

Method and System for Modelling or Transforming a Process

Background and Field of the Invention

5 This invention relates to method and system for modelling or transforming a process.

Complexity of business environment has been growing at an exponential pace. On the other hand, information systems that support a business have evolved from a centralised, mono platform into a decentralised, hybrid eco-system. Changes in
10 business landscape as a result of international effort to liberalise the free exchange of people, information, products and money in the global market, has asserted a much greater opportunity and challenge to the robustness and adaptivity of a system solution. As a result, it is common for an IT infrastructure of an organisation not to be unified with the demands of the business environment.

15

It is an object of the present invention to method and system for modelling or transforming a process which alleviates at least one of the disadvantages of the prior art and/or to provide the public with a useful choice.

20 Summary of the Invention

In a first aspect of the invention, there is provided a computer implemented method of transforming a process into a software application for performing a sequence of tasks of the process, the method comprising

25 (i) providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of

predetermined choice objects are representations of physical objects or information objects required for performing the interrelated task;

(ii) receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the sequence of tasks for constructing the process, the process activities being associated with a mapping module which maps each process activity to at least one executable program for performing the process activity; and

(iii) creating the software application for the process based on the mapped executable programs.

10

An advantage of the described embodiment of the invention is that with predetermined process commands and choice objects, this ensures repeatability and certainty in constructing the process. In this way, the method provides a technology enabler which allows organisations to define their core business model or process in an orderly, structured and simple way. The constructed process is then used to create the software application for performing the interrelated tasks of the process and as a result, achieves mapping or unification between a process, such as a business process/model, and information technology.

15

The constructed process may comprise more than one sequence of tasks, and the method may comprise converting each sequence of tasks as a sequence of events, and for each sequence of events, creating the software application based on the mapped executable programs associated with the process activities which defines the tasks corresponding to each sequence. This may involve the step of tracing paths from an end point to a start point of the constructed process to determine the sequence of events.

25

Preferably, the process flow comprises six sub-processes, and the method may comprises defining each of the interrelated tasks of the six sub-processes based on the predetermined process commands and choice objects to construct the process flow. The six sub-processes may be selected from the group consisting of: (i) representation and
5 customer interaction, (ii) translation, unification and optimization, (iii) service handling, (iv) configuration, (v) composition and computation, and (vi) intelligence. The method may further comprise the step of mapping relationship between the six sub-processes, the process commands and the choice objects in a 3-dimensional space. This would enable clear visualisation of the relationships of these components.

10

The predetermined process commands may be Fischer verbs or some other suitable predetermined terms/phrases.

The method may further comprise the step of generating a document which describes
15 the sequence of tasks of the process flow based on the process activities. In this way, the process flow may be documented for review and analysis.

The above method is particularly useful, although not exclusively, for a business process.

20

In a second aspect of the invention, there is provided a computer implemented method of transforming a process into a document for describing a sequence of tasks of the process, the method comprising

(i) providing a plurality of predetermined process commands and choice objects for
25 selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of

predetermined choice objects are representations of physical objects or information objects required for performing the task;

(ii) receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing

5 the sequence of tasks for constructing the process; and

generating a document which describes the sequence of tasks of the process flow based on the process activities.

In a third aspect of the invention, there is provided apparatus for transforming a process
10 into a software application for performing a sequence of tasks of the process, the apparatus comprising

(i) a representation interface for providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a
15 corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for performing the task;

(ii) an input interface for receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process
20 activities for suitably describing the sequence of tasks for constructing the process, each process activity being associated with a mapping module which maps the process activity to at least one executable program for performing the process activity; and

(iii) a creator for creating the software application for the process based on the mapped executable programs.

In a fourth aspect of the invention, there is provided apparatus for transforming a process into a document for describing a sequence of tasks of the process, the apparatus comprising

- 5 (i) a representation interface for providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for performing the task;
- 10 (ii) an input interface for receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the sequence of tasks for constructing the process; and
- iii) a generator for generating a document which describes the sequence of tasks of the process flow based on the process activities.

15

In a fifth aspect of the invention, there is provided a graphic modelling system comprising a mapping of choice objects, each choice object being a representation of a physical or information object; the mapping being arranged to illustrate contextual and/or spatial relationships between the choice objects; and means for transforming the mapping of choice objects into different user interfaces in accordance with user selection.

20

In this way, the graphic modelling system may provide an universal template for transforming the mapping of choice objects into the different user interfaces.

25

Preferably, the choice objects are represented by circles and the relationships are represented by arrows. Advantageously, the mapping is represented in 3-dimensional.

The transformation may include a plurality of converters for respective ones of the user interfaces.

- 5 The graphic modelling system may further comprise a transformation interface for receiving a first user interface; and a corresponding first user interface converter for extracting a structure of the first user interface and for converting the first user interface into the mapping of choice objects.

- 10 The mapping of choice objects may be used by a mapping module to identify at least one executable program for performing a process activity defined by at least some of the choice objects of the mapping.

In a sixth aspect of the invention, there is provided a method of converting between different user interfaces, the method comprising:

- 15 providing a plurality of user interfaces for selection by a user;
extracting a structure of a first user interface and converting the first user interface to an universal graphical representation based on the extracted structure; and
extracting a structure of the universal graphical representation and converting the universal graphical representation to a selected one of the plurality of user interfaces
20 based on the extracted structure of the universal graphical representation.

The sixth aspect may thus have the advantage of easily transforming or representing one user interface into/with a different user interface.

- 25 The universal graphical representation may include a mapping of choice objects. Each choice object may be a representation of a physical or information object defined in the

first user interface and the mapping may be arranged to illustrate contextual and/or spatial relationships between the choice objects.

Preferably, the graphical representation represents the choice objects with circles and the relationships are represented by arrows. Advantageously, the mapping is represented in 3-dimensional.

The plurality of user interface may include objected oriented user interface, application user interface, and web-based user interface.

10

In a seventh aspect, there is provided a user interface conversion system, the system comprising

a plurality of user interfaces for selection by a user;

a first converter configured to extract a structure of a first user interface and to convert the first user interface to an universal graphical representation based on the extracted structure; and

a second converter configured to extract a structure of the universal graphical representation and to convert the universal graphical representation to a selected one of the plurality of user interfaces based on the extracted structure of the universal graphical representation.

20

The universal graphical representation may include a mapping of choice objects. Each choice object may be a representation of a physical or information object defined in the first user interface and the mapping may be arranged to illustrate contextual and spatial relationships between the choice objects.

25

Preferably, the graphical representation may represent the choice objects with circles and the relationships are represented by arrows. Advantageously, the mapping is represented in 3-dimensional. The plurality of user interface may include objected oriented user interface, application user interface, and web-based user interface.

5

In an eighth aspect of the invention, there is provided an executable software module comprising

a main code portion when executed by a computer causes execution of an executable program for performing a task;

10 an input code portion for defining parameters for use by the program; and

a configurable code portion configurable by a user and is executable by a computer to augment the task and without affecting the performance of task.

15 Such a module may thus provide flexibility in expanding the functions of the task with ease.

20 The application may be external to the executable software code, and the main code portion may store a link to the external application. The configurable code portion may include a pre-rule portion arranged to be executed prior to the main code portion, and a post-rule portion arranged to be executed after the main code portion.

