(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 February 2005 (03.02.2005)

PCT

(10) International Publication Number
WO 2005/010749 A2

(51) International Patent Classification[7]: G06F 9/318

(21) International Application Number:
PCT/US2004/022481

(22) International Filing Date: 14 July 2004 (14.07.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/621,772 17 July 2003 (17.07.2003) US

(71) Applicant *(for all designated States except US)*: **RAYTHEON COMPANY** [US/US]; 870 Winter Street, Waltham, MA 02451 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **O'BRIEN, Wayne, Patrick** [US/US]; 507 N. Roosevelt Boulevard, C315, Falls Church, VI 22044 (US).

(74) Agent: **ICHIYE, Keiko**; Baker Botts LLP, 2001 Ross Avenue, Dallas, TX 75201-2980 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

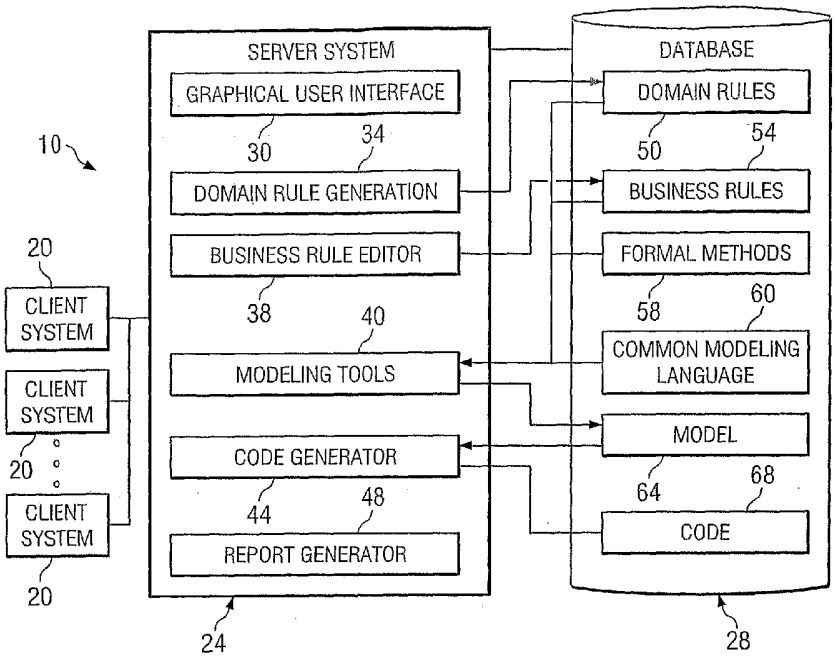
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for all designations*

[Continued on next page]

(54) Title: DESIGNING COMPUTER PROGRAMS



(57) Abstract: Designing a computer program includes accessing invariant domain rules and displaying variable business rules. One or more business rules are selected in response to a user selection and are customized. The business rules are associated with a procedure, and the domain rules are associated with the procedure. A model representing the procedure is displayed, and a code corresponding to the procedure is generated to design a computer program.

WO 2005/010749 A2

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DESIGNING COMPUTER PROGRAMS

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to the field of computers and more specifically to designing computer programs.

BACKGROUND OF THE INVENTION

Designing a computer program may involve generating a set of domain rules that set out the requirements for the computer program. Known techniques for designing computer programs typically involve generating a limited set of domain rules. A limited set of domain rules, however, may result in inefficient and ineffective computer program design. Consequently, know techniques for designing computer programs may be unsatisfactory in certain situations.

SUMMARY OF THE INVENTION

In accordance with the present invention, disadvantages and problems associated with previous techniques for designing computer programs may be reduced or eliminated.

According to one embodiment of the present invention, designing a computer program includes accessing invariant domain rules and displaying variable business rules. One or more business rules are selected in response to a user selection and are customized. The business rules are associated with a procedure, and the domain rules are associated with the procedure. A model representing the procedure is displayed, and a code corresponding to the procedure is generated to design a computer program.

According to another embodiment of the present invention, managing rules for designing a computer program includes accessing a number of rules. The rules are analyzed to separate domain rules from business rules, where each domain rule is invariant and each business rule is variable. The business rules are stored. A business rule is provided from the stored business rules in response to a request for the business rule.

Certain embodiments of the invention may provide one or more technical advantages. A technical advantage of one embodiment may be that a complete set of domain rules is determined prior to generation of the final code. The complete set may be used to define a problem space, providing for more efficient and effective design of a computer program.

Certain embodiments of the invention may include none, some, or all of the above technical advantages. One or more other technical advantages may be readily apparent to one skilled in the art from the figures, descriptions, and claims included herein.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and its features and advantages, reference is now made to the following description, taken in conjunction with the accompanying drawings, in which:

FIGURE 1 is a block diagram illustrating one embodiment of a system for designing a computer program;

FIGURE 2 is a flowchart illustrating one embodiment of a method for designing a computer program;

FIGURE 3 is a diagram illustrating one embodiment of a relationship between high-level artifacts stored at database;

FIGURE 4 is a diagram illustrating one embodiment of a structural view;

FIGURE 5 is a diagram of one embodiment of a behavioral view; and

FIGURE 6 is a flowchart illustrating one embodiment of a method for generating domain rules.

DETAILED DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention and its advantages are best understood by referring to FIGURES 1 through 6 of the drawings, like numerals being used for like and corresponding parts of the various drawings. As used in this document, "each" refers to each member of a set or each member of a subset of a set.

FIGURE 1 is a block diagram illustrating one embodiment of a system 10 for designing a computer program. In general, system 10 is involved in the generation of a substantially complete set of domain rules to define the problem space of the computer program design. Invariant domain rules and customizable business rules are used to design the program.

According to the illustrated embodiment, system 10 includes one or more client systems 20, a server system 24, and a database 28 coupled as shown in FIGURE 1. A client system 20 allows a user to communicate with server system 24 to design computer programs. Server system 24 manages applications for designing computer programs, such

as a graphical user interface 30, a domain rule generation module 34, a business rule editor 38, modeling tools 40, a code generator 44, and a report generator 48. Database 28 stores data that may be used by server system 24. Database 28 may include, for example, domain rules 50, business rules 54, formal methods 58, a common modeling language 60, a model 64, and code 68.

In operation, domain rule generation module 34 generates a substantially complete set of domain rules 50 that includes invariant rules that may be used to define a domain. Business rule editor 38 is used to customize business rules 54 for a particular computer program. Modeling tools 40 may use domain rules 50, business rules 54, and formal methods 58, which may be expressed according to common modeling language 60, in order to generate model 64. A user may manipulate model 64 to design the computer program. Code generator 44 generates code 68 according to model 64, and may also operate to check code 68 and model 64 for syntax compliance and for consistency. Report generator 48 may be used to generate a report describing the computer program.

According to one embodiment, system 10 may operate to design a computer program using object-oriented technology. According to object-oriented technology, a computer program may be viewed as a collection of discreet objects representing entities that are self-contained collections of data structures and routines that interact with other objects. Object-oriented programming involves both analysis and design. Object-oriented analysis identifies component objects and system requirements, and describes how the component objects and system requirements interact to perform specific tasks. Typically, analysis attempts to reuse existing solutions. Design involves creating a computer program in which the objects may be efficiently adapted to perform the tasks.

According to the illustrated embodiment, client systems 20 may allow one or more users to concurrently design a computer program. Users may include designers, stakeholders, or any other person or identifier identifying a person. Stakeholders may include engineers from any of a number of fields such as the network, hardware, software, human factors, or database fields. Graphical user interface 30 allows users of client systems 20 to access applications of server system 24.

