(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6692289号
(P6692289)
(45)発行日　令和2年5月13日(2020.5.13)　(24)登録日　令和2年4月16日(2020.4.16)

(51)Int.Cl.　　F I
G06F 8/20 (2018.01)　　G06F 8/20
G06F 8/73 (2018.01)　　G06F 8/73

請求項の数 13　(全 22 頁)

(21)出願番号　特願2016-254283(P2016-254283)
(22)出願日　平成28年12月27日(2016.12.27)
(65)公開番号　特開2018-106556(P2018-106556A)
(43)公開日　平成30年7月5日(2018.7.5)
審査請求日　平成31年3月26日(2019.3.26)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　110000176
一色国際特許業務法人
(72)発明者　是木　玄太
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　前岡　淳
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　鹿糠　秀行
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内

最終頁に続く

(54)【発明の名称】画面情報生成装置、画面情報生成方法、及びプログラム

(57)【特許請求の範囲】
【請求項１】
ディスプレイ、メモリ、及び前記ディスプレイと前記メモリとに通信可能に接続されているプロセッサ、を備え、
前記プロセッサは、
サービス提供装置によって実行され、前記サービス提供装置がユーザからの入力を処理することを可能にするソースコードを取得し、
ユーザインタフェースの画面を定義する画面定義規則を前記メモリから読み出し、
前記ユーザインタフェースの特定の画面が前記ユーザインタフェースの他の画面にいつ遷移するかを定義する画面遷移関数規則を取得し、
前記画面定義規則により前記ソースコードを解析して特定の画面を識別し、
前記画面遷移関数規則により前記ソースコードを解析して特定の画面遷移を識別し、
識別された前記特定の画面と識別された前記特定の画面遷移とを含む画面遷移情報により画面遷移表を生成し、
前記ディスプレイに前記画面遷移表を表示し、
前記画面定義規則は、前記ソースコードに含まれているコントローラに関する記述と、前記コントローラに対応するコントローラクラスと、を特定し、
前記プロセッサが、前記コントローラクラスに基づき、画面の遷移を行うハンドラである遷移ハンドラに関する情報を取得し、
前記画面遷移表は、前記遷移ハンドラに関する情報を含む、

画面情報生成装置。
【請求項２】
請求項１に記載の画面情報生成装置であって、
前記プロセッサは、前記遷移ハンドラの遷移先となる画面の情報が前記画面遷移関数規則に含まれていない場合、当該遷移ハンドラに関する情報を前記画面遷移情報から除外する、
画面情報生成装置。
【請求項３】
請求項１に記載の画面情報生成装置であって、
前記プロセッサは、前記ソースコードを解析することにより前記遷移ハンドラが実際に使用されているか否かを判定し、前記遷移ハンドラが実際に使用されていない場合、当該遷移ハンドラに関する情報を前記画面遷移情報から除外する、
画面情報生成装置。
【請求項４】
請求項３に記載の画面情報生成装置であって、
前記プロセッサは、
前記ソースコードを解析することにより、前記遷移ハンドラが静的な画面遷移を行うハンドラであるか否かを判定し、
前記ソースコードに定義されている前記遷移ハンドラが静的な画面遷移を行うものである場合、前記遷移ハンドラの遷移先となる画面毎に当該遷移ハンドラが実際に使用されているかを判定する、
画面情報生成装置。
【請求項５】
請求項４に記載の画面情報生成装置であって、
前記プロセッサは、画面遷移関数の引数に遷移先の画面の識別子が即値で指定されている場合に前記遷移ハンドラが静的な遷移を行うハンドラであると判定する、
画面情報生成装置。
【請求項６】
請求項４に記載の画面情報生成装置であって、
前記プロセッサは、画面遷移関数をラッピングした遷移ハンドラと前記画面遷移関数規則の引数とが同じ変数を使用しており、かつ、遷移先の画面の識別子を即値で指定して前記遷移ハンドラを呼び出している場合に前記遷移ハンドラが静的な画面遷移を行うハンドラであると判定する、
画面情報生成装置。
【請求項７】
請求項３乃至６のいずれか一項に記載の画面情報生成装置であって、
前記プロセッサは、前記遷移ハンドラが実際に使用されているか否かを、前記ソースコードの画面記述文に記述されている所定のタグの属性値として前記遷移ハンドラが指定されているか否かに基づき判定する、
画面情報生成装置。
【請求項８】
請求項３乃至６のいずれか一項に記載の画面情報生成装置であって、
前記プロセッサは、前記遷移ハンドラが、同一の画面に関する単数または複数の画面記述文の複数箇所で使用されている場合、同一の遷移先の画面への遷移で使われている同一の遷移ハンドラを重複した画面遷移情報として、前記重複した画面遷移情報の出力を行わない、
画面情報生成装置。
【請求項９】
請求項１乃至３のいずれか一項に記載の画面情報生成装置であって、
前記プロセッサは、前記ソースコードを解析することにより、前記遷移ハンドラが静的

な画面遷移を行うハンドラであるか否かを判定し、
前記ソースコードに定義されている遷移ハンドラが静的な画面遷移を行わないものである場合に前記画面遷移情報を補完する、
画面情報生成装置。
【請求項１０】
請求項１乃至３のいずれか一項に記載の画面情報生成装置であって、
前記プロセッサは、前記ソースコードを解析することにより、前記遷移ハンドラが静的な画面遷移を行うハンドラであるか否かを判定し、
前記ソースコードに定義されている遷移ハンドラが静的な画面遷移を行うものであり、かつ、複数の遷移先の画面を有している場合に前記画面遷移情報を補完する、
画面情報生成装置。
【請求項１１】
請求項１乃至３のいずれか一項に記載の画面情報生成装置であって、
前記プロセッサは、前記画面遷移情報に基づき、前記ユーザインタフェースの画面の夫々の間の関係を示す情報を視覚的に示す情報である画面遷移図を生成する、
画面情報生成装置。
【請求項１２】
プロセッサとメモリを含む情報処理装置が、
サービス提供装置によって実行され、前記サービス提供装置がユーザからの入力を処理することを可能にするソースコードを取得するステップと、
ユーザインタフェースの画面を定義し、前記ソースコードに含まれているコントローラに関する記述と前記コントローラに対応するコントローラクラスとを特定する画面定義規則を前記メモリから読み出すステップと、
前記ユーザインタフェースの特定の画面が他の画面にいつ遷移するかを定義する画面遷移関数規則を取得するステップと、
前記画面定義規則により前記ソースコードを解析して特定の画面を識別するステップと、
前記画面遷移関数規則により前記ソースコードを解析して特定の画面遷移を識別するステップと、
前記コントローラクラスに基づき、画面の遷移を行うハンドラである遷移ハンドラに関する情報を取得するステップと、
識別された前記特定の画面と識別された前記特定の画面遷移とを含む画面遷移情報により、前記遷移ハンドラに関する情報を含む画面遷移表を生成するステップと、
前記画面遷移表を表示するステップと、
を実行する、画面情報生成方法。
【請求項１３】
プロセッサとメモリを含む情報処理装置に、
サービス提供装置によって実行され、前記サービス提供装置がユーザからの入力を処理することを可能にするソースコードを、取得する機能と、
ユーザインタフェースの画面を定義し、前記ソースコードに含まれているコントローラに関する記述と前記コントローラに対応するコントローラクラスとを特定する画面定義規則を前記メモリから読み出す機能と、
前記ユーザインタフェースの特定の画面が他の画面にいつ遷移するかを定義する画面遷移関数規則を取得する機能と、
前記画面定義規則により前記ソースコードを解析して特定の画面を識別する機能と、
前記画面遷移関数規則により前記ソースコードを解析して特定の画面遷移を識別する機能と、
前記コントローラクラスに基づき、画面の遷移を行うハンドラである遷移ハンドラに関する情報を取得する機能と、
識別された前記特定の画面と識別された前記特定の画面遷移とを含む画面遷移情報によ