25 The pre-rule portion may be configured to transform data received the software module into a format compatible for use by the main code portion. The post-rule portion may be configured to transform an output of the software module into a format suitable for processing by another executable software module. The input code portion may include a model identifier for uniquely identifying the software module. The input code portion may also include an instance identifier for identifying an executing instance of the

software module. The input code portion may include choice objects defined by a user which are representations of physical or information objects.

In a ninth aspect of the invention, there is provided a computer implemented method of transforming a process into a software application for performing a sequence of tasks of the process, the method comprising

5 providing a plurality of executable software modules, each software module being mapped to respective process activities for defining the sequence of tasks and includes a main code portion when executed by a computer causes execution of a corresponding

10 executable program for performing one of the sequence of tasks

receiving selections of process activities by a user for defining the sequence of tasks;

associating the selection of process activities with a respective mapped software module;

creating the software application based on the mapped software modules according to

15 the sequence of tasks.

The main code portion of a said software module may provide a link to the corresponding executable program which resides out of said software module.

The computer implemented method may further comprise providing a mapping table for

20 mapping the plurality of executable software modules to corresponding executable programs. Preferably, more than one software module is mapped to a said corresponding executable program.

The process activities may comprise a plurality of predetermined process commands

25 and choice objects for selection by a user. Each of the plurality of predetermined process commands may define an action to be performed for a corresponding task, and

each of the plurality of predetermined choice objects may be representations of physical objects or information objects required for performing the task.

Each of the software modules may comprise a configurable code portion configurable by
5 a user and may be executable by a computer to augment the task performed by the corresponding executable program and without affecting the performance of task.

The computer implemented method may further comprise linking the mapped software modules to create the software application; and configuring the configurable code
10 portion of the mapped software modules to enable the linking.

The configurable code portion may include a pre-rule portion arranged to be executed prior to the main code portion, and a post-rule portion arranged to be executed after the main code portion. The pre-rule portion may be configured to transform data received
15 the software module into a format compatible for use by the main code portion.

In a tenth aspect of the invention, there is provided a computer implemented method of transforming a process into a software application for performing a series of interrelated tasks of the process, a plurality of predetermined process commands and choice objects
20 being provided for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding interrelated task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for performing the interrelated task, the method comprising:

25 selecting combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the series of interrelated tasks for constructing the process, each process activity being associated

with a mapping module which maps the process activity to at least one executable program for performing the process activity; and wherein the software application for the process is created based on the mapped executable programs.

5 Brief Description of the Drawings

An embodiment of the invention will now be described, by way of example, with reference to the accompanying drawings in which,

10 Figure 1 is a broad overview of a modelling system of a preferred embodiment of the present invention which comprises a BPSOD module, a CDM module and a RAPS module;

 Figure 2 is a block diagram illustrating six basic process units of the BPSOD module 12 of Figure 1;

15 Figure 3 is a flow chart illustrating general steps for defining a process of withdrawing money from a bank over-the-counter;

 Figure 4a is a table which provides examples of process commands used in the flow chart of Figure 3;

 Figure 4b is a table which provides examples of choice objects used in the flow
20 chart of Figure 3;

 Figure 5 shows a completed construction of the process "Representation and Customer Interaction" for the withdrawal of money process of Figure 3;

 Figure 6 shows a completed construction of the entire withdrawal of money process of Figure 3;

25 Figure 7 is a 3-dimensional relationship diagram between the six basic process units of Figure 2, choice objects and verbs of Figure 3;

Figure 8 is an expanded flow chart showing the process of Figure 5 in more detailed steps;

Figures 9a, 9b and 9c show sequence of scenarios based on the flow chart of Figure 8;

5 Figure 10 is an example of an object structure for a voucher, one of the choice objects provided by the CDM module of Figure 1;

Figure 11 shows an organisational map of the choice object "voucher" of Figure 10;

10 Figure 12 is a pictorial representation illustrating multiple modes of user interface provided by the CDM module of Figure 1;

Figure 13a shows a structure of a process atom and Figure 13b shows a process molecule formed from a number of process atoms of Figure 13a;

Figure 14 shows an example of how a software application comprises a number of different process atoms of Figure 13;

15 Figure 15 is an example of an object structure of "customer identity", one of the choice objects provided by the CDM module of Figure 1;

Figure 16 is an example of an object structure of "withdrawal information", one of the choice objects provided by the CDM module of Figure 1;

20 Figure 17 is an example of an object structure of "Account Info/Balance", one of the choice objects provided by the CDM module of Figure 1;

Figure 18 is an example of an object structure of "Cash Voucher", one of the choice objects provided by the CDM module of Figure 1;

25 Figure 19 is a look-up-table illustrating how selections of process commands of Figure 4a and choice objects of Figure 4b are mapped to respective process atoms of the RAPS module of Figure 1,

Figure 20 is a graphical 3D representation (object universe) of choice objects of the CDM module of Figure 1;

Figures 21 and 22 are magnified and further magnified views respectively of portion E of Figure 20;

Figure 23 is a magnified view of the portion F of Figure 22;

Figures 24 to 26 illustrate how the object universe of Figure 20 is rotatable about
5 a vertical plane to show the 3D representation of the choice objects; and

Figures 27 and 28 are magnified and further magnified views of Figure 26.

Detailed Description of the Preferred Embodiment

10 It is known that the nature of a business is dynamic, interactive, decisive and adaptive. On the other hand, nature of an information technology (IT) system is virtual and possesses unlimited possibilities (probably constrained by advances of technology) with no physical constraints. Due to overt properties and behaviour of the nature of the business and IT system, a direct mapping from "real world" to "virtual world" is difficult.
15 Thus, the preferred embodiment of the present provides a unified platform for the business and technical realms to eliminate communication gaps from these two domains.

Figure 1 is a broad overview of a modelling system 10 of a preferred embodiment of the
20 present invention which comprises a BPSOD module 12, a CDM module 14 and a RAPS module 16. The modelling system 10 includes an executor 11 such as a main processor 11 which controls the operations of the BPSOD module 12, the CDM module 14 and the RAPS module 16. It should be appreciated that each of these modules may have their own processor, rather than a common one.

25

An advantage of the modelling system 10 is that it provides an end-to-end framework and technology enabler that allows organisations to define their core business needs

and processes in an orderly, structured and simple way. Such a modelling system 10 also allows businesses to respond more rapidly to changes in market demand and continuously improves internal processes to reduce cost and improve service quality. The system 10 also fosters delegation of work and clear definition of accountability down to each single process helps to highlight operational risks. These advantages would
5 become clearer with the following explanation on each of the modules 12,14,16 of the system 10.

BPSOD (Business Process and Strategy on Demand)

10

An overview of the different components of the BPSOD module 12 is shown in Figure 2. BPSOD provides a structured framework to define a business process based on the following rules:

15 (i) Business processes are formulated as algebraic structure comprising a series of activities connected by arrows;

(ii) Each activity is a combination of a process command 100 and a choice object 102.

The process command 100 is part of a list of predetermined standard commands or
20 verbs which are used to describe the activity or task to be performed. The choice object 102 describes a physical object in a real/physical world or an information object in an information/virtual world. The structure of the choices and objects are clearly defined. Each activity has a set of structured property that further qualifies and explains the details of the activity.

25

Examples of standard commands are Fischer verbs and these are listed in Figure 4a and it has been found that these Fischer verbs are able to describe adequately tasks or

activities relating to banking, retail or the airline industry. It should be clarified that the process command 100 is an action which may be defined by six characteristics. For most cases, the process command 100 is a verb which satisfies various rules of mathematical composition.