Domain rule generation module 34 generates domain rules 50. Domain rules 50 comprise invariant rules that define a domain that may be used to determine a problem space and a solution space. A substantially complete set of domain rules may anticipate substantially all possible applications of the domain, and may provide a framework from

which substantially all solutions of the solution space may be generated. Domain rules 50 may be selected according to any suitable procedure to generate any suitable problem space, and may be generated through successive iterations of the design process.

TABLE 1 lists examples of domain rules 50 from accounting theory.

5

TABLE 1

Domain Rule	Prescriptive Description
Duality	Offset each increment to resources with a corresponding decrement, and vice versa. Characterize increments by transferring in purchases and cash receipts and the corresponding decrements by transferring out sales and cash disbursements.
Accounting Equation	Ensure Assets = Liabilities + Owner's Equity.
Income Equation	Ensure Revenues – Expenses = Net Income (or Net Loss).
Accounting Period	Ensure transaction effective date between Accounting Period End Date and Accounting Period Begin Date.
Accrual	Calculate portion of expense or revenue based on when the corresponding purchase or sale event occurred, not when cash is received or disbursed.
Realization	Recognize when the expense occurred based on when the physical item or service was received.
Matching	Match revenue that occurred in an accounting period with associated expenses.
Money Measurement	Provide a common unit of measure for all calculations by translating all measurements into monetary units.
Entity	Define the boundaries of the organization for which accounts are kept and reports are made.
Going Concern	Prepare financial reports based on the assumption that the organization will continue its current operations indefinitely, not based on current liquidation value.
Cost	Value assets based on original cost, not current value and not value adjusted for inflation or deflation.
Consistency	Do not change the accounting method for a kind of event or asset from one accounting period to the next.

Domain Rule	Prescriptive Description
Conservatism	Recognize revenues and gains slower than expenses and losses.
Materiality	Do not measure or record events that are insignificant, applying consistency and conservatism to determine significance.

TABLE 2 lists examples of domain rules 50 from military theory.

TABLE 2

Domain Rule	Prescriptive Description
Set the Objective	Direct every military mission toward a clearly defined, decisive, and attainable objective. The proper objective in battle is the destruction of the enemy's combat forces. To do this, however, subordinate commanders must be given terrain objectives toward which they move.
Take the Offensive	Seize, retain, and exploit the initiative. Offensive action is the most effective and decisive way to attain a clearly defined common objective.
Mass the Effects	Mass the effects of synchronizing the employment of overwhelming combat power at the decisive place and time to gain the objective. Achieve military superiority at the decisive place and time. Mass is generally gained by Maneuver.
Use Forces Economically	Employ all combat power available in the most effective way possible to gain the objective. Allocate to secondary efforts minimum essential combat power.
Maneuver Combat Power	Place the enemy in a position of disadvantage through the flexible application of combat power. Position your military resources to favor the accomplishment of your mission.
Use Unity of Command	Designate a single decision maker responsible for all activities related to an operation. Focus all activity upon a single objective.
Be Secure	Never permit the enemy to acquire an unexpected advantage.

Domain Rule	Prescriptive Description
Use Surprise	Strike the enemy at a time, at a place, or in a manner for which he is unprepared. Accomplish your purpose before the enemy can effectively react.
Use Simplicity	Prepare clear, uncomplicated plans and clear, concise orders to ensure thorough understanding.

Domain rule generation module 34 may identify the boundaries of a domain, and determine commonalities and variations among systems that meet the requirements. The boundaries and requirements may be defined for a domain in terms of the domain rules and functions. Functions may include related subfunctions, methods, processes, and procedures.

Business rules 54 comprise rules that may be customized to fit a particular application. Business rules may include, for example, rules of engagement, mission plans, or resource management rules. While domain rules 50 are invariant, business rules 54 are customizable. As an example, domain rules may comprise the principles of war, while business rules 54 comprise the rules of engagement.

TABLE 3 presents example business rules for accounting theory.

TABLE 3

Business Rule	Prescriptive Instructions
Financial – Compliance with budget policies	If a request results in the total encumbered dollars greater than the budget, reject the request.
Operational – Compliance with authorization policies	If the amount for a purchase order exceeds the signature authority of the purchasing agent, reject the purchase order.
Regulatory – Compliance with tax law and regulation	If an asset type specified for a subsidiary ledger account does not match the asset type for the depreciation method, reject the transaction.
Fraud – Compliance with legal and policy requirements	Select all transactions for a specified subsidiary ledger account for a specified time period exceeding a specified dollar amount, then process the details of those transactions to detect patterns of fraudulent activity.

TABLE 4 presents example business rules for military theory.

TABLE 4

Rules of Engagement	Prescriptive Instructions
Use armed force as the last resort	When possible, the enemy will be warned first and allowed to surrender. Armed civilians will be engaged only in self-defense. Civilian aircraft will not be engaged without approval from above division level unless it is in self-defense.
Avoiding harming civilians unless necessary to save US lives	If possible, arrange for the evacuation of civilians prior to any US attack. If civilians are in the area, do not use artillery, mortars, armed helicopters, AC-130s, tube- or rocket-launched weapons, or M551 main guns against known or suspected targets without the permission of a ground maneuver commander, LTC or higher. If civilians are in the area, all air attacks must be controlled by a FAC or FO. If civilians are in the area, close air support (CAS), white phosphorus, and incendiary weapons are prohibited without approval from above division level. If civilians are in the area, do not shoot except at known enemy locations. If civilians are not in the area, you can shoot at suspected enemy locations.
Avoid harming civilian property unless necessary to save US lives	Public works such as power stations, water treatment plants, dams, or other utilities may not be engaged without approval from above division level. Hospitals, churches, shrines, schools, museums, or other historical or cultural site will not be engaged except in self-defense.
Treat all civilians and their property with respect and dignity	Before using privately owned property, check to see if any publicly owned property can be used. No requisitioning of civilian property without permission of a company-level commander and without giving a receipt. If an ordering officer can contract for the property, then do not requisition it.

	No looting. Do not kick down doors unless necessary. Do not sleep in their houses. If you must sleep in a privately owned building, have an ordering officer contract for it.
Control civilians engaged in looting	Senior person in charge may order warning shots. Use minimum force, but not deadly force, to detain looters. Defend Panamanian (and other) lives with minimum force including deadly force when necessary.

Business rules 54 may be maintained at database 28 and customized by business rules editor 38. As an example, business rules 54 may be stored in a table, and a user may define a specific business rule 54 by revising the table. Business rule editor 38 may be used to perform security audits on business rules 54, analyze business rules 54, check new rules before adding them to database 28, and apply business rules 54 to decision support tools.

Formal methods 58 are rules that may be specified using a formal syntax such as first or second order logic or set theory. Graphical user interface 30 may be used to represent formal methods using a visual model.

Modeling tools 40 generate model 64 that represents the computer program under design, and may include, for example, nodes representing objects with operations performed by the computer program and branches representing relations among the objects. Code 68 comprises the code that executes the computer program represented by model 64.

Modeling tools 40 may comprise, for example, modeling tools provided by RATIONAL SOFTWARE such as RATIONAL ROSE REAL-TIME (RRT) modeling and code generation tool. Modeling tools 40 may also include tools that deal with requirements management, configuration management, testing, performance optimization, and documentation.

Model 64 may be actively linked to code 68 such that modeling tools 40 may provide dynamic views of model 64 to aid in the design of the computer program. For example, as code 68 is being run, a node of model 64 corresponding to code 68 that is being run may be highlighted, for example, the node may be displayed in a green color. As another example, if an inconsistency is found in code 68, a node of model 64

corresponding to code 68 having the inconsistency may be highlighted. For example, the node may be displayed in a red color. Visual indicators provided as the code executes may allow for visual verification and validation of code 68. Visual verification and validation used in conjunction with publish-and-subscribe interfaces may provide for assurance of preserving interoperability.

