConclusion/QualityAnswer/SystemName = null
/code = "HF2-T1"
```

Outputs are described in the table of FIG. 146B. The Quality Conclusion contains a vocabulary code. The code designation may be retrieved from dmMotion vocabulary by the consumer (EDR). The vocabulary may also hold the mapping between the code and the relevant category e.g. as shown in FIG. 153. Attribute names and values are described in the table of FIG. 146C.

An example SmartWatch KM for the LVS function evaluation method’s monitoring task may be as follows:

Inputs may include:

- “Evaluation of LVS Function” KM evaluation result
- Decision logic, in words and in pseudocode, may be as follows:

In words:

When always
Then
Action: Store Result to EDR
```

(8871) The result stored in the EDR is the “Evaluation of LVS Function” (clinical) KM evaluation result as-is.
In pseudo code:

```
When i=1
Then
Create Evaluation Result - Action Invocation Activity:
ActionInvocationActivity/ActionInvocation/actionId = "StoreToEDR"
/ActionInvocationParameter/alias = "Evaluation of LVS Function"
/сотrend = null
/extension = null
/InformationModelSSID = $EvaluationOfLVSFunctionER.SSId
/value = $EvaluationOfLVSFunctionER
```

Outputs may include EvaluationOfLVSFunctionAction, which may be of the Action Invocation Activity type.
CTS: Retrieve list of document types ("LVS Function Evaluation")

Triggering Rule may include:

```
CTS list=
  @code="11522-0"; @codeSystemId="2.16.840.1.113883.6.1"
  @code="18745-0"; @codeSystemId="2.16.840.1.113883.6.1"
```

KNEO may be:

[0872]

```
CTS list=
  @code="11522-0"; @codeSystemId="2.16.840.1.113883.6.1"
  @code="18745-0"; @codeSystemId="2.16.840.1.113883.6.1"
```

KFW may include:

```
CTS list=
  @code="11522-0"; @codeSystemId="2.16.840.1.113883.6.1"
  @code="18745-0"; @codeSystemId="2.16.840.1.113883.6.1"
```

HF-3

[0873] An example, also termed herein HF-3, of ACEI or ARB for LVSD is now described with reference to FIGS. 147-150.

A Monitoring Task for ACEI or ARB for LVSD, may for example be as follows:

Triggering Rules, each of which may be implemented in XML, may include some or all of:

- Identify changes in ACEI or ARB Medications for specific Encounter—list of codes e.g. as described below with reference to the triggering rule for these medications.
- Identify changes in ACEI or ARB Allergies for specific Patient—list of codes e.g. as described below with reference to the triggering rule for these allergies.
- Identify negated OR cancelled HF Problem events for specific patient—e.g. as per exit task triggering rules for HF patient classification as described above.
- Identify changes in ACEI or ARB Contraindication Conditions—list of codes e.g. as described below with reference to the triggering rule for these contraindication conditions.

ACEI or ARB for LVSD KM:

[0874] Inputs/Query Parameters are described in the table of FIG. 147. The list of codes for the first and second rows in the table may be as described herein for the CTS of the ACEI or ARB for LVSD monitoring task (KNEO for ACEI and ARB medications). The list of codes for the third row in the table may be as described herein for the CTS of the ACEI or ARB for LVSD monitoring task (KNEO for diastolic dysfunction conditions). The list of codes for the fourth row in the table may be as described herein for the CTS of the ACEI or ARB for LVSD monitoring task (KNEO for contraindications conditions). The list of codes for the fifth and sixth rows in the table may be as described herein for the CTS of the ACEI or ARB for LVSD monitoring task (KNEO for ACEI and ARB allergies).

Decision logic, in words and in pseudocode, may be as follows:

In words:
An example of a suitable decision table (in words) is provided in FIG. 148. Codes translations may be as shown in FIG. 153. In pseudo code:

When
Patient has Diastolic Dysfunction?

Diastolic Dysfunction Conditions DRI is not empty

Conditions/Condition/value(code,codeSystem) Member in CTS list- 0

ACEI Medication Prescribed During Encounter?

ACEI Medications DRI is not empty

Substances/Administration/Substance/Administration/Medication/code(code,codeSystem) Member in CTS list e.g. as described herein with reference to the KFW of the ACEI and ARB medications.

ARB Medication Prescribed During Encounter?

ARB Medications DRI is not empty

Substances/Administration/Substance/Administration/Medication/code(code,codeSystem) Member in CTS list- e.g. as described herein with reference to the KFW of the ACEI and ARB medications.

ACEI Allergy?

ACEI Allergies DRI is not empty

Allergies/Allergy/intolerance/value(code,codeSystem) Member in CTS list - e.g. as described herein with reference to the KFW of the ACEI and ARB allergies.

ARB Allergy?

ARB Allergies DRI is not empty

Allergies/Allergy/intolerance/value(code,codeSystem) Member in CTS list e.g. as described herein with reference to the KFW of the ACEI and ARB allergies.

ACEI or ARB Contraindications Conditions?

ACEI and ARB Contraindication Conditions DRI is not empty

Conditions/Condition/value(code,codeSystem) Member in CTS list - e.g. as described herein with reference to the KFW of the contraindication conditions for the ACEI and ARB.
Set Evaluation Result (IsPerformed) to True/False according to the table of FIG. 148.
Set Evaluation Result (IsRequired) to True/False according to the table of FIG. 148.
Outputs are described in the table of FIG. 149.
The Quality Conclusion contains a vocabulary code. The code designation may be retrieved from dbMotion vocabulary by the consumer (EDR). The vocabulary may also hold the mapping between the code and the relevant category e.g. as shown in FIG. 153.
Attribute names, values and other characteristics are set out in the table of FIG. 150. An example of a SmartWatch KM for the monitoring task of the ACEI or ARB for LVSD is now described.

Inputs may include:
- "ACEI or ARB for LVSD" KM evaluation result

The result stored in the EDR is the "ACEI or ARB for LVSD" (clinical) KM evaluation result as-is.

In pseudocode:

When l=1
Then
Create Evaluation Result - Action Invocation; Action Invocation(Activity = "StoreToEDR"
/Action Invocation/Parameter.alias = "ACEI or ARB for LVSD"
/rootId = null
/extension = null
/Information Model SSI = $ACEIorARForLVSD.ESSID
/value = $ACEIorARForLVSD

Outputs may include ACEIorARBforLVSD.Act, of the Action Invocation Activity type.
An example CTS for the monitoring task of the ACEI or ARB for LVSD is now described.
KFW may comprise:

ACEI CTS list=
  named query="product/GetLocalCodesForConceptHierarchy"
  Parameters=
    @code="372733002"
    @codeSystemId="2.16.840.1.113883.6.96"
ARB CTS list=
  named query="product/GetLocalCodesForConceptHierarchy"
  Parameters=
    @code="372913009"
    @codeSystemId="2.16.840.1.113883.6.96"

KFW may comprise:

ACEI CTS list=
  named query="product/GetConceptHierarchy"
  Parameters=
    @code="372733002"
    @codeSystemId="2.16.840.1.113883.6.96"
ARB CTS list=
  named query="product/GetConceptHierarchy"
  Parameters=
    @code="372913009"
    @codeSystemId="2.16.840.1.113883.6.96"

Contraindication conditions: list of contraindication conditions (ACEI and ARB) may be retrieved.

Triggering Rule may be:

KFW may comprise:

Diastolic dysfunction conditions: list of diastolic dysfunction conditions (subset of HF conditions) may be retrieved.

KFW may comprise:

Temp

[0881] An example sub-classification for temp discharged HF patients is now described with reference to FIGS. 151-1523, including triggering rules and other characteristics for entry and exit tasks. Motivation—to support a business requirement to continue monitoring discharged patients for 24 hours after the discharge. This sub-classification is a workaround for Dynamic Triggering Rule Parameters. Parent Classification—Admitted HF Patients.

An example Entry Task for the sub-classification for temp discharged HF patients is now described.

Triggering Rules: Identify Completed Inpatient Encounters in specific Facility for specific Patient e.g. as per third and fourth triggering rules for entry task of admitted HF patients sub-classification, described above.

