Journal of Simulation (2015), 1–17

www.palgrave-journals.com/jos/

Business process simulation in the context of enterprise engineering

Y Liu* and J Iijima

Tokyo Institute of Technology, Tokyo, Japan

Traditional business process-based discrete event simulation is not sufficiently powerful to support change-related business process reengineering (BPR). The simulation methods are inadequate for describing large, complex systems and are difficult to change so as to simulate new BPR designs. The limitations are caused by (1) focusing on workflow; (2) low-abstraction conceptual models; and (3) a lack of separate opinions on design and implementation. Our research is based on enterprise engineering, a construction-based approach, rather than workflow. By clarifying the differences between and dependencies of the ontological and implementation models, this research proposes a generic framework for generating a modularized and component-based simulation model with increased reusability as well as with the capacity to make controllable changes in enterprise simulation. The method is assumed to assist BPR by reducing complexity and through its focus on enterprise engineering-based design thinking and human-centered interactions.
Journal of Simulation advance online publication, 9 January 2015; doi:10.1057/jos.2014.35

Keywords: enterprise ontology; DEMO; conceptual modelling; business process simulation; enterprise engineering; BPR

1. Background

Business process reengineering (BPR) concerns enterprise changes. However, many researchers have argued that the high failure rate of many real-life BPR projects is attributable to the lack of tools for evaluating the effects of designed solutions before their implementation (Tumay, 1996; Paolucci *et al*, 1997). On one hand, traditional process models (PMs), for example, UML, BPMN, flowcharts and EPC, are widely used in business process modelling. However, such models are criticized for their weakness in inspecting and analysing business processes (Vergidis *et al*, 2008). For example, these models are said to be weak at providing the necessary means for identifying bottlenecks and for performance analysis or for generating alternative improved business processes in terms of specified objectives.

On the other hand, business process simulation (BPS) (Scholz-Reiter *et al*, 1999) is considered a powerful tool to assist in change analysis and effectiveness evaluation due to its ability to measure performance, test alternatives and engage in processes (Greasley and Barlow, 1998). On the basis of a number of reviews (Aguilar-Savén, 2004; Netjes, 2006; Jahangirian *et al*, 2010), simulation for business process design and engineering is the second most important area in simulation studies (following only scheduling).

However, current simulation methods are weak at describing large, complex systems. As some researches (Barber *et al*, 2003; Jahangirian *et al*, 2010) have noted, most business process improvement projects consider only a single process without a holistic view of the enterprise, and complexity is increased when small individual PMs are joined into a large hierarchical construct. Thus, it is inefficient to utilize these methods in process change analysis that concerns the entire enterprise.

Another limitation of BPS is the complexity of changing models to simulate new designs in BPR (Greasley, 2003). Simulation is useful in comparing 'as-is' and 'to-be' models to validate the effects of change and ensure the completeness of the model, but it has limited ability to design a 'to-be' model. Most of the BPS literature restricts itself to comparing the before and after conditions and provide little information on the redesign process itself (Reijers and Liman-Mansar, 2005).

Moreover, as Valiris and Glykas (2004) noted, most of the reengineering methodologies 'lack the formal underpinning to ensure the logical consistency of the generation of the improved business process models'. This issue leads to the lack of a systematic approach that can guide a process designer through a series of repeatable steps to achieve process redesign. Vergidis *et al* (2008) argued that 'a structural and repeatable methodology that could be generally applied to business process modelling and improvement was never established'. This situation is a problem for business process modelling as well as for BPS.

The authors consider the reasons for the limitations described above from two aspects: (1) workflow viewpoint and (2) the low abstraction levels in conceptual modelling.

(1) Workflow viewpoint Many of the existing BPS studies take a naive view of business processes. According to Davenport *et al* (1994) and Hammer and Champy (1993), business processes are defined as structured, measured sets of input–output activities that produce value for a particular customer. In contrast to the

*Correspondence: Y Liu, Department of Industrial Engineering and Management, Tokyo Institute of Technology, Ookayama 2-12-1-W9-66, Meguro-ku, Tokyo 152-8552, Japan.

traditional system view, which emphasizes what is produced from 'a black box', process emphasizes how things are done from a business logic perspective (Davenport *et al*, 1994). Wang and Brooks (2007) showed that the most widely used representation technique for BPS is the flowchart (used by 63% of simulation modellers). Other modelling methods, such as BPMN, UML and EPC, are also widely used. All of these modelling tools are founded on the workflow viewpoint, which aims to represent the sequences of real-world work. Thus, the workflow-based discrete event simulation is more widely used in BPS than is agent-based simulation, although the latter is considered a promising method for analysing enterprise as a social system (Siebers *et al*, 2010).

The advantage of the workflow viewpoint is clear: it can help us understand how things are done in a step-by-step manner. However, this method occasionally leads us to over analyse the details such that we are not able to grasp the full image of the system: It is difficult to analyse tens of A4-paper-based flowcharts or UML models to understand why actions are performed, what the relationships are and, correspondingly, how to make changes that can confirm the consistency of the system before and after. Workflow-based simulation by default has the same problem: it occasionally delves too deeply into process details on 'how to?' but is weak in analysing 'why?' and 'what?'. Moreover, models from the workflow viewpoint are typically non-modularized but described as a sequence of activities. Oriented from this viewpoint, it is difficult to conceive of a systematic approach that can guide a process re-designer through a series of repeatable steps. Meanwhile, enterprise is not considered as a whole, and human interactions are not emphasized when considering workflow. In summary, the workflow viewpoint neither reduces the complexity of modelling nor facilitates change. Workflow-based BPS is not an adequate solution for supporting enterprise reengineering.

(2) Low abstraction levels in conceptual modelling Conceptual modelling is regarded as the most important and difficult step, but it is also the least investigated step in simulation, especially in BPS. As indicated by Banks *et al* (2013), there are surprisingly few books and academic papers on the subject of conceptual models of enterprise-related simulation. Robinson (2006) defined conceptual modelling as the 'representation of the abstracted world, expressed by means of diagrams and written text'. Turnitsa *et al* (2010) provided an extended definition: 'a formal specification of a conceptualization' and 'an ontological representation of the simulation that implements it'. One example of an ontology-based conceptual model for simulation is the system entity structure (SES) proposed by Zeigler *et al* (2000) and utilized in a number of Zeigler's discrete event simulation studies. However, SES emphasizes only the system's data structure. That is, it is considered weak at describing enterprises as social systems. Thus, it is less applicable in the context of business processes. In contrast, most of the conceptual modelling methods used in process-related simulation consider not the ontology level but the implementation level of enterprise, for example, flowcharts, BPMN, UML or onto-UML (Guizzardi and Wagner, 2012). These models do have semantic definitions of business processes related to how systems work. They are widely used because the simulations are entirely at the implementation levels with which they best match. However, as was criticized in multiple studies (Chen, 1976; Salimifard and Wright, 2001), these models do not describe enterprise functionality, and thus they are difficult to be used by management for decision-making support. Moreover, there are no clear definitions of highly abstracted enterprise ontology. These weaknesses make it difficult or impossible to apply one conceptual model in different implementations that support the same ontology structure, as is required in business process redesign and reengineering. Furthermore, the unstructured and non-ontological conceptual model leads to low modularity, low reusability and uncontrollable changes in simulation modelling, especially when the simulation is employed for BRP.

To avoid the limitations of workflow-based business process modelling and simulation, the authors took a construction viewpoint on enterprise in the context of enterprise engineering (Dietz *et al*, 2013).

Enterprise engineering is an integrated set of disciplines for building or changing an enterprise, its processes and its systems (Martin, 1995), in which an enterprise is considered an intentionally created cooperative of human beings with a certain societal purpose (Dietz and Hoogervorst, 2012; Dietz *et al*, 2013) and with highly organized complexity (Hoogervorst, 2009). In other words, it analyzes enterprise as an interrelated, highly complex social system rather than as a workflow. Enterprise engineering considers that low BPR success rates occur when enterprise phenomena are not comprehensively understood, and cannot be addressed adequately, hence the nature of necessary changes cannot be determined (Dietz *et al*, 2013). Furthermore, 'an enterprise must operate as a unified and integrated whole' (Dietz *et al*, 2013). Successful reengineering follows design thinking, which concerns understanding the enterprise and its strategic intentions that are to be operationalized, as well as arranging for these to happen (Dietz *et al*, 2013).

The core theory related to design thinking is enterprise ontology, Design and Engineering Methodology for Organization (DEMO) (Dietz, 2006). DEMO is a human-centered, highly abstracted ontology that describes how an enterprise is constructed. It answers such questions as 'What is produced?' in its *fact model* (FM); 'Why?' and 'Who?' in its *construction model* (CM); and 'How?' in its PM and *action model* (AM). DEMO assists in grasping the essence of an enterprise by looking inside its daily activities, like an X-ray (Dietz *et al*, 2013). It abstracts the real world without considering implementation details to allow for an understanding of the essence of the system. In other words, by defining a DEMO model, we can understand the core, stable part of an enterprise. This higher-level abstraction reduces the complexity of modelling social systems and aids in understanding the strategic intentions that are to be operationalized in the design phase.

An enterprise is an implementation of its ontological model, and there could be different possible implementations for the

same ontology in achieving the same goal. In most cases, BPR relates to implement the same enterprise ontology differently. However, ontology is not sufficient for simulation. We use simulation to understand implementation problems such as optimization or as-is/to-be analysis. These are closely related to how to arrange the ontology to happen because ontology could have different implementations in the same strategic instances.

To take advantage of enterprise ontology, the authors proposed DEMO++, an expanded DEMO with an implementation model, as a conceptual modelling method for BPS. There are two parts: ontology and implementation. The former describes how the enterprise is constructed, and the latter describes how the interesting part of an ontological model is implied. It is a method for analysing, executing and evaluating business processes that reduces the complexity in modelling and simulating large systems but focuses on implementing the interesting part. DEMO++ is fully modularized such that the generated simulation is entirely component-based, with features of controllable changes and reusability. It supports BPS, especially when simulation is used in top-down enterprise reengineering.