Different models 64 may present different views of the computer program. Models 64 may include, for example, a domain model that establishes the context of the program, a business model that establishes an abstraction of an organization associated with the program, a use case model that establishes the program's functional and non-functional requirements, and an analysis model that establishes a conceptual design of the program. The models may be actively linked with each other to reflect aspects of each other. For example, domain model may be actively linked with a lower-level model, such that the lower-level model requirements reflect the requirements of the domain model. Examples of views are described in more detail with reference to FIGURES 3 through 5.

Domain rules 50, business rules 54, and formal methods 58 may be expressed according to common modeling language 60, which provides for a common representation for data used by system 10. Common modeling language 60 may comprise, for example, the Unified Modeling Language (UML) supported by OBJECT MANAGEMENT GROUP.

Common modeling language 60 may be used to represent artifacts of the program design from semantic broadly-stated requirements through syntactic operating or executing components, and may be used to express artifacts from various stages of program design. Stages may include the early stages of design, for example, a request to automate an operation or to change an existing automated system, which are typically expressed as narrative descriptions. Subsequent phases such as concept exploration and definition, requirements analysis, program design and verification, and software coding and testing may also be expressed using common modeling language 60. Common modeling language 60 provides for artifacts that are understandable to users at any stage of design. Accordingly, users may determine whether the requirements have been captured by the program, and inconsistencies between stages may be more effectively resolved.

Code generator 44 in conjunction with modeling tools 40 may be used to iteratively generate code 68 for a computer program. Modeling tools 40 may be used to generate model 64 from which code 68 is generated at an iteration. Model 64 may be

modified and new code 68 may be generated at successive iterations. At each iteration, detail may be added or requirements may be adjusted. Each iteration generates executable code 68 that may be tested in order to provide early views of the program, which may be used to confirm the proper operation of the program. Early feedback may serve to reduce risks by identifying problems early in the process.

Code generator 44 may include a debugger that may be used to check the syntax of code 68 and may also be used to detect logical inconsistencies between model 64 and code 68. Debugger may also be used to check whether code 68 correctly implements model 64 and satisfies formal statements.

Client system 20 and server system 24 may each operate on one or more computers and may include appropriate input devices, output devices, mass storage media, processors, memory, or other components for receiving, processing, storing, and communicating information according to the operation of system 10. As used in this document, the term "computer" refers to any suitable device operable to accept input, process the input according to predefined rules, and produce output, for example, a personal computer, work station, network computer, wireless telephone, personal digital assistant, one or more microprocessors within these or other devices, or any other suitable processing device.

Client system 20 and server system 24 may be integrated or separated according to particular needs. For example, the present invention contemplates the functions of both client system 20 and server system 24 being provided using a single computer system, for example, a single personal computer. If client system 20 and server system 24 are separate, client system 20 may be coupled to server system 24 using one or more local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), a global computer network such as the Internet, or any other appropriate wire line, wireless, or other links.

Database 28 may be local to or remote from server system 24, and may be coupled to server system 24 using one or more local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), a global computer network such as the Internet, or any other appropriate wire line, wireless, or other links. Artifacts of database 28 may be actively linked in order to allow for more efficient generation of products from the artifacts. The active links may be used to integrate analysis, design, and implementation of computer programs.

Modifications, additions, or omissions may be made to the system without departing from the scope of the invention. Moreover, the operation of the system may be performed by more or fewer modules. For example, the operation of modeling tools 40 and code generator 44 may be performed by one module, or the operation of modeling tools 40 may be performed by more than one module. Additionally, functions may be performed using any suitable logic comprising software, hardware, other logic, or any suitable combination of the preceding.

FIGURE 2 is a flowchart illustrating one embodiment of a method for designing a computer program. The method begins at step 70, where model 64 is displayed to a user. Domain rules 50 are accessed at step 71. The model represents a computer program designed by the user. Business rules 54 are displayed to the user at step 72. A business rule 54 is selected at step 74 in response to a selection by the user. Business rule 54 is customized at step 76 in response to selections by the user.

Business rule 54 is associated with model 64 at step 78. If a next business rule is to be selected at step 80, the method returns to step 74, where the next business rule is selected. If no next business rule is to be selected at step 80, the method proceeds to step 84. Code 68 is generated from model 64 at step 86. Modifications may be performed to model 64 during each iteration.

If there is a next iteration at step 88, the method returns to step 70, where business rules 54 are displayed. If there is no next iteration at step 88, the method proceeds to step 90, where the results are reported. The results may include code 68 generated from the finalized model 64. After reporting the results, the method terminates.

Modifications, additions, or omissions may be made to the method without departing from the scope of the invention. Additionally, steps may be performed in any suitable order without departing from the scope of the invention.

FIGURE 3 is a diagram 100 illustrating the relationship between high-level artifacts stored at database 28. Diagram 100 may be displayed at client system 20. Folders 102 represent packages that may be used to collect elements including other packages in order to organize the development project. Arrows 104 represent a relationship among packages. For example, arrows 104 may represent cross-referencing among packages. Notes 106 represent information used during the development project. Lines 105 represent the connection to a folder to which notes 106 apply, and may be used to represent the association between business rules and a model.

According to the illustrated embodiment, model elements database folder 102a is related to a use cases folder 102b, a behavioral views folder 102c, a deployed system folder 102d, an operating or executing components folder 102e, and a structural view folder 102f. Deployed system folder 102d references operating or executing components 102e, which in turn references behavioral views folder 102c, which references use cases folder 102b.

A sources of requirements note 106a supplies information to a requirements identification methodology and a requirements manager 106b, which in turn supplies information to use cases folder 102b. Integrated documentation and configuration management tools 106c supplies abilities and information to model elements database 102a, and integrated target platform tools 106d supplies information and abilities to operating or executing components 102e.

FIGURES 4 and 5 illustrates example views that may be presented by modeling tools 40. FIGURE 4 is a diagram illustrating one embodiment of a structural view 120. FIGURE 5 is a diagram of one embodiment of a behavioral view 140. Views 120 and 140 are different views of the same program. As an example, derived active class 2 of structural view 120 corresponds to the instance of derived active class 2 of behavioral view 140.

FIGURE 6 is a flowchart illustrating one embodiment of a method for generating domain rules. The method generates domain rules by capturing the domain rules in use cases and analyzing the domain rules in use case realizations. Domain rules may be generated for domains such as accounting information systems and command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) systems.

Domain rules are collected at step 200. An example of a domain rule for the accounting information system may include ensuring that debits and credits are balanced, and a domain rule for the C4ISR system may include place the enemy in a position of disadvantage through the flexible application of combat power. The domain rules may be gathered according to the Rational Unified Process (RUP) and the Unified Development Process (USDP).

Requirements are extracted from supplemental sources at step 202. Supplemental sources may include, for example: explicit definitions of the domain; traditional subdomains, functions, methods, processes, and procedures; and established processing

cycles, business processes, or patterns. Explicit definitions of the domain may include, for example, natural or legislated laws, standards or regulations promulgated by professional organizations, or works by widely recognized researchers or practitioners. A domain may be divided into subdomains in accordance with the functions associated with the domain.

Processing cycles and patterns suggest what the function should accomplish and how the functions and the components within them should interact. TABLE 5 lists examples of processing cycles for an accounting information system.

TABLE 5

Cycle Name	Description
Expenditures and cash disbursements	Purchase requisitions, purchase orders, supplier or vendor invoices, receiving reports, written checks.
Sales and cash receipts	Customer orders, bill of lading invoices, customer checks, remittance advices.
Payroll	Time cards, paychecks.
Production	Materials requisition, labor time cards, production order, operations list.
Facilities	Documents supporting the purchase of property, plants, and equipment.
General Ledger	Adjusting, closing, and correcting entries and input from various feeder cycles, for example, expenditure and sales.
Financing	Capital raising, for example, bank notes, bond agreements, common stock issuances.
Investment	Stocks, bonds, CDs, repurchase agreements.