Is Discharged HF Patient KM:

[0882] Inputs/Query Parameters may be as per row 1 in the table of FIG. 145A.

Decision logic, in words and in pseudocode, may be as follows:

In words:
When Encounter discharge date is from the last 24 hours
Then Return IsDischargedHF Encounter = True
Else Return IsDischargedHF Encounter = False
In pseudo code:

```
When (Encounter/Encounter/statusCode/code = "completed" AND
Encounter/Encounter/statusCode/code/system = "2.16.840.1.113883.5.1.4") OR
Encounter/Encounter/statusCode/code = "null")
AND
Encounter/Encounter/effectiveTime_end + 24 hours > Now
Encounter status concept may be retrieved from the CTS (design-time)
Then Set Evaluation Result to True
```

This rule may be exactly as the first part of the Is Eligible HF Encounter KM e.g. as described above with reference to “Is eligible HF encounter KM” pertaining to the entry task of the sub-classification for admitted heart failure patients. Therefore it may be useful to reuse this logic i.e. the Is Eligible HF Encounter KM may be divided into 2 KMs/rules that one of them is this one.

Outputs may include is Discharged HF Encounter, whose type is Boolean Conclusion ER.

Attribute names, values, and other characteristics are set out in the table of FIG. 151.

**SmartWatch KM**

**[0883]** Inputs may include:

- “Is Discharged HF Patient” KM evaluation result (Boolean Conclusion ER)

Decision logic, in words and in pseudocode, may be as follows:

```
In words:
When "Is Discharged HF Patient" KM evaluation result is True
Then
Add member to population "Temp Discharged HF Patients", Step in classification

In pseudo code:

```
When "Is Discharged HF Patient" KM evaluation result = True
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classificationId = "TempDischargedHFPatients"
/move = "In"
```

Outputs are set out in the table of FIG. 152.

An example Exit Task for the sub-classification for temp discharged HF patients is now described.

Triggering Rules may include at least the following:

**Temporal Trigger Discharge Date+24 Hours**

**[0884]** Is Discharged HF Patient KM: as for the entry task. An example SmartWatch KM for the exit task may be as follows:

Inputs:

**[0885]** “Is Discharged HF Patient” KM evaluation result (Boolean Conclusion ER)

Decision logic, in words and in pseudocode, may be as follows:

```
In words:
When "Is Discharged HF Patient" KM evaluation result is False
Then
Remove member from population "Temp Discharged HF Patients", Step out classification