り、前記遷移ハンドラに関する情報を含む画面遷移表を生成する機能と、
前記画面遷移表を表示する機能と、
を実現するためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、画面情報生成装置、画面情報生成方法、及びプログラムに関する。
【背景技術】
【０００２】
特許文献１には、「開発したアプリケーションに対し、開発元となる画面設計書を元に、自動でその整合性をチェックする機能を提供する。データ項目整合性摘出ツールは、画面設計書と入力項目および画面遷移を定義するソースコードを入力とし、ツールを実行することで、それぞれ同一のフォーマットであるドキュメント中間ファイルＸＭＬおよびソースコード中間ファイルＸＭＬに変換する。変換後、２つのＸＭＬを比較し、差異があるものについて、項目整合性抽出結果ファイルに出力する。」と記載されている。
【先行技術文献】
【特許文献】
【０００３】
【特許文献１】特開２００９－１２９０３８号公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】
ソフトウェアの開発や保守においてはソースコードとドキュメント（システム設計書、プログラム仕様書、画面仕様書等）の乖離がしばしば問題となる。例えば、Ｗｅｂアプリケーションの開発や保守に際しては画面まわりの仕様変更が頻繁に行われるが、ソースコードが追加／変更等された場合でもドキュメントの修正が行き届かないことが多い。そしてこのようにして生じた乖離を解消するには通常は多大な労力が必要になる。
【０００５】
特許文献１では、画面遷移を定義するソースコードに基づきソースコードと画面設計書との差異を検出している。しかし同文献の方法は画面遷移を定義するソースコードの存在を前提としており、画面遷移に関する記述がソースコードの様々な箇所に分散しているような場合には必ずしも適用することができない。
【０００６】
本発明はこうした背景に鑑みてなされたものであり、ソフトウェアの開発や保守に際して用いるドキュメントの管理を支援する、画面情報生成装置、画面情報生成方法、及びプログラムを提供することを目的としている。
【課題を解決するための手段】
【０００７】
本発明のうちの一つは、画面情報生成装置であって、ディスプレイ、メモリ、及び前記ディスプレイと前記メモリとに通信可能に接続されているプロセッサ、を備え、前記プロセッサは、サービス提供装置によって実行され、前記サービス提供装置がユーザからの入力を処理することを可能にするソースコードを取得し、ユーザインタフェースの画面を定義する画面定義規則を前記メモリから読み出し、前記ユーザインタフェースの特定の画面が前記ユーザインタフェースの他の画面にいつ遷移するかを定義する画面遷移関数規則を取得し、前記画面定義規則により前記ソースコードを解析して特定の画面を識別し、前記画面遷移関数規則により前記ソースコードを解析して特定の画面遷移を識別し、識別された前記特定の画面と識別された前記特定の画面遷移とを含む画面遷移情報により画面遷移表を生成し、前記ディスプレイに前記画面遷移表を表示し、前記画面定義規則は、前記ソースコードに含まれているコントローラに関する記述と、前記コントローラに対応するコントローラクラスと、を特定し、前記プロセッサが、前記コントローラクラスに基づき、

画面の遷移を行うハンドラである遷移ハンドラに関する情報を取得し、前記画面遷移表は、前記遷移ハンドラに関する情報を含む。

【０００８】

その他、本願が開示する課題、及びその解決方法は、発明を実施するための形態の欄、及び図面により明らかにされる。

【発明の効果】

【０００９】

本発明によれば、ソフトウェアの開発や保守に際して用いるドキュメントの管理を支援することができる。

【図面の簡単な説明】

【００１０】

【図１】情報処理システムの概略的な構成を示す図である。

【図２】情報処理装置のハードウェア構成を示す図である。

【図３】画面情報生成装置の機能及び情報記憶部が記憶するデータを示す図である。

【図４】画面定義関数表の一例である。

【図５】コントローラ登録関数表の一例である。

【図６】画面遷移関数表の一例である。

【図７】画面遷移表の一例である。

【図８】遷移ハンドラ使用確認表の一例である。

【図９】画面遷移情報生成処理を説明するフローチャートである。

【図１０】（ａ）は画面定義に関するソースコードの一例であり、（ｂ）はコントローラの登録に関するソースコードの一例である。

【図１１】遷移ハンドラの定義に関するソースコードの一例である。

【図１２】フィルタリング処理を説明するフローチャートである。

【図１３】遷移ハンドラを利用するソースコードの一例である。

【図１４】補完処理を説明するフローチャートである。

【図１５】画面遷移図の一例である。

【発明を実施するための形態】

【００１１】

以下、実施形態について適宜図面を参照しつつ詳細に説明する。

【００１２】

図１に実施形態として説明する情報処理システム１の概略的な構成を示している。同図に示すように、情報処理システム１は、ソフトウェアの開発や保守が行われる開発／保守サイド２（システム開発センタ等）に設けられる一つ以上のソフトウェア開発保守装置１０と、ソフトウェアが運用される運用サイド３（システムセンタ、データセンタ等）に設けられる一つ以上のサービス提供装置２０とを含む。ソフトウェア開発保守装置１０及びサービス提供装置２０は、いずれも情報処理装置（コンピュータ）である。ソフトウェア開発保守装置１０とサービス提供装置２０とは、ＬＡＮ（Local Area Network）やインターネット等の通信手段５を介して通信可能に接続されている。

【００１３】

サービス提供装置２０は、例えば、ソフトウェアを実行もしくは提供することによりユーザサイドの情報処理装置（Ｗｅｂクライアント等）に対してサービスを提供する情報処理装置（Ｗｅｂサーバ、アプリケーションサーバ等）である。サービス提供装置２０は、例えば、クラウドシステムにより提供されるクラウドサーバ等のように仮想的に実現されるものであってもよい。サービス提供装置２０は、例えば、ソフトウェアを実行することにより実現される、コンテンツデータ（文字データ、動画データ、静止画データ等）を記憶するコンテンツデータ記憶機能やＷｅｂクライアント等からのリクエストに応じてコンテンツデータを配信するコンテンツデータ配信機能等を備える。

【００１４】

ソフトウェア開発保守装置１０は、サービス提供装置２０が実行するソフトウェアの開

発や保守に用いられる情報処理装置（コンピュータ）である。ソフトウェア開発保守装置１０では、サービス提供装置２０が実行するソフトウェアの開発や保守を支援する、所定のフレームワークに準拠したソフトウェアが動作している。またソフトウェア開発保守装置１０は、例えば、サービス提供装置２０によって実行されるソフトウェアのソースコードやドキュメント（画面仕様書等、システム設計書、プログラム仕様書）の管理を行う。本実施形態では、上記フレームワークとして、例えば、AngularJS(登録商標)を想定しているが、フレームワークの種類は必ずしも限定されない。

【００１５】

図２にソフトウェア開発保守装置１０やサービス提供装置２０の実現に用いられるハードウェア構成の一例（情報処理装置３０）を示している。同図に示すように、情報処理装置３０は、プロセッサ１１、主記憶装置１２、補助記憶装置１３、入力装置１４、出力装置１５、及び通信装置１６を備える。これらは図示しないバス等の通信手段を介して互いに通信可能に接続されている。

【００１６】

プロセッサ１１は、例えば、ＣＰＵ（Central Processing Unit）やＭＰＵ（Micro Processing Unit）を用いて構成されている。プロセッサ１１が、主記憶装置１２に格納されているプログラムを読み出して実行することにより、ソフトウェア開発保守装置１０の機能やサービス提供装置２０の機能が実現される。主記憶装置１２は、プログラムやデータを記憶する装置であり、例えば、ＲＯＭ（Read Only Memory）、ＲＡＭ（Random Access Memory）、不揮発性半導体メモリ（ＮＶＲＡＭ（Non Volatile RAM））等である。

【００１７】

補助記憶装置１３は、例えば、ハードディスクドライブ、ＳＳＤ（Solid State Drive）、光学式記憶装置（ＣＤ（Compact Disc）、ＤＶＤ(Digital Versatile Disc)等）、ストレージシステム、ＩＣカード、ＳＤメモリカードや光学式記録媒体等の記録媒体の読取／書込装置、クラウドサーバの記憶領域等である。補助記憶装置１３に格納されているプログラムやデータは主記憶装置１２に随時ロードされる。補助記憶装置１３は、例えば、ネットワークストレージのように通信手段を介して接続するものであってもよい。

【００１８】

入力装置１４は、外部入力を受け付けるユーザインタフェースであり、例えば、キーボード、マウス、タッチパネル等である。出力装置１５は、処理経過や処理結果等の各種情報を提供するユーザインタフェースであり、例えば、画面表示装置（液晶モニタ、ＬＣＤ（Liquid Crystal Display）、グラフィックカード等）、印字装置等である。尚、例えば、通信装置１６を介して他の装置との間で情報の入力や出力を行う構成、即ち、通信装置１６が、入力装置１４や出力装置１５として機能する構成としてもよい。

【００１９】

通信装置１６は、ＬＡＮやインターネット等の通信手段を介して行われる他の装置との間の通信を実現する有線方式または無線方式の通信インタフェースであり、例えば、ＮＩＣ（Network Interface Card）や無線通信モジュール等である。