TABLE 6 lists examples of patterns for the C4ISR system.

TABLE 6

Name	Description
Plan	Translation of higher Commander's intent into specific Courses Of Action (COAs) in a compressed plan cycle for preparation and execution by subordinate elements. Define battle space areas of operation for synchronization and control. Generate alternate courses of action and evaluate against most likely and dangerous adversary actions. Develop synchronized schedule of tasks and activities for subordinates to prepare and execute. Develop integrated, combined effect operations plan to include all the battlefield functional areas.
Prepare	Activities by the unit before execution to improve its ability to conduct the planned operation, including plan refinement, force protection, rehearsal, reconnaissance, integration and coordination of warriors and resources, inspections, and movement to planned locations.
Execute	Apply combat power to accomplish the planned mission, exercise control through assessment of battle space to understand the situation in order to make execution and adjustment decisions for battle management.
Assess	Monitor and evaluate the current situation and progress of an operation on a continual basis throughout planning, preparation and execution. Evaluate against criteria of success to make decisions and adjustments.

The requirements are allocated to use cases at step 204. The requirements may include functional and non-functional domain rules. Functional domain rules comprise rules that are implemented by a specific function, and non-functional domain rules comprise system properties that are not directly implemented by a specific function. Non-functional domain rules may be allocated to use cases along with other non-functional requirements for realization, and may be allocated to use cases that they affect. For example, a performance requirement may be allocated to a use case that has functional requirements that would affect the performance criteria. If the affected use cases are subsequently realized, the non-functional requirements may be allocated to the analysis classes as tagged values, constraints, or documentation notations.

Requirements are traced to use cases at step 206 to determine if the domain rules have been allocated to appropriate use cases. The use cases are realized through collaborations at step 210 in order to identify any gaps at the next level. If capabilities from the supplemental sources seem to go beyond the domain rules, it may be determined whether implicit domain rules are imbedded in the capabilities or if the capabilities are unnecessary, for example, they lie outside the domain, they are redundant, or they are obsolete.

Use cases may be modified or added at step 208. The addition or modification may occur at this iteration or subsequent iterations. Use cases are realized at step 210 by identifying analysis classes and creating collaborations. Use cases may be realized for some requirements before other requirements. For example, use cases may be realized for requirements related to requests for initiating the development of the program. These requirements may be solution-oriented, and may tend to focus on a specific application. The other requirements, however, may be considered in order to complete the problem space. Domain rules are allocated to analysis classes at step 212.

Commonalties of analysis classes are identified at step 214. Commonalties are identified by determining the analysis classes that appear in multiple use cases. Stable variability of the analysis classes are identified at step 216 may be identified through subsequent analysis and design of the common classes. Business rules that capture the volatile variability of the program are determined at step 218. After determining the business rules, the method terminates.

Modifications, additions, or omissions may be made to the method without departing from the scope of the invention. Additionally, steps may be performed in any suitable order without departing from the scope of the invention.

Certain embodiments of the invention may provide one or more technical advantages. A technical advantage of one embodiment may be that a complete set of domain rules is determined prior to generation of the final code. The complete set may be used to define a problem space, providing for more efficient and effective design of a computer program.

Although an embodiment of the invention and its advantages are described in detail, a person skilled in the art could make various alterations, additions, and omissions without departing from the spirit and scope of the present invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for designing a computer program, comprising:

accessing a plurality of domain rules, each domain rule being invariant;

displaying a plurality of business rules, each business rule being variable;

selecting one or more business rules of the plurality of business rules in response to a user selection;

customizing the one or more business rules;

associating the one or more business rules with a procedure;

associating the domain rules with the procedure;

displaying a model representing the procedure; and

generating a code corresponding to the procedure in order to design a computer program.

2. The method of Claim 1, further comprising:

collecting the domain rules and the business rules;

allocating the domain rules and the business rules to a plurality of use cases;

realizing the use cases; and

assessing the domain rules and the business rules in accordance with the realization.

3. The method of Claim 1, further comprising:

checking a syntax of the code; and

providing a notification if a syntax error is detected.

4. The method of Claim 1, further comprising:

checking a logical consistency of the code; and

providing a notification if a logical inconsistency is detected.

5. The method of Claim 1, further comprising:

checking a compatibility between the model and the code; and

providing a notification if an inconsistency is detected.

6. The method of Claim 1, wherein the model is expressed according to a common modeling language.

7. Logic for designing a computer program, the logic embodied in a medium and operable to:

access a plurality of domain rules, each domain rule being invariant;

display a plurality of business rules, each business rule being variable;

select one or more business rules of the plurality of business rules in response to a user selection;

customize the one or more business rules;

associate the one or more business rules with a procedure;

associate the domain rules with the procedure;

display a model representing the procedure; and

generate a code corresponding to the procedure in order to design a computer program.

8. The logic of Claim 7, further operable to:

collect the domain rules and the business rules;

allocate the domain rules and the business rules to a plurality of use cases;

realize the use cases; and

assess the domain rules and the business rules in accordance with the realization.

9. The logic of Claim 7, further operable to:

check a syntax of the code; and

provide a notification if a syntax error is detected.

10. The logic of Claim 7, further operable to:

check a logical consistency of the code; and

provide a notification if a logical inconsistency is detected.

11. The logic of Claim 7, further operable to:

check a compatibility between the model and the code; and

provide a notification if an inconsistency is detected.

12. The logic of Claim 7, wherein the model is expressed according to a common modeling language.

13. A system for designing a computer program, comprising:

a database operable to store a plurality of domain rules, each domain rule being invariant; and

a server coupled to the database and operable to:

display a plurality of business rules, each business rule being variable;

select one or more business rules of the plurality of business rules in response to a user selection;

customize the one or more business rules;

associate the one or more business rules with a procedure;

associate the domain rules with the procedure;

display a model representing the procedure; and

generate a code corresponding to the procedure in order to design a computer program.

14. The system of Claim 13, the server further operable to:

collect the domain rules and the business rules;

allocate the domain rules and the business rules to a plurality of use cases;

realize the use cases; and

assess the domain rules and the business rules in accordance with the realization.

15. The system of Claim 13, the server further operable to:

check a syntax of the code; and

provide a notification if a syntax error is detected.

16. The system of Claim 13, the server further operable to:

check a logical consistency of the code; and

provide a notification if a logical inconsistency is detected.

17. The system of Claim 13, the server further operable to:

check a compatibility between the model and the code; and

provide a notification if an inconsistency is detected.

18. The system of Claim 13, wherein the model is expressed according to a common modeling language.

19. A system for designing a computer program, comprising:

means for accessing a plurality of domain rules, each domain rule being invariant;

means for displaying a plurality of business rules, each business rule being variable;

means for selecting one or more business rules of the plurality of business rules in response to a user selection;

means for customizing the one or more business rules;

means for associating the one or more business rules with a procedure;

means for associating the domain rules with the procedure;

means for displaying a model representing the procedure; and

means for generating a code corresponding to the procedure in order to design a computer program.

20. A method for designing a computer program, comprising:

collecting a plurality of domain rules, allocating the domain rules to a plurality of use cases, realizing the use cases, assessing the domain rules in accordance with the realization, and accessing the domain rules, each domain rule being invariant;

displaying a plurality of business rules, each business rule being variable;

selecting one or more business rules of the plurality of business rules in response to a user selection;

customizing the one or more business rules;

associating the one or more business rules with a procedure;

associating the domain rules with the procedure;

displaying a model representing the procedure, the model expressed according to a common modeling language;

generating a code corresponding to the procedure in order to design a computer program;

checking a syntax of the code, and providing a notification if a syntax error is detected;

checking a logical consistency of the code, and providing a notification if a logical inconsistency is detected; and

checking a compatibility between the model and the code, and providing a notification if an inconsistency is detected.