In pseudo code
```

```
When "Is Discharged HF Patient" KM evaluation result = False
Then
Create Evaluation Result - Classification Decision Activity:
ClassificationDecisionActivity/ClassificationDecision/classificationId = "TempDischargedHFPatients"
/move = "Out"
```

Outputs are set out in the table of FIG. 152.

Example vocabulary Codes for the embodiment of FIGS. 144A-152B are set out in the table of FIG. 153.

**[0886]** FIG. 154a is a simplified functional block diagram of a high level architecture of a smart agent system constructed and operative in accordance with certain embodiments of the present invention. In FIG. 154a, it is appreciated that EMR is also termed EHR herein. Data Analyzer Desktop Agent is also termed EHR Agent Host or SmartAgent host herein. Agent Viewer is also termed EHR Agent User Interface or SmartAgent user interface herein. A VPO (Virtual Patient Object) is an Object, in the sense of object-oriented programming, that is retrieved by a Patient data services functionality and stores a patient’s clinical History. A VPO Analyzer is also termed EHR Agent Orchestration Web Service herein. A VPO Business Domain is also termed Clinical Data Web Service or “Patient Clinical Data Services” or “Patient data services” herein. The term “WPF” refers to a conventional computer-software graphical subsystem for rendering user interfaces such as but not limited to, for Windows-based applications, Windows Presentation Foundation.

**[0887]** The term “PPOL” refers to a Patient-Provider-Organization Link such as but not limited to that provided in DBMotion’s commercially available HIE. Generally, a medical domain typically comprises interconnected network of entities (e.g. as shown in FIG. 172) such as Providers (e.g., medical staff, clinician, nurse) providing care services to Patients, at medical Organizations (e.g., clinic, hospital). The information about relations (Links) between medical entities is an integral part of healthcare information which facilitates intelligent services. For example, a PPOL may list patients being treated by a specific clinician, describe an organizational hierarchy of a medical unit, and return the PCP of a specific patient. A Providers-Patients-Organizations Link (PPOL) computerized service typically provides and manipulates information about relations (Links) between medical entities. A PPOL typically manages different entities and the relations between the entities in the medical domain, such as—for example—relationships between a patient’s GP and the physician’s office manager. A PPOL typically does one or more of: connects to a conventional EMPI (Enterprise Master Patient Index) patient registry; provides a unique ID for patient and provider identities (aka cluster); stores provider-provider relations, e.g. office manager; stores provider-pa-
tient relations, e.g. PCP based on ADT messages; and acts as a providers’ registry or connects to an existing such registry. **[0888]** The smart agent system of FIG. 154a typically includes a client and a server, each including some or all of the illustrated blocks, interacting suitably e.g. as shown. The SmartAgent client application typically interacts with EMR activities and recognizes EMR context e.g. user, patient and workflow e.g. as described herein in detail. The PPOL module typically performs identity management including harmonization of different identities of entities. The CTS terminologies and vocabularies module typically harmonizes local terminologies used in various data sources to the ontology used by the HiES e.g. the HiES commercially available from DhMotion, Isreal. The security authority block typically authenticates and authorizes user access to patients’ clinical data. The “get VPO” block typically retrieves patient data (e.g. VPOs—virtual patient objects) from repositories, which may be harmonized and/or federated, in the HiES. **[0889]** The term “smart agent” is used herein to include an HiES-EMR bridging system according to any of the embodiments shown and described herein, which facilitates cooperation between HiES and a population of one or more EMRs with which the HiES interacts. For example, the smart agent may perform any or all of the following operations:

a. identify an application which has opened on a workstation on which the HiES client is installed, as a medical service provider application e.g. EMR. This may occur because the EMR is compatible with a context sharing standard that the smart agent also supports or may be screen captured. Alternatively, APIs in the operating system may be available which define (assign an operating system name to) the relevant screen and/or fields within the screen, enabling the smart agent to identify the application.

b. identify health care providing user e.g. because user logs in with password, or by capturing user’s name from the screen, or e.g. authenticating the user using single-sign on functionality.

c. assuming that the EMR is currently working on the medical record of a particular patient, the smart agent may find the patient identifier, e.g. by screen capture or by context sharing. Typically, when an EMR enters a medical record, the smart agent retrieves the patient identifier by first recognizing the EMR page through which the patient user accesses the medical records and then, e.g. using prior knowledge re the location of the patient identifier on that page, capturing the user identifier.

d. optionally, filter available HiES information, e.g. using suitable “attention rules”, to select suitable information to provide to the user identified in (b), pertaining to the patient identified in (c). Typically, attention rules filter out either irrelevant information or superfluous (repetitive) information, or both.

e. display all, or some (as filtered in (d)) information pertaining to the patient identified in (c). Typically, all information available from all EMRs with which the HiES interacts, is available to be displayed, unless filtered. Typically, this information is displayed to the health care providing user via an HIIE data importer such as a table or button that takes the user to an HIIE portal.

Examples of the operation of steps (d) and (e):

**[0890]** (i) if the user presses on the lab page in her or his EMR, the lab page may be identified by the hovering smart agent. The smart agent may then display to the user only lab results which are not present in her EMR, optionally blinking to indicate that such exist and are available for viewing. **[0891]** (ii) if the user presses on a medicine page in her or his EMR, the hovering smart agent may intercept that the user is prescribing penicillin to a patient called Susan Smith and may then display only relevant information such as all allergies known for Susan Smith, or the subset of Susan Smith allergies pertinent to penicillin.

**[0892]** (iii) The user presses on a lab result page and sends Susan Smith to do a lab test. The smart agent may intercept this and blink or otherwise indicate that Susan has already done this lab test.

**[0893]** The SmartAgent user interface and behavior is designed to be floated on top of an instance of an EMR and not to interrupt the regular user workflow.

**[0894]** A Floating application within User, Patient and System Context, e.g. as shown in FIG. 156a, may be used as follows:

1. The User opens a patient record in the EMR.
2. A HiES client agent (SmartAgent) installed on the client machine “captures” the patient identifier (MRN), the User Context (Username/Role), and the System Context (SystemID) and calls the HiES’s VPO Analyzer web service, which identifies the System, the User and the Patient.
3. The User is authorized and the Patient is found in the HiES.
4. The client SmartAgent gets the response and presents a Floating Application. The Floating button includes a link to launch the HiES’s Viewer with the user and patient context.
5. The User clicks the button and seamlessly accesses the HiES’s Clinical Viewer.

**[0895]** The VPO (Virtual Patient Object) Analyzer of FIG. 154a typically performs smart evaluations on the VPO in the context of the User, the Patient and the System, in order to provide more relevant and needed information to the User. One of the VPO Analyzer’s methods is “Exclude System Data”. This method excludes data from the VPO that already exists in the User’s EMR. This results in “clean” data (excluding what user can already see in the EMR) that is presented within the Results or Viewer Panels.

**[0896]** A Semantic Search method is now described:

1. The user looks for data on Diabetes (say) in the HiES. To do so, the User enables the Search option in the floating toolbar and types the phrase “Diab.”
2. Search suggestions are presented and user selects the “Diabetes” Suggestion.
3. As a result, the Results and Navigation pane opens and presents the results for Diabetes from the Patient’s VPO, organized according to Clinical Aspects (Medications, Problems, Population Membership, etc).
4. The User clicks the “Diabetes” population.
5. The Diabetes View is opened in the View panel.

**[0897]** A process operative to Launch an HiES Viewer and CareBoard is now described. Any information found is presented in the Data and Navigation panel. The information is organized according to the different clinical aspects (Laboratory, Medications, etc.) and evaluation aspects (Population membership, Metrics, Notifications, Alerts etc.). The clinical aspects and actual presented data are links to the relevant page in the HiES’s Clinical Viewer or Collaborate. Example: Under the Lab Results menu, the user sees a result for HbA1c from last week. Beside the result there are two buttons: one opens the Labs Clinical View and the other opens the Lab Results Page with the HbA1c history.
FIG. 154b is a table summarizing an example set of functional requirements of a content/VPO analyzer included in the apparatus shown and described herein.

FIG. 154c is a table summarizing an example set of functional requirements of a content capturing and sharing functionality included in the apparatus shown and described herein.

FIG. 154d is a table summarizing an example set of functional requirements of a semantic search functionality included in the apparatus shown and described herein.

FIG. 154e is a table summarizing an example set of functional requirements of a floating application included in the apparatus shown and described herein.

FIG. 155a is a table summarizing an example set of non-functional auditing, security and localization requirements of apparatus shown and described herein.

FIG. 155b is a table summarizing an example set of non-functional topological and pre-requisite requirements of apparatus shown and described herein.

FIG. 155c is a table summarizing an example set of non-functional performance requirements of apparatus shown and described herein.

FIG. 155d is a table summarizing an example set of non-functional reusability and integrability requirements of apparatus shown and described herein.

In an OnPage. Load mode of operation, FIG. 156a is an example user interface useful in entering a Patient File and launching the SmartAgent system provided according to certain embodiments of the present invention.

Floating Application Small Panel: An example user interface for a FloatingClosed functionality is illustrated in FIG. 156b. An example object table useful in understanding the functionality of the user interface of FIG. 156b is illustrated in FIG. 156c. Footnotes in the first column of the object table of FIG. 156c refer to suitably marked locations in the user interface of FIG. 156d, respectively.

An example user interface for a FloatingSearchOpen functionality is illustrated in FIG. 156d.

FIG. 157a illustrates the SmartAgent application hovering on top of an example EMR (Allscripts Sunrise—commercially available EMR) in an expanded mode.

FIG. 157b is an enlarged view of the expanded SmartAgent panel.

An example object table useful in understanding the functionality of the user interface of FIG. 157b is illustrated in FIG. 157c. Footnotes in the first column of the object table of FIG. 157c refer to suitably marked locations in the user interface of FIG. 157b, respectively.

FIG. 157d is an Object Table.

FIG. 158 is a user interface for a laboratory screenshot of a preview panel useful in accordance with certain embodiments of the present invention.

Example Attention Rule Definitions for a Clinical Content Specification are now described. The smart agent typically uses attention rules to determine whether or not to provide a user with an indication that aims to direct her or his attention to the fact that information relevant to her or him is available in the HIS. The indication may include a highlight color in the background of the Name e.g. as shown in FIG. 159a. A rule can be built from a set of predefined filter types, such that a suitable predefined combination of the rules triggers the Attention indication and the relevant data presented by default. FIG. 159b is a table presenting an example set of basic rules. FIG. 159c is a table presenting an example set of rule combinations. The indication may consider the filter types of FIG. 159c, which are combinations of the basic rules of FIG. 159b.

Example rules are as follows:

Rule 1: Exclude my E.H.R. data:
User is alerted if there is information in the HIES which does not exist in the user’s EMR. Typically, the HIES stores an indication, for each information item, of the EMR which provided that information item, allowing information items provided by a user’s EMR to be filtered out and not displayed to the user.

Rule 2: Exclude data irrelevant to workflow.

A health care providing user opens his EMR at a page pertaining to laboratory, medicine, Procedures, Allergies, Vital signs, Pathologies, Imaging results, Clinical documents, immunizations, Problems, Diagnosis, or any other EMR functionality. The Agent hovers over the EMR application, intercepts the type of page opened, and selects from among the HIE information items available, only the ones which are relevant, using predetermined criteria, to the current EMR functionality.