The remainder of this paper is organized as follows. First, the research design is introduced in Section 2. Then, background knowledge on DEMO is explained with a case study in Section 3; the expanded DEMO++ is defined in Section 4. This approach is validated in Section 5 with a model built in AnyLogic. Finally, a brief discussion and future work are given in Section 6.

2. Research design

Zeigler *et al* (2000) suggested a framework for discrete event modelling and simulation. A *conceptual model* is built through a collection of assumptions about system components and the interactions between them, which involves some degree of abstraction of systems operations. The *operational model* is the executable model that implements the conceptual model. Our research is designed following Zeigler's conceptual and operation model structures. There are three levels:

- ontology;
- conceptual models (expanded ontology with implementation);
- operation models.

At the ontology level, we use DEMO to describe the real world at a high level of abstraction. DEMO has been proven to be able to support discrete event systems in a number of studies (Barjis *et al*, 2001; Barjis *et al*, 2002; Barjis, 2007, 2008, 2010).

At the conceptual level, we define a DEMO-based conceptual model called DEMO++ that expands and integrates ontology with implementation models.

At the operation level, we choose AnyLogic (XJ Technologies, 2009) as the execution environment to test DEMO++. There are different types of simulation environments. AnyLogic is selected for its excellent user interface and hybrid simulation support capability. Since for the purpose of enterprise reengineering and enterprise transformation, we need to consider not only business activities from the workflow view but also actor interactions from the agent view and possibly the effectiveness and efficiency of the full system from the system dynamics view to obtain the full image of how the system works and the potential effects of change. AnyLogic is the only simulation software with powerful support for hybrid simulation that combines discrete events, an agent base, system dynamics and other methods.

The entire structure is validated with a case study on Buono Pizza.

3. DEMO

3.1. Enterprise ontology

Enterprise ontology DEMO is a core theory of enterprise engineering. The goal is to offer a new understanding of enterprises such that one is able to look through the distracting and confusing actual appearance of an enterprise into its essence (Dietz *et al*, 2013).

DEMO was founded based on *ψ theory*, which considers an organization an interaction of individual social subjects. A *subject* 'enters into and complies with commitments regarding the products/services that they bring about in cooperation' (Dietz and Hoogervorst, 2014). A *product* is an independently existing fact (eg, 'pizza order #002 has been delivered'). Subjects bring about products by performing *production acts* (or Pacts for short). Meanwhile, subjects enter into and make commitments towards each other regarding products by performing *coordination acts* (Cacts for short). The effects of Cacts/Pacts are called Cfacts/Pfacts. Cacts and Pacts occur in universal patterns called *transactions*. A transaction involves two subjects, an *initiator* who generates a request and an *executor*, who produces products (Dietz and Hoogervorst, 2014). Transactions are the elementary (essential) organizational building blocks of enterprises (Perinforma, 2012). Enterprises have dozens of different processes, such as for production, purchasing and logistics. Despite their different natures, they all share the same underlying transaction patterns, with similar coordination and production routines (Dietz *et al*, 2013). A *basic transaction pattern* is described in Figure 1. A coordination process begins when an initiator 'requests' (rq) a product. The executor responds to the request by 'promising' it (pm) and then produces it. After he/she 'states' (st) that the product has been produced, in response to this event, the initiator 'accepts' (ac) the produced product. The four *intentions* (rq, pm, st, ac) are basic steps for a successful transaction. The effect of each intention leads to some state change of the world, for example 'proposition requested', 'result produced' (Dietz and Hoogervorst, 2014).

To obtain a full image, the ontological model of an organization is divided into four sub-models that describe different aspects of the complete model (Figure 2). The CM, located at the top of the triangle, is the most concise, describing how transactions and actor roles are composed to construct a system; PM describes the detailed causal relationships and constructions in processes; FM describes the objects and facts related to the process; and

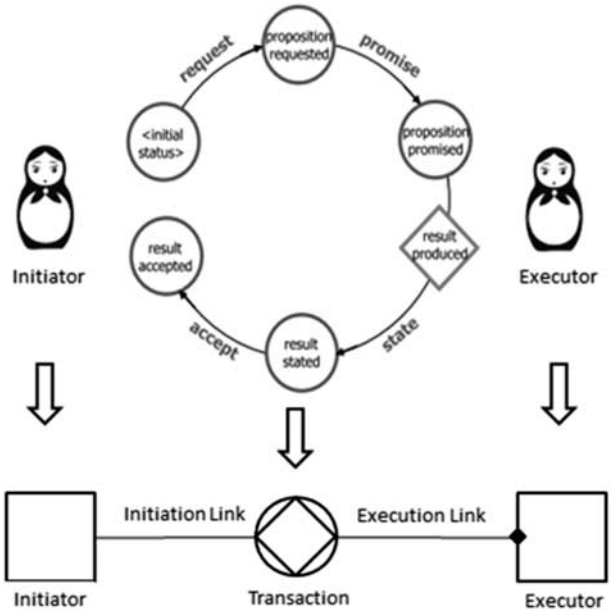

Figure 1 Basic transaction pattern.

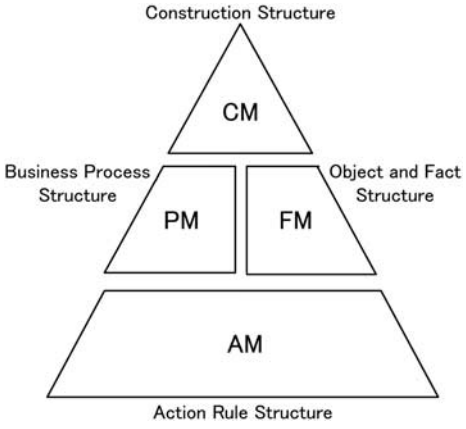

Figure 2 DEMO aspect models.

AM describe the action rules for actor roles. With these models, DEMO proposes a consistent, coherent, concise, comprehensive and essential (C4E) representation.

Details of the DEMO aspect models are explained by analysing a real-life pizza shop in the following subsections.

3.2. Case description: Buono Pizza

Buono, a pizza shop, is located in a small city in Japan, with one manager, Mr. Inoue, and four young adults on a part-time basis.

They use an order-taking system to help manage the orders. Thirty percent of the orders are taken by phone, 60% are from the Internet and only 10% are directly handled at the store counter. For all three methods, ordering takes a minimum of 1 min, 3 min on average and occasionally up to 5 min.

The order is entered in the IT system, an empty box is pulled and a label with the customer's name, address, order and phone number printed on it is applied to the box (this duration is included in the ordering time). Normally, an available worker reads the waiting orders in the IT system and prepares the pizza dough. This step takes an average of 2 min. The staff will then add the requested toppings; this step takes an average of 3 min. This step can only be performed if the table is available; the table can fit four pizzas. Three ovens are available. Baking takes 6 min, and the time to place the pizza in the oven is negligible.

When the pizza exits the oven, it rests on the packing table. An available worker takes the pizza and puts it in the box with the label printed on it. This step takes an average of 1 min. Then, an available deliverer will deliver the prepared pizza to the customer's location and receive the payment. Buono's delivery area is within a driving distance of 10 min (5–10 min one way). The four staff members in the store can do all of the work, but at least one staff person always remains in the store.

As a courtesy, if the wait time from placing the order to delivery is more than 30 min, Buono will provide a free pizza in your next order.

However, Mr. Inoue found that too many customers were not receiving their orders within 30 min (more than 20%). Therefore, it was necessary to find possible solutions to reduce the free pizza requirement. He hoped that simulation could provide some advice.

3.3. DEMO construction model

In DEMO, every transaction is of some type, called a *transaction type* (T for short). Each T involves two *actor roles*, who are authorized to commit or produce the facts generated by the transactions. An actor role acts as either the initiator or executor of a transaction (cf Section 3.1). Actor roles are either elementary or composite: *elementary actor roles* (A for short) are actor roles within the system of focus, and they can be the executors of only one transaction type; *composite actor roles* (CA for short) represent actor roles that are not focused. A composite actor role can execute more than one Ts. Ts and related actor roles (A and CA) are described in the *organization construction diagram* (OCD) of the DEMO CM. As shown in Buono's OCD (Figure 3), we abstracted four transaction types—T1, T2, T3 and T4. A1 (the order completer) is an elementary actor role as the executor of T1 and the initiator of T2, T3 and T4. CA1 (the customer) is a composite actor role because his/her behaviour is out of the scope of focus. Transaction types and actor roles are connected by links. There are two types of links: An *initiator link* is a link from an initiator (source of the link) to its transaction type (the target of the link), represented as a line. An example is the link from CA1 to T1 shown in Figure 3. An *executor link* is a link from a transaction type (the source of the link) to its executor (the target of the link), represented as a line with a black diamond at the end. An example is the link from T1 to A1, shown in Figure 3.

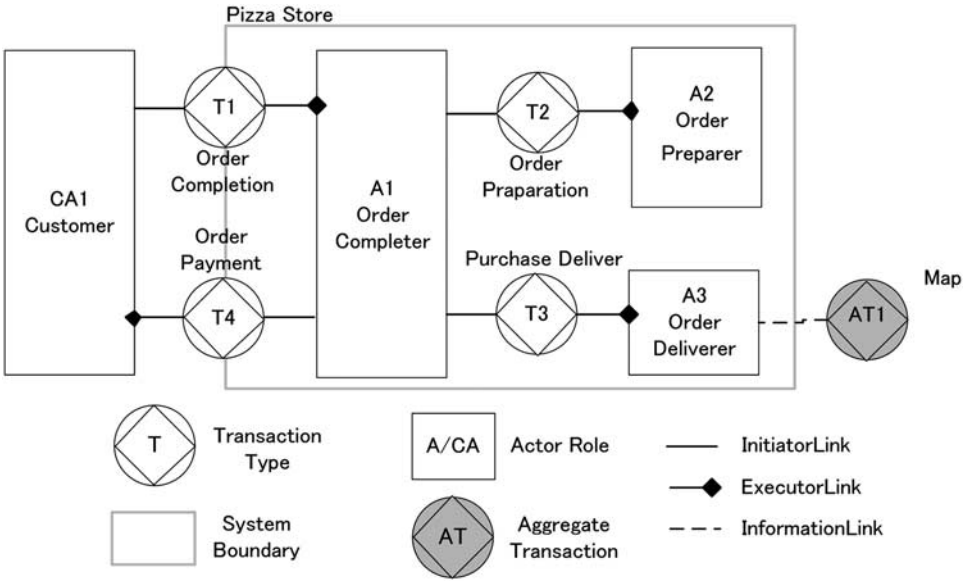

Figure 3 Buono Pizza's OCD.