21. A method for managing rules for designing a computer program, comprising:

accessing a plurality of rules;

analyzing the rules to separate a plurality of domain rules from a plurality of business rules, each domain rule being invariant, each business rule being variable;

storing the business rules; and

providing a business rule from the stored business rules in response to a request for the business rule.

22. The method of Claim 21, further comprising:

customizing the provided business rule;

associating the customized business rule with a procedure; and

generating a code corresponding to the procedure in order to design a computer program.

23. The method of Claim 21, further comprising:

associating the domain rules with a procedure; and

generating a code corresponding to the procedure in order to design a computer program.

24. The method of Claim 21, further comprising:

allocating the domain rules and the business rules to a plurality of use cases;

realizing the use cases; and

assessing the domain rules and the business rules in accordance with the realization.

25. A system for managing rules for designing a computer program, comprising:

a database operable to store a plurality of rules; and

a server coupled to the database and operable to:

analyze the rules to separate a plurality of domain rules from a plurality of business rules, each domain rule being invariant, each business rule being variable;

store the business rules; and

provide a business rule from the stored business rules in response to a request for the business rule.

26. The system of Claim 25, wherein the server is further operable to:

customize the provided business rule;

associate the customized business rule with a procedure; and

generate a code corresponding to the procedure in order to design a computer program.

27. The system of Claim 25, wherein the server is further operable to:

associate the domain rules with a procedure; and

generate a code corresponding to the procedure in order to design a computer program.

28. The system of Claim 25, wherein the server is further operable to:

allocate the domain rules and the business rules to a plurality of use cases;

realize the use cases; and

assess the domain rules and the business rules in accordance with the realization.

29. Logic for managing rules for designing a computer program, the logic embodied in a medium and operable to:

access a plurality of rules;

analyze the rules to separate a plurality of domain rules from a plurality of business rules, each domain rule being invariant, each business rule being variable;

store the business rules; and

provide a business rule from the stored business rules in response to a request for the business rule.

30. The logic of Claim 29, further operable to:

customize the provided business rule;

associate the customized business rule with a procedure; and

generate a code corresponding to the procedure in order to design a computer program.

31. The logic of Claim 29, further operable to:

associate the domain rules with a procedure; and

generate a code corresponding to the procedure in order to design a computer program.

32. The logic of Claim 29, further operable to:

allocate the domain rules and the business rules to a plurality of use cases;

realize the use cases; and

assess the domain rules and the business rules in accordance with the realization.

33. A system for managing rules for designing a computer program, comprising:

means for accessing a plurality of rules;

means for analyzing the rules to separate a plurality of domain rules from a plurality of business rules, each domain rule being invariant, each business rule being variable;

means for storing the business rules; and

means for providing a business rule from the stored business rules in response to a request for the business rule.

34. A method for managing rules for designing a computer program, comprising:

accessing a plurality of rules;

analyzing the rules to separate a plurality of domain rules from a plurality of business rules, each domain rule being invariant, each business rule being variable;

allocating the domain rules and the business rules to a plurality of use cases;

realizing the use cases;

assessing the domain rules and the business rules in accordance with the realization;

storing the business rules;

providing a business rule from the stored business rules in response to a request for the business rule;

customizing the provided business rule;

associating the customized business rule with a procedure;

associating the domain rules with the procedure; and

generating a code corresponding to the procedure in order to design a computer program.

35. A method for initiating display of a view of a computer program design, comprising:

accessing a plurality of artifacts of a computer program design, each artifact of the plurality of artifacts expressed using a common modeling language;

receiving a selection of a first view from a plurality of views, each view of the plurality of views associated with a display of a subset of the plurality of artifacts;

organizing a first subset of artifacts for display according to the first view, the first subset comprising a particular artifact;

initiating display of the first subset of artifacts according to the first view;

receiving a selection of a second view from the plurality of views;

organizing a second subset of artifacts for display according to the second view, the second subset comprising the particular artifact; and

initiating display of the second subset of artifacts according to the second view.

36. The method of Claim 35, wherein an artifact of the plurality of artifacts comprises a requirement of the computer program design.

37. The method of Claim 35, wherein:

receiving the selection of the first view further comprises receiving a selection of a high-level artifact view; and

organizing the first subset of artifacts further comprises organizing a plurality of high-level artifacts of the plurality of artifacts according to the high-level artifact view.

38. The method of Claim 35, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a structural view; and

the second view comprises the structural view.

39. The method of Claim 35, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a behavioral view; and

the second view comprises the behavioral view.

40. The method of Claim 35, wherein:

the first view comprises a structural view, the structural view comprising an active class; and

the second view comprises a behavioral view, the behavioral view comprising the active class.

41. A system for initiating display of a view of a computer program design, comprising:

a database operable to store a plurality of artifacts of a computer program design, each artifact of the plurality of artifacts expressed using a common modeling language; and

a server coupled to the database and operable to:

receive a selection of a first view from a plurality of views, each view of the plurality of views associated with a display of a subset of the plurality of artifacts;

organize a first subset of artifacts for display according to the first view, the first subset comprising a particular artifact;

initiate display of the first subset of artifacts according to the first view;

receive a selection of a second view from the plurality of views;

organize a second subset of artifacts for display according to the second view, the second subset comprising the particular artifact; and

initiate display of the second subset of artifacts according to the second view.

42. The system of Claim 41, wherein an artifact of the plurality of artifacts comprises a requirement of the computer program design.

43. The system of Claim 41, wherein the server is further operable to:

receive the selection of the first view by receiving a selection of a high-level artifact view; and

organize the first subset of artifacts by organizing a plurality of high-level artifacts of the plurality of artifacts according to the high-level artifact view.

44. The system of Claim 41, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a structural view; and

the second view comprises the structural view.

45. The system of Claim 41, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a behavioral view; and

the second view comprises the behavioral view.

46. The system of Claim 41, wherein:

the first view comprises a structural view, the structural view comprising an active class; and

the second view comprises a behavioral view, the behavioral view comprising the active class.

47. Logic for initiating display of a view of a computer program design, the logic embodied in a medium and operable to:

access a plurality of artifacts of a computer program design, each artifact of the plurality of artifacts expressed using a common modeling language;

receive a selection of a first view from a plurality of views, each view of the plurality of views associated with a display of a subset of the plurality of artifacts;

organize a first subset of artifacts for display according to the first view, the first subset comprising a particular artifact;

initiate display of the first subset of artifacts according to the first view;

receive a selection of a second view from the plurality of views;

organize a second subset of artifacts for display according to the second view, the second subset comprising the particular artifact; and

initiate display of the second subset of artifacts according to the second view.

48. The logic of Claim 47, wherein an artifact of the plurality of artifacts comprises a requirement of the computer program design.

49. The logic of Claim 47, further operable to:

receive the selection of the first view by receiving a selection of a high-level artifact view; and

organize the first subset of artifacts by organizing a plurality of high-level artifacts of the plurality of artifacts according to the high-level artifact view.

50. The logic of Claim 47, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a structural view; and

the second view comprises the structural view.

51. The logic of Claim 47, wherein:

the first view comprises a high-level artifact view, the high-level artifact view comprising a behavioral view; and

the second view comprises the behavioral view.

52. The logic of Claim 47, wherein:

the first view comprises a structural view, the structural view comprising an active class; and

the second view comprises a behavioral view, the behavioral view comprising the active class.