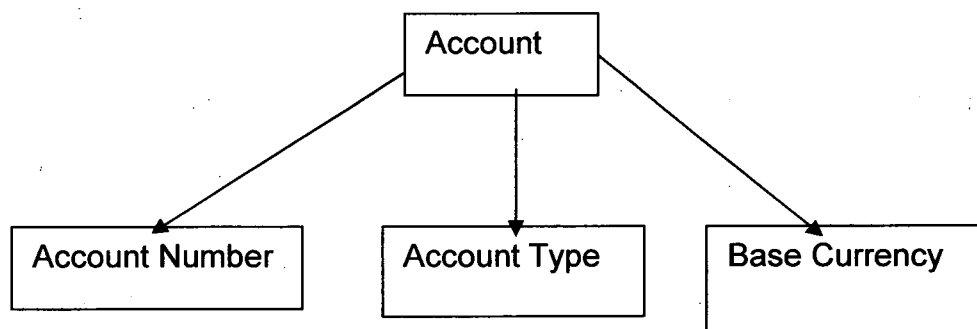
5

Examples of choice objects 102 for selection by the user are listed in Figure 4b.

By using a standard vocabulary (process command 100 and choice objects 102) which is selected by users to define the activity, this ensures repeatability and thus, allows
10 organisations to define a process in an orderly, structured and simple way. Each process command 100 is associated with a software program configured to perform a task associated with the Process Command.

Each choice object 102 in the BPSOD module 12 is associated with an object structure
15 defined in the CDM module 14. Take an example of "Open Account" as an activity, "Open" is the process command 102 (see Figure 4a) and "Account" is the choice object 102. In the CDM module 14, the choice object "Account" is defined as an object structure which corresponds to a structure of information such as
->Account Number, Account Type, Base Currency. To represent this graphically:

20



25

Each object structure defined in the CDM module 14 is associated with physical data in a real world, either directly or indirectly. In the latter case, the object structure may correspond to another choice object in the CDM module 14, and the another choice object may correspond with a further choice object etc, before ending with an associated physical data in the real world.

(iii) A business processes in the modelling system 10 is defined by six types of purpose driven sub-processes performed by corresponding basic process units or sub-modules of the BPSOD module 12, namely

10 1) Representation and customer interaction 18a – Interface sub-module 18: refer to user interface, input/output devices such as Point-of-Sale systems (POS), Automated Teller Machines (ATM); mobile phones, printouts or equipment for interacting or communicating with customers or clients etc.;

 2) Unification, Optimization and translation 20a – Translator and Optimizer 20:
15 serve the purpose of mapping different message format to another format and route to the correct recipient. For example, this sub-process may convert an input from the representation and customer interaction sub-process and convert it into another language which can be understood by the subsequent processes. As a further example, there could be different input devices from the representation and customer interaction
20 sub-process and the translation, unification and optimization sub-process then unifies the inputs from the different input devices and converts the inputs into a format suitable for subsequent processing;

 3) service handling 22a – Service Handler 22: this sub-module 22 provides the requested service On Demand and is triggered by events. This sub-module 22
25 comprises computer programs that run independently to serve a specific purpose for example, interest computation, update customer address etc.;

4) configuration 24a – Configurator 24: this sub-module systematically groups system parameters that serve similar purposes, for example, loan product configuration, GL posting transaction code. The configurator 24 controls the system's behaviour and how the system 10 processes the requests for sub-processes (3) and (5);

5) computation and composition 26a – Computation sub-module 26: this sub-process refers to batch processing that is triggered ON TIME basis due to availability of information and it is optimum to process the request on batch basis;

6) business intelligence 28a – Intelligence sub-module 28: this sub-module 28 performs applications like MIS reporting, data warehouse analytics that look at hind-sight history in order to monitor and improve the process based on after-the-fact information.

Each of the basic business units 18,20,22,24,26,28 includes an input and an output linked by structures therebetween. Each of these basic business units represents a group of unique characteristics that serve specific purposes in the most effective and efficient way. As a result, it helps to optimize resource utilization and reduce total cost of ownership (TCO). The grouping of the activities of a process flow into the six sub-processes 18a,20a,22a,24a,26a,28a or segments provides a clear and end-to-end picture of how a business process executed by its stakeholder is related to components in a complex information technology (IT) system. Typical business process documentation captures either from a business perspective or from a system's perspective. By the use of the six segment approach, this allows business users and technical experts to complement each other's knowledge about how a process is done from the user-end to the background process in a machine and vice-versa.

A simple business process of a customer withdrawing money over a bank counter will be used to describe how the BPSOD module 12, and specifically the six basic process units 18,20,22,24,26,28, are used to define the business process. Broadly, the business

process comprises the following sequence of activities: the customer walks into a branch of a bank and fills up a withdrawal form. He then presents the withdrawal form to a bank officer over the counter. The bank officer authenticates the identity of the customer, verify the account status and balance, and if the information tallies with the bank's records, the bank officer proceeds to dispense the cash to the customer together with a receipt.

Based on the rules of the BPSOD module 12, each business process is defined by the six sub-processes 18a,20a,22a,24a,26a,28a explained earlier and a user has to define the withdrawal process based on the six sub-processes and Figure 3 shows an overview of the process flow.

A first step is to define the first sub-process "Customer Engagement and Delivery" 18a and at step 300, the user identifies the activities that make up the process. At step 302, the user defines each activity using process commands 100 and choice objects 102 stored in a BPSOD database 310 and these are used as inputs to the BPSOD module 12 (see Figure 2).

For example, the user would select the process command "Capture" from the list in Figure 4a plus the choice object "Customer identity" to describe the process of the bank officer obtaining the customer identity. The user then continues to select a command from the list of process commands in Figure 4a and choice objects in Figure 4b which best describes subsequent activities under the first sub-process 18a and a completed sub-process 18a is shown in Figure 5. In other words, the user has defined the first sub-process "Representation and Customer Interaction" 18a as the following activities or tasks:

- a) Capture Customer Identity 500;

- b) Enter Withdrawal information 502;
- c) Validate Withdrawal information 504;
- d) Calculate Balance 506;
- e) Print Voucher 508; and
- 5 f) Keep Voucher copy 510.

At step 304, the user moves on to define the activities for the next sub-process "Unification Optimization Translation" 20a. Again, at step 300, the user identifies the activities which make up this sub-process and at step 302, proceeds to select the process commands 100 from Figure 4a which describes the activity and likewise the choice objects 102. In this embodiment, and referring to Figure 6, these are defined by the user under Unification Optimization and Translation 20a:

- a) Receive withdrawal information 620 – this receives the withdrawal information from the first sub-process;
- 15 b) Validate withdrawal information 622 – this validates the source of the withdrawal information;
- c) Translate withdrawal information 624 – this converts the withdrawal information into a suitable format for processing the next sub-process 22a;
- d) Route withdrawal information 626 – this routes the validated and translated (if
20 necessary) withdrawal information to the next sub-process 22a.

The user then defines the third sub-process, Service Handling 22a, and these are defined by the user as:

- a) Validate account information 628 – this validates the account information from
25 the withdrawal information;
- b) Write records 630 – write a transaction record of the withdrawal into a database;

- c) Update records 632 - update records of account information for example, balances;
- d) Send Response 634 – send new account information such as balance to the first sub-process “Representation and Customer Interaction” 18a.