【００２０】

図３にソフトウェア開発保守装置１０の一つとして機能する画面情報生成装置１００の機能及び当該画面情報生成装置１００が記憶するデータを示している。

【００２１】

同図に示すように、画面情報生成装置１００は、画面遷移情報生成部２１０、画面遷移図生成部２３０、及び情報記憶部２５０の各機能（機能部、処理部）を備える。また画面遷移情報生成部２１０は、画面定義情報取得部２１１、画面遷移関連情報取得部２１２、フィルタリング部２１３、及び画面遷移情報補完部２１４を備える。これらの機能は、例えば、プロセッサ１１が、主記憶装置１２や補助記憶装置１３に格納されているプログラムを読み出して実行することにより実現される。またこれらの機能は、例えば、画面情報生成装置１００が備えるハードウェア（ＡＳＩＣ（Application Specific Integrated Circuit）等）によって実現される。尚、画面情報生成装置１００は、これらの機能以外に

、例えば、オペレーティングシステム、デバイスドライバ、ＤＢＭＳ（DataBase Management System）等が動作するものであってもよい。

【００２２】

同図に示すように、情報記憶部２５０は、画面定義関数表２５１（画面定義関数情報）、コントローラ登録関数表２５２、画面遷移関数表２５３（画面遷移関数情報）、画面遷移表２５４、遷移ハンドラ使用確認表２５５、及びソースコード２７０の各データを記憶する。情報記憶部２５０は、これらのデータを、例えば、ファイルシステムやＤＢＭＳによって管理する。尚、これらのデータは、必ずしも画面情報生成装置１００が常に記憶していなくてもよい。また画面情報生成装置１００が、サービス提供装置２０等の他の情報処理装置と通信することにより必要なデータを随時取得する構成としてもよい。

【００２３】

ソースコード２７０は、サービス提供装置２０の機能を実現するソフトウェアの実行コードの生成元となるデータである。ソースコード２７０は、所定の言語（ＨＴＭＬ（Hyper Text Markup Language）、スクリプト（script）等）で記述された記述文を含む。尚、本実施形態では、ソースコード２７０として、AngularJS(登録商標)やTypeScript(登録商標)向けのものを想定しているが、ソースコード２７０の種類は必ずしも限定されない。

【００２４】

画面遷移情報生成部２１０は、遷移元の画面（以下、遷移元画面と称する。）に関する情報（遷移元画面情報）と遷移先の画面（以下、遷移先画面と称する。）に関する情報（遷移先画面情報）との関係を示す情報（以下、画面遷移情報と称する。）を含む画面遷移表２５４を生成する。尚、画面遷移表２５４の詳細については後述する。

【００２５】

画面定義情報取得部２１１は、ソースコード２７０から画面の定義に関する情報（以下、画面定義情報と称する。）を取得し、取得した情報を解析してその結果を画面遷移表２５４に反映する。

【００２６】

画面遷移関連情報取得部２１２は、ソースコード２７０から画面の遷移に関する情報（以下、画面遷移関連情報と称する。）を取得し、取得した情報を解析してその結果を画面遷移表２５４に反映する。

【００２７】

フィルタリング部２１３は、画面遷移情報のフィルタリング（無効な内容の排除や除去等）を行う。フィルタリング部２１３が行う処理の詳細については後述する。

【００２８】

画面遷移情報補完部２１４は、画面遷移情報の補完（情報の追加や補充等）を行う。画面遷移情報補完部２１４が行う処理の詳細については後述する。

【００２９】

画面遷移図生成部２３０は、画面遷移表２５４に基づき、後述する画面遷移図１５００を生成する。

【００３０】

図４に、情報記憶部２５０が記憶する画面定義関数表２５１の例を示している。画面定義関数表２５１には、ソースコード２７０に記述されている、画面を定義する関数（以下、画面定義関数と称する。）に関する情報（以下、画面定義関数情報と称する。）が管理されている。

【００３１】

同図に示すように、画面定義関数表２５１は、サービス名４１１、メソッド名４１２、画面ＩＤ（引数位置）４１３、画面ＨＴＭＬ４１４（引数位置４１４１、プロパティ名４１４２）、及びコントローラＩＤ４１５（引数位置４１５１、プロパティ名４１５２）の各項目を有する一つ以上のレコードで構成されている。画面定義関数表２５１の各行は一つの画面定義関数に対応している。

【００３２】

上記項目のうち、サービス名４１１には、画面定義関数のサービス名（例えば、AngularJS(登録商標)が提供するサービスの名称）が、メソッド名４１２には、画面定義関数のメソッド名が設定される。本例では、サービス名４１１に「$stateProvider」が、メソッド名４１２に「state」が、夫々設定されている。

【００３３】

画面ＩＤ（引数位置）４１３には、当該画面定義関数における画面ＩＤ（画面の識別子）の引数位置を示す情報が設定される。本例では、画面ＩＤ（引数位置）４１３に「０」が設定されている。

【００３４】

画面ＨＴＭＬ４１４（引数位置４１４１）には、当該画面定義関数における、画面記述文（本例ではＨＴＭＬ）を含むオブジェクトの引数位置を示す情報が設定される。本例では、画面ＨＴＭＬ４１４（引数位置４１４１）に「１」が設定されている。また画面ＨＴＭＬ４１４（プロパティ名４１４２）には、上記オブジェクトのプロパティ名が設定される。本例では、画面ＨＴＭＬ４１４（プロパティ名４１４２）に「templateUrl」が設定されている。

【００３５】

コントローラＩＤ４１５（引数位置４１５１）には、当該画面定義関数における、コントローラＩＤを含むオブジェクトの引数位置を示す情報が設定される。本例では、コントローラＩＤ４１５（引数位置４１５１）に「１」が設定されている。またコントローラＩＤ４１５（プロパティ名４１５２）には、上記オブジェクトのコントローラＩＤのプロパティ名が設定される。本例では、コントローラＩＤ４１５（プロパティ名４１５２）に「controller」が設定されている。

【００３６】

尚、コントローラは、例えば、ＭＶＣモデル（Model View Controller model）におけるコントローラである。本実施形態では、説明の簡単のため、画面定義関数で定義される１つの画面が１つの画面記述文と１つのコントローラとを用いて構成されている場合を例として説明するが、画面定義関数で定義される画面の構成は必ずしも例示するものに限定されない。

【００３７】

図５に、情報記憶部２５０が記憶するコントローラ登録関数表２５２の例を示している。コントローラ登録関数表２５２には、コントローラを登録する関数（以下、コントローラ登録関数と称する。）に関する情報が管理されている。

【００３８】

同図に示すように、コントローラ登録関数表２５２は、サービス名５１１、メソッド名５１２、コントローラＩＤ（引数位置）５１３、及びコントローラクラス名の引数位置５１４（引数位置５１４１、配列内位置５１４２）の各項目を有する一つ以上のレコードで構成されている。コントローラ登録関数表２５２の各行は一つのコントローラ登録関数に対応している。

【００３９】

サービス名５１１には、コントローラ登録関数のサービス名が、メソッド名５１２にはコントローラ登録関数のメソッド名が、夫々設定される。本例では、サービス名５１１に「regService」が、メソッド名５１２に「regController」が、夫々設定されている。

【００４０】

コントローラＩＤ（引数位置）５１３には、当該コントローラ登録関数における、コントローラＩＤの引数位置を示す情報が設定される。本例では、コントローラＩＤ（引数位置）５１３に「０」が設定されている。

【００４１】

コントローラクラス名の引数位置５１４（引数位置５１４１）には、当該コントローラ登録関数において、コントローラクラス名を含む配列の引数位置を示す情報が設定される。本例では、引数位置５１４（引数位置５１４１）に「１」が設定されている。またコン

トローラクラス名の引数位置５１４（配列内位置５１４２）には、上記配列内のコントローラクラス名の配列内の位置を示す情報が設定される。本例では、引数位置５１４（配列内位置５１４２）に「lastIndex」が設定されている。

【００４２】

図６に、情報記憶部２５０が記憶する画面遷移関数表２５３の例を示している。画面遷移関数表２５３には、画面の遷移を行う関数（以下、画面遷移関数と称する。）に関する情報が管理されている。

【００４３】

同図に示すように、画面遷移関数表２５３は、サービス名６１１、メソッド名６１２、遷移先画面ＩＤ６１３（引数位置６１３１、遷移ハンドラが静的な遷移を行うものであるか否かの判定（内容６１３２、使用６１３３））、及びラッピング階層数６１４の各項目を有する一つ以上のレコードで構成されている。画面遷移関数表２５３の各行は一つの画面遷移関数に対応している。

【００４４】

サービス名６１１には、画面遷移関数のサービス名が、メソッド名６１２には画面遷移関数のメソッド名が設定される。本例では、サービス名６１１に「$state」が、メソッド名６１２に「go」が、夫々設定されている。