Table 1 Buono Pizza's Transaction Product Table

Transaction type	*Product type*
T1 order completion	P1 order has been completed
T2 order preparing	P2 order has been prepared
T3 order delivery	P3 order has been delivered
T4 order payment	P4 order has been paid

Transactions of the same type regard products of the same type, called *product type* (P for short). Transaction type and the corresponding product types are described in the *transaction product table* (TPT) of the CM. As shown in Buono's TPT (Table 1), there are four product types, P1, P2, P3 and P4, defined according to the four transaction types.

Aggregate transaction type (AT for short) represents a transaction type that belongs to the system environment. With Buono, an aggregate transaction type AT1 contains map information that is obtained externally for delivery. All of the generated facts (cf Section 3.1) are stored in either Ts or ATs. These Ts and ATs are *information banks*. The actor roles can access the information banks to set information. Links between actor roles and information banks are called *information links*, for example, the link between A3 and AT1. Initiator and executor links can both serve as information links.

3.4. DEMO process model

CM briefly describes how an organization is constructed. The PM located below the CM in Figure 2 describes transaction details as well as how they are interrelated.

In the PM, transactions are expanded into transaction patterns, in which the routines and effects of acts (described as facts) are defined. There are three types of transaction patterns: basic, standard and complete.

A *basic transaction pattern* (cf Section 3.1) describes the simplest 'happy path' to accomplishing a transaction, including the following intentions: rq, pm, st and ac.

A *standard transaction pattern* considers both the 'happy path' and exceptions, in which the transaction may be stopped or redone. In addition to the basic transaction pattern, there are four new intention types: {decline (dc), quit (qt), reject (rj), stop (sp)}. As described in Figure 4, a transaction begins when its initiator requests a product. The executor responds to the request with a decision on whether to promise or decline the request following some action rule (cf Section 3.6): If a request is declined, the process will enter a negotiation stage, and as a result of negotiation, the initiator can choose to re-request or to cease cooperating. The situation is the same when a statement is rejected by the initiator: the executor can choose whether to restate or stop the process (Dietz, 2006). In transaction patterns, a link from an act (source of the link) to a fact (target of the link) is called a *fact link*, indicating that one fact is the effect of the act, represented as a line. A link from a fact (source of the link) to an act (target of the link) is called a *response link*, indicating that an act is the reaction to the fact, represented as an arrow line.

A *complete transaction pattern* concerns not only the 'happy path' and exceptions but also cancellations of request, promise, state and accept. To simplify the problem, we use the standard transaction pattern to define the possible states of a transaction in this research.

Moreover, in the PM, the waiting and causal relationships between transactions are described in the *process structure diagram* (PSD) shown in Figure 5.

Waiting relationship indicates the conditions of an act. As represented in Figure 5, acts (expressed as small rectangles) of a

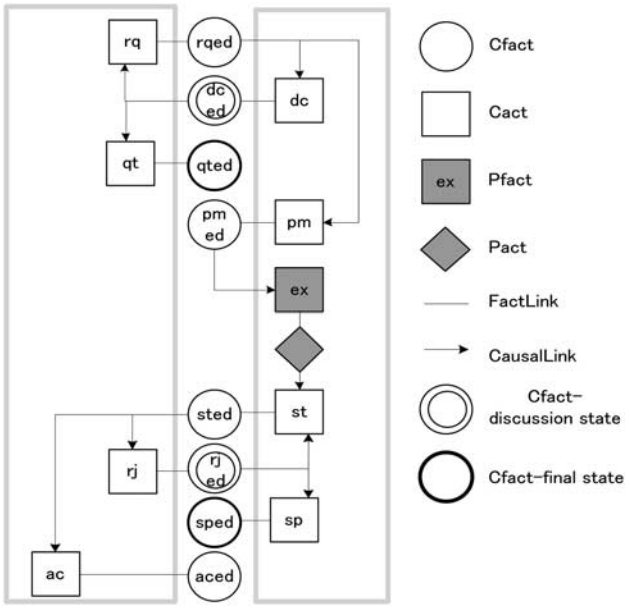

Figure 4 Standard transaction pattern.

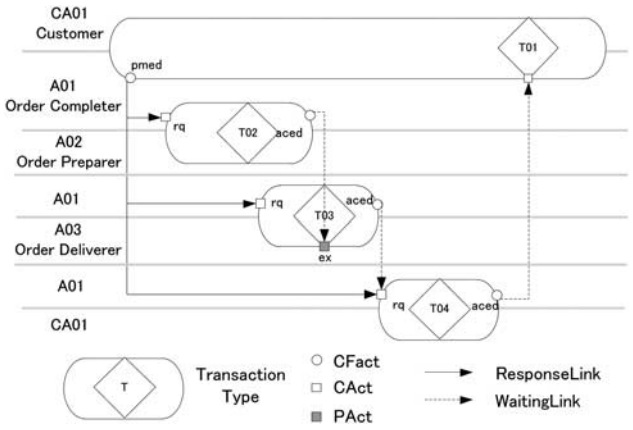

Figure 5 PSD of Buono Pizza.

transaction type may need to wait for some facts (expressed as small circles) to be created. For example: the act 'execution of order delivery' (T3/ex) must wait for the fact 'pizza preparation accepted' (T2/aced). The waiting relationship is represented by a dashed arrow from fact (source of the link) to act (target of the link), called the *waiting link*.

Causal relationship indicates that a fact causes another transaction type to be initiated. As represented in Figure 5, when the order for completion is promised (T1/pmed), new transaction types T2 (order preparation), T3 (order delivery) and T4 (order payment) are initiated. Causal relationships are represented by an arrow from fact (source of the link) to act (target of the link), called the *causal link*. In the PSD, acts and facts are described only when there are causal or waiting relationships between them. The others are hidden within transaction type as part of the transaction pattern.

3.5. DEMO fact model

The PM takes the process and state view when analysing an organization. The FM, located at the same level as the PM, describes a different aspect: the object and fact structures represented in the *object fact diagram* (OFD). As shown in Figure 6, an *object* is an identifiable individual component. Objects are always of some type, called *object type* (O for short). For example, in the case of Buono, 'Order', 'Person' and 'Pizza' are the object types. A relationship between object types is called a *fact type*. In the example, a fact type between 'Order' and 'Pizza' is 'order O contains Pizzas P'. An instance of fact type expresses an elementary state of the world. Fact type is represented as an *OO link* in the OFD.

When an object type is related to a transaction's production process, it is connected with the responding product type, representing possible stages of the object type. For example, 'Order' is involved in all four production processes, T1, T2, T3 and T4. Thus, the possible stages of 'Order' are P1, P2, P3 and P4, meaning that an order must be prepared (P2), delivered (P3), paid for (P4) and then completed (P1). This relationship is represented as the *OP link* in the OFD.

3.6. DEMO action model

An AM located at the bottom of the triangle shown in Figure 2 describes action rules. An action rule is expressed as a *crispie*. Crispies are founded on finite automaton theory (Hopcroft *et al*, 2006), which entails a finite set of states and a finite set of state transitions.

With crispies, the world is in some state at every point of time. *State* (S) is defined as a set of facts including both Cfacts and Pfacts. At any moment, the crispie releases an *agenda* (tasks to perform), each item on which is a pair, for instance, *c, t*, where *c* is an instance of Cfact (eg, T3(o)/ rqed) and *t* is the settlement time, at which point the creator will expect the event to be responded to by a crispie (eg, t_{T3rq}). The crispie responds to the event by evaluating a particular function, called a *rule base* (R). The evaluation result is a set of reactions to the Cfact (Dietz, 2006). An example of an AM for actor role A3 is as follows:

```
WHEN Order Delivery for Order (o) is requested,
IF it can be promised,
THEN promise Order Delivery for Order (o)

WHEN Order Delivery for Order (o) has been promised,
IF map has been prepared,
THEN execute Order Delivery for Order (o) and
        State Order Delivery for Order (o)
```

Action rules are defined for each actor role. The entire enterprise is a crispie net constructed by actor roles.

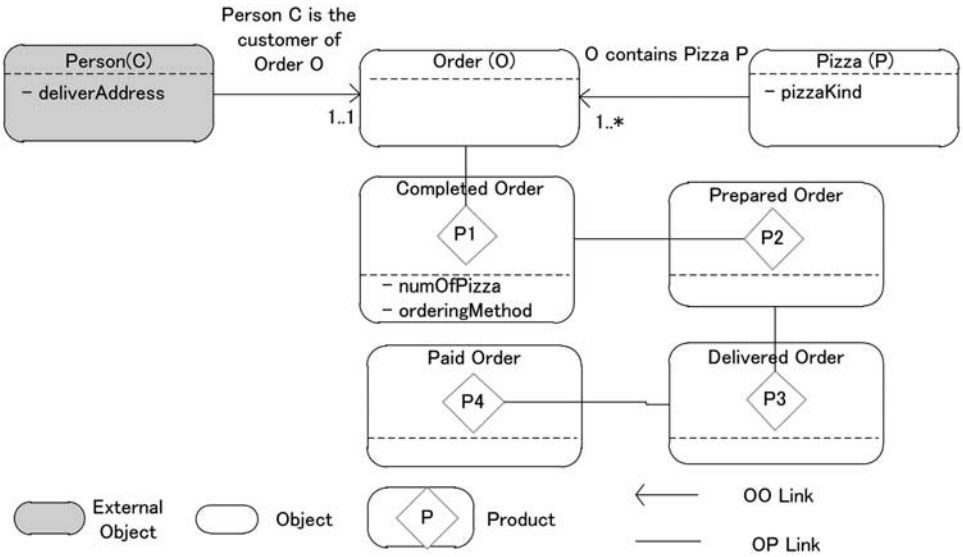

Figure 6 OFD of Buono Pizza.