53. A system for initiating display of a view of a computer program design, comprising:

means for accessing a plurality of artifacts of a computer program design, each artifact of the plurality of artifacts expressed using a common modeling language;

means for receiving a selection of a first view from a plurality of views, each view of the plurality of views associated with a display of a subset of the plurality of artifacts;

means for organizing a first subset of artifacts for display according to the first view, the first subset comprising a particular artifact;

means for initiating display of the first subset of artifacts according to the first view;

means for receiving a selection of a second view from the plurality of views;

means for organizing a second subset of artifacts for display according to the second view, the second subset comprising the particular artifact; and

means for initiating display of the second subset of artifacts according to the second view.

54. A method for initiating display of a view of a computer program design, comprising:

accessing a plurality of artifacts of a computer program design, each artifact of the plurality of artifacts expressed using a common modeling language, an artifact of the plurality of artifacts comprising a requirement of the computer program design;

receiving a selection of a first view from a plurality of views, each view of the plurality of views associated with a display of a subset of the plurality of artifacts, the first view comprising a high-level artifact view, the high-level artifact view comprising a structural view and a behavioral view;

organizing a first subset of artifacts for display according to the first view, the first subset comprising a particular artifact;

initiating display of the first subset of artifacts according to the first view;

receiving a selection of a second view from the plurality of views, the second view comprising at least one of the structural view and the behavioral view;

organizing a second subset of artifacts for display according to the second view, the second subset comprising the particular artifact; and

initiating display of the second subset of artifacts according to the second view.

FIG. 1

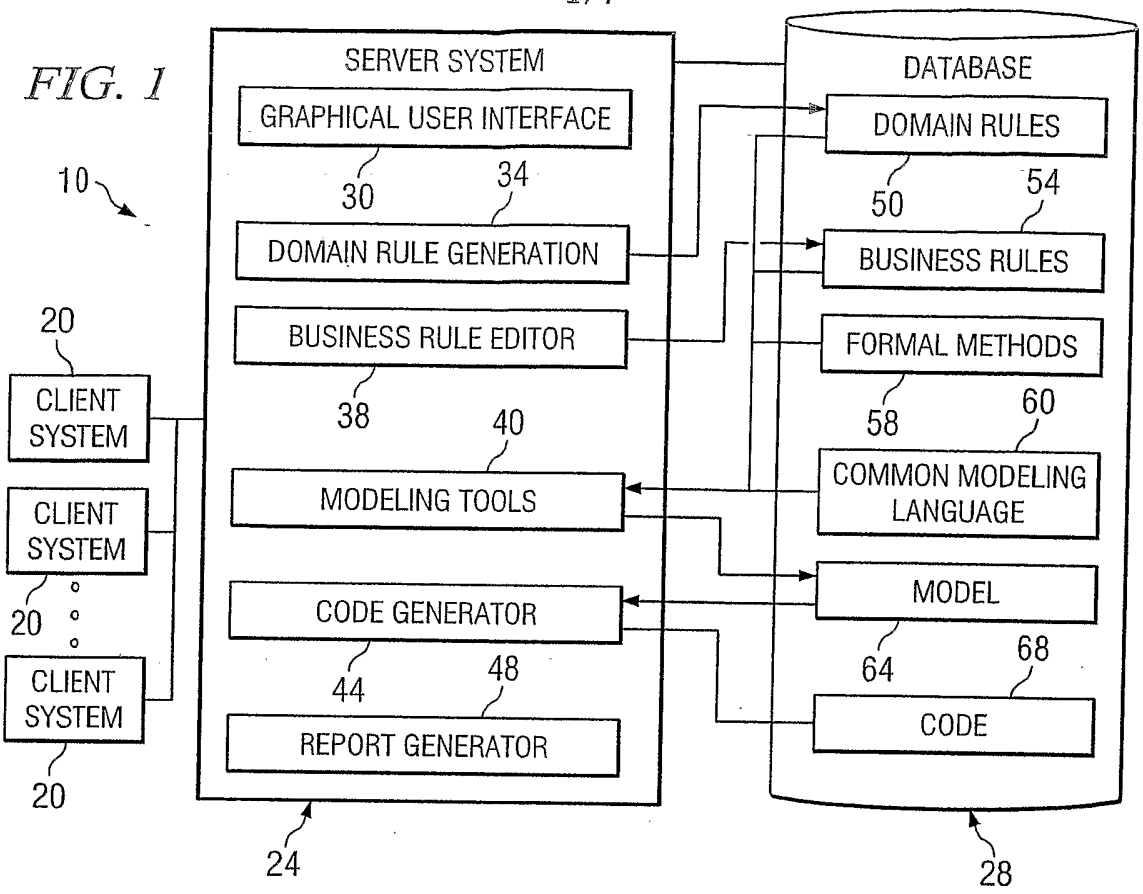

SERVER SYSTEM
GRAPHICAL USER INTERFACE
30
34
DOMAIN RULE GENERATION
BUSINESS RULE EDITOR
38
40
MODELING TOOLS
CODE GENERATOR
44
48
REPORT GENERATOR
24
DATABASE
DOMAIN RULES
50
54
BUSINESS RULES
FORMAL METHODS
58
60
COMMON MODELING LANGUAGE
MODEL
64
68
CODE
28
10
20
CLIENT SYSTEM
CLIENT SYSTEM
20
CLIENT SYSTEM
20