【００４５】

遷移先画面ＩＤ６１３（引数位置６１３１）には、画面遷移関数における遷移先の画面の画面ＩＤ（以下、遷移先画面ＩＤと称する。）の引数位置を示す情報が設定される。本例では、遷移先画面ＩＤ６１３（引数位置６１３１）に「０」が設定されている。

【００４６】

遷移先画面ＩＤ６１３（遷移ハンドラが静的な遷移を行うものであるか否かの判定（内容６１３２））には、画面遷移関数をラッピングした遷移ハンドラが静的な遷移を行うものであるか否かを判定する方法を示す情報が設定される。遷移先画面ＩＤ６１３（遷移ハンドラが静的な遷移か否かの判定（使用６１３３））には、上記方法を使用するか否かを示す情報（その方法を使用する場合は「○」、その方法を使用しない場合は「×」）が設定される。同図では、上記方法として、画面遷移関数の引数に遷移先画面ＩＤが即値で指定されているか否かに基づき判定する方法と、画面遷移関数をラッピングした遷移ハンドラと画面遷移関数の引数が同じ変数を使用し、かつ、遷移ハンドラの引数を即値で使用しているかに基づき判定する方法、の２つを例示している。

【００４７】

尚、静的な遷移とは、プログラムの実行時に遷移先が予め決まっている遷移のことである。また以下の説明において、静的な遷移でない遷移のことを動的な遷移とも称する。即ち、動的な遷移とは、遷移先が予め決まっておらずプログラムの実行時に遷移先が決定される遷移のことである。

【００４８】

ラッピング階層数６１４には、遷移関数を一階層でラッピングしている関数を遷移ハンドラとするか否かを示す情報が設定される。本例では、ラッピング階層数６１４に「１」が設定されている。

【００４９】

図７は、情報記憶部２５０が記憶する画面遷移表２５４の一例である。画面遷移表２５４には、遷移元画面に関する情報と、遷移先画面に関する情報との関係を示す情報が管理されている。尚、画面遷移表２５４は遷移元画面を単位として構成されている。また本例では、画面遷移表２５４には、画面情報生成装置１００が解析の対象とする全て範囲の遷移元画面の情報が管理されているものとする。

【００５０】

同図に示すように、画面遷移表２５４は、遷移元情報７１及び遷移先情報７２の各項目を含む一つ以上のレコードで構成されている。同図に示すように、遷移元情報７１は、遷移元画面７３を項目として有する。また遷移先情報７２は、遷移先画面７４及び遷移ハン

ドラ７５の各項目を有する。
【００５１】
遷移元画面７３は、遷移元画面ＩＤ７３１、遷移元画面ＨＴＭＬ７３２、コントローラＩＤ７３３、及びコントローラクラス７３４の各項目を有する。
【００５２】
このうち遷移元画面ＩＤ７３１には、遷移元画面の画面ＩＤが設定される。本例では、遷移元画面ＩＤ７３１に「top」が設定されている。
【００５３】
遷移元画面ＨＴＭＬ７３２には、遷移元画面のソースコード（画面記述文）を特定する情報（画面記述文のファイル名等）が設定される。本例では、遷移元画面ＨＴＭＬ７３２に「top.html」が設定されている。
【００５４】
コントローラＩＤ７３３には、当該遷移元画面のコントローラの識別子（以下、コントローラＩＤと称する。）が設定される。本例では、コントローラＩＤ７３３に「topCtrl」が設定されている。
【００５５】
コントローラクラス７３４には、コントローラＩＤ７３３で特定されるコントローラのクラス名が設定される。本例では、コントローラクラス７３４に「TopCtrl」が設定されている。
【００５６】
同図に示すように、遷移先画面７４は、遷移先画面ＩＤ７４１、画面定義有無７４２、有効／無効７４３、及び遷移条件７４４の各項目を有する。
【００５７】
このうち遷移先画面ＩＤ７４１には、遷移先画面の画面ＩＤが設定される。本例では、遷移先画面ＩＤ７４１に画面ＩＤとして「news」、「travel」、「map」、「blog」、「game」、「sports」、「movie」が設定されている。
【００５８】
画面定義有無７４２には、画面の定義が存在するか否かを示す情報（画面の定義がある（存在する）ことを示す「○」または画面の定義が無いことを示す「×」）が設定される。
【００５９】
有効／無効７４３には、当該遷移先画面の有効／無効を示す情報（有効であることを示す「○」または無効であることを示す「×」）が設定される。本例では、画面定義有無７４２に「○」が設定されており、かつ、使用７５３に「○」が設定されている場合に有効／無効７４３に「○」が設定され、それ以外の場合は有効／無効７４３に「×」が設定される。
【００６０】
遷移条件７４４には、遷移ハンドラが動的な遷移を行うものであり、かつ、複数の画面への遷移を行うものである場合に、画面を遷移させる条件（以下、遷移条件と称する。）を示す情報が設定される。
【００６１】
同図に示すように、遷移ハンドラ７５は、ハンドラ名７５１、静的遷移７５２、及び使用７５３の各項目を含む。
【００６２】
ハンドラ名７５１には、当該遷移元画面に定義されている遷移ハンドラの名称（以下、ハンドラ名と称する。）が設定される。
【００６３】
静的遷移７５２には、遷移ハンドラが静的な遷移を行うものであるか動的な遷移を行うものであるかを示す情報（静的な遷移を行うものであれば「○」、動的な遷移を行うものであれば「×」）が設定される。

【００６４】
使用７５３には、遷移ハンドラが実際に使用されているか否かを示す情報（使用されていれば「○」、使用されていなければ「×」）が設定される。
【００６５】
本例の場合、ハンドラ名７５１が「transToNews」の遷移ハンドラは、「news」という画面への静的な遷移を行うハンドラであり、画面定義有無７４２には「○」が、使用７５３には「○」が、夫々設定されており、その結果として、有効／無効７４３には「○」が設定されている。尚、「transToNews」は「news」画面にのみ遷移させるハンドラであるので遷移条件７４４は設定されていない。
【００６６】
またハンドラ名７５１が「transToTravel」の遷移ハンドラは、「travel」という画面への静的な遷移を行うハンドラであるが、画面定義有無７４２に「×」が設定されているため、有効／無効７４３には「×」が設定されている。また「transToTravel」は、「travel」画面にのみ遷移させるハンドラであるので遷移条件７４４は設定されていない。
【００６７】
またハンドラ名７５１が「transToMap」の遷移ハンドラは、「map」という画面に静的な遷移を行うハンドラであるが、使用７５３に「×」が設定されているため有効／無効７４３には「×」が設定されている。
【００６８】
ハンドラ名７５１が「transToX」の遷移ハンドラは、「blog」または「game」という画面に動的な遷移を行うハンドラであり、「blog」、「game」のいずれの画面への遷移も有効である。本例ではいずれの画面についても画面定義有無７４２に「○」が設定されており、かつ、使用７５３に「○」が設定されているので、いずれの画面についても有効／無効７４３には「○」が設定されている。尚、この遷移ハンドラは動的な遷移を行うものであり、従って「blog」または「game」のいずれの画面に遷移するかはソフトウェアの実行時に決定される。同図に示すように、「blog」には、遷移条件７４４として、画面ＨＴＭＬに表示されているプルダウンメニューで”blog”が指定されている時を意味する内容が、また「game」には、遷移条件７４４として、画面ＨＴＭＬに表示されているプルダウンメニューで”game”が指定されている時を意味する内容が、夫々設定されている。
【００６９】
ハンドラ名７５１が「transToY」の遷移ハンドラは、「sports」、「movie」という画面に静的な遷移を行うハンドラである。この遷移ハンドラは、「sports」については使用７５３に「○」が設定されているが、「movie」については使用７５３に「×」が設定されている。そのため、「sports」については有効／無効７４３に「○」が設定され、「movie」については有効／無効７４３に「×」が設定されている。同図に示すように、「sports」には、遷移条件７４４として、画面ＨＴＭＬに表示されているプルダウンメニューで”sports”が指定されている時を意味する内容が、また「movie」には、遷移条件７４４として、画面ＨＴＭＬに表示されているプルダウンメニューで”movie”が指定されている時を意味する内容が、夫々設定されている。
【００７０】
図８に遷移ハンドラ使用確認表２５５の一例を示している。遷移ハンドラ使用確認表２５５には、遷移ハンドラが実際に使用されているか否かを判定する際に用いる情報が管理されている。本例では、遷移ハンドラ使用確認表２５５に、画面記述文（ＨＴＭＬ文）内のどの部分と遷移ハンドラが対応づけられるかという情報（以下、遷移ハンドラ利用確認情報と称する。）が管理され、タグ８１１「button」と属性８１２「ng-click」との対応を示す情報が管理されている。
【００７１】
図９は、画面情報生成装置１００が画面遷移表２５４を生成する際に行う処理（以下、画面遷移情報生成処理Ｓ９００と称する。）を説明するフローチャートである。以下、同図とともに画面遷移情報生成処理Ｓ９００について説明する。尚、画面遷移情報生成処理