4. DEMO++

DEMO++ is an ontology-based conceptual model for simulation. Following Zeigler's framework (2000), DEMO++ is designed as a hieratical structure of components. The meta-model of DEMO++ is represented in Figure 7. There are two parts in DEMO++: ontology and implementation:

Ontology (in the left of Figure 7) describes the ontology of an enterprise: (1) transaction types; (2) causal and waiting relationships between transaction types; (3) initiators and executors of transaction types; (4) information banks; and (5) information access between actor roles and information banks. In ontology, the components on transaction types (T), aggregate transaction types (AT), object types (O), elementary actor roles (A) and composite actor roles (CA) are defined.

Implementation (in the middle of Figure 7) describes the implementation of ontology by considering how the execution steps of ontological acts are defined, how they are related to resources and what is the cooperation delay of ontology in the real world. In implementation components on resource type (R) and ActorRoleComponent are defined.

Both ontology and implementation are contained and connected in the component *Main* (in the right of Figure 7).

Each component contains ports, including both an *input port*, through which the component can obtain events or information, and an *output port*, through which the component can send events or information to the others. The *inter connection* (IC) is a connection between the output and input ports through which components are connected so they can communicate. The *external input connection* (EIC) is a connection between the input ports of hieratical components through which the obtained event or information could be transferred from a component to its subcomponents. The *external output connection* (EOC) is a connection between the output ports of hieratical components through which generated events or information can be sent out.

4.1. Ontology

O (Object type). O is a component of object type whose state will be changed by transactions. It follows the concept of object type in DEMO with the definitions shown in Figure 7 (O). With Buono, three objects are defined (cf Figure 6). The object 'Order' is described in Figure 8 for illustration.

- *name* is the full name of the object type (eg, 'Order');
- *type* indicates the short form of the type name (eg, 'O');
- *properties* are obtained from the properties or related fact types defined in the OFD.
- *phases* and *transitions* are defined on the left side of Figure 8, a state transition diagram. *Phases* are an object's possible stages. Derived from the OPLink in OFD, there are four phases for the object type 'Order': P1, P2, P3 and P4. These phases indicate that an order must be prepared, delivered and paid for before it can be completed. *Transition* is the phase change.

T (Transaction type). T is a component of transaction type that is derived from the transaction type (cf Section 3.3) in DEMO. According to the meta-model (T in Figure 7), the following must be defined for each transaction type, as shown in Figure 9 (table Transaction):

- *type* is the transaction type (eg, T1);
- *name* is the description of the transaction type (eg, 'Order completion' for T1);
- *initiator* and *executor* follow the DEMO definitions;
- *object* is the related objects.

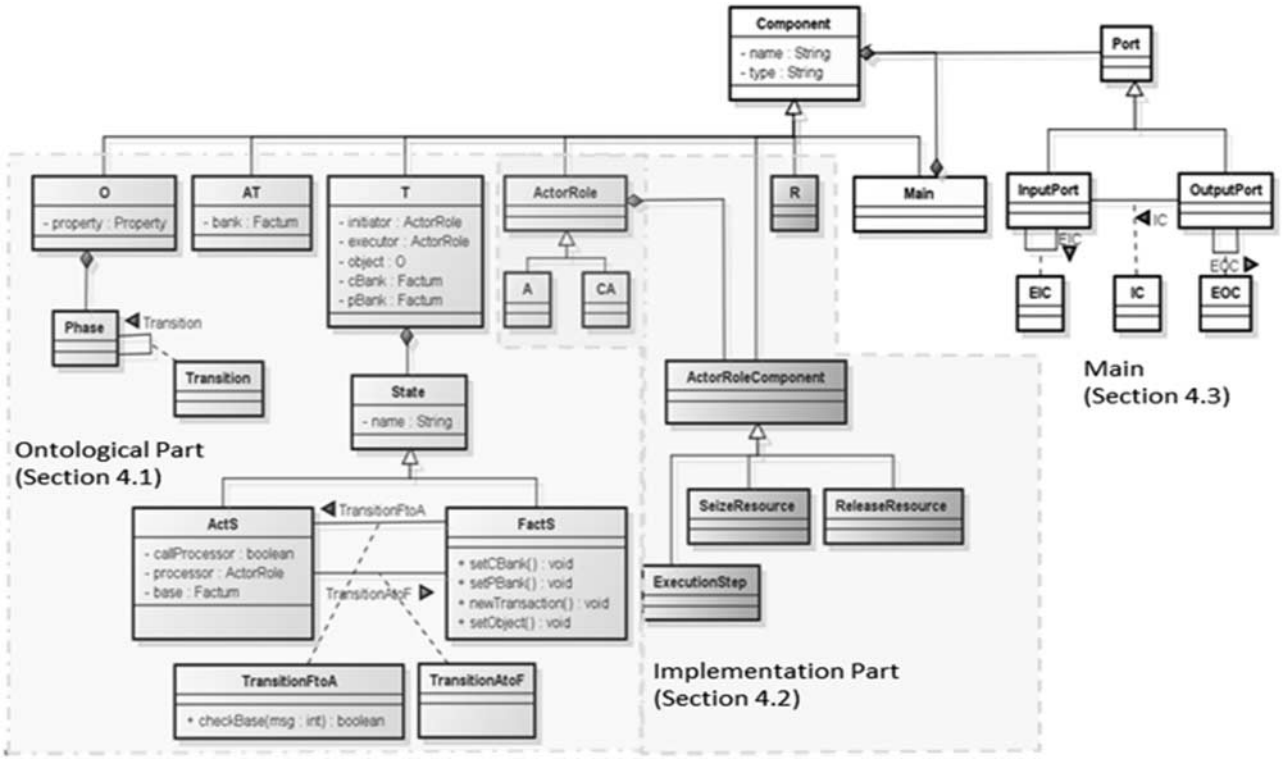

Figure 7 Meta-model of DEMO++.

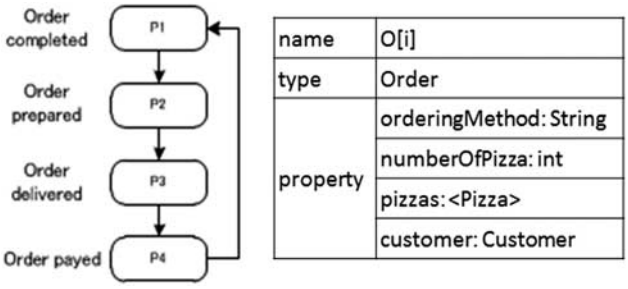

Figure 8 Object type component 'order'.

A transaction type is considered not only a set of acts but also a collection of generated facts in the information bank. Information banks need to be defined for each transaction, including cBank and pBank:

- *cBank* contains all of the Cfacts generated by the transaction type. Each bank item is a type of fact, defined as a tuple <type, status, time> (also called a factum), indicating whether an instance of this Cfact type has been settled (*status* is true) or not and the settlement time *t*. For example, <rqedT1, true, t_1> means that Cfact instance 'rqedT1' was settled at time t_1.
- *pBank* contains all of the Pfacts generated by the transaction type. pBank items are formatted in the same manner as cBank items.

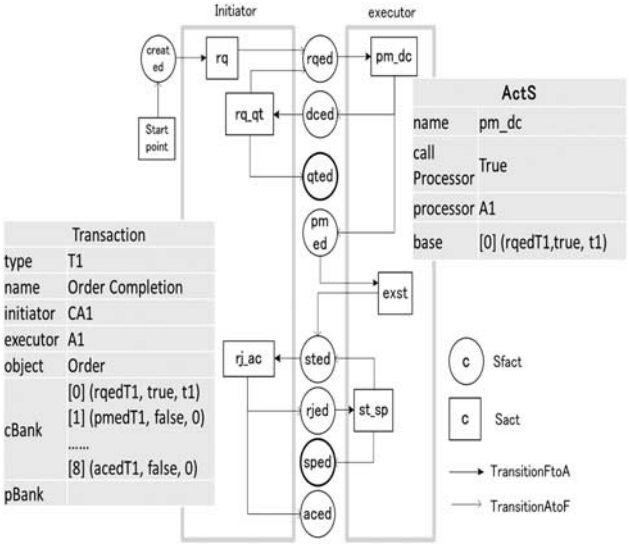

Figure 9 Transaction type component 'T1'.

Following DEMO's standard transaction pattern, a state transition pattern is defined for each T, as shown in Figure 9. There are two types of states and two types of transitions between states defined:

- *ActS*: indicating the state of acting, represented as a rectangle. In DEMO, acts are abstracted ontological concepts. To combine

ontology with implementation, each ActS in DEMO++ is related to a processor. As presented in Figure 9 (table ActS), the *processor* is the corresponding actor role who takes responsibility for making decisions and implying the ontological act. Ontology can either be expanded into implementation or not, depending on whether resource utilization, communication flows or other operation-related details must be examined. The shift is controlled by the Boolean property *callProcessor*: a 'true' means the ontological act will be processed with a feedback of generated results after some delay. According to the DEMO AM, an automaton can process ahead only when all of the state conditions are satisfied. *Base* contains the status of all Cfacts and Pfacts necessary to perform an act. An act can be performed only when all items in its 'base' are satisfied. Base items are formatted as factum. For example, the required state condition for act pmT1 is 'T1 has been requested'. This condition is represented as a base item < rqedT1, true, t1 > of ActS 'pm_dc'. By using the base, we can check the multiple state conditions required to perform an act.

- *FactS* indicates the state after an act is performed, called the fact state and represented as a circle. For FactS, we must update the effects of an act with a number of functions, including: update banks of the transaction type, update the states of related objects, and add new transaction instances if necessary.
- *TransitionFtoA* is a transition from FactS to ActS. It is triggered when all of the items in the act's base are true;
- *TransitionAtoF* is a transition from ActS to FactS. It is triggered by a particular message.

All transaction types are modelled following a state transition pattern, with different names, types, initiators, executors, objects, banks and bases for ActS, and reactions for FactS.

Several types of ports are defined for transaction types corresponding to different types of links in DEMO:

- *Port type i*: Input and output port, connected to the initiator.
- *Port type e*: Input and output port, connected to the executor.
- *Port type w_*: Input port, connected to other transaction types (derived from the waiting link).
- *Port type o_*: Output port, connected to other transaction (derived from the waiting link and causal link).
- *Port type i_*: Input port, connected to transaction types (derived from the causal link).
- *Port type info*: Input and output port, connected to the actor roles that can access this bank.