Rule 3: New since last seen.

Smart agent generates an alert if information which is new, relative to the point in time at which a particular health care providing user last looked at the HIES.

FIG. 160a is a table of an example set of presentation use cases.

FIG. 160b is a table of an example set of context interception use cases.

FIG. 160c is a table of an example set of system health use cases.

FIG. 160d is a table of an example set of data preparation, configuration & deployment, and extension development use cases.

The high level architecture of a smart agent system constructed and operative in accordance with certain embodiments of the present invention is illustrated in the simplified functional block diagram of FIG. 154a. The apparatus of FIG. 154a may include some or all of:

1. VPO analyzer—an addition to the VPO Web Services, is typically operative to filter and highlight pieces in the VPO which are considered important to be presented. It is typically based on GetVPO functionality and rules on those.

2. SmartAgent application. A client-installed application. Typically comprises some or all of:
   a. SmartAgentHost—a windows try application that connects all the dots on the client machine
   b. SmartAgentPresentation—SmartAgent presentation Module (WPF) application

The Context interception library—Library Screen capturing interfaces. Its job is to capture events from EMR application such as patient context, and pass it on to the SmartAgentHost.

Certain user context interception sequences, some or all of which may be included in the model, are illustrated in the diagram of FIG. 161, which includes an Interaction example—User context interception sequence. FIG. 161 and other figures herein are generated in Enterprise Architect UML format; it is appreciated that this does not limit the scope of the invention and is merely utilized as one possible format for demonstrating one possible example set of data structures and computerized processing methods useful for implementing the present invention.

An example use case model is now described with reference to FIGS. 162, 163a-163c, 164-166.
A Use Case Model Overview (Use case diagram) is illustrated in FIG. 162. “Perspective” Overview’s are illustrated in FIGS. 163a and 163b including an actors overview diagram and a use case context diagram, respectively.

Regarding the configuration and deployment functionality of FIG. 162:

Configure & Personalization presentation options may include, inter alia: Presentations, skin vs EHR, Presentation rules and Base query, each of which is now described in detail according to respective embodiments of the present invention:

Presentation skins refers to an ability to customize the SmartAgent application appearance to have a plurality of different “look and feel”s. Typically, end users want the EMR agent to appear on top of a given EMR with a look and feel which is similar to that of the EMR.

Presentation Rules—refers to how the SmartAgent behaves to a given attention rule result—such as blinking frequency of alerts, or whether to blink and/or to expand the SmartAgent to view Clinical Data.

Base Query refers to filtering to obtain a VPO. Filtering can be by Time, or which clinical Aspects to obtain (e.g. filter to obtain only labs, or All Clinical Aspects).

For the “deploy updates to agent” use case, the system may update to the smart agent applications scattered all over the network (and out of network for community clinics). The smart agent may be installed on any suitable platform, such as a Citrix box. For the “install agent on Citrix box” use case, setting up the agent may be controlled by configuring Citrix sessions, or by network login script.

A Context Interception (Use Case diagram) is illustrated in FIG. 163c, which is suitable for implementing the context interception functionality of FIG. 162. Context interception may include identification, e.g. as described below, of some or all of: (a) application, (b) patient e.g. as per FIG. 164 described herein, (c) user, e.g. as per FIG. 167 described herein, (d) workflow context, (e) patient interaction, (f) user-managed user interactions, e.g. as per FIG. 167 described herein, and (g) user details e.g. as per FIG. 166 described herein. User identification may include interception of the users’ details, and interception of user-managed user interactions.

Re “Identify Application” use case shown in FIG. 162: In this UC the system intercepts the application (EMR) which the user is using. For example, is the user working with Allscripts MyWay, Cerner etc. This context may be captured from the EMR screen or provided by the EMR. This same application can be used for multiple instances. For example a Cerner app may be used in different “regions” in UPMC—Cerner H1, H2, H3. The instance may be used when applying content rules such as “exclude mine”.

Re “Identify Patient” use case shown in FIG. 162: In this UC the system intercepts the patient that the user is currently looking at in the EMR. This context may be captured from the EMR screen or provided by the EMR.

Re additional “Identify Patient” use case shown in FIG. 162: In this UC the system intercepts the patient that the user is currently looking at in the EMR. This context may be captured from the EMR screen or provided by the EMR.

FIG. 164 is a Sequence diagram of an “Identify Patient” Interaction.

Re the “Identify User” use case shown in FIG. 162: In this UC the system intercepts the user that is using the EMR in order for the health information exchange system (HIEX) to present data to that user, according security privileges as configured in the EMR. This context may be captured from the EMR screen or provided by the EMR.

FIG. 166 is an Activity diagram of an “Intercept User Details” functionality.

FIG. 167 is a Use Case diagram of an “Identify User” functionality which is useful e.g. in conjunction with the “Identify User—managed user Interaction” use case shown in FIG. 162.

Re the “Identify Workflow Context” use case shown in FIG. 162: In this UC the system intercepts the workflow in which the user is on the EMR side. For example, is the user on “labs” tab or on “medication” tab? This context may be captured from the EMR screen or provided by the EMR. An advanced scenario here is responding to “free” selection on the screen, e.g. —user selects “Hgb” on the screen and history graph needs to be shown.

Turning now to the Data Preparation functionality in FIG. 162:

Re the “Apply Rules” use case: Typically, acting on patient data, the system decides which records are to be presented using attention rules e.g. as described herein. The system is typically able to add more rules and adjust existing rules. Example rules include: —“exclude my EMR data” and —“exclude what i saw”, also termed herein the “new since last seen” rule.

Re the “Prepare Patient Data for presentation” use case: Typically, the system prepares data to be presented to the user. This includes fetching the data as well as filtering out irrelevant records, based on rules. This typically works in conjunction with the VPO analyzer of FIG. 16z.

Re the Extension Developments functionality in FIG. 162:

The “Develop & Configure EMR support” UC typically describes a possible need of the system to grow and support new EMRs, new versions of known EMRs as well as customized versions of known EMRs. The support is focused on the interception of elements from the EMR screen.

Re the Presentation functionality in FIG. 162:

The “Control floating application state” may for example include Minimize, expand, size, position, dock, close etc. In the launch health information exchange system (HIEX) Viewer UC the system acts merely as an entry point to the existing viewer (Clinical Viewer, Collaborate). The launch may or may not include context (user, patient, app). For example—navigation menu for medications (even though no medication record is currently shown). In the “Perform Search” UC the user searches for records within the patient record. The search can either be on the codified data, or free search over text (notes within acts and the content of clinical documents). In the “Present Clinician Data” UC the system presents data from the HIEX that is not related to the patient viewed in the EMR. For example—“my recent events” from a Collaborate functionality in the HIEX, “my admitted patients” and so on.

In the Present Patient Data UC the system presents patient data to the user. In the “Send To My EMR” UC the user selects act to be sent to his EMR and the system delivers them to the EMR to be presented there.

The GetVPO service of FIG. 162 is an Actor which represents clinical services or Web Services which provides the ability to query and retrieve Patient clinical Data.

An example Design Model is shown in FIG. 168. FIG. 169 is a logical diagram of the smart agent of FIG. 168.
The screen capturing functionality of FIG. 169 typically comprises a library that implements an IOC (Inversion Of Control) implementation of the Screen capturing interfaces. Its job includes capture of events from an EMR application such as patient context, and passing the patient content on to the AgentHost of FIG. 169. FIG. 171 is a logical diagram of the ScreenCapturing functionality of FIG. 169.

The ScreenCapturingEHRCContextInterceptor of FIG. 171 may comprise an interceptor based on conventional Screen Capturing technology which may operate the dlls and convert to Net events as appropriate.

The ScreenCapturingEHRCContextInterceptor of FIG. 171 may comprise any interceptor based on suitable Screen Capturing technology such as that commercially available from screenscraper studio, having a http presence on the world wide web at screen-scraper.deskexperience.com/ which is a software tool including a development kit that is capable of screen capturing actions.

The SmartAgentHost functionality of FIG. 19 typically is installed on the client machine, hosts the presentation layer and orchestrates the interaction between the presentation layer and the screen interception e.g. screen capturing. A logical diagram of controllers is shown in FIG. 174. FIG. 174 illustrates the engine and the corresponding configurations that execute and orchestrate the various interceptors shown and described herein. Typically, the apparatus of FIG. 174 is useful in conjunction with the screen capturing functionality of FIGS. 169 and 171. FIG. 174 is particularly useful in understanding an example design of context interception, its classes and methods, and scenarios the context interceptor may support, such as but not limited to: user log-in, enter patient file, exit patient file, all according to certain embodiments of the invention. The diagram of FIG. 173 describes, for the above scenarios, an initial step of getting the context and passing it to an EMR agent, so as to call VPO services, search patient and security, all in accordance with certain embodiments of the invention.

Regarding the EHRCContextManager of FIG. 174, it is typically the responsibility of this class to orchestrate the context switches, bridging between context interceptors, presentation parts, and windows events, such as but not limited to:

a. Pass on events to the presentation about patient, user, application and workflow context switches
b. Receive events from presentation such as user requests to hide/close/float the window and make sure to respond correctly to those events, for example by stopping the interceptors.

c. Make sure the window moves together with the EMR window.

The multiplicity of this class may be expected to be one per EMR window instance. (e.g. two Cerner windows would be assigned with two different context managers.

Operations, some or all of which may be performed by the InterceptorsFactory functionality of FIG. 174, are set out in the table of FIG. 175.

The RuntimeManager of FIG. 174 typically hosts the entire system. Its responsibilities include some or all of the following:

1. Manage (start/stop/monitor) the different managers.

2. Orchestrate the process of discovering new EMR window, creating a ContextManager and context interceptor for it.

A logical diagram of ContextInterception is illustrated in FIG. 174. The InterceptorConfiguration of FIG. 176 typically comprises an object which may be delivered to each interceptor when it is started. The configuration is stored and controlled. The InterceptorConfiguration may hold a configuration of the Intercept in the context of the EMR it intercepts. An example configuration is as follows:

- Connection to server
- To allow an EMR Agent work with a database such as a dbMotion database an address of the server, e.g. dbMotion server, is typically set up. The Server address is typically a URL which may for example have an http:// prefix followed by, say, //ApplicationServer:9150/dbMotion/SmartAgent
- View window position:

  - EMR Agent shell window position may be defined relatively to EMR application window. The View can be anchored to one of the sides of EMR window. Suitable parameters may specify relative position such as some or all of the following parameters as defined in FIG. 184:
  - EMR angle—angle of EMR window to be used as a start point for computing View position
  - View angle (Shell angle)—angle of View to be positioned relatively to EMR angle
  - Horizontal offset—horizontal offset of View angle from EMR angle (can be negative to position View angle to the left from EMR angle)
  - Vertical offset—vertical offset of View angle from EMR angle (can be negative to position View angle above EMR angle)

  - Each time EMR window position or size is changed, EMR Agent typically updates corresponding shell window position e.g. according to the parameters above.

- View position parameters may be specified in an InterceptionConfig.xml file under a ShellConfig tag. Here is an example of suitable ShellConfig values:

```xml
<ShellConfig
  EhrAngles="RightTop" ShellAngles="RightTop"
  HorizontalOffset="10" VerticalOffset="10" />
```

- These parameters may be specified per EMR.
- c. SmartAgent Floating Application Attention rules may be as follows:

- The EMR Agent view title area may be used to notify the user when context dependent data is delivered to the Agent from a suitable database e.g. dbMotion database. The notification may have some or all of the following 3 parameters:

- Blinking number—how many time title area blinks after some data is delivered
Blinking rise time—duration of each blink
Blinking pause time—delay between blinks
These parameters can be changed in a suitable file e.g. “dBMotion.SmartAgent.Client.Viewer.dll.config”. There follows an example fragment of the configuration file with default values:

```
<setting name="BlinkingNumber" serializeAs="String">
    <value>5</value>
</setting>
<setting name="BlinkingRiseTime" serializeAs="String">
    <value>00:06:00,2000000</value>
</setting>
<setting name="BlinkingPauseTime" serializeAs="String">
    <value>00:06:00</value>
</setting>
```

These parameters may be specified globally.

The IEHRContextInterceptor of FIG. 178 is an interface; the responsibility of implementer of this interface typically includes discovery of context changes to a given EMR window instance. The window’s assigned context manager may be subscribed to the interceptor’s events.

The IEHREventInterceptor of FIG. 178 typically comprises an “abstract” interface which only defines the minimal requirements.

The Impl package of FIG. 178 includes out-of-box interception capabilities focused on interception of some or all of: application launch, application position change, internal proxies to external interceptors, including, e.g. active EMR or screen capturing vendor.

Regarding the IEHRLaunchInterceptor functionality of FIG. 178, the responsibility of the implementer of this interface is to intercept the event of launching a known EMR application. Once launched, a dedicated EHRClearInterceptor may be created to the specific window.

The runtime manager may be subscribed to the OnApplicationOpened event.

The IEHRWindowstateInterceptor of FIG. 178 typically comprises an interface operative to allow the application to stick into the EMR window and track it as it moves.

Referring again to FIG. 168, the VPOAnalyzer is an addition to the VPO “family” e.g. clinical data services as described herein and is operative to filter and highlight important pieces in the VPO to be presented. It is based on GetVPO functionality and rules pertaining to that functionality, typically including some or all attention rules applicable to the patient clinical data retrieved as described herein in detail.

Example Framework and Contracts pertaining to the VPOAnalyzer may be appreciated with reference to FIG. 177. The AnalyzedPatientRecord of FIG. 177 typically provides a Data Contract for the VPO analyzer of FIG. 154a, containing the VPO as well as the markup—the highlighted elements of the VPO. The AnalyzerRuleHighlights of FIG. 177 typically holds the VPO markup highlights for an analyzer rule. The HighlightedElement of FIG. 177 typically allows the rule to mark / highlight VPO elements to be presented according to their logic, for example—those that do not come from a given EMR. The SystemIdentity of FIG. 177 typically comprises a data contract package which may optionally be part of PPOL of FIG. 168.

FIG. 178 is an example Logical diagram for the SystemIdentity functionality of FIG. 177.

Service & business layers may include the apparatus shown in the logical diagram of FIG. 179. The IVpoDataSource of FIG. 179 may comprise an interface which abstracts away the details of how the VPO is retrieved. A BAC-based data source for deployment outside “business” areas is provided and a business-agent version may be used when integrated into business access services hence it can utilize internal features. The SystemLinksManager of FIG. 179 may be part of the PPOL of FIG. 15 and typically comprises a manager responsible of providing configuration of system links and the relation between them. The VPOAnalyzerManager of FIG. 179 may comprise a manager/controller responsible of orchestrating the VPO analysis process.

FIG. 180 is an example logical diagram of the rules functionality of FIG. 179. The AnalyzerRuleFactory functionality of FIG. 180 typically comprises a singletone class, serving as a factory for analyzer rules, based on configuration. The configuration may be based on CareManagement group, which in turn may allow configuring different profiles and personalization. The CareEventsGetterRule functionality of FIG. 180 typically comprises an analyzer “rule” which gets CareEvents based on a CareEvents query (which events are of interest? how far back?). It may merge its results to the upper level. The OutsideEHRRecordsRule functionality of FIG. 180 typically comprises an analyzer which highlights those records which are out of reach in a given EMR. The rule may be based on system links configuration, which may be part of the PPOL of FIG. 168. The UnseenRecordsVPOAnalyzerRule functionality of FIG. 180 typically highlights those records the user have not seen yet, based on information about entry to health information exchange system (HIE) applications, for example PLV records. The VPOGetterRule functionality of FIG. 180 typically comprises an analyzer “rule” which is based on VPO query and a set of child analyzer rules which further analyze the fetched VPO. The VPOGetterRule functionality of FIG. 180 typically merges its results to the upper level.

An example Use Case (UC) Realization process is illustrated in the Identify Patient UC realization sequence diagram of FIG. 181. An example server operative to Identify Patient UC realization is illustrated in the sequence diagram of FIG. 182.

A particular advantage of certain embodiments of the present invention is that the EMR’s code need not be changed because the apparatus of the present invention is able to intercept whatever context it employs from the EMR. Any suitable method may be employed for intercepting EMR context, such as but not limited to:
a. use of a context management protocol such as CCOW.
b. screen capturing (also termed herein “screen scraping”) c. providing a specialized context interception adaptor, also termed herein “interceptor” for a particular EMR taking advantage of a specific context sharing capability which that particular EMR has. Typically, the architecture of the system shown and described herein includes one or more adaptors for context interception also termed herein “interceptors”, e.g. as shown in FIG. 176, and therefore any such special adaptor can easily be incorporated into the architecture of the system shown and described herein, as an additional or substitute adaptor in FIG. 176.

“Manual Intervention” typically includes a set of features in the Smart Agent which enable the system to locate, within a patient record, an item of information that may be
relevant for the end user in the context of the workflow. As an example, a health provider-user may be prescribing a new medication to the patient. According to certain embodiments, capturing the prescribed medication enables to search for a drug interaction or an allergy within the HIE patient record. It is appreciated that finding the relation between the prescribed medication in a given EMR and the ability to identify kind of relations to other clinical artifacts within the patient record, are not trivial, e.g. because the clinical concepts terminology is unknown to the HIE. Also, since the terminology used for the medication is unknown, there may be no ability to manage the relations to other clinical concepts.

[0992] The CTS subsystem of dbMotion's commercially available health information exchange system typically provides capabilities that make intervention in the EMR based on local terminology possible, including one or more of the following, all e.g. as described herein with reference to any or all of FIGS. 2-3 and 27-39:

[0993] 1. The dbMotion CTS maintains semantic and/or ontological relationships between Local EMR terminologies to a basic ontology e.g. as employed by dbMotion.

[0994] 2. The dbMotion Ontology contains relationships, e.g. of medication concepts and their related contraindications. Also, allergy clinical ontology concepts are related to relevant medication clinical concepts.

[0995] 3. The CTS facilitates handling of ontology relations (e.g. relations between clinical concepts), typically by using only a single, given, pre-defined medical terminology, while "translating" all terms or codes in all other medical terminologies into and out of the given terminology. All ontological relationships are represented in terms of, and typically only in terms of, the given medical terminology. For example, the code that corresponds to Medication of Penicillin typically has a relation in the ontology to many other clinical concepts which also pertain to Penicillins, such as side effects thereof. In this context, the CTS facilitates querying for all the Penicillins ontology clinical concepts, using a given terminology's clinical concept. In summary, while many medical terminologies each include a multiplicity of medical codes, and terms may be supported by the systems shown and described herein, typically, ontological/semantic relationships between concepts are represented (e.g. only as) relationships between terms in the pre-defined medical terminology; whereas ontological/semantic relationships between terms, say, A and B, within a medical terminology X other than the pre-defined medical terminology (also termed herein "the ontology") are derived by translating both A and B from terminology X into the pre-defined medical terminology, and identifying any semantic relationships between the translations of these. Typically, even if new relationships which are imported into the system are expressed in terms of a medical terminology X other than the pre-defined medical terminology, nonetheless, the relevant terms in X are first translated into the pre-defined medical terminology, so as to express the new relationships solely in terms of the pre-defined medical terminology, and only then, the new relationships, so expressed, are saved for future use.

[0996] An example CTS functionality is described above with reference to FIGS. 18A-105. A commercially available CTS functionality is provided in DBmotion's E.H. R. product. Embodiments of CTS functionality are also described below.