Ｓ９００は、例えば、ユーザが画面情報生成装置１００に対して所定の起動処理を行ったことを契機として開始される。

【００７２】

同図に示すように、まず画面定義情報取得部２１１が、画面定義関数表２５１を参照しつつソースコード２７０を解析し、その結果（遷移元画面ＩＤ７３１、遷移元画面ＨＴＭＬ７３２）を画面遷移表２５４に反映する（Ｓ９１１）。また画面定義情報取得部２１１が、コントローラ登録関数表２５２を参照しつつソースコード２７０を解析し、その結果（コントローラＩＤ７３３、コントローラクラス７３４）を画面遷移表２５４に反映する（Ｓ９１２）。これらの処理（Ｓ９１１、Ｓ９１２）について、図１０（ａ）、（ｂ）に示すソースコード２７０、図４に示す画面定義関数表２５１、及び図５に示すコントローラ登録関数表２５２を例として具体的に説明する。

【００７３】

まず画面定義情報取得部２１１は、図４に示す画面定義関数表２５１の画面ＩＤ（引数位置）４１３に設定されている「０」という情報に基づき、図１０（ａ）に示す画面定義のソースコード２７０から、０番目の引数位置に設定されている「top」を遷移元画面ＩＤ７３１として取得する。

【００７４】

続いて、画面定義情報取得部２１１は、図４に示す画面定義関数表２５１の画面ＨＴＭＬ４１４（引数位置４１４１）に設定されている「１」、及び画面ＨＴＭＬ４１４（プロパティ名４１４２）に設定されている「templateUrl」に基づき、図１０（ａ）に示す画面定義のソースコード２７０から、「templateUrl」というプロパティ名のオブジェクトの１番目の引数位置に設定されている「top.html」を遷移元画面ＨＴＭＬ７３２として取得する。

【００７５】

続いて、画面定義情報取得部２１１は、図４に示す画面定義関数表２５１のコントローラＩＤ４１５（引数位置４１５１）に設定されている「１」、及びコントローラＩＤ４１５（プロパティ名４１５２）に設定されている「controller」に基づき、図１０（ａ）に示すソースコード２７０から、「controller」というプロパティ名のオブジェクトの１番目の引数位置に設定されている「topCtrl」をコントローラＩＤ７３３として取得する。

【００７６】

続いて、画面定義情報取得部２１１は、コントローラＩＤ７３３として取得した「topCtrl」と、コントローラ登録関数表２５２のコントローラＩＤ（引数位置）５１３に設定されている「０」という情報とから、図１０（ｂ）に示すコントローラの登録に関するソースコード２７０を特定する。そして、画面定義情報取得部２１１は、コントローラクラスの引数位置５１４（引数位置５１４１、配列内位置５１４２）に設定されている情報に基づき、図１０（ｂ）に示すソースコード２７０から、1番目の引数位置にある配列の最後のインデックス位置の文字列「TopCtrl」をコントローラクラス７３４として取得する。

【００７７】

画面定義情報取得部２１１は、以上のようにして取得した情報（遷移元画面ＩＤ７３１、遷移元画面ＨＴＭＬ７３２、コントローラＩＤ７３３、及びコントローラクラス７３４）を遷移元情報７１（遷移元画面７３）として画面遷移表２５４に反映する。

【００７８】

図９に戻り、続いて、画面遷移関連情報取得部２１２が、画面遷移表２５４に記載されている遷移元情報７１の一つ（遷移元画面ＩＤ７３１で区別されるレコードのうちの一つ）を選択する（Ｓ９１３）。

【００７９】

続いて、画面遷移関連情報取得部２１２は、選択中の遷移元情報７１のコントローラクラス７３４から特定されるコントローラクラスを解析して遷移先画面ＩＤ７４１、ハンドラ名７５１、及び静的遷移７５２の内容を取得し、取得した内容を遷移先情報７２として

画面遷移表２５４に反映する（Ｓ９１４）。Ｓ９１４の処理について、図７に示す画面遷移表２５４、図１１の遷移ハンドラの定義に関するソースコード２７０、及び図６の画面遷移関数表２５３、を例として具体的に説明する。

【００８０】

まず画面遷移関連情報取得部２１２は、コントローラクラス７３４に設定されているコントローラクラスの図６の画面遷移関数表２５３に記載されている画面遷移関数を、当該画面遷移関数表２５３のラッピング階層数６１４に設定されている階層でラッピングしている関数を遷移ハンドラとして取得する。また画面遷移関連情報取得部２１２は、画面遷移関数の０番目（図６の引数位置６１３１に設定されている「０」から取得）の引数を解析し、引数が即値であれば、遷移ハンドラは静的な遷移を行うハンドラであると判定する。図１１のソースコード２７０の場合、画面遷移関連情報取得部２１２は、「transToNews」、「transToTravel」、「transToMap」、「transToX」、「transToY」を遷移ハンドラとして取得する。

【００８１】

また画面遷移関連情報取得部２１２は、遷移ハンドラが内部で利用している、画面遷移関数表２５３に記載されている関数（本例では、サービス名６１１は「$state」、メソッド名６１２が「go」の関数）の使用箇所を解析することにより、遷移ハンドラが静的な遷移を行うものであるか否かを判定し、遷移ハンドラから遷移先の画面（遷移先画面ＩＤ７４１）を取得する。例えば、遷移ハンドラが、画面遷移関数表２５３に記載されている関数（本例では、サービス名６１１は「$state」、メソッド名６１２が「go」の関数）を、０番目の引数を即値として利用している場合、画面遷移関連情報取得部２１２は、遷移ハンドラは静的な遷移を行うものと判定する。また０番目の引数を即値を指定する以外の方法で利用している場合、画面遷移関連情報取得部２１２は、遷移ハンドラは動的な遷移を行うものと判定する。図１１のソースコード２７０の場合、画面遷移関連情報取得部２１２は、「transToNews」、「transToTravel」、「transToMap」、「transToY」については静的な遷移を行うハンドラであると判定し、「transToX」については動的な遷移を行うハンドラであると判定する。画面遷移関連情報取得部２１２は、静的な遷移を行うハンドラであると判定した遷移ハンドラについては該当する画面遷移表２５４の静的遷移７５２に「○」を設定する。また画面遷移関連情報取得部２１２は、動的な遷移を行うハンドラであると判定した遷移ハンドラについては該当の画面遷移表２５４の静的遷移７５２に「×」を設定する。

【００８２】

遷移ハンドラが静的な遷移を行うものである場合、画面遷移関連情報取得部２１２は、遷移ハンドラが内部で利用する、画面遷移関数表２５３に記載されている関数（本例では、サービス名６１１は「$state」、メソッド名６１２が「go」の関数）の０番目の引数を遷移先画面ＩＤ７４１として取得する。図１１に示すソースコード２７０の場合、「transToNews」の遷移先画面ＩＤ７４１は「news」、「transToTravel」の遷移先画面ＩＤ７４１は「travel」、「transToMap」の遷移先画面ＩＤ７４１は「map」、「transToY」の遷移先画面ＩＤ７４１は「sports」又は「movie」となる。尚、「transToX」は動的な遷移を行うハンドラであるので実際の遷移先はプログラムの実行時に決定される。後述するように、画面遷移関連情報取得部２１２は、遷移先画面ＩＤ７４１の内容（本例では「blog」、「game」）を例えばユーザに入力させる。尚、Ｓ９１４処理の時点では「transToX」の遷移先画面ＩＤ７４１の内容は設定されていないものとする。遷移先画面ＩＤ７４１の具体的な設定方法については後述する。

【００８３】

このように、画面遷移関連情報取得部２１２は、画面定義情報取得部２１１が特定したコントローラクラスを解析して効率よく画面の遷移を行う遷移ハンドラを特定する。また画面遷移関連情報取得部２１２は、遷移ハンドラが静的な遷移を行うハンドラであるのか動的な遷移を行うハンドラであるのかを判定し、画面遷移表２５４の遷移先情報７２の内容を適切に設定する。