AT (Aggregate transaction type). AT is derived from DEMO aggregate transaction types. It is used as an information bank when an actor requests external information. As represented in Figure 7 (AT), only bank is defined. Here, bank shares the same meaning as pBank in T: it contains the facts.

AT is connected with the actor role through port type *info*:

- *Port info*: Input and output port, connected to the actor roles who can access this bank.

Actor role (A and CA). Actor role is a bridge that connects ontology with implementation, as shown in Figure 7 (ActorRole). At the ontological level, *actor role* describes responsibilities and authorities, whereas at the implementation level, *actor role* defines the details of executing ontological acts. In other words, as the initiator or executor of a transaction, the actor role is defined as the processor of the ontological acts for implementation.

Through ports, actor roles are linked with transactions as the processors of corresponding acts or with information banks as information users. There are three types of ports defined for actor role:

- *Port type i*: Input and output port, linked with the transactions initiated by it.
- *Port type e*: Input and output port, linked with the transactions executed by it.
- *Port type info*: Input and output port, linked with the information bank.

4.2. Implementation

In implementation, DEMO is expanded with implementation details: ActorRoleComponent and Resource. The meta-model of implementation is given in Figure 7.

ActorRoleComponent. *ActorRoleComponent* indicates the subcomponents of an actor role who defines the implementation details, including the following:

- *ExecutionStep*: The blocks for describing business process details. For example, for actor role A1 (Figure 10), implementation process pmT1 is defined, describing how to promise, what the required resources (Staff) are for keeping the promise, and what steps (Order Taking) must be taken to complete this coordination.
- *SeizeResource:* A block for seizing resources if necessary.
- *ReleaseResource:* A block for releasing resources if necessary.

R (Resource type)

- R is utilized in transactions. There are two types of resources: *operator* (an actor who takes responsibility) and *operand* (utilized by the operator). Both can be seized or released by the actor role. We define the *actor plays actor role* through this seize-release resource procedure. Seize an operator resource indicates that the actor role is played by this type of resource.

Discussion on implementing ontology. Figure 10 describes the implementation model of A1 (the order completer) and A2 (the baker) at Buono. For A1, two different implementations are available for promising the 'order complete': at the window or by phone, which requires staff resources, or through the online

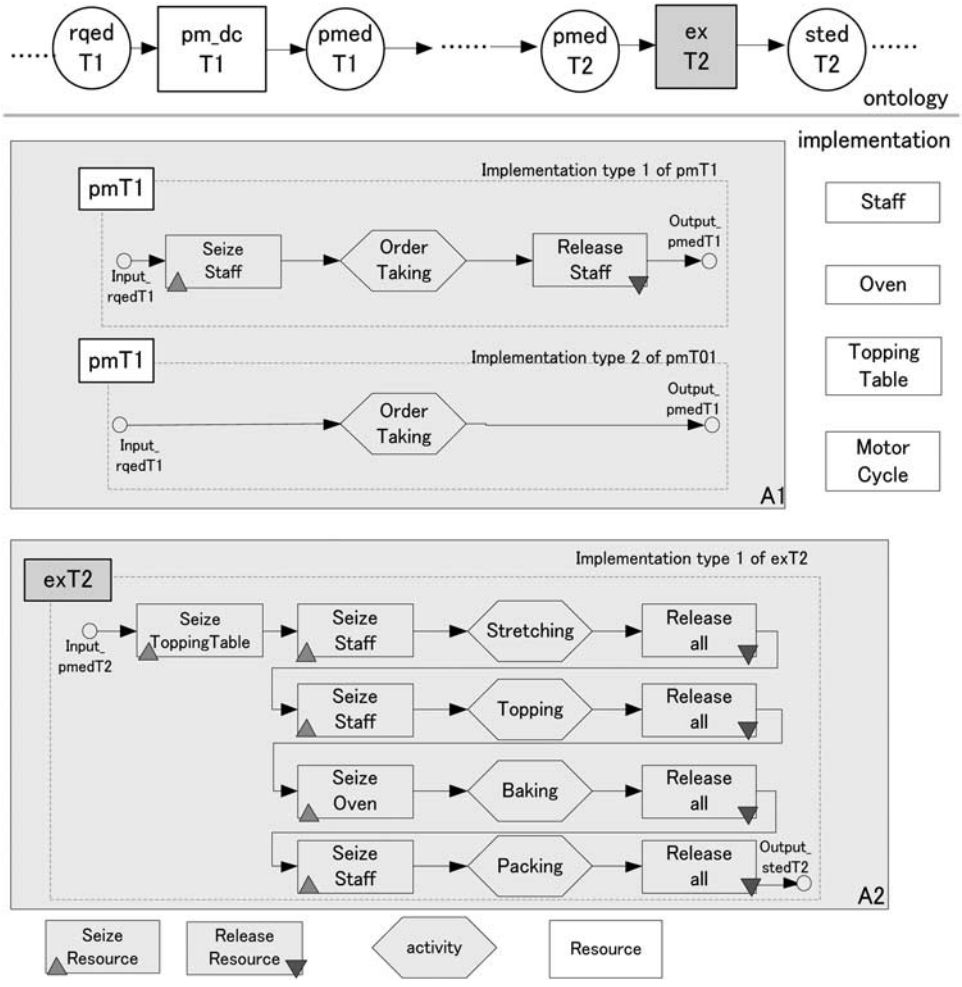

Figure 10 Examples of implementation models.

ordering system, which has no resource requirement. For A2, the detailed execution process of preparing the pizza is defined: stretching, topping, baking and packing, with resource requirements.

Ontology could be implied differently. The key factors of different implementations are concluded from the following two aspects:

(1) Who plays the actor role?
Actors are an organization's resources. The actor playing the actor role is represented as an actor role's seizing one type of resource (operator) if necessary. As described in the actor–actor role mapping in Table 2, one type of actor (resource) can play different ontological actor roles: at Buono, any staff member can play all three elementary actor roles, A1, A2 and A3. Meanwhile, one actor role can be played by different actors (resources): for example, if there are two types of staff members in the store, staff type one can perform only the duties inside the store, such as taking orders and preparing pizza; staff type two responds by delivering the pizza. Then, we can define another implementation plan: staff type one can play actor roles A1 and A2, and staff type 2 can play actor role A3, as shown in Table 3.
Actor–actor role mapping explains how humans interact by playing actor roles. Although the ontology remains the same, different human interactions during implementation will affect the effectiveness of communication in cooperation and resource utilization and hence the entire process. In our method, it is simple to change the actor–actor role mapping plan to illustrate the effects of different implementation plans in the simulation model.

(2) How to play the actor role?
Implementation relates to how to apply ontology in an enterprise; we can have different implementation plans for

Table 2 Actor–actor role mapping table plan one

Actor	*Actor role*		
	A1 Order completer	*A2 Order preparer*	*A3 Order deliverer*
Stuff	A	A	A

the same ontological design. As illustrated in Figure 10, there are two implementations defined for A1, corresponding to the same ontological act pmT1.

In reality, all of the possible implementation plans could be defined independently in the corresponding actor roles, so we can choose different approaches to compare the effects. The changes are controlled: implementation changes will not affect the ontological level, and the ontological changes will not affect implementation if it is not necessary. For example, all of the execution phases within T2 could be outsourced to other companies. In this situation, we only need to change the implementations within A2, with no change to the ontological model. Similarly, if we need to enable customer payment through the Internet using a credit card, it is not necessary to wait for delivery to be accepted (acedT3) to request the payment (rqT4). We can change the ontological model to make 'acedT3' a condition of 'exT1' instead of 'rqT4'. However, this ontological change does not change the implementation of any acts.

Table 3 Actor-actor role mapping table plan two

Resource	*Actor role*		
	A1 Order completer	*A2 Order preparer*	*A3 Order deliverer*
Stuff type 1	A	A	
Stuff type 2			A

4.3. Main

Main contains both ontological and implementation components, with inter-connections (ICs) between the components defined.

An example of component 'Main' for Buono is given in Figure 11. Component ports are connected by ICs. ICs between transaction types are derived from the PSD's causal and waiting links. ICs between transaction types and actor roles are derived from the OCD initiator, executor and information links. ICs between actor role and aggregate transactions are derived from the OCD information link.

4.4. Mapping with DEMO concepts

DEMO++ follows the enterprise engineering concept, with separate design and realization phases. The ontology describes the core, stable essence of an enterprise, whereas the implementation model describes how to bring the essence into reality. DEMO++ focuses on not only workflow but also the human aspects of cooperation. It is modularized so that all changes are controllable for the purpose of reengineering.

All of the ontological components are derived from DEMO. The mapping is concluded and listed in Table 4. This mapping process could be semi-automatically completed using the MDD4MS framework, as presented in previous

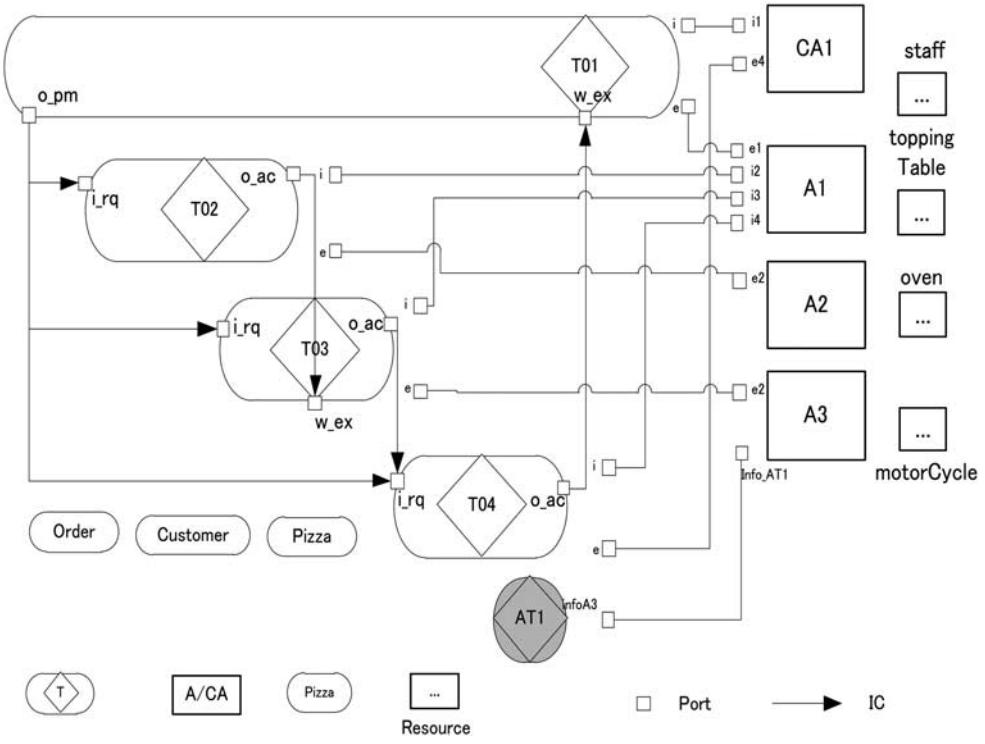

Figure 11 Example of main component.