5

It should be appreciated that the first three sub-processes 18a,20a,22a may be considered as “front-end” and are performed in real-time. Subsequent sub-processes may be considered the “back-end” and may be event or time triggered, rather than carried out in real-time.

10

The next sub-process “Configuration” 24a is defined by the user as “Refer information” 636. This sub-process defines the rules and configuration for the third sub-process “Service Handling” 22a for example, how to validate account information 628 in the third sub-process 22a.

15

The fifth sub-process is computation and composition 26a and the following process commands 100 and choice objects 102 are defined:

- a) compute interest 638;
- b) compute service fees 640 – service fees incurred in the user performing the service of withdrawing the money;
- c) Generate reports 642 – the reports include account entries;
- d) Distribute reports 644 – selecting this distributes the reports to relevant parties;

20

The sixth sub-process is the business intelligence 26a process in which the first to fifth sub-processes are reviewed and analysed and here, the user has selected the following process commands 100 and choice objects 102:

25

- a) Generate compliance reports 646 – based on the analysis, this combination of process command and choice object generates reports for regulation compliance to improve the overall process;
- b) Distribute compliance reports 648;
- 5 c) Print compliance reports 650;
- d) Verify compliance reports 652.

The completed definition of the sub-processes is shown in Figure 6 and the process of Figure 3 ends, with the end result being that the cash is handled over to the customer.

10

When the user completes defining the activities for the six basic process units, the user completes constructing the entire process flow for the withdrawal of money. It should be apparent that having such a structured framework, this cultivates the user to properly align and sort activities according to their characteristics so as to classify the activities

15 into the respective six basic process units.

20

25

At this stage, it is possible to construct a 3-dimensional relationship diagram 700 between the six basic process units 702, the choice objects 102 and the process commands 100 which define the activities, as shown in Figure 7. Through the use of the relationship diagram 700, the user can identify any activity which is inconsistent between its purpose and its underlying characteristics. For example, if the relationship diagram shows the choice object “customer identity” is related to the verb “Capture” and also the second basic process unit of “unification, optimization and translation”, this should raise alarm bells since the customer identity should be captured by the Interface module 18 and not by the Translator and optimizer 20. The 3-dimensional relationship diagram 700 depicts all possible combinations of the three parameters – process units 702, choice objects 102 and process commands 100. Each combination (illustrated as a point in

space – see for example point 701 in Figure 7) is what may be selected by the user in constructing a process, and is associated with a computer program for performing the activity defined by the process command 100 of the combination.

- 5 The next step is to transform the constructed process flow into “scenarios”. To do this, the system 10 employs a reverse path tracing method by identifying the end point and starting point and reverse tracing the paths from the end point to the starting point. In the example of Figure 6, the end point would be “cash in my hand” and the reverse tracing would trace backwards through each of the sub-processes starting from “Intelligence” to
- 10 “Composition and computation” and so on until “Customer Engagement Delivery”.

For ease of explanation, the first process “Representation and customer interaction” performed by the Interface module 18 is used as an example to explain how the reverse path tracing works to transform the process into scenarios. However, it would be

15 apparent that the same process applies to the other sub-processes or basic process unit 20,22,24,26,28.

Figure 5 represents an ideal construction of the flow of the first basic process unit. However, in reality, there are check points or decision points along the way such that

20 options are available depending on the circumstances, and Figure 8 shows an expanded flow of Figure 5. Steps 800 to 804 correspond with the first three steps of Figure 5. As part of the validation step 804, at step 806, the customer identity is checked against the bank’s database and if such a customer exists, the process goes to the step 808. On the other hand, if such a customer identity does not exist, the flow jumps to step 816 in

25 which the withdrawal request is rejected.

At step 808, the amount to withdraw is checked against the customer's balance and if the amount to withdraw is less than or equal to the balance, the process proceeds to step 810 in which the balance is calculated and then the voucher is printed at step 812 and the customer keeps the voucher at step 814. On the other hand, if the amount to
5 withdraw is greater than the balance, the flow jumps to step 818 in which the customer is asked to re-enter the withdrawal amount and then step 808 is performed again.

As it can be appreciated, there are different permutations depending on whether the identity of the customer matches with the banks' records and whether the withdrawal
10 amount is appropriate or not. Based on these permutations, the system 10 transforms the process of Figure 8 into sequences of scenarios based on these permutations and these are shown in Figures 9a to 9c. Figure 9a shows the scenarios which take place when the amount entered by the user is checked at step 808 to be appropriate (i.e. the amount is less than the balance) and the ID is verified at step 806 to be valid. Figure 9b
15 shows the scenarios which take place when the amount entered by the user is checked at step 808 to be inappropriate (i.e. the amount is more than the balance) with correct ID. Figure 9c illustrates the scenarios which take place when the ID is checked at step 806 to be incorrect.

20 As explained earlier, each process command 100 is associated with a software program for carrying out the task described by the command and based on the sequence of tasks/events which make up the various possible scenarios of Figures 9a-9c, the system 10 transforms the flow automatically into a software application for performing the tasks/events of each scenario. For example, in the scenario of Figure 9a, capture
25 customer identity would call a software program associated with "capture" which prompts the bank officer to input the identity of the customer into the system 10, and "enter" would call up another software program for the purpose of entering the

withdrawal information. In this way, the process of withdrawing money has been transformed from a "business world" to the "IT/virtual" world through the construction of the process flow based on the six sub-processes or basic process units.

- 5 The system 10 may also provide other outputs, for example, generation of a document which explains the possible scenarios obtainable from the business process of withdrawal money. The scenarios which are derived as shown in Figures 9a-9c can also be used as comparisons with each other for missing elements or process breaks to verify the integrity of the process flow. Further, with the systematic approach of
- 10 constructing the process, the process is broken down into different tasks and events and this is helpful to identify non-compliance in the process flow. As shown in Figure 9a-9c, it is also possible to represent a process flow in different scenarios which makes understanding and appreciating the complexity of the process flow much easier.
- 15 As it can be appreciated, the BPSOD module 12 benefits an organisation in managing, analyzing and optimizing business processes in an effective and efficient way. The benefits may be realised via the following functions and features:
- a) standardised vocabulary to describe the activity and process;
 - b) maintain a chronology log of changes in business processes;
 - 20 c) comparison of as-is and to-be process prior to actual implementation;
 - d) capability of defining business rules to verify processes in order to identify potential vulnerability which could be exposed as a risk factor. The system helps organisations to proactively identify possible exposures ex-ante.
 - e) generate printable documents;
 - 25 f) display business processes in intuitive, multiple view, and interactive graphical approach;

- g) powerful and interactive search of business processes. Since a standard vocabulary is used to define or model a process, it is possible to search based on a specific process command 100 or specific choice object 102 to call out processes which use the specific process command or choice object. Further, the search may be conducted on combinations of process commands 100 and choice objects 102 or sequence of combinations of process commands and choice objects. The search may also be conducted based on specific stakeholder involved in the process (a stakeholder is a person or group responsible for the outcome of the process);
- h) navigation of large process repository via process and purpose map;
- i) simulation of business process flow by iterating through the flow of activities and inter-related business process in an animated environment;
- j) manage, group/sort, versioning, assign and import/export business process projects for collaboration and work delegation;
- k) analyse processes and activities, such as estimating cost, lead time and work load, tracing dependency and lineage of processes etc. For example, based on the scenarios of Figures 9a-9c, each activity of a scenario may be associated with properties such as time and costs required to perform the activity. As a result, it is possible to estimate the costs and time (or other characteristics) for performing the entire process.