[0997] If some or all of features 1-3 above are provided, the SmartAgent is typically operative to present the user with an alert on an allergy to a prescribed medication by intercepting the medication name, locating its concepts in the CTS terminology as a local concept, locating the baseline ontology concept it relates to, locating the related ontology concepts of allergy and, using a given VPO, resolving if an individual patient meets this intervention rule. This is an example of use of the semantics framework provided in the commercially available dbMotion health information exchange system, in order to perform workflow context interception.

[0998] Embodiments of CTS functionality are now described:

[0999] While there is value in bringing together data from multiple source systems into a unified virtual patient record with each source-system providing data using its own "local terminology" codes, there are many limitations in viewing the resulting hodgepodge of terminologies. By converting all the data into a common set of standards-based codes, using CTS functionality, the results are much more meaningful.

[1000] The term Semantic Interoperability refers to the ability to translate data from multiple and varied Healthcare IT systems and then to organize such data in a meaningful way for display and analysis. This facilitates creation of a meaningful "virtual patient record" comprised of data from multiple source systems that can still be coherently organized and understood. Semantic Interoperability facilitates assimilation of data that originated in one system, into a different system.

[1001] The Semantic Interoperability capabilities may have some or all of the following target audiences:

[1002] End users who benefit from the meaningful translation of data to a common terminology set and the organization of the data in a meaningful way.

[1003] Vocabulary administrators tasked with maintaining the code sets used by an organization's healthcare IT systems and the mapping/translation between them.

[1004] Data experts responsible for understanding and leveraging data stored in a HIE system.

[1005] SDK developers who need to create applications that access semantically harmonized data.

[1006] Terminology of Semantic Interoperability used herein includes:

[1007] Code Set Mapping: Translation among code sets associated with various Healthcare IT data encoding standards. There are a variety of code sets commonly used by various vendor systems. For example, one EMR might encode medications using the Medispan code set, while another will use the NDC code set, another the Multum code set, and still another the RxNorm code set. Similar differences (with different sets of alternate standards) are found in the code sets used for Conditions, Labs, Immunizations, Allergies, Diagnoses, etc. Even when a common code set is used by two systems, there are often differences in the versions of the code set, and in the subset of codes actually used from within the standard code set. Moreover, even when sharing a code-set and version, there can be specific customizations and variations used by different organizations or departments.
[1008] Semantic Ontology: A set of code sets (typically including versions and subsets) used to store clinical/administrative data, together with defined relationships among the various codes that are represented and managed. An ontology includes relationships between data elements, as opposed to mere translating of varying code sets used to represent data in clinical or administrative domains. Relationships may include “grouping” (e.g., the fact that a large set of RxNorm codes for various forms of Penicillin can all be grouped together in certain contexts e.g. from a point of view of side-effects or indications), and/or inter-relationships (contradictions, interchangeability, etc.). Relationships can be within one administrative/clinical domain (e.g. the therapeutic classes of medications) or across domains (e.g. a relationship between a condition and the medication used to treat it).

[1009] Baseline Ontology and Local Terminologies: An HIE e.g. that commercially available from DBMotion, may be pre-configured with a Baseline Semantic Ontology (also termed herein a Baseline Ontology). The pre-configured or “out of the box” (OOB) Baseline Ontology can be customized and configured by a user. The Baseline Ontology typically represents a common language into which all data aggregated from source systems by dbMotion is translated (“Local to Baseline Translations”) and from which it is again translated (“Baseline to Local Translation”), if required, in the course of semantically meaningful data export (termed herein “semantic export”) to a destination system in a way that is consistent with the terminology used by that destination system. The terminologies used by the source systems and the destination systems are termed herein Local Terminologies.

[1010] Typically, the Baseline Ontology is also used to group data and optionally to enable at least one of decision support, alerting, and population management. When doing Local to Baseline Translation, a record of the original Local Terminology encoding is typically maintained for reference. It may then be a configurable option whether data is displayed in Local Terminology, Baseline Ontology, or both.

According to some embodiments, the HIE does not import/export information about the relationships between codes: The Local to Baseline Translations and the Baseline to Local Translations are concerned only with mapping terminology (codes). The broader Semantic Ontology is exclusively in the Baseline Ontology—it can be customized if desired, but it is not impacted by the source or destination systems that are interconnected by the HIE system.

Vocabulary Manager to Common Terminology Service

[1011] HIE’s Semantic Interoperability may be provided by a Vocabulary Manager (VM) sub-system. Terminology translation support may be limited to one-way translation (Local to Baseline Translation) and Semantic Ontology may be limited to basic grouping capabilities.

[1012] A Common Terminology Service (CTS) may alternatively or in addition provide Semantic Interoperability by providing a Semantic Ontology functionality that provides both translation services and semantic relationship knowledge services throughout the HIE.

[1013] In commercially available dbMotion HIE’s Release 4.2, CTS services are used by components associated with the dbMotion Collaborator application, as well as by Semantic Export capabilities introduced in the context of the Allscripts Community Record features powered by dbMotion Release 4.x. Some modules may utilize an earlier dbMotion Vocabulary sub-system. A CTS Synchronization Service enables the CTS to co-exist side by side with a Vocabulary Manager system. The CTS Synchronization Service may synchronize a newer CTS Baseline Ontology with the Vocabulary Manager system (CTS to VM) and also synchronize Local Terminology mappings managed within the VM with the CTS (VM to CTS).

[1014] If there are Baseline codes that are not part of the Baseline Ontology (referred to as Proprietary Baseline Terminologies), these are typically mapped to the CTS Baseline Ontology concepts using a structured Baseline Mapping file provided by the HIE.

[1015] A CTS may support inter alia any or all of the following use-case scenarios:

[1016] Semantically harmonize aggregated medical/administrative data from multiple source systems e.g. by translating the source system local terminology to the Baseline Ontology (Local to Baseline Translation), as well as by using the Baseline Ontology to hierarchically organize certain data areas to facilitate grouping, filtering and sorting features.

[1017] Semantically export data from the virtual patient record and its Baseline Ontology to the Local Terminology used by a destination system (e.g. a particular EMR) (Baseline to Local Translation).

[1018] Maintain bi-directional mapping between different Local Terminologies and the terminology used by the (dbMotion e.g.) Baseline Ontology.

[1019] Provide services and infrastructure for semantic navigation capabilities e.g. by using associations between various concepts belonging to different clinical domains (e.g. a Lab test related to an Allergy, related to the contraindication).

CTS may use either or both of the following Semantic Content Types:

- Baseline Ontology, which includes:
  - The baseline terminology—selection of code sets (typically including the version and sub-set of codes used) for all relevant clinical/administrative domains;
  - relationships (primarily hierarchy/grouping relationships) among those codes within and across domains.

- Terminology Maps: bi-directional code mappings from Local Terminology sets to the terminology of the Baseline Ontology. Some mappings are provided pre-configured and are referred to as “out of the box” (OOB) mappings. Additional customer-specific mappings and/or modifications to the OOB terminology maps may be provided to match the specific usage of a given Local Terminology by a given customer source system. The HIE may automatically identify unmapped codes that it encounters.

[1022] The Baseline Ontology Terminology may for example be assembled by combining selected sub-sets of different well-known standard terminology systems (SMONED, RxNorm, LOINC, etc). The specific subset of codes in the Baseline Ontology from within a given standard may be selected based on suitable criteria such as but not limited to some or all of the following:

[1023] Based on concepts most frequently used concepts by EMRs
[1024] Based on anticipated Ontology use cases (such as concepts related to Diabetes)

[1025] Compliency with pre-defined requirements

[1026] The Table of FIG. 185a is an example of origins of Baseline Ontology Terminology Components in an example HIE (the DBMotion HIE).

[1027] The Baseline Ontology Relationships or concept-relationship aspect of the Baseline Ontology may be maintained using a suitable toolset for ontology management such as OWL (Web Ontology Language), endorsed by the Worldwide Web Consortium (W3C).

[1028] Relationships supported in the CTS may for example include some or all of the following:

[1029] Classification (Hierarchy):

[1030] PARENT_CLASS_OF: A concept that is the Parent of a related classification concept.

[1031] SUB_CLASS_OF: A concept that is the Child of a related classification concept.

[1032] Mapping:

[1033] EQUALSX_TO: Equivalent meaning of two concepts.