Table 4 DEMO++ and DEMO mapping table

	DEMO++	*DEMO*	*Implementation*
Main	t	t	
	AT	AT	
	A&CA	A&CA	
	IC	Causal link	
		Waiting link	
		Information link	
		Initiator link	
		Executor link	
T	ActS	Standard	
	FactS	transaction	
	Transition FtoA	pattern	
	Transition AtoF		
	Port type i	InitiatorLink. target (CM)	
	Port type e	EexcutorLink .source (CM)	
	Port type o_	CausalLink .source (PM)	
		WaitingLink .source (PM)	
	Port type i_	CausalLink .target (PM)	
	Port type w_	WaitingLink .target (PM)	
	Port type info	InformationLink (CM)	
	cBank/pBank	AM	
AT	Port type info	InformationLink (CM)	
A&CA	Port type info	InformationLink (CM)	
	Port type i	InitiatorLink .source (CM)	
	Port type e	EexcutorLink .target (CM)	
O	O	Object (FM)	
	property	property (FM)	
		OOLink (FM)	
	state	product (FM)	
ActorRole			Actor role
component			component
R			Resource type

research (Cetinkaya *et al*, 2011; Cetinkaya *et al*, 2012; Liu and Iijima, 2014).

5. Simulation in AnyLogic

The simulation based on DEMO++ was conducted in AnyLogic. The main model is represented in Figure 12, following Main component in DEMO++.

Object type 'Order' shows the possible states of this object type, which are shared by all instances. When we examine one instance of 'Order', for example, the instance 'o5' is represented by a rectangle in Figure 12, and its state transition chart is as presented in Figure 13: At the moment, this object is in state 'P2', which is related to transaction T2.

Figure 14 shows the state transition pattern of transaction type T1. At the moment, the instance T1 (o5) is in 'exT2', meaning that order (o5) is in preparation. The implementation model of exT2 is defined in actor role A2, including stretching, topping, baking and packing (Figure 15).

On the basis of the description of the Buono Pizza case, we establish experimental parameters, as shown in Figure 16. The simulation result (Figure 17) shows that with the current number of resources and processes, approximately 25% of customers wait for more than 30 min to receive their pizzas. By optimization (Figure 18), we find that this percentage could be greatly reduced if there were six staff members and four vehicles available. This observation suggests that Buono's problem is mainly a function of resource allocation. Because they were already using an online ordering system, no significant changes were required in their business processes.

6. Discussion and future research

The only unchanged thing is change itself. However, traditional BPSs are inadequate for describing complex systems; it is difficult to change models to simulate reengineering, and the reengineering process cannot be repeated because of the lack of systematic approaches. These limitations are caused by using the workflow viewpoint and by the low abstraction levels in conceptual modelling, as well as by indistinguishable opinions on design and implementation. These concerns are limitations not only in BPS but also in BPR.

Enterprise engineering is a promising research area for its complexity-reducing capability, design thinking viewpoint and human-centered interaction. Enterprise engineering makes reengineering modularized and controllable. It takes enterprise as a whole and investigates its core structures with separate design and implementation. Meanwhile, it follows a human-centered viewpoint, considering the social aspect of enterprise as well as its processes. However, the advantages of enterprise engineering have not been fully investigated in simulation for enterprise reengineering. A DEMO-based Petri net model (Dietz and Barjis, 2000) explained how we can use ontology as an executable model. However, ontology only is not sufficient for enterprise reengineering simulation because many of the changes are related to implementation. The main problem for those who use enterprise ontology in their work is how to connect ontology with implementation.

The proposed DEMO++ is a conceptual model that expands and combines ontology with implementation in the context of enterprise engineering to assist in BPS. It concerns not only the abstracted essence of enterprise but also its implementation. In DEMO++, business process is controlled in its ontology; decision making and implementations are called for only when necessary during implementation. The separately defined core and

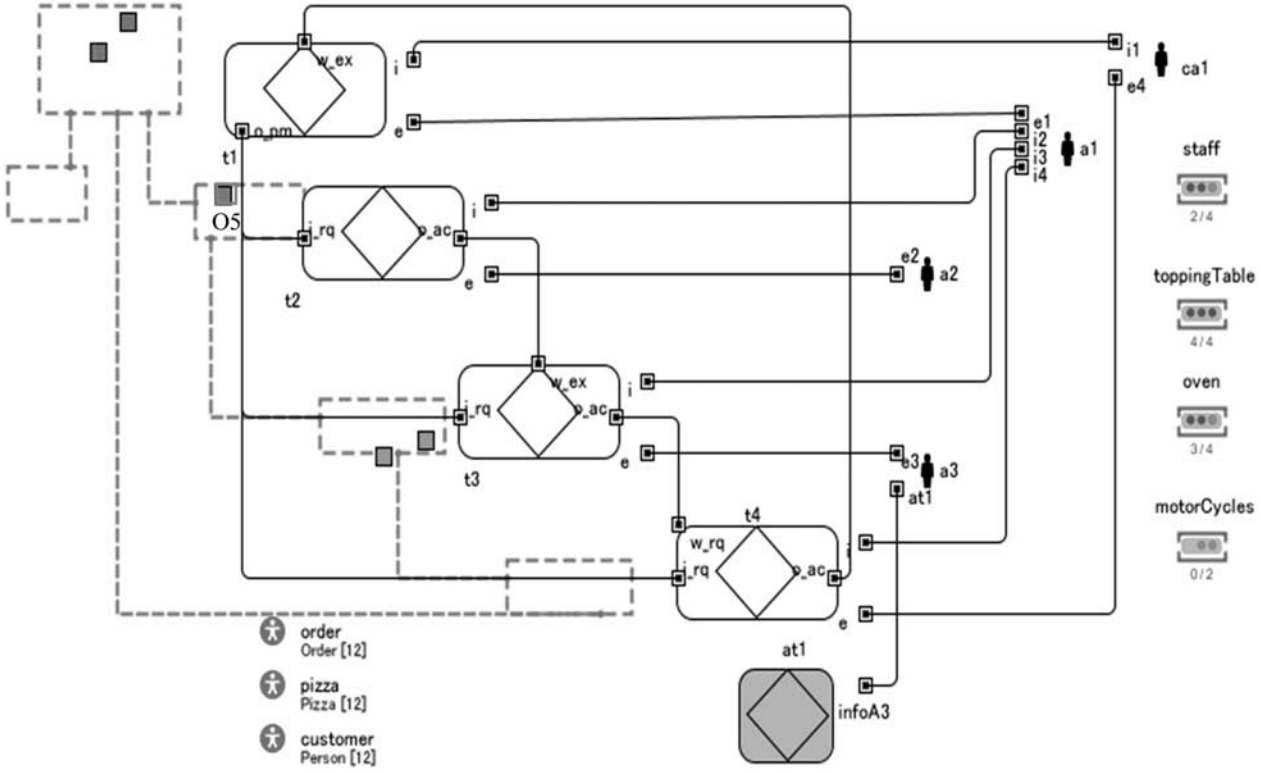

Figure 12 Main of Buono Pizza.

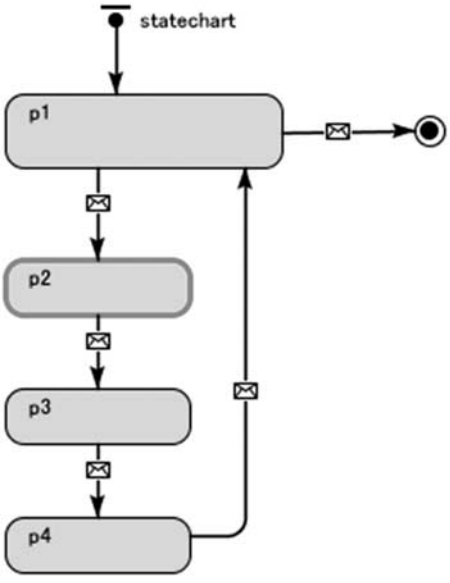

Figure 13 State transition diagram of order instance 'o5'.

implementations reduce the complexity of modelling complex systems and make simulation models more flexible and agile.

Meanwhile, DEMO++-based conceptual models make changes controllable. In some situations, the ontological model itself must be changed. These are always fundamental changes in BPR; in most cases, the changes only involve providing an alternative implementation plan for the same ontological objective. Regardless of the change types, the changes are within certain components.

Compared with traditional workflow-based business process modelling and simulation methods, DEMO++ is not only a way of simulating but also a way of analysing. This method can be well connected with management for analysing problems, seeking solutions and evaluating alternative plans for enterprise reengineering.

Another contribution of this research is considered from the practice standpoint. By clarifying the differences and dependencies between the ontological and implementation models, our methodology proposed a generic framework for generating a modularized, component-based simulation model with increased reusability. The proposed components were developed as an AnyLogic DEMO++ library and can be reused in other DEMO++ based simulations.