Choice Determination Method (CDM)

A main function of the CDM module 14 is to organise the choice objects 102 which are used by the BPSOD module. A choice object 102 represents a physical object or an information object.

Take for example, the activity "Print Voucher" with "Voucher" as the choice object 102.

Figure 10 shows an example of such a voucher 30 with the following fields:

- a) Name 32: Lawrence
- b) Withdrawal amount 34: \$100
- 5 c) Balance 36: \$900
- d) Date and Time 38: 1 March 2010

Figure 11 shows an object structure or organisational map 40 of a database for the choice object "Voucher" 30. As it can be appreciated from Figure 11, there are two
10 formats V101 and V102 and the user needs to perform a sub-selection of which format to choose. In this case, V101 is the format selected to give the output of the voucher illustrated in Figure 10.

For the system 10 to know where to retrieve the information associated with each choice
15 object, it should be appreciated that this is via the BPSOD module 12 in which in the sub-process 18a, when "Capture Customer identity" 500 is executed, a software program in RAPS associated with the process command "Capture" captures the entered customer identity and the identity is defined in the CDM module as a relationship model shown in Figure 15. The captured information is then pulled and used as information to
20 update the voucher in the third sub-process "Service Handling" 22a and in this case the customer's name "Lawrence" was entered in the first sub-process 18a and thus, this name appears in the voucher of Figure 10.

A similar procedure applies for the other choice objects "Withdrawal amount" 34,
25 "balance" 36 and "date and time" 38, and each of these are defined in CDM as illustrated in Figures 16 to 18. It would be appreciated that the choice objects are represented with circles and the relationships are represented by arrows.

Basically, the choice object organisation map of the CDM module 14 (such as the one shown in Figure 15) defines the source or origin of information which are used by the RAPS module. In other words, the CDM module 14 functions as a linkage between the BPSOD module 12 and the RAPS module 16. Also, the CDM module 14 may be used as a standalone or independent module to model and analyse choice objects and their structures. Through the mapping, CDM allows an organisation to construct, analyse and synthesize the universe of choice objects which reflects the organisation's core business.

The following functionalities are provided by CDM:

- Clear definition of choice objects 102 through the mapping;
- Delineate the complete structure of choice objects 102 which not only define the choice objects in words, but also defining relationship of each choice object with other choice objects to illustrate contextual and spatial relationships between the choice objects 102;
- Organise the choice objects to align to core business. For example, a bank's core business may be considered to "finance" customers and receiving "deposits" from "customers". Thus, when a user organises the choice objects, he has to align the choice objects as a "child" to one of these choices "finance", "deposit" and "customer".;
- Holistic map of enterprise model that unifies IT-eco system;
- Tracing of dependency between objects and between processes and objects which can be appreciated from a 3-dimensional representation discussed below;
- Compare structure change of choice objects 102 for different versions. Using Figure 11 as an example, it is possible to compare and appreciate the different

voucher structures and formats between the two types V101 and V102. Further hierarchical relationships between choice objects 102 may be presented;

- Generate standardized documents;
- 3-dimensional representation of object hierarchy and this may be termed as an object universe 104 which is shown in Figure 20. The object universe 104 comprises a mapping of all the choice objects 102 and the spatial and contextual relationships between the choice objects 102 are represented by arrows 108, such as the choice object "customer identity" shown in Figure 15. Figures 21 and 22 are magnified and further magnified views respectively of the portion E of Figure 20 and Figure 23 is a magnified view of the portion F of Figure 22. As it may be appreciated from Figure 23, the choice objects 102 which have closer relationships are clustered together to form clusters 106. This makes appreciation of the choice objects 102 and the relationships with other choice objects 102 easier.

Figures 24 to 26 show how the object universe 104 of Figure 20 may be rotated about a vertical plane to show the 3D representation of the object universe 104 and thus, the relationships between the choice objects 102. Figures 27 and 28 are magnified and further magnified views of Figure 26 to show the clusters 106 and choice objects 102 in the third dimension.

It is also envisaged that the 3-dimensional effect may be achieved through 3-D stereoscopic view to analyse the data structure.

Searching based on this 3-dimensional representation utilizes a concept of interrelationship between choice objects. Taking Figure 15 as an example, when a user searches for biometric identity, the CDM module 14 would show "Customer Information", "Fingerprints" and "Iris Pattern" as closer spatial proximity to each

other, as compared with the other sub-objects such as "Document identity" or "Name". The spatial relationship in the representation illustrates how search results are interrelated and from the 3-dimensional representation, one can appreciate that "Fingerprints" and "Iris Patterns" are sub-concepts of "Biometric Identity" and "Customer Identity. In this way, hierarchical perspectives and topological distributions showing how "close" the choice objects are related can be displayed.

- Auto construction of the CDM 3-dimensional representation directly from application databases. The CDM includes converters configured to convert between corresponding application databases and the CDM organisational structures, and if an application database is converted into the CDM organisational structure, then the applicant database may also be presented in 3-dimensional. This is further elaborated below.

An important feature of the CDM module is that it is able to generate multiple modes of user interfaces. A pictorial representation is shown in Figure 12, which shows a reduced size and simplified version of the CDM organisational map 30 of Figure 11 with eight different interfaces – Webpage 120, GUI screen 121, Data Dict (Data Dictionary) 122, SQL 123, ERD (Entity Relationship Diagram) 124, DFD (Data Flow Diagram) 125, Use case 126, Doc/Excel 127.

It is appreciated that different user interfaces created using different software applications (i.e. the data or codes of each application) comprise structures. For example, HTML code for a webpage may have 3 layers of structures "Title", "Header" and "Body" of the HTML code.

Thus, the CDM converters include a webpage-CDM converter 128 which extracts the structure of the HTML code (i.e. webpage 120), converts and represents the structure of the HTML code in CDM as a CDM organisational map i.e. in circles and arrows, and the CDM module 14 is further configured to convert the converted CDM organisational map into an application user interface, for example SQL. In this case, the CDM converters include a SQL-CDM converter 129 configured to extract the structure of the converted CDM organisational map and convert the structure and content into an equivalent SQL code.

- 10 As it can be appreciated from the above explanation, the CDM module 14 functions as a common or universal platform or graphical representation for the different application databases 120-127, so that conversion between the different application databases is performed via the CDM module 14. In this way, the CDM module 14 is able to achieve conversion efficiency (as compared to converting between user interfaces directly) because the converters may be designed or written in advance since the conversion to or from is always from the CDM module 14.

With the CDM module 14, it is also possible to link choice objects automatically based on preset rules, when it may not be possible using the original user interface. For example, the CDM module may be organised by using product names or names in the choice objects. The preset rule may specify that names of choice objects starting with a particular alphabet should be linked together and in this way, relationship between choice objects in the CDM module 14 may be established automatically. Also, if the choice objects have a common identification, for example, from a particular region or country, it is possible to create the entire map of the CDM module 14 based on the preset rules.

Rules and Purpose Section (RAPS)

The RAPS module 16 comprises a depository of process “atoms” 40 or building blocks of executable software codes, which are reusable and specific purpose programs.