[1034] Cross domain relations:

[1035] VACCINE_AGAINST: Cross domain relation. An Immunization concept that should prevent/immunize against the related condition concept.

[1036] Inner domain relations:

[1037] RxNorm

[1038] INGREDIENT_OF: An ingredient concept is the content of the related medication concept.

[1039] HAS_INGREDIENT: A medication concept has the related ingredient concept.

[1040] SNOMED

[1041] HAS_FINDING_SITE: A site concept for where the related condition occurs.

[1042] HAS_CAUSATIVE_AGENT: A disease concept is caused by the child causes.

[1043] In order for the CTS to provide its translation services, it may be supplied with Terminology Maps for translating bi-directionally between the Local Terminologies used by various source and destination healthcare IT systems and the Baseline Ontology. The CTS may for example be pre-configured with a set of OOB (out-of-the-box) Terminology Maps that map between Local Terminologies used by Allscripts EMRs and the Baseline Ontology. Examples of such maps, e.g. for the DBMotion HIE, are set out in the table of FIG. 185a.

[1044] Any or all of the following methods may be supported for building Terminology Maps:

[1045] OOB Terminology Maps: Terminology maps can be purchased or developed, using conventional Terminologies consultancy techniques, and provided as OOB (out-of-the-box) Terminology Maps.

[1046] Customer Provided Terminology Maps: A Vocabulary Manager sub-system of the HIE may include an “Import Tool” that can import Excel-based Terminology Maps provided by customers. These may then be automatically synchronized to the new CTS sub-system.

[1047] Surfacing of Unmapped Codes: During the data aggregation process, a Vocabulary Manager sub-system may automatically surface unmapped codes that it encounters and provide a facility for mapping them to the Baseline Ontology.

[1048] Semantic Interoperability may facilitate Semantic Export. Semantic Interoperability may make it easier to view aggregated data within a Clinical Viewer even if it only worked “one way” (from the terminology of a source system to our common baseline terminology). A CTS typically however provides full round-trip translation services, allowing the HIE to move data from one system to another in a meaningful way e.g. by providing Semantic Export capabilities into Allscripts EMRs. Data that flows into an Allscripts EMR via a Allscripts Community Record solution can be displayed in the “language” of that EMR, regardless of the source system from which it came.

[1049] Semantic Interoperability typically provides a “central hub” type system as opposed to “myriad point-to-point” systems. Rather than creating mappings between every potential pair of relevant terminology code sets, a standards-based baseline may be provided from which and to which everything is translated. This means that far fewer mappings are used to translate “anything to anything”. Analogous is the difference between using a dictionary to translate every language to and from every other language and using just one dictionary (per each of n languages) from each language into a “universal common language” and back, which effectively translates between any 2 languages.

[1050] A difference between Terminology Services and Ontology Services is that Terminology typically is limited to translating codes from one set to another, whereas Ontology also refers to relationships between those codes e.g. the hierarchy/grouping relationship that enables grouping together clinically related medications, allergies, etc. “Ontology relationships” may also be defined to enable a CTS to respond to a query such as “give me all the medications, conditions, and lab tests associated with Diabetes”. When an HIE operates exclusively through a central Baseline Ontology, the HIE can define these relationships only once in the terminology of the Baseline Ontology and can then use the relationships for any Local Terminology used by any other source system.

[1051] According to some embodiments, Semantic Interoperability is not “point-to-point”. Thus, instead of mapping from and to every possible combination of source and destination code sets, only one mapping is performed, from any source/destination code set to/from the Baseline Ontology.

[1052] According to some embodiments, Semantic Interoperability is not limited to “terminology services” and instead provides an “ontology” of relationships between concepts within a given clinical domain (e.g. a hierarchy of medications), as well as across clinical domains (e.g. all medications, conditions, and labs associated with Diabetes).

[1053] According to some embodiments, Semantic Interoperability focuses most frequently used codes and most important, as pre-defined relationships among codes. This allows efficient investment in maintaining mappings and reducing the almost infinite potential task of “mapping everything” to a manageable mapping agenda.

[1054] One implementation of the CTS (Common Terminology Services) functionality described herein is the DBMotion HIE’s software module that powers its Semantic Interoperability capabilities.

[1055] FIG. 186 is a simplified functional block diagram illustration of an embodiment of the invention showing a smart agent which may be constructed and operative in accordance with any of the embodiments shown and described.
herein, operative in conjunction with an HIE which may comprise any conventional HIE such as but not limited to DBMotion's HIE; and one or more conventional EMRs (E.H. R.'s).

[1056] The smart agent of FIG. 186 may include a client and server as shown; however, functionalities of these two may be interchanged as desired. Any suitable functional connection between the SmartAgent and the Entity Identity Management (EMPI) may be provided such as that described above with reference to FIGS. 164 and/or 168. Any suitable functional connection between the SmartAgent and Security Services may be provided such as that described above with reference to FIGS. 167 and/or 168. Any suitable functional connection between the SmartAgent and Patient Data Services may be provided such as that described above with reference to FIG. 168. Any suitable interceptors may be employed such as the Context Interceptor processor described above with reference to FIG. 166. The Client may be based in part on the apparatus of FIG. 169 including the ScreenScraping functionality which may include a context interceptor, SmartAgent Host and SmartAgent presentation client as described herein. The SmartAgent Host may optionally be based on the Context Interceptor described hereinabove with reference to any or all of FIGS. 170, 174 and 176. A suitable object model of the context which the context interceptor API of FIG. 186 may employ, is described above with reference to FIG. 170. The Web service component of the server may for example comprise the SmartAgent Presentation Server of FIG. 169 as described above. The VPO analyzer component of the server may for example be implemented in accordance with the apparatus of FIG. 177.

[1057] It is appreciated that terminology such as “mandatory”, “required”, “need” and “must” refer to implementation choices made within the context of a particular implementation or application described herewithin for clarity and are not intended to be limiting since in an alternative implantation, the same elements might be defined as not mandatory and not required or might even be eliminated altogether.

[1058] It is appreciated that software components of the present invention including programs and data may, if desired, be implemented in ROM (read only memory) form including CD-ROMs, EPROMs and EEPROMs, or may be stored in any other suitable typically non-transitory computer-readable medium such as but not limited to disks of various kinds, cards of various kinds and RAMs. Components described herein as software may, alternatively, be implemented wholly or partly in hardware, if desired, using conventional techniques. Conversely, components described herein as hardware may, alternatively, be implemented wholly or partly in software, if desired, using conventional techniques.

[1059] Included in the scope of the present invention, inter alia, are electromagnetic signals carrying computer-readable instructions for performing any or all of the steps of any of the methods shown and described herein, in any suitable order; machine-readable instructions for performing any or all of the steps of any of the methods shown and described herein, in any suitable order; program storage devices readable by machine, tangibly embodying a program of instructions executable by the machine to perform any or all of the steps of any of the methods shown and described herein, in any suitable order; a computer program product comprising a computer useable medium having computer readable program code, such as executable code, having embodied therein, and/or including computer readable program code for performing, any or all of the steps of any of the methods shown and described herein, in any suitable order; any technical effects brought about by any or all of the steps of any of the methods shown and described herein, when performed in any suitable order; any suitable apparatus or device or combination of such, programmed to perform, alone or in combination, any or all of the steps of any of the methods shown and described herein, in any suitable order; electronic devices each including a processor and a cooperating input device and/or output device and operative to perform in software any steps shown and described herein; information storage devices or physical records, such as disks or hard drives, causing a computer or other device to be configured so as to carry out any or all of the steps of any of the methods shown and described herein, in any suitable order; a program pre-stored e.g. in memory or on an information network such as the Internet, before or after being downloaded, which embodies any or all of the steps of any of the methods shown and described herein, in any suitable order, and the method of uploading or downloading such, and a system including server/s and/or client/s for using such; and hardware which performs any or all of the steps of any of the methods shown and described herein, in any suitable order, either alone or in conjunction with software. Any computer-readable or machine-readable media described herein is intended to include non-transitory computer- or machine-readable media.