A limitation of this research is that we used the standard transaction pattern in describing transactions. However, in certain real-world cases, we would need to simulate exceptions, such as cancel and redo. The standard transaction would need to be expanded to a complicated transaction pattern (with decline, reject and cancellation) to make the simulation more realistic and comprehensive.

Although the DEMO++ AnyLogic components are useable, we still need to create the simulation model manually, with many

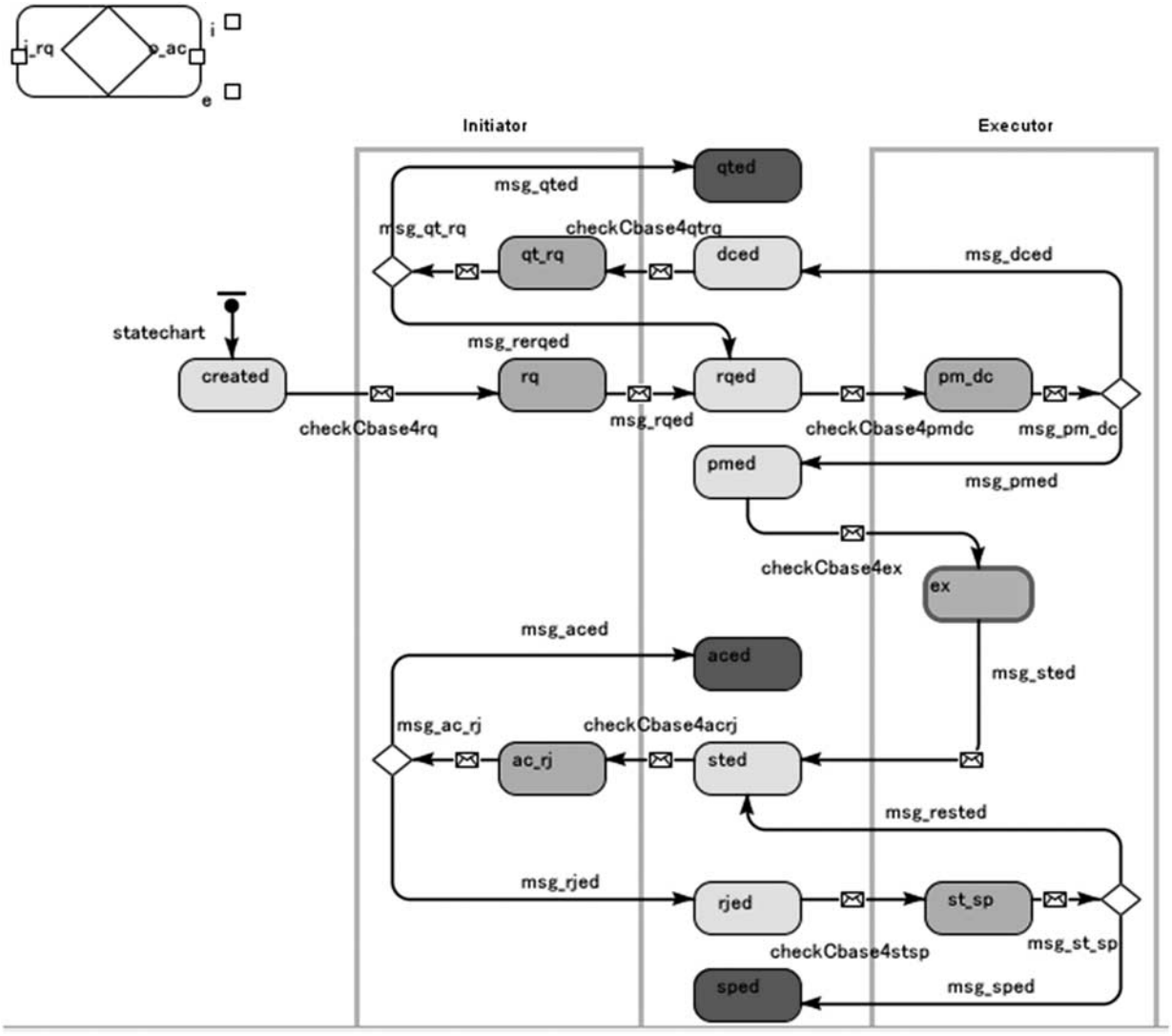

Figure 14 Transaction pattern of transaction T1.

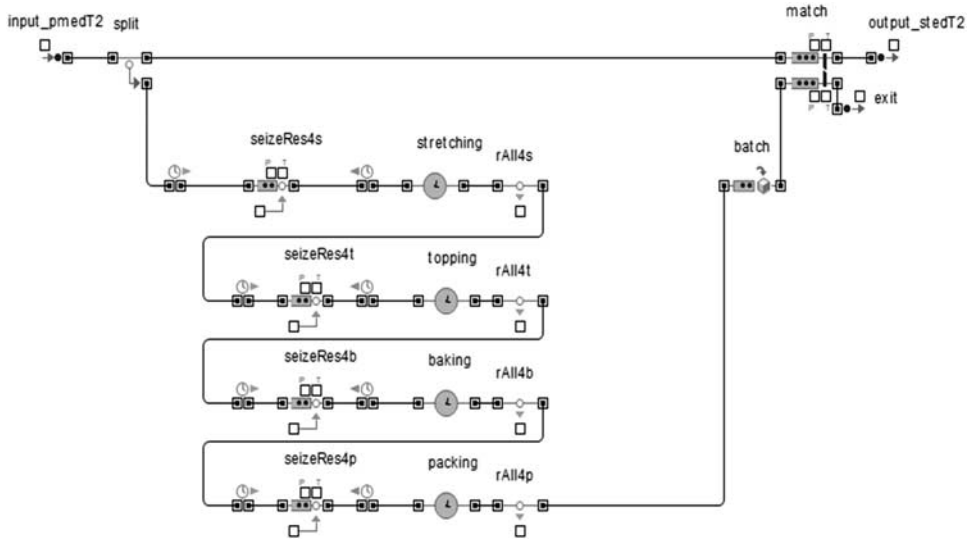

Figure 15 Implementation phase of exT2.

repeated modification tasks. On the basis of the meta-model given in this research and on previous studies (Liu and Iijima, 2014), we will convert a semi-automatic transformation tool from DEMO to DEMO++ to an AnyLogic model to reduce the complexity of simulation modelling based on DEMO++ in our future research.

In reality, pizza is mainly related to resource allocation issues. These are quite simple and traditional problems that can also be easily solved by process-based simulation models. This case was only utilized to explain the basic concepts of DEMO++. In the context of enterprise engineering, DEMO++ can provide more ability than mere simulation. It is better at analysing and simulating complex business processes and cooperation that the other traditional simulation models cannot well support. Improving this method further by applying it in real-world BPR projects is also planned as our next research target.

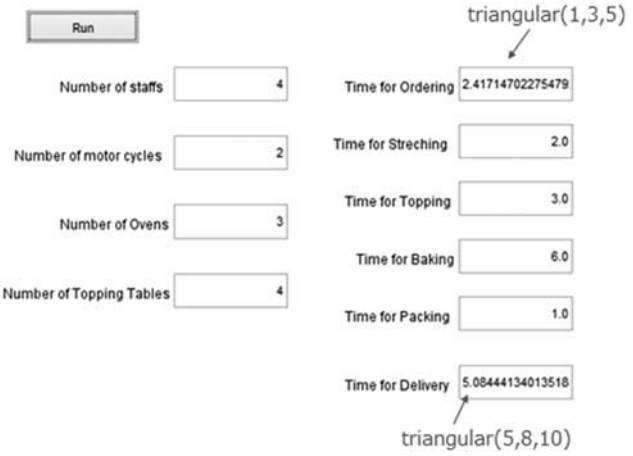

Figure 16 Experimental parameter setting.

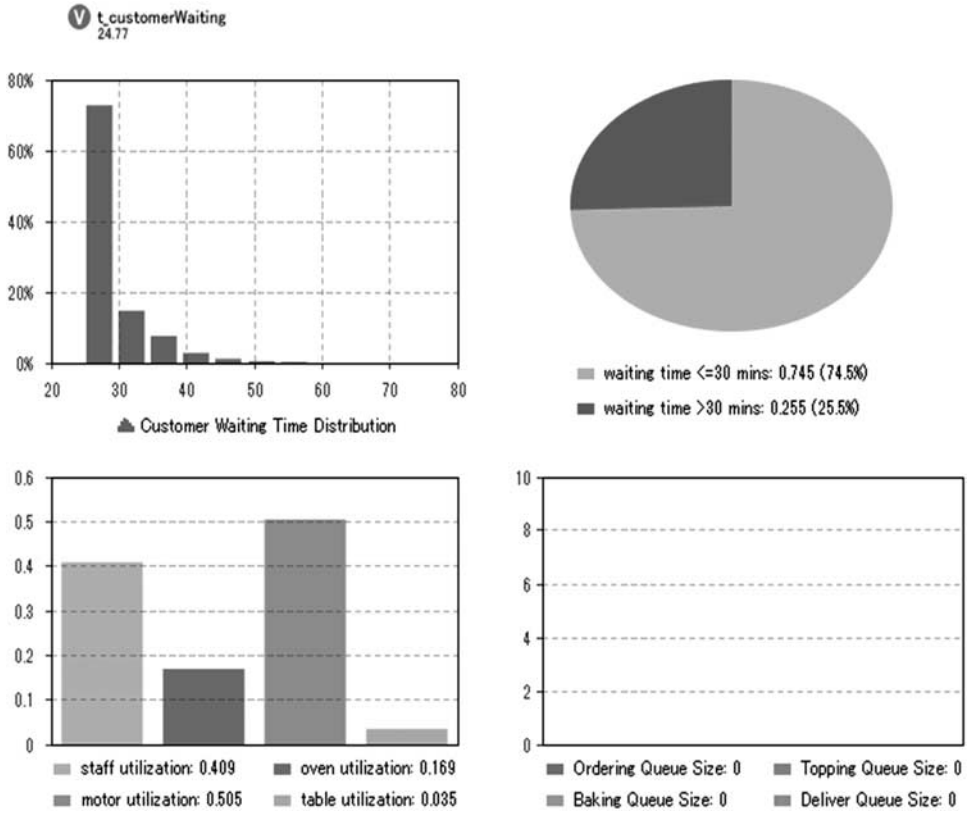

Figure 17 Simulation result.