5

An example of a structure of the process atom 40 is shown in Figure 13a for performing the command “validate” withdrawal info 504 of Figure 5 which includes the following elements:

a) Pre-Rules 41 – Together with post-rules (see below), Pre-Rules augment the functions of the process atom 40. Pre-rules (or pre-processing rules) is used to transform data received by the process atom 40 to a format which is recognised and processed by the process atom 40. The Pre-Rules may also enrich the data and precalculation of data. The pre-rules also keep track of looping and looping conditions to invoke the process atoms iteratively.

15 b) Input segment 43

This provides the input data for processing by the process atom 40 and includes:

i) Model ID – this refers to a unique model identifier or name to identify the process atom 40. In this case, the model ID identifies the process atom 40 for the “validate” process command. There is a minimum of one process atom in a sequence model – there is no upper limit of the cardinality of a sequence model.

ii) Instance ID - this refers to a unique instance identifier to identify an executing instance of a sequence of the process atom

iii) Data – this is the information required to carry out the validation and includes “Customer Name”, “Amount”, “Date and Time”, “Balance etc.”

25

c) Processing segment 47 –

- i) Process ID – this refers to a unique process identifier to identify an executing instance of a process atom in an executing instance of a sequence of atoms.
 - ii) Process Type – this identifies programming language of the process, for example, Java, Fortran, .NET etc.
 - iii) URL – the URL provides information to define where and how to invoke the underlying process of the molecule and in this case, the “validate” process. The “where” identifies where the executable application is located which may reside in another hardware different from where the process atom 40 resides. The “how” provides the method to communicate with programming language defined in the Process Type.
- d) Output segment 47
- i) This provides the output of the process atom 40 after being processed by the processing segment 45. It should be appreciated that the input segment 43 feeds the data into the processing segment 45 and the output of the process atom 40 is provided at the output segment 47. In this case, the result of the validation is provided in this segment 47.
 - e) Post-rules 49 – Post-rules (or post-processing rules) transform dataset output of the process atom to another format. The post-processing rules also perform logic computation to control which process atom receives the dataset.

The process atom 40 defines or represents a specific purpose executable program. The actual executable program (i.e. the program code) may reside within or outside of the RAPS module 16. As explained above, the process atom 40 includes detailed information of the actual executable program, such as method used to invoke the program, location of where the program resides, input dataset and output dataset

required by the program. The RAPS module 16 may be configured to combine a number of process atoms to create a "process molecule" 42 – analogous to chemistry where a molecule is made up of a number of atoms.

5 Figure 13b shows an example of how four process atoms 40a, 40b, 40c, 40d are combined to form the process molecule 42. Each of the process atoms 40a, 40b, 40c, 40d has a similar structure as the process atom reference by numeral 40, and input segments 44a, 50, 44b, 44c and output segments 46a, 52, 46b, 46c are similar to the input segment 43 and output segment 47 of the process atom 40. Figure 13b only shows
10 part of the process atom 40a, 40b, 40c, 40d in the molecule 42 (i.e. without showing the Pre-Rules, Post-Rules etc).

The process atom 40b functions as a control gate which has a processing segment 48 defining a Rule ID and Rule Type with two outputs "T" for True or "F" for False and the
15 Rule ID and Rule Type determine which path should data received by the input segment 50 and output by the output segment 52 travel from the process atom 40a to the next process atom (i.e. either process atom 40b or 40c). Although not shown, the control gate 48 also includes a set of pre-rules and post-rules similar in purpose as the pre-rules and post-rules of the process atom.

20

The process molecule has a flexibility of combining different process atoms 40a, 40b, 40c, 40d to perform a desired activity or task and this flexibility would be appreciated from Figure 14. Figure 14 shows an example of how applications or process molecules A, B, C and D comprise a number of different process atoms. The elements
25 marked by A1, A2, AB1, B1, BC1, C1 are different process atoms similar in structure as process atom 40, although each of which defines a software program when executed is configured to perform a particular task or function which may be different from the

process atom 40. Application A is created based on three such atoms A1, A2 and AB1, and Application B comprises the process atoms AB1 and B1, and so on and so forth. Since both Applications A and B comprises a different set of process atoms, they perform separate functions. As it can be appreciated, by selecting and combining
5 different process atoms 40, higher level application programs are formed and the ingenuity is that no recompilation of the process atoms 40 is needed. This is because the RAPS module 16 performs virtual mapping between a process atom 40 and the corresponding executable program. The virtual mapping allows the RAPS module 16 to “see” and access the executable program, but not the other way around. The RAPS
10 module 16 access the executable program via the executor 11 which is configured to interface the executable program and the RAPS module 16 so as to pass dataset between the two parties. Since the executable program is a standalone program, the executable program does not need to be “aware” of the existence of the RAPS module 16 or the other process atoms. In an executing environment, each atom in a process
15 molecule 42 does not communicate directly with each other. Each of the process atoms has a two way communication with the RAPS module 16 via the executor 11, and hence the RAPS module 16 decides what dataset to pass to which atom and which atom will be invoked and when it would be invoked. A sequence of process atoms (i.e. the process molecule 42) in the RAPS module 16 is a “virtually” compiled collection of
20 programs. The RAPS module 16 does not need all the process atoms to be compiled into a single program.

For businesses, the process atoms and the RAPS module can achieve significant increases in efficiency and effectiveness. RAPS is an excellent way of implementing
25 process integration across multiple platforms through the use of the executor 11. Instead of one executor 11, there may be a set of executors developed for the purposes of facilitating communication for programs in different software platforms to communicate

with the RAPS module 16. The RAPS module 16 provides the ability for an organisation to quickly adapt or create new business processes because of the flexibility of the process atom.

5 Reference is now made to the embodiment of Figure 6 to explain further how the process atom is derived. Each of the process atoms is associated with a pre-developed program and how the pre-developed programs are associated with the process commands and choice objects are based on a look-up table. As illustrated in Figures 4a and 4b, a list of process commands and choice objects are provided for selection by a
10 user. Each of the possible combinations of the process commands and choice objects are mapped in a look-up table to a respective process atom for carrying out the defined actions or activities. An example of a look-up table is shown in Figure 19 with the row A, B, C, D etc representing the choice objects of Figure 4b and the column P1, P2, P3, P4 etc representing the process commands of Figure 4a. The references ATM1, ATM2,
15 ATM3 etc represent process atoms having a similar structure as the process atom 40 which are mapped to possible combinations of choice objects A, B, C, D etc and process commands P1, P2, P3, P4 etc...

In other words, when the user selects the process command and choice objects to
20 define actions or activities of a business process such as the one in Figure 5, each of the combination of process command and choice objects is already mapped to a process atom for carrying out the defined action or activity.

The pre-developed programs are developed based on the process commands and
25 choice objects which are available for selection by a user, and for ease of explanation, let's assume that 100 of such programs are pre-developed and named as PGM1...PGM100. This set of programs may encompass different types of computer

program operating on different software platforms, and located in different physical or virtual machines.

Next, a set of process atoms similar to the process atom 40 is created from the set of programs and for ease of explanation, let's assume that 150 process atoms are created and named as ATM1...ATM150. Multiple atoms ATM1....ATM150 may be mapped to a same program e.g. ATM5 and ATM8 are mapped to PGM7. Although ATM5 and ATM8 are mapped to the same program, they have different pre and post processing rules 41,49 that made them distinct to each other. The mapping between process atoms and pre-developed programs may be named MAP1.