[1060] Any computations or other forms of analysis described herein may be performed by a suitable computerized method. Any step described herein may be computer-implemented. The invention shown and described herein may include (a) using a computerized method to identify a solution to any of the problems or for any of the objectives described herein, the solution optionally include at least one of a decision, an action, a product, a service or any other information described herein that impacts, in a positive manner, a problem or objectives described herein; and (b) outputting the solution.

[1061] Features of the present invention which are described in the context of separate embodiments may also be provided in combination in a single embodiment. Conversely, features of the invention, including method steps, which are described for brevity in the context of a single embodiment or in a certain order may be provided separately or in any suitable subcombination or in a different order; “e.g.” is used herein in the sense of a specific example which is not intended to be limiting. Devices, apparatus or systems shown coupled in any of the drawings may in fact be integrated into a single platform in certain embodiments or may be coupled via any appropriate wired or wireless coupling such as but not limited to optical fiber, Ethernet, Wireless LAN, HomePNA, power line communication, cell phone, PDA, Blackberry GPRS, Satellite including GPS, or other mobile delivery. It is appreciated that in the description and drawings shown and described herein, functionalities described or illustrated as systems and sub-units thereof can also be provided as methods and steps therewithin, and functionalities described or illustrated as methods and steps therewithin can also be provided as systems and sub-units thereof. The scale used to illustrate various elements in the drawings is merely exemplary and/or appropriate for clarity of presentation and is not intended to be limiting.
1. A method for using a health information exchange system which stores patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

for each individual EMR within said first plurality of EMRs, performing a computerized context interception process using a processor to intercept context from the individual EMR and to identify therewithin an event whereby a health provider using said individual EMR calls up an individual patient's record from said individual EMR; and

responsive to identification of said event, using a computerized output device for providing patient record data, pertaining to said individual patient, to said health provider.

2. A method according to claim 1 wherein said patient record data includes at least one information item which is tagged to indicate a source EMR which provided said information item to said health information exchange system and wherein said patient record data provided to said health provider is filtered so as to filter out said at least one information item if said information item's source EMR equals said individual EMR.

3. A method according to claim 1 wherein said intercepting and identifying includes using a context management protocol.

4. A method according to claim 3 wherein said context management protocol comprises CCOW.

5. A method according to claim 1 wherein said health provider is served by a display screen when using said individual EMR and wherein said identifying includes capturing information from said display screen.

6. A method according to claim 1 wherein said identifying includes:

pre-storing a patient identifier location for each EMR within said first plurality of EMRs; and

capturing said individual patient's patient identifier from said patient identifier location.

7. A method according to claim 1 wherein said providing patient record data includes subscribing to events exchanged between the individual EMR and an operating system serving the EMR, ascertaining in real time at least one current EMR display operation and mimicking the EMR display operation when providing the patient record data.

8. A method according to claim 7 wherein said EMR display operation comprises at least one of a translation along a display screen, a minimization of size and a maximization of size.

9. A method according to claim 1 wherein said intercepting and identifying includes screen-capturing EMR context.

10. A method according to claim 5 wherein said patient record data includes at least one information item which is tagged to indicate a category of information to which said information item pertains, and wherein said capturing includes capturing screen information indicative of a category of information currently being viewed by the health provider, and wherein said patient record data presented to said health provider is filtered so as to filter out said at least one information item pertaining to a category of information other than the category of information currently being viewed by the health provider.

11. A method according to claim 10 wherein said category of information includes at least one of the following group of categories: laboratory information, medicines, allergies, procedures, vital signs, pathologies, imaging results, clinical documents, immunizations, problems, and diagnosis.

12. A method for using a health information exchange system, storing patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

for each individual EMR within said first plurality of EMRs, using a processor to automatically identify an event whereby a health provider using said individual EMR calls up an individual patient's record from said individual EMR; and

using a processor to apply at least one predetermined rule involving intercepted EMR context and using a computerized output device to provide an alerting indication drawing the health provider's attention to at least some of said patient record data according to said predetermined rule.

13. A method according to claim 12 wherein said generating an alerting indication comprises presenting patient record data, pertaining to said individual patient, to said health provider.

14. A method according to claim 12 wherein said predetermined rule comprises alerting said health provider's attention if said patient record data includes at least one item of information not included in said individual EMR.

15. A method according to claim 12 wherein said rule involves health-provider specific context and wherein said applying includes identifying said health provider's identifier.

16. A method according to claim 14 wherein items of information included in said patient record data are stored in computerized storage, in association with respective information source tags indicating an information source supplying each said item, and wherein a determination is made, based on said information source tags, of whether or not said patient record data includes at least one item of information not included in said individual EMR.

17. A method according to claim 16 wherein a list of information sources supplying an individual EMR is stored in computerized storage and wherein said determination includes determining whether all items in said patient record data are supplied by one of said individual EMR, and an information source appearing in the list of information sources supplying the individual EMR.

18. A method according to claim 15 wherein said predetermined rule comprises alerting said health provider's attention if said patient record data includes at least one item of information which was not seen by the health provider identified by said identifier when he was last served by said health information exchange system.

19. A method according to claim 15 wherein said identifying said health provider's identifier employs single-sign-on authentication of said health provider.

20. A method according to claim 15 wherein said identifying said health provider's user name includes capturing said health provider's identifier from a screen display employed by said health provider.

21. A method according to claim 1 wherein said intercepting and identifying includes providing an architecture which interacts with at least one context interception adaptor and,
when operating in conjunction with a specific EMR having a specific context sharing capability, using an adapter adapted to utilize said specific context sharing capability as said context interception adaptor.

22. A method according to claim 1 wherein said context intercepted includes an indication of at least one display characteristic characterizing a graphic user interface being generated by the EMR and wherein said providing patient record data includes generating a graphic user interface for displaying said patient record data which shares said display characteristic.

23. A method according to claim 22 wherein said at least one display characteristic includes at least one of: a color characteristic, a texture characteristic, a text font, a text size, and a characteristic of an icon such as a button.

24. A system for using a health information exchange system, storing patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the system comprising:

a record-call up identifying processor operative, for each individual EMR within said first plurality of EMRs, to automatically identify an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and

an intercepted context rule applying processor applying at least one predetermined rule involving intercepted EMR context and controlling a computerized output device to provide an alerting indication drawing the health provider’s attention to at least some of said patient record data according to said predetermined rule.

25. A system for using a health information exchange system which stores patient record data regarding a multiplicity of patients, to serve a first plurality of EMRs each interacting with an EMR community including a set of at least one EMR, the method comprising:

a record-call up identifying processor operative, for each individual EMR within said first plurality of EMRs, for performing a computerized context interception process using a processor to intercept context from the individual EMR and to identify therewithin an event whereby a health provider using said individual EMR calls up an individual patient’s record from said individual EMR; and

a record-call up identifying processor-driven output device controller operative, responsive to identification of said event, to control a computerized output device for providing patient record data, pertaining to said individual patient, to said health provider.

* * * * *