【００８４】
図９に戻り、続いて、フィルタリング部２１３が、Ｓ９１４で取得された遷移先情報７２をフィルタリングする（Ｓ９１５）。尚、この処理（以下、フィルタリング処理Ｓ９１５と称する。）の詳細については後述する。
【００８５】
続いて、画面遷移情報補完部２１４が、Ｓ９１４で取得した画面遷移情報を補完する（Ｓ９１６）。尚、この処理（以下、補完処理Ｓ９１６と称する。）の詳細については後述する。
【００８６】
続いて、画面遷移関連情報取得部２１２が、画面遷移表２５４に記載されている遷移元情報７１のうち未選択のものがあるか否かを判定する。画面遷移関連情報取得部２１２が未選択の遷移元情報７１があると判定した場合（Ｓ９１７：ＹＥＳ）、処理はＳ９１３に戻り、画面遷移関連情報取得部２１２は、未選択の遷移元情報７１について以上と同様の処理（Ｓ９１４以降の処理）を繰り返す。画面遷移関連情報取得部２１２が未選択の遷移元情報７１がないと判定した場合（Ｓ９１７：ＮＯ）、画面遷移情報生成処理Ｓ９００は終了する。
【００８７】
図１２は、図９のフィルタリング処理Ｓ９１５の詳細を説明するフローチャートである。以下、同図とともにフィルタリング処理Ｓ９１５について説明する。
【００８８】
まずフィルタリング部２１３は、画面遷移表２５４から遷移先画面７４の一つ（遷移元画面ＩＤ７３１で区別されるレコードの遷移先画面ＩＤ７４１で区別される遷移先画面の一つ）を選択する（Ｓ１２１１）。
【００８９】
続いて、フィルタリング部２１３は、選択中の遷移先画面が画面遷移表２５４に遷移元画面７３として含まれているか否か（選択中の遷移先画面７４が画面遷移表２５４に存在するか否か）を判定する（Ｓ１２１２）。フィルタリング部２１３が、選択中の遷移先画面が画面遷移表２５４に遷移元画面７３として含まれていると判定した場合（Ｓ１２１２：ＹＥＳ）、処理はＳ１２１３に進む。一方、フィルタリング部２１３が、選択中の遷移先画面が画面遷移表２５４に遷移元画面７３として含まれていないと判定した場合（Ｓ１２１２：ＮＯ）、処理はＳ１２１４に進む。
【００９０】
Ｓ１２１３では、フィルタリング部２１３は、選択中の遷移先画面の画面遷移表２５４の画面定義有無７４２に「○」を設定する（Ｓ１２１３）。その後、処理はＳ１２１５に進む。またＳ１２１４では、フィルタリング部２１３は、選択中の遷移先画面の画面遷移表２５４の画面定義有無７４２に「×」を設定する（Ｓ１２１４）。その後、処理はＳ１２１５に進む。
【００９１】
Ｓ１２１５では、フィルタリング部２１３は、選択中の遷移先画面の遷移ハンドラ７５が実際に使用されているか否かを判定する。フィルタリング部２１３が選択中の遷移先画面の遷移ハンドラ７５が実際に使用されていると判定した場合（Ｓ１２１５：ＹＥＳ）、処理はＳ１２１６に進む。フィルタリング部２１３が選択中の遷移先画面の遷移ハンドラ７５が実際に使用されていないと判定した場合（Ｓ１２１５：ＮＯ）、処理はＳ１２１７に進む。
【００９２】
ここでフィルタリング部２１３は、遷移ハンドラ７５が実際に使用されているか否かの判定を、選択中の遷移先画面の遷移元画面ＨＴＭＬ７３２で特定されるソースコード２７０を解析することにより行う。具体的には、フィルタリング部２１３は、遷移ハンドラ使用確認表２５５のタグ８１１と属性８１２の組み合わせが上記のソースコード２７０に記述されており、上記属性８１２として遷移ハンドラ７５が使用されている場合は当該遷移

ハンドラ７５が実際に使用されていると判定する。尚、フィルタリング部２１３は、遷移ハンドラが同一の画面に関する単数または複数の画面記述文の複数箇所で使用されている場合、同一の遷移先画面への遷移で使われている同一の遷移ハンドラは同じものとして画面遷移情報の重複した出力を行わない。これにより冗長な情報が画面遷移情報として含まれてしまうのを防ぐことができる。

【００９３】

Ｓ１２１６では、フィルタリング部２１３は、画面遷移表２５４の使用７５３に「○」を設定する。その後、処理はＳ１２１８に進む。Ｓ１２１７では、フィルタリング部２１３は、画面遷移表２５４の使用７５３に「×」を設定する。その後、処理はＳ１２１８に進む。

【００９４】

ここで例えば図１３に示すソースコード２７０の例では、「button」タグと「ng-click」属性の組み合わせで「transToNews」、「transToTravel」、「transToX」、及び「transToY」という４つの遷移ハンドラが使用されているが、図７に示すように「transToY」については「sports」または「movie」の２つの画面に遷移する可能性があるので、フィルタリング部２１３は夫々について遷移ハンドラが使用されているか否かを判定する。例えば、図１１のソースコード２７０の例では「var a=10」で変数「ａ」が「10」に設定されているため画面遷移先が「movie」の遷移は起こり得ず、フィルタリング部２１３は「movie」については遷移ハンドラ「transToY」が実際に使用していると判定し、使用７５３に「○」を設定する。またフィルタリング部２１３は「sports」については遷移ハンドラ「transToY」は実際に使用されていないと判定し、使用７５３に「×」を設定する。

【００９５】

図１２に戻り、Ｓ１２１８では、フィルタリング部２１３は、画面遷移表２５４に記載されている遷移先画面７４のうち未選択のものがあるか否かを判定する。フィルタリング部２１３が未選択のものがあると判定した場合（Ｓ１２１８：ＹＥＳ）、処理はＳ１２１１に戻り、未選択の遷移先画面７４について以上と同様の処理（Ｓ１２１２以降の処理）を繰り返す。フィルタリング部２１３が未選択のものがないと判定した場合（Ｓ１２１８：ＮＯ）、フィルタリング処理Ｓ９１５は終了し、処理は図９の補完処理Ｓ９１６に進む。

【００９６】

図１４は、図９の補完処理Ｓ９１６を説明するフローチャートである。以下、同図とともに補完処理Ｓ９１６について説明する。

【００９７】

まず画面遷移情報補完部２１４が、画面遷移表２５４から遷移ハンドラ７５を一つ（遷移元画面ＩＤ７３１で区別されるレコードのハンドラ名７５１で区別される遷移ハンドラの一つ）選択する（Ｓ１４１１）。

【００９８】

続いて、画面遷移情報補完部２１４は、画面遷移表２５４を参照しつつ、選択中の遷移ハンドラが静的な遷移を行うものであるか否かを判定する（Ｓ１４１２）。画面遷移情報補完部２１４が、選択中の遷移ハンドラは静的な遷移を行うものであると判定した場合（Ｓ１４１２：静的遷移）、処理はＳ１４１５に進む。一方、画面遷移情報補完部２１４が、選択中の遷移ハンドラは静的な遷移を行うものでない（動的な遷移を行うものである）と判定した場合（Ｓ１４１２：動的遷移）、処理はＳ１４１３に進む。

【００９９】

Ｓ１４１３では、画面遷移情報補完部２１４は、画面遷移表２５４の動的な遷移の遷移先画面の候補となる遷移先画面ＩＤ７４１を補完する。この補完は、例えば、ポップアップ画面等を介してユーザから画面ＩＤの入力を受け付ける、予め候補となる画面ＩＤをソースコード２７０内にコメント等として記述しておきこれを利用して補完する、といった方法で行われる。

【０１００】

Ｓ１４１４では、画面遷移情報補完部２１４は、Ｓ１４１３で補完された遷移先画面ＩＤ７４１が画面遷移表２５４の遷移元画面ＩＤ７３１に含まれているか（補完された遷移先画面が実際に存在するか）否かを判定する。画面遷移情報補完部２１４が、補完された遷移先画面ＩＤ７４１が画面遷移表２５４の遷移元画面ＩＤ７３１に含まれていると判定した場合（Ｓ１４１４：ＹＥＳ）、処理はＳ１４１５に進む。画面遷移情報補完部２１４が、補完された遷移先画面ＩＤ７４１が画面遷移表２５４の遷移元画面ＩＤ７３１に含まれていないと判定した場合（Ｓ１４１４：ＮＯ）、処理はＳ１４１３に戻る。