With MAP1 developed, this is where the look-up table of Figure 19 is constructed. As explained earlier, the look-up table of Figure 19 is a mapping between an individual activity (i.e. a combination of process command and choice object) to one and only one process atom, although multiple activities may be mapped to the same process atom. The look-up table of Figure 19 may be reference as MAP2. For example, the activities "Compute Simple Interest" and "Compute Compound Interest" may both be mapped to the process atom ATM8 (it should be appreciated that both activities have a common process command "Compute" and different choice objects "Simple Interest" and "Compound Interest".)

When a user constructs a process (say PR1) by selecting activities in the BPSOD module from the set of available combinations of process commands and choice objects, the user also links these activities in sequential order, similar to what is illustrated in Figures 5 and 6.

The RAPS module 16 then creates a new system based on the constructed process PR1. Firstly, each activity in PR1 is mapped or associated with a process atom using MAP2. Secondly, the sequential order of the activities in PR1 is then converted to a sequence order for the process atoms (say SEQ1). Thirdly, the pre-developed programs PGM1...PGM100 that correspond to each atom are invoked from MAP1 according to SEQ1.

It should also be appreciated that the structure of the process atom 40 makes it very easy to add more functions, through a process called augmentation. For example, take the example of the activity "validate withdrawal info", which is to be performed by process atom ATM1. The URL of the process segment of the process atom ATM1 points to a pre-developed program which resides in another server which is adapted to carry out the activity "validate withdrawal info". However, the user now wants to carry out a batch process of "validate withdrawal information". An inefficient way would be to rewrite the pre-developed program for such a function or create a new pre-developed program for such a function. However, this would mean having to recompile the rewritten program or the new program which is inefficient.

Instead, the present embodiment proposes a more efficient way of carrying out such a function through the use of the pre-rule 41 and post-rule 49 to accommodate the new function. For example, the pre-rule 41 and post-rule 49 of the process atom ATM1 may be modified by the user to create a counter and loop instructions to carry out the batch validation thus, leaving the pre-developed program intact. In the alternative, a new process atom may be created with suitable changes to the pre-rule 41 and post-rule 49 but still pointing to the same pre-developed program. With these ways, an advantage is that this does not affect the execution of the pre-developed program which resides in another hardware. It should also be appreciated that the pre-rule and post-rule may be

adapted to carry out other functions, other than what is described here. It would thus be appreciated that the pre-rule 41 and post-rule 49 are a configurable portion of the process atom 40 and these may be configured by a user to augment the task performed by the pre-developed program.

5

As it can be appreciated, the process atoms 40 provide the links (via URL in process segment 45) to executable pre-developed programs but exceptions may occur such that there is no response from the executable program. This may happen in four possible scenarios:

- 10 1) the executable program failed to execute because of an error and is aware that an exception has occurred. The executable program then sends an error message to the executor 11;
- 2) the executable program crashes and did not inform the executor 11 ;
- 3) the executable program goes in an infinite computation loop and could not
15 respond to the request from RAPS module 16.
- 4) Similar to scenario (1) above, but the error message is not received by the executor 11.

When exceptions occur, the executor 11 may perform error recovery and two methods
20 are proposed here. For example, a process atom R1 is running and its process segment requires it to jump to a pre-developed program which resides in another server. The pre-developed program did not respond and this causes an exception to occur. Under the circumstances, the executor 11 may do one of two things. A first option is to write to an error file and hold the instruction of the process atom R1 in abeyance and attempt to
25 recover the pre-developed program. When the pre-developed program is active again, use the error file to run the pre-developed program and update the records.

An alternative is to use redundancy. For example, for mission critical pre-developed programs, a copy of such programs are created and stored as backup. If the original program fails to execute, then the back-up copies are used. Such redundancy is useful in handling exceptions but it does take up memory space.

5

As it can be appreciated above, a business process automation using the RAPS module 16 is able to compose multiple process atoms to create process molecules; hence creating a completely new business software application for end users. The RAPS module 16 also allows a programmer to describe a new business process from a collection of process building blocks formed by reusable, purpose specific process atoms. Through the use of RAPS, this reduces cost of developing new applications and overall quality because applications are now constructed from a set of tested software components, instead of created from scratch.

10

15 The RAPS module 16 provides a graphical framework for describing logic to control and coordinate process atoms participating in a process flow. The framework is interpreted and executed by the RAPS engine (i.e. executor 11, although it may be a separate and independent engine). The engine coordinates all of the activities of the process, and controls the system's corrective activities when exceptions occur.

20

The RAPS module liberates integration and process tasks from underlying processes and objects so that the processes and objects may be more effectively changed, managed and optimised by providing a unified integration framework for program-to-program and people-to-people integration.

25

The described embodiment should not be construed as limitative. For example, although the described embodiment relates to a process for withdrawing money, the present invention may also be applied to other processes.

- 5 Having now fully described the invention, it should be apparent to one of ordinary skill in the art that many modifications can be made hereto without departing from the scope as claimed.

CLAIMS

1. A computer implemented method of transforming a process into a software application for performing a sequence of tasks of the process, the method comprising
- 5 comprising
- (i) providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical
- 10 objects or information objects required for performing the task;
- (ii) receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the sequence of tasks for constructing the process, the process activities being associated with a mapping module which
- 15 maps each process activity to at least one executable program for performing said process activity; and
- (iii) creating the software application for the process based on the mapped executable programs.
- 20 2. A computer implemented method according to claim 1, wherein the constructed process comprises more than one sequence of tasks, and the method comprises converting each sequence of tasks as a sequence of events, and for each sequence of events, creating the software application based on the mapped executable programs associated with the process activities which defines the
- 25 tasks corresponding to each sequence.

3. A computer implemented method according to claim 2, further comprising the step of tracing paths from an end point to a start point of the constructed process to determine the sequence of events.
- 5 4. A computer implemented method according to any preceding claim, wherein the process flow comprises six sub-processes, and the method comprises defining each of the interrelated tasks of the six sub-processes based on the predetermined process commands and choice objects to construct the process flow.
- 10 5. A computer implemented method according to claim 4, wherein the six sub-processes is selected from the group consisting of: (i) representation and customer interaction, (ii) translation, unification and optimization, (iii) service handling, (iv) configuration, (v) composition and computation, and (vi)
15 intelligence.
6. A computer implemented method according to claim 4 or 5, further comprising the step of mapping relationship between the six sub-processes, the process commands and the choice objects in a 3-dimensional space.
- 20 7. A computer implemented method according to any preceding claims, wherein the predetermined process commands are Fischer verbs.
8. A computer implemented method according to any preceding claim, further
25 comprising the steps of generating a document which describes the sequence of tasks of the process flow based on the process activities.

9. A computer implemented method according to any preceding claim, wherein the process is a business process.
10. A computer implemented method of transforming a process into a document for
5 describing a sequence of tasks of the process, the method comprising
(i) providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical
10 objects or information objects required for performing the task;
(ii) receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the sequence of tasks for constructing the process; and
15 generating a document which describes the sequence of tasks of the process flow based on the process activities.
11. Apparatus for transforming a process into a software application for performing a sequence of tasks of the process, the apparatus comprising
20 (i) a representation interface for providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for
25 performing the task;
(ii) an input interface for receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form

respective process activities for suitably describing the sequence of tasks for constructing the process, each process activity being associated with a mapping module which maps the process activity to at least one executable program for performing the process activity; and

- 5 (iii) a creator for creating the software application for the process based on the mapped executable programs.