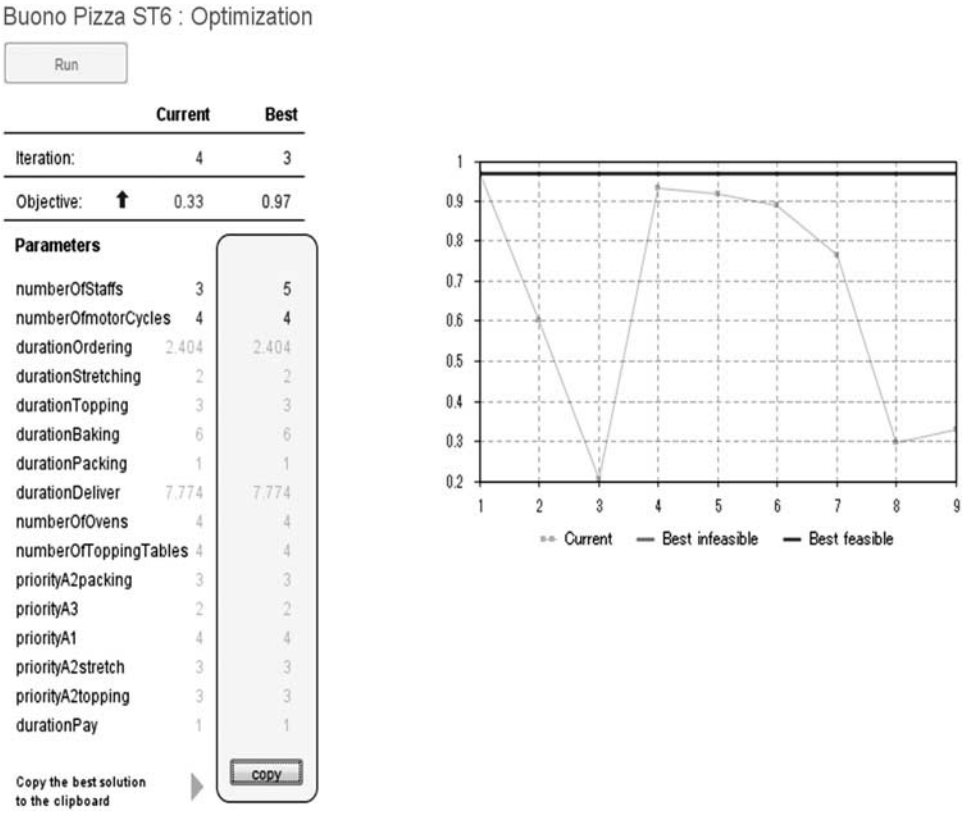

Figure 18 Optimization result.

References

Aguilar-Savén RS (2004). Business process modelling: Review and framework. *International Journal of Production Economics* **90**(2): 129–149.

Banks C, Filho JP, de Moura J and Santini B (2013). Framework for specifying a discrete-event simulation conceptual model. *Journal of Simulation* **7**(1): 50–60.

Barber KD, Dewhurst FW, Burns RLDH and Rogers JBB (2003). Business-process modelling and simulation for manufacturing management: A practical way forward. *Business Process Management Journal* **9**(4): 527–542.

Barjis J (2007). Automatic business process analysis and simulation based on DEMO. *Enterprise Information Systems* **1**(4): 365–381.

Barjis J (2008). The importance of business process modeling in software systems design. *Science of Computer Programming* **71**(1): 73–87.

Barjis J (2010). Collaborative, participative, and interactive modeling and simulation in systems engineering. In: Wainer GA (ed). *Proceeding: Spring Simulation Multi-conference 2010.* SCS Publishing House: San Diego, CA, pp 119–124.

Barjis J, Dietz JLG and Liu K (2001). Combing the DEMO methodology with semiotic methods in business process modeling. In: Liu K, Clarke RJ, Andersen PB and Stamper RK (eds). *Information, Organisation and Technology; Studies in Organizational Sementics,* Springer: Dordrecht, The Netherlands, pp 213–246.

Barjis J, Dietz JLG and Galatonov T (2002). Language based requirements engineering combined with Petri nets. In: Halpin T, Siau K and Krogstie J (eds). *Proceedings of the Seventh CAiSE/IFIP-WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design EMMSAD'02 (in conjunction with CAiSE'02),* Microsoft Research: Toronto, pp 1671–1678.

Cetinkaya D, Verbraeck A and Seck MD (2011). MDD4MS: A model driven development framework for modeling and simulation. In: *Proceedings of the 2011 Summer Computer Simulation Conference* Den Haag, the Netherlands, Society for Computer Simulation International San Diego, CA.

Cetinkaya D, Verbraeck A and Seck MD (2012). Model transformation from BPMN to DEVS in the MDD4MS framework. In: *Proceeding TMS/DEVS '12 Proceedings of the 2012 Symposium on Theory of Modeling and Simulation—DEVS Integrative M&S Symposium*. Society for Computer Simulation International San Diego, CA: Orlando, FL.

Chen P (1976). The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems* **1**(1): 9–36.

Davenport TH, Young E and Stoddard DB (1994). Reengineering: Business change of mythic proportions? *MIS Quarterly* **18**, (Issues&Opinions: Myths About Reengineering) pp. 121–127.

Dietz JLG (2006). *Enterprise Ontology*. Springer: Berlin Heidelberg.

Dietz JLG and Barjis J (2000). Petri net expressions of DEMO process models as a rigid foundation for requirements engineering. In: *ICEIS 2000*. INSTICC: Stafford, UK, pp 267–274.

Dietz JLG and Hoogervorst JAP (2012). The principles of enterprise engineering. In: Albani A, Aveiro D, and Barjis J (eds). *EEWC 2012, LNBIP 110*. Springer: Delft, The Netherlands, pp 15–30.

Dietz JLG and Hoogervorst J (2014). *Theories in Enterprise Engineering Memorandum 5: The PSI Theory*. Working Paper, http://www.ciaonetwork.org/uploads/eewc2014/EE-theories/TEEM-5 PSI v2.pdf.

Dietz JLG *et al* (2013). The discipline of enterprise engineering. *International Journal of Organisational Design and Engineering* **3**(1): 86.

Greasley A (2003). Using business-process simulation within a business-process reengineering approach. *Business Process Management Journal* **9**(4): 408–420.

Greasley A and Barlow S (1998). Using simulation modelling for BPR: Resource allocation in a police custody process. *International of Operations &Production Management* **18**(9/10): 978–988.

Guizzardi G and Wagner G (2012). Tutorial: Conceptual simulation modeling with onto-UML. In: *Proceedings of the 2012 Winter Simulation Conference*, IEEE: Berlin, Germany.

Hammer M and Champy J (1993). *Reengineering the Corporation*. Nicolas Brealey: London.

Hoogervorst JAP (2009). *Enterprise Governance and Enterprise Engineering*. Springer: Diemen, The Netherlands.

Hopcroft JE, Motwani R and Ullman JD (2006). *Introduction to Automata Theory, Languages, and Computation,* 3rd edn, Addison-Wesley: Boston, MA.

Jahangirian M, Eldabi T, Naseer A, Stergioulas LK and Young T (2010). Simulation in manufacturing and business: A review. *European Journal of Operational Research* **203**(1): 1–13.

Liu Y and Iijima J (2014). Automatic model transformation for enterprise simulation. In: Aveiro D, Tribolet J and Gouveia D (eds). *EEWC 2014*. Springer: Funchal, Portugal.

Martin J (1995). *Using the Seven Principles of Enterprise Engineering to Align People, Technology and, Strategy*. American Management Association: New York.

Netjes M (2006). Business process simulation—A tool survey. In: *In Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*. CPN'05: Aarhus, Denmark.

Paolucci E, Bonci F and Russi V (1997). Redesigning organisations through business process re-engineering and object-orientation. In: *Proceedings of the European Conference on Information Systems*, IEEE: Cork, UK.

Perinforma APC (2012). *The Essence of Organisation*. Sapio Enterprise Engineering: Leidschendam, The Netherlands.

Reijers HA and Liman-Mansar S (2005). Best practices in business process redesign: An overview and qualitative evaluation of successful redesign heuristics. *The International Journal of Management Science* **33**(58): 283–306.

Robinson S (2006). Conceptual modeling for simulation: Issues and research requirements. In: *Proceeding WSC '06 Proceedings of the 38th conference on Winter simulation*, Winter Simulation Conference: Monterey, CA, pp 792–800.

Salimifard K and Wright M (2001). Petri net-based modelling of workflow systems: An overview. *European Journal of Operational Research* **134**(3): 664–676.

Scholz-Reiter B, Stahlmann HD and Nethe A (1999). *Process Modelling*. Springer: Berlin Heidelberg.

Siebers PO, Macal CM, Garnett J, Buxton D and Pidd M (2010). Discrete-event simulation is dead, long live agent-based simulation! *Journal of Simulation* **4**(3): 204–210.

Tumay K (1996). Business process simulation. In: *Proceedings of the 1996 Winter Simulation Conference*, IEEE: Coronado, CA, pp 93–98.

Turnitsa C, Padilla JJ and Tolk A (2010). Ontology for modeling and simulation. In: *Proceedings of the 2010 Winter Simulation Conference*, IEEE: Baltimore, MD, pp 643–651.

Valiris G and Glykas M (2004). Business analysis metrics for business process redesign. *Business Process Management Journal* **10**(4): 445–480.

Vergidis K, Tiwari A and Majeed B (2008). Business process analysis and optimization: Beyond reengineering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **38**(1): 69–82.

Wang W and Brooks RJ (2007). Empirical investigations of conceptual modeling and the modeling process. In: *Proceedings of the 2007 Winter Simulation Conference*, IEEE: Washington DC, pp 762–770.

XJ Technologies (2009). AnyLogic home page, http://www.xjtek.com/, accessed 5 December 2012.

Zeigler BP, Kim TG and Praehofer H (2000). *Theory of Modeling and Simulation*. 2nd edn. Academic Press: San Diego, CA.

Received 12 July 2013;
accepted 28 October 2014 after two revisions