FIG. 2

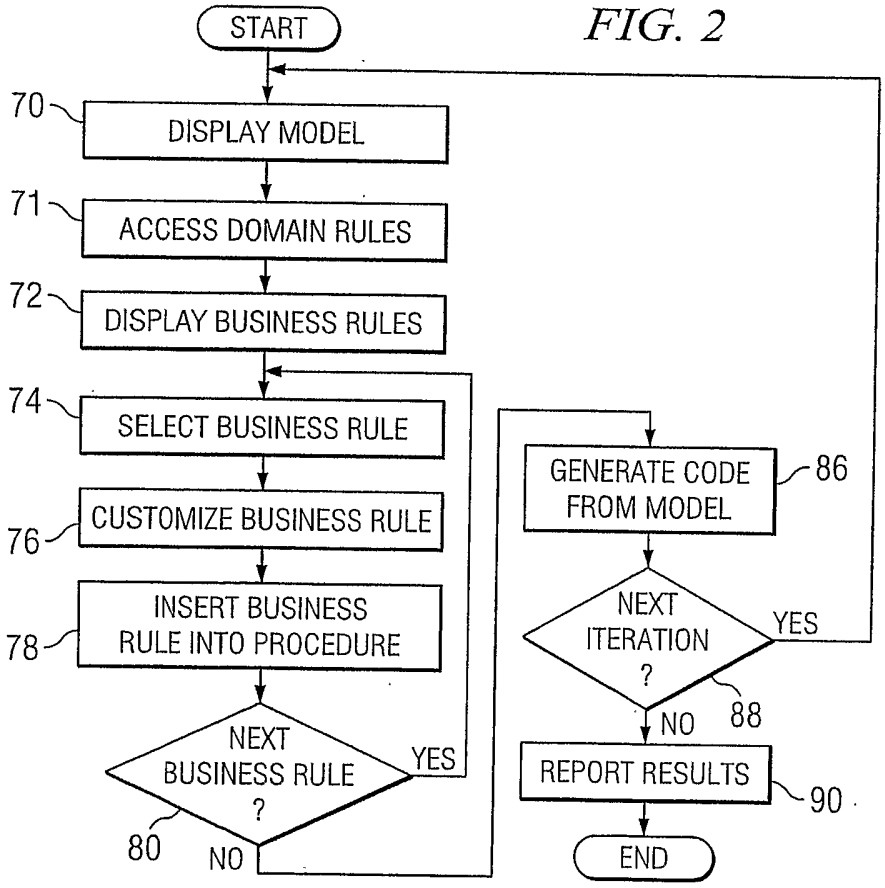
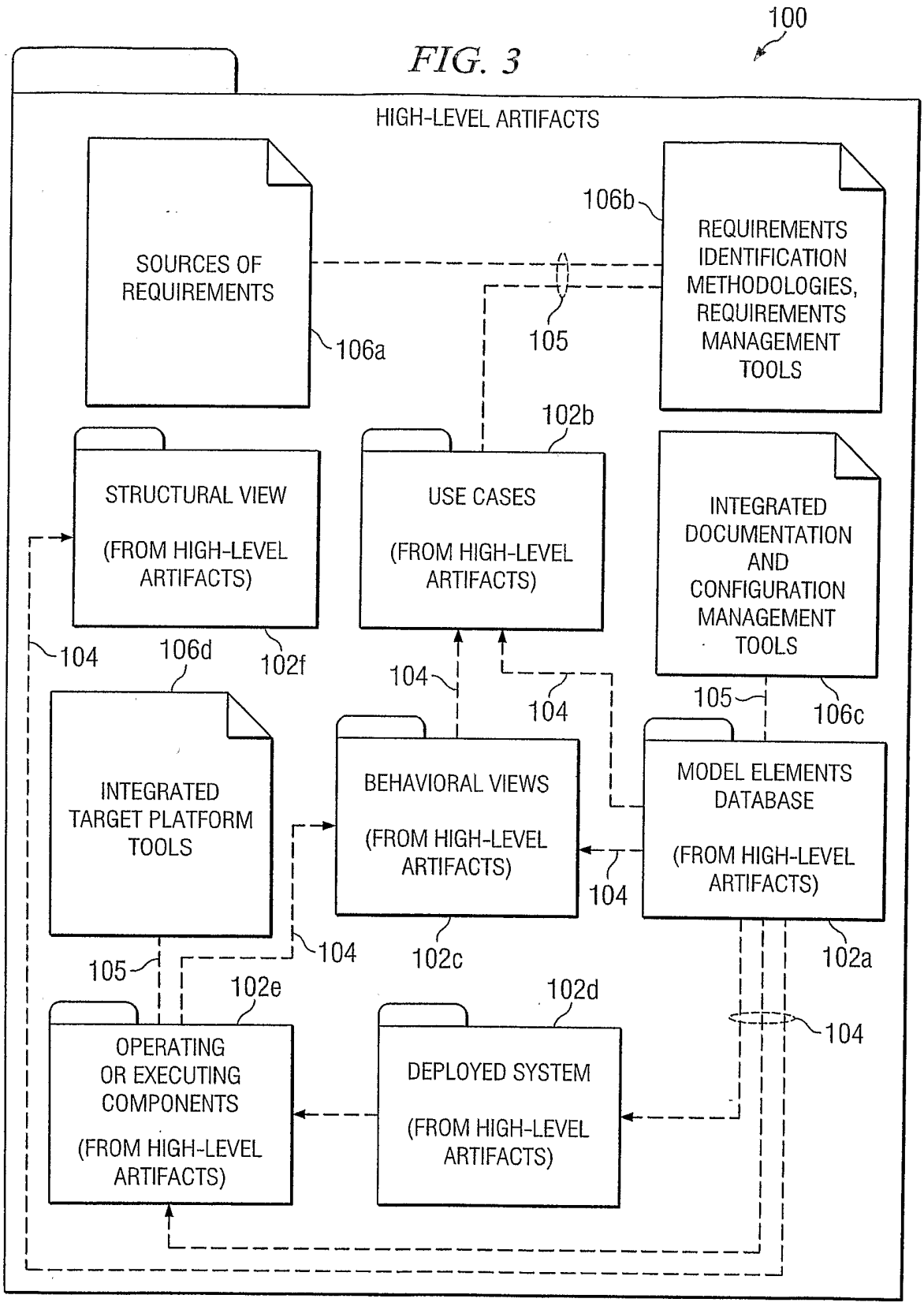

START
70
DISPLAY MODEL
71
ACCESS DOMAIN RULES
72
DISPLAY BUSINESS RULES
74
SELECT BUSINESS RULE
76
CUSTOMIZE BUSINESS RULE
78
INSERT BUSINESS RULE INTO PROCEDURE
NEXT BUSINESS RULE ?
80
YES
NO
GENERATE CODE FROM MODEL
86
NEXT ITERATION ?
88
YES
NO
REPORT RESULTS
90
END

FIG. 3


100
HIGH-LEVEL ARTIFACTS
SOURCES OF REQUIREMENTS
106a
106b
REQUIREMENTS IDENTIFICATION METHODOLOGIES, REQUIREMENTS MANAGEMENT TOOLS
105
102b
STRUCTURAL VIEW
(FROM HIGH-LEVEL ARTIFACTS)
USE CASES
(FROM HIGH-LEVEL ARTIFACTS)
INTEGRATED DOCUMENTATION AND CONFIGURATION MANAGEMENT TOOLS
104
106d
102f
104
104
105
106c
INTEGRATED TARGET PLATFORM TOOLS
BEHAVIORAL VIEWS
(FROM HIGH-LEVEL ARTIFACTS)
MODEL ELEMENTS DATABASE
(FROM HIGH-LEVEL ARTIFACTS)
104
104
105
102c
102a
102e
102d
104
OPERATING OR EXECUTING COMPONENTS
(FROM HIGH-LEVEL ARTIFACTS)
DEPLOYED SYSTEM
(FROM HIGH-LEVEL ARTIFACTS)