【０１０１】

Ｓ１４１５では、画面遷移情報補完部２１４は、画面遷移表２５４に基づき、選択中の遷移ハンドラの遷移先画面が複数であるか否かを判定する。画面遷移情報補完部２１４が、遷移ハンドラの遷移先画面が複数であると判定した場合（Ｓ１４１５：ＹＥＳ）、処理はＳ１４１６に進む。一方、画面遷移情報補完部２１４が、遷移ハンドラの遷移先画面が複数でないと判定した場合（Ｓ１４１５：ＮＯ）、処理はＳ１４１７に進む。

【０１０２】

Ｓ１４１６では、画面遷移情報補完部２１４は、選択中の遷移ハンドラの遷移先画面毎に遷移条件を補完する。この補完は、例えば、ポップアップ画面等を介してユーザから遷移条件の入力を受け付ける、予め遷移条件をソースコード２７０内にコメント等として記述しておきこれを利用して補完する、といった方法で行われる。

【０１０３】

Ｓ１４１７では、画面遷移情報補完部２１４は、選択中の遷移ハンドラについて画面遷移表２５４の遷移が有効か無効か否かを判定し、判定した結果を有効／無効７４３に反映する。前述したように、本例の場合、画面遷移情報補完部２１４は、画面定義有無７４２に「○」が設定され、かつ、使用７５３の欄に「○」が設定されている場合に有効／無効７４３に「○」を設定し、それ以外は使用７５３の欄に「×」を設定する。

【０１０４】

Ｓ１４１８では、画面遷移情報補完部２１４は、未選択の遷移ハンドラがあるか否かを判定する。画面遷移情報補完部２１４が未選択の遷移ハンドラがあると判定した場合（Ｓ１４１８：ＹＥＳ）、処理はＳ１４１１に戻り、画面遷移情報補完部２１４は、未選択の遷移ハンドラ７５について以上と同様の処理（Ｓ１４１２以降の処理）を繰り返す。画面遷移情報補完部２１４が未選択の遷移ハンドラがないと判定した場合（Ｓ１４１８：ＮＯ）、補完処理Ｓ９１６は終了し、処理は図９のＳ９１７に進む。

【０１０５】

尚、画面遷移図生成部２３０は、以上のようにして生成された画面遷移表２５４を利用して、遷移元画面情報と遷移先画面情報との関係を視覚的に示す情報である画面遷移図を生成する。画面遷移図生成部２３０は、例えば、ユーザインタフェースを介したリクエストに応じて画面遷移図を生成する。

【０１０６】

図１５に画面遷移図の一例を示す。同図に示す画面遷移図１５００を参照することで、ユーザは、例えば、画面ＩＤが「top」の画面から、画面ＩＤが「news」の画面へ、「transToNews」ハンドラを介して遷移し、画面ＩＤが「top」の画面から、画面ＩＤが「blog」または「game」の画面に「transToX」ハンドラを介して遷移することを容易に把握することができる。またユーザは、例えば、「プルダウンが”blog”の時」という遷移条件を満たした場合に「blog」画面に遷移し、「プルダウンが”game”の時」という遷移条件を満たした場合に「game」画面に遷移することを容易に把握することができる。また画面遷移図１５００によれば、ユーザは、例えば、画面ＩＤが「top」の画面から画面ＩＤが「sports」の画面への遷移は「transToY」ハンドラを介して行われることも容易に把握することができる。尚、この場合の遷移条件は「aが5未満の時」であるが、前述したように遷移先の画面が１つのみ有効であるため、本例では遷移条件は記載していない。

【０１０７】

以上に説明したように、本実施形態の画面情報生成装置１００は、画面定義関数表２５

１、コントローラ登録関数表２５２、及び画面遷移関数表２５３に基づき、ソースコード２７０から画面定義情報及び画面遷移関連情報を自動的に取得し、遷移元画面情報と遷移先画面情報との関係を示す情報を含む画面遷移情報（画面遷移表２５４）を生成する。このため、例えば、画面の遷移に関する情報がソースコード２７０の様々な箇所に分散して記述されている場合でも効率よく画面の遷移に関する情報を得ることができる。また例えば、画面情報生成装置１００が生成した画面遷移表２５４とソースコード２７０とを比較すれば両者が乖離している箇所を容易に特定することができ、ソフトウェアの開発や保守に際して用いるドキュメントを効率よく管理することができる。

【０１０８】

またフィルタリング部２１３は、画面遷移表２５４に遷移ハンドラの遷移先となる画面を定義する情報が含まれていない場合や遷移ハンドラが実際に使用されていない場合、当該遷移ハンドラに関する情報を画面遷移情報から除外するので、ユーザは有効な情報を効率よく取得することができる。またフィルタリング部２１３は、遷移ハンドラが静的な遷移を行うものであるか動的な遷移を行うものであるかを自動的に判定して不要な情報を除去するので適切かつ効率よく画面遷移情報を生成することができる。また画面遷移情報補完部２１４は、必要な情報を自動的に補完し、もしくは、ユーザに要求して補完するので、効率よく必要な情報を画面遷移情報に補完することができる。

【０１０９】

以上、本発明について実施の形態に基づき具体的に説明したが、本発明は上記の実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。例えば、上記の実施の形態は本発明を分かりやすく説明するために詳細に説明したものであり、必ずしも説明した全ての構成を備えるものに限定されるものではない。また、上記実施形態の構成の一部について、他の構成の追加・削除・置換をすることが可能である。

【０１１０】

また上記の各構成、機能部、処理部、処理手段等は、それらの一部または全部を、例えば、集積回路で設計する等によりハードウェアで実現してもよい。また上記の各構成、機能等は、プロセッサがそれぞれの機能を実現するプログラムを解釈し、実行することによりソフトウェアで実現してもよい。各機能を実現するプログラム、テーブル、ファイル等の情報は、メモリやハードディスク、ＳＳＤ（Solid State Drive）等の記録装置、またはＩＣカード、ＳＤカード、ＤＶＤ等の記録媒体に置くことができる。

【０１１１】

また上記の各図において、制御線や情報線は説明上必要と考えられるものを示しており、必ずしも実装上の全ての制御線や情報線を示しているとは限らない。例えば、実際にはほとんど全ての構成が相互に接続されていると考えてもよい。

【０１１２】

また以上に説明した画面情報生成装置１００の各種機能部、各種処理部、各種データベースの配置形態は一例に過ぎない。各種機能部、各種処理部、各種データベースの配置形態は、画面情報生成装置１００が備えるハードウェアやソフトウェアの性能、処理効率、通信効率等の観点から最適な配置形態に変更し得る。

【０１１３】

また前述したデータベースの構成（スキーマ（Schema）等）は、リソースの効率的な利用、処理効率向上、アクセス効率向上、検索効率向上等の観点から柔軟に変更し得る。

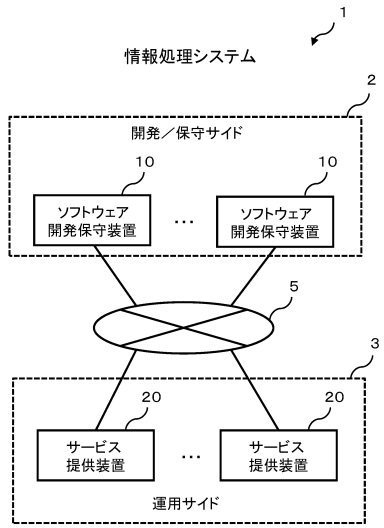
【符号の説明】

【０１１４】

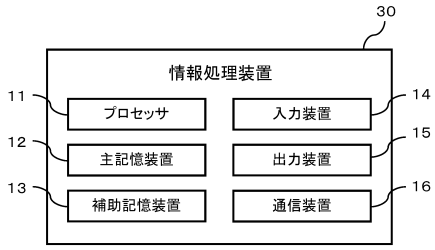
３０　情報処理装置、画面情報生成装置１００、２１０　画面遷移情報生成部、２１１　画面定義情報取得部、２１２　画面遷移関連情報取得部、２１３　フィルタリング部、２１４　画面遷移情報補完部、２３０　画面遷移図生成部、２５０　情報記憶部、２５１　画面定義関数表、２５２　コントローラ登録関数表、２５３　画面遷移関数表、２５４　画面遷移表、２５５　遷移ハンドラ使用確認表、Ｓ９００　画面遷移情報生成処理、Ｓ９