12. Apparatus for transforming a process into a document for describing a sequence of tasks of the process, the apparatus comprising

- 10 (i) a representation interface for providing a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for performing the task;

(ii) an input interface for receiving input which selects combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the sequence of tasks for constructing the process; and

- 20 (iii) a generator for generating a document which describes the sequence of tasks of the process flow based on the process activities.

13. A graphic modelling system comprising

- 25 a mapping of choice objects, each choice object being a representation of a physical or information object; the mapping being arranged to illustrate contextual and/or spatial relationships between the choice objects; and

means for transforming the mapping of choice objects into different user interfaces in accordance with user selection.

14. A graphic modelling system according to claim 13, wherein the choice objects
5 are represented by circles and the relationships are represented by arrows.
15. A graphic modelling system according to claim 13 or 14, wherein the mapping is represented in 3-dimensional.
- 10 16. A graphic modelling system according to any of claims 13 to 15, wherein the transformation means include a plurality of converters for respective ones of the user interfaces.
- 15 17. A graphic modelling system according to any of claims 13 to 16, further comprising a transformation interface for receiving a first user interface; and a corresponding first user interface converter for extracting a structure of the first user interface and for converting the first user interface into the mapping of choice objects.
- 20 18. A graphic modelling system according to any of claims 13 to 17, wherein the mapping of choice objects are used by a mapping module to identify at least one executable program for performing a process activity defined by at least some of the choice objects of the mapping.
- 25 19. A method of converting between different user interfaces, the method comprising:
providing a plurality of user interfaces for selection by a user;

extracting a structure of a first user interface and converting the first user interface to an universal graphical representation based on the extracted structure; and

extracting a structure of the universal graphical representation and converting the universal graphical representation to a selected one of the plurality of user interfaces based on the extracted structure of the universal graphical representation.

20. A method according to claim 19, wherein the universal graphical representation includes a mapping of choice objects; each choice object being a representation of a physical or information object defined in the first user interface; the mapping being arranged to illustrate contextual and/or spatial relationships between the choice objects.
21. A method according to claim 20, wherein the graphical representation represents the choice objects with circles and the relationships are represented by arrows.
22. A method according to claim 20 or 21, wherein the mapping is represented in 3-dimensional.
23. A method according to any of claims 19 to 21, wherein the plurality of user interface includes objected oriented user interface, application user interface, and web-based user interface.
24. A user interface conversion system, the system comprising a plurality of user interfaces for selection by a user;

a first converter configured to extract a structure of a first user interface and to convert the first user interface to an universal graphical representation based on the extracted structure; and

a second converter configured to extract a structure of the universal graphical representation and to convert the universal graphical representation to a selected one of the plurality of user interfaces based on the extracted structure of the universal graphical representation.

25. A user interface conversion system according to claim 24, wherein the universal graphical representation includes a mapping of choice objects, each choice object being a representation of a physical or information object defined in the first user interface; the mapping being arranged to illustrate contextual and spatial relationships between the choice objects.
26. A user interface conversion system according to claim 25, wherein the graphical representation represents the choice objects with circles and the relationships are represented by arrows.
27. A user interface conversion system according to claim 25 or 26, wherein the mapping is represented in 3-dimensional.
28. A user interface conversion system according to any of claims 24 to 27, wherein the plurality of user interface includes objected oriented user interface, application user interface, and web-based user interface.
29. An executable software module comprising

a main code portion when executed by a computer causes execution of an executable program for performing a task;

an input code portion for defining parameters for use by the program; and

a configurable code portion configurable by a user and is executable by a computer to augment the task and without affecting the performance of task.

30. An executable software module according to claim 29, wherein the application is external to the executable software code, and the main code portion stores a link to the external application.

31. An executable software module according to claim 29 or 30, wherein the configurable code portion includes a pre-rule portion arranged to be executed prior to the main code portion, and a post-rule portion arranged to be executed after the main code portion.

32. An executable software module according to claim 31, wherein the pre-rule portion is configured to transform data received the software module into a format compatible for use by the main code portion.

33. An executable software module according to claim 31 or claim 32, wherein the post-rule portion is configured to transform an output of the software module into a format suitable for processing by another executable software module.

34. An executable software module according to any of claims 29 to 33, wherein the input code portion includes a model identifier for uniquely identifying the software module.

35. An executable software module according to claim any of claims 29 to 34, wherein the input code portion includes an instance identifier for identifying an executing instance of the software module.
- 5 36. An executable software module according to any of claims 29 to 35, wherein the input code portion includes choice objects defined by a user which are representations of physical or information objects.
- 10 37. A computer implemented method of transforming a process into a software application for performing a sequence of tasks of the process, the method comprising
- 15 providing a plurality of executable software modules, each software module being mapped to respective process activities for defining the sequence of tasks and includes a main code portion when executed by a computer causes execution of a corresponding executable program for performing one of the sequence of tasks
- receiving selections of process activities by a user for defining the sequence of tasks;
- 20 associating the selection of process activities with a respective mapped software module;
- creating the software application based on the mapped software modules according to the sequence of tasks.
- 25 38. A computer implemented method according to claim 37, wherein the main code portion of a said software module provides a link to the corresponding executable program which resides out of said software module.

39. A computer implemented method according to claim 38, further comprising providing a mapping table for mapping the plurality of executable software modules to corresponding executable programs.
- 5 40. A computer implemented method according to claim 39, wherein more than one software module is mapped to a said corresponding executable program.
41. A computer implemented method according to any of claims 37 to 40, wherein the process activities comprises a plurality of predetermined process commands and choice objects for selection by a user, each of the plurality of predetermined
10 process commands defining an action to be performed for a corresponding task, and wherein each of the plurality of predetermined choice objects are representations of physical objects or information objects required for performing the task.
- 15 42. A computer implemented method according to any of claims 37 and 40, wherein each of the software modules comprises a configurable code portion configurable by a user and is executable by a computer to augment the task performed by the corresponding executable program and without affecting the
20 performance of task.
43. A computer implemented method according to claim 42, further comprising linking the mapped software modules to create the software application; and configuring the configurable code portion of the mapped software modules to
25 enable the linking.
44. A computer implemented method according to claim 42 or 43, wherein the

configurable code portion includes a pre-rule portion arranged to be executed prior to the main code portion, and a post-rule portion arranged to be executed after the main code portion.

- 5 45. A computer implemented method according to claim 44, wherein the pre-rule portion is configured to transform data received the software module into a format compatible for use by the main code portion.
- 10 46. A computer implemented method of transforming a process into a software application for performing a series of interrelated tasks of the process, a plurality of predetermined process commands and choice objects being provided for selection by a user, each of the plurality of predetermined process commands defining an action to be performed for a corresponding interrelated task, and wherein each of the plurality of predetermined choice objects are representations
- 15 of physical objects or information objects required for performing the interrelated task, the method comprising:
- 20 selecting combinations of the plurality of process commands and the plurality of choice objects to form respective process activities for suitably describing the series of interrelated tasks for constructing the process, each process activity being associated with a mapping module which maps the process activity to at least one executable program for performing the process activity; and wherein the software application for the process is created based on the mapped executable programs.