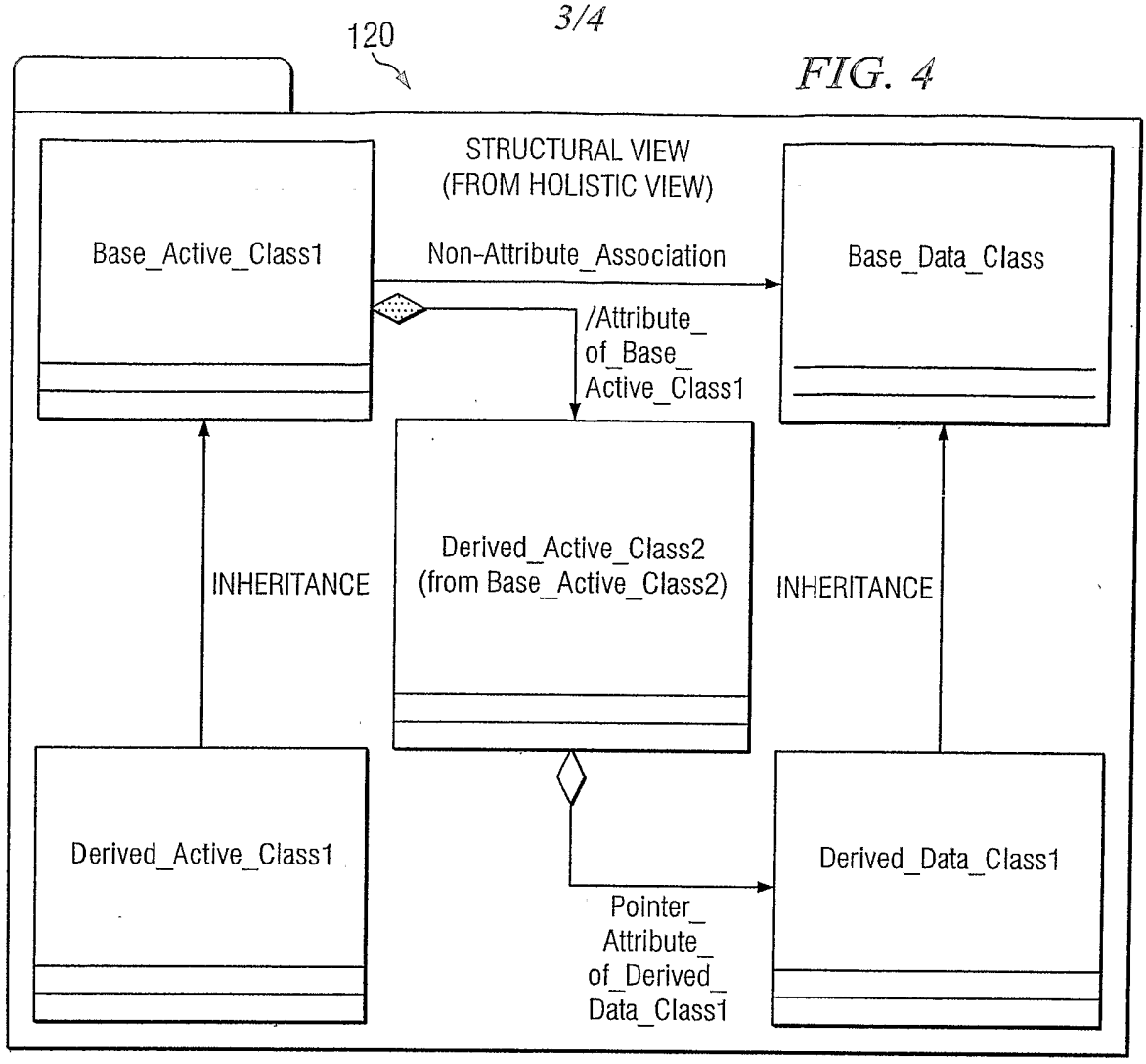

120
STRUCTURAL VIEW
(FROM HOLISTIC VIEW)
Base_Active_Class1
Non-Attribute_Association
Base_Data_Class
/Attribute_
of_Base_
Active_Class1
Derived_Active_Class2
(from Base_Active_Class2)
INHERITANCE
INHERITANCE
Derived_Active_Class1
Derived_Data_Class1
Pointer_
Attribute_
of_Derived_
Data_Class1

FIG. 4

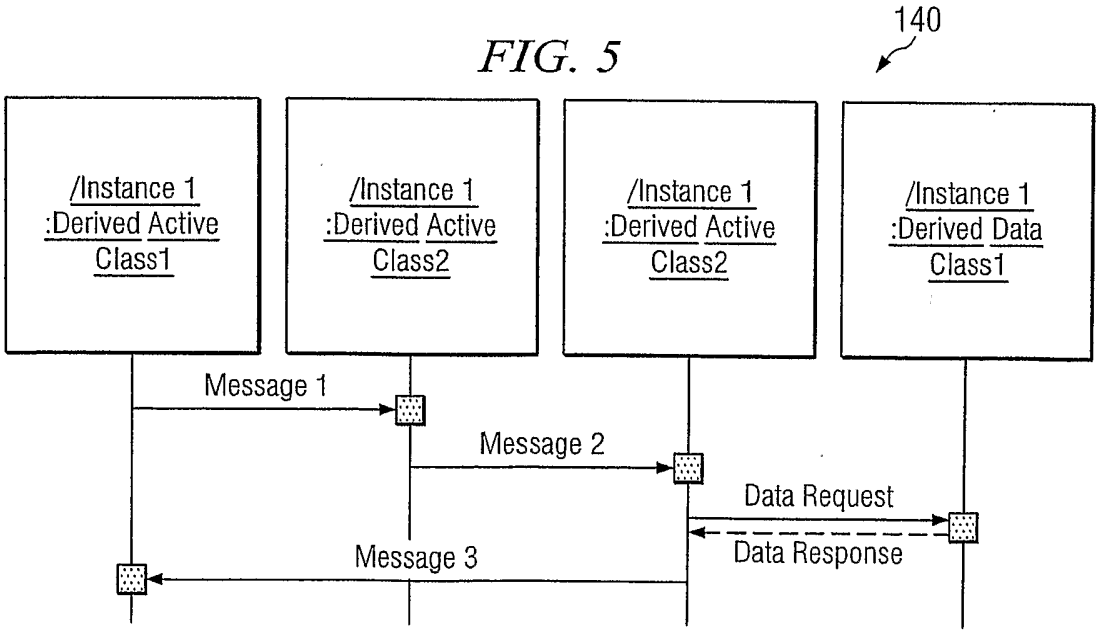
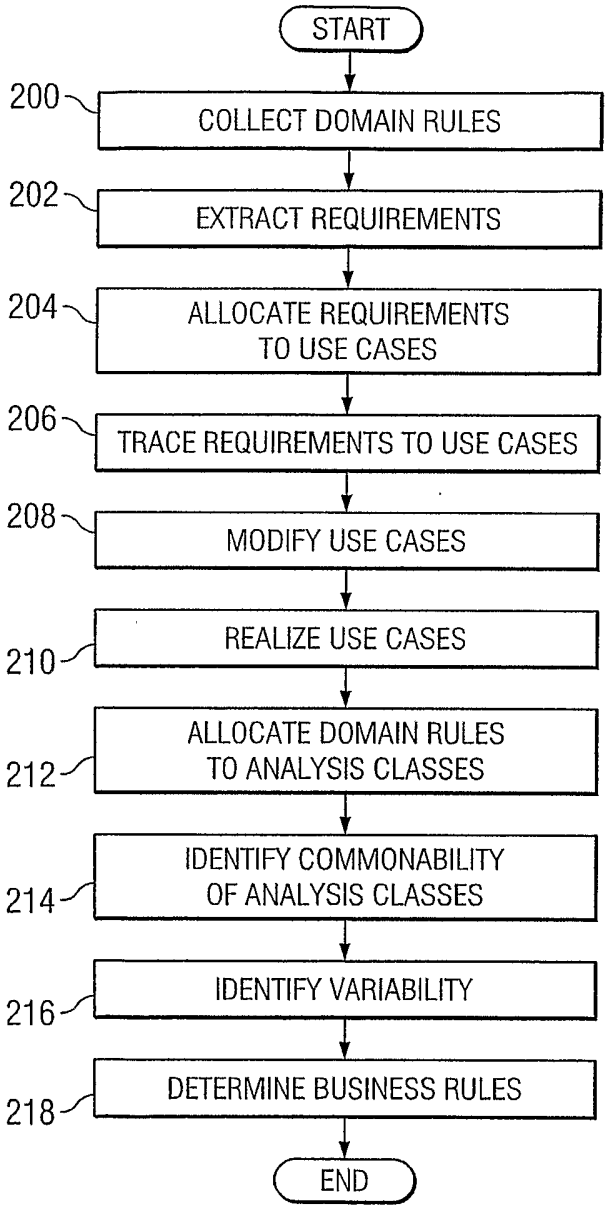

140
/Instance 1
:Derived Active
Class1
/Instance 1
:Derived Active
Class2
/Instance 1
:Derived Active
Class2
/Instance 1
:Derived Data
Class1
Message 1
Message 2
Data Request
Message 3
Data Response

FIG. 5

FIG. 6


START
200
COLLECT DOMAIN RULES
202
EXTRACT REQUIREMENTS
204
ALLOCATE REQUIREMENTS TO USE CASES
206
TRACE REQUIREMENTS TO USE CASES
208
MODIFY USE CASES
210
REALIZE USE CASES
212
ALLOCATE DOMAIN RULES TO ANALYSIS CLASSES
214
IDENTIFY COMMONABILITY OF ANALYSIS CLASSES
216
IDENTIFY VARIABILITY
218
DETERMINE BUSINESS RULES
END