１５　フィルタリング処理、Ｓ９１６　補完処理

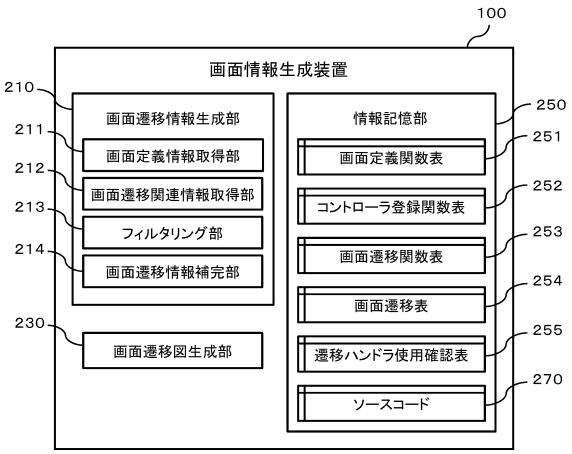
【図１】



【図２】



【図３】



【図４】

画面定義関数表251

411 サービス名	412 メソッド名	413 画面ID	4141 414 画面HTML	4142	4151 415 コントローラID	4152
		引数位置	引数位置	プロパティ名	引数位置	プロパティ名
$stateProvider	state	0	1	templateUrl	1	controller
…	…	…	…	…	…	…

【図５】

コントローラ登録関数表252

511 サービス名	512 メソッド名	513 コントローラID	5141 コントローラクラスの引数位置	514	5142
		引数位置	引数位置	配列内位置	
regService	regController	0	1	lastIndex	
…	…	…	…	…	

【図６】

画面遷移関数表253

611 サービス名	612 メソッド名	6131 遷移先画面ID 引数位置	613 遷移ハンドラが静的遷移か否かの判定 内容	6132	6133 使用	614 ラッピング階層数
$state	go	0	画面遷移関数の引数が即値		○	1
			画面遷移関数をラッピングした遷移ハンドラと画面遷移関数の引数が同じ変数を使用し、かつ、遷移ハンドラの引数が即値		×	
…	…	…	…			…

【図７】

画面遷移表254

71 遷移元情報			73	74 遷移先情報				72	75	
遷移元画面				遷移先画面					遷移ハンドラ	
遷移元画面ID	遷移元画面HTML	コントローラID	コントローラクラス	遷移先画面ID	画面定義有無	有効/無効	遷移条件	ハンドラ名	静的遷移	使用
top	top.html	topCtrl	TopCtrl	news	○	○	–	transToNews	○	○
				travel	×	×	–	transToTravel	○	○
				map	○	×	–	transToMap	○	×
				blog	○	○	プルダウンが"blog"の時	transToX	×	○
				game	○	○	プルダウンが"game"の時			
				sports	○	○	aが5以上の時	transToY	○	○
				movie	○	×	aが5未満の時	transToY	○	×
..	..	..	..	…	…	…	…	…	..	..

731 732 733 734 741 742 743 744 751 752 753

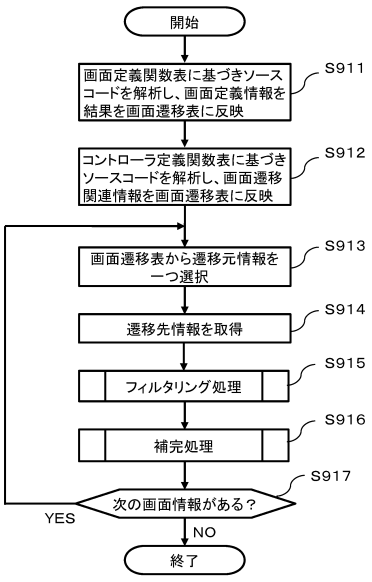
【図８】

遷移ハンドラ使用確認表255

811 タグ	812 属性
button	ng-click
…	…

【図９】

画面遷移情報生成処理S900



【図１０】

画面定義に関するソースコード

270

```
$stateProvider
.state( 'top' , {
    …
    'templateUrl' :  'top.html'
    'controller' :  'topCtrl' ,
    …
});
```

(a)

コントローラの登録に関するソースコード

270

```
regService.regController( 'topCtrl' ,
    [ '$scope' ,  '$state' , TopCtrl]
);
```

(b)

【図１１】

遷移ハンドラの定義に関するソースコード

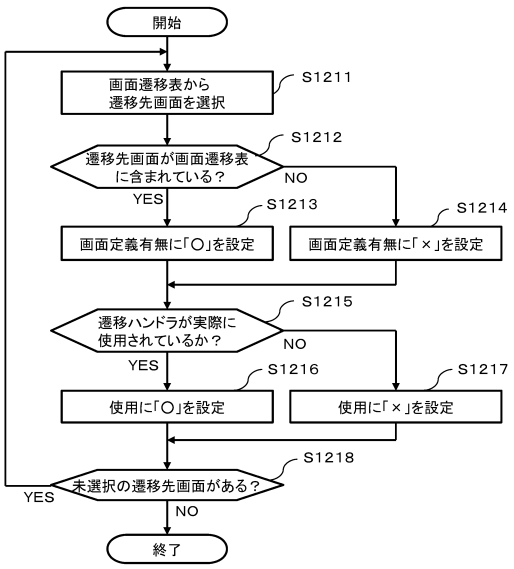
TopController.ts　270

```
class TopCtrl {
    constructor(…) {

        $scope.transToNews = () => {
            $state.go( 'news' );
        };
        $scope.transToTravel = () => {
            $state.go( 'travel' );
        };
        $scope.transToMap = () => {
            $state.go( 'map' );
        };
        $scope.transToX = (toScreen) => {
            $state.go(toScreen);
        };
        $scope.transToY = () => {
            var a = 10;
            if(a >= 5) {
                $state.go( 'sports' );
             } else {
                $state.go( 'movie' );
             }
        };
        …
    }
}
```

【図１２】

フィルタリング処理S915



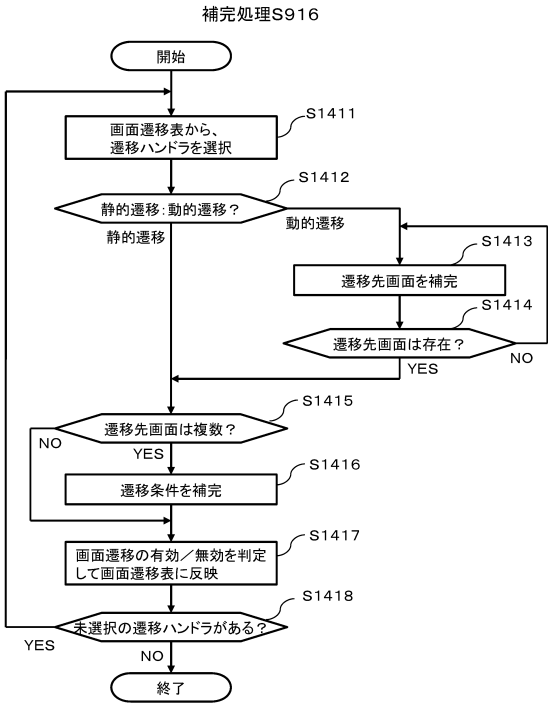
【図１３】

遷移ハンドラを使用するソースコード

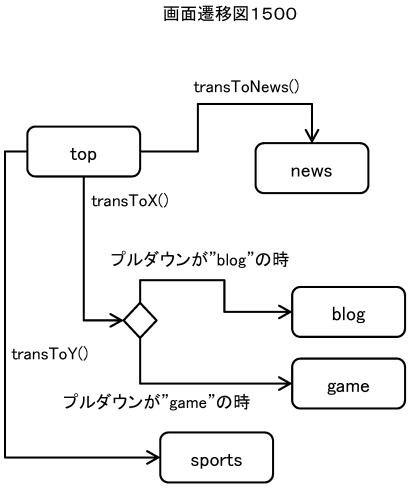
top.html　270

```
…
<button ng-click="transToNews()">
   ニュース画面へ
</button>
<button ng-click="transToTravel()">
   旅行画面へ
</button>
<button ng-click="transToX()">
  その他の画面Xへ
</button>
<button ng-click="transToY()">
  その他の画面Yへ
</button>
…
```

【図１４】


補完処理S916
開始
画面遷移表から、
遷移ハンドラを選択
S1411
静的遷移:動的遷移？
S1412
静的遷移
動的遷移
遷移先画面を補完
S1413
遷移先画面は存在？
S1414
NO
YES
遷移先画面は複数？
S1415
NO
YES
遷移条件を補完
S1416
画面遷移の有効／無効を判定
して画面遷移表に反映
S1417
未選択の遷移ハンドラがある？
S1418
YES
NO
終了

【図１５】


画面遷移図1500
transToNews()
top
news
transToX()
プルダウンが”blog”の時
blog
transToY()
game
プルダウンが”game”の時
sports

フロントページの続き

(72)発明者　中村　秀樹
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内

審査官　塚田　肇

(56)参考文献　特開２００１－０７５７８９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　８／２０
Ｇ０６Ｆ　　８／７３