

# Dynamic Composition and Analysis of Modern Service-Oriented Information Systems

Habib Abdulrab, Eduard Babkin, and Jeremie Doucy

**Abstract** Despite all the advantages brought by service-oriented architecture (SOA), experts argue that SOA introduces more complexity into information systems rather than resolving it. The problem of service integration challenges modern companies taking the risk of implementing SOA. One of important aspects of this problem relates to dynamic service composition, which has to take into account many types of information and restrictions existing in each enterprise. Moreover, all the changes in business logic should also be promptly reflected. This chapter proposes the approach to solution of the stated problem based on such concepts as model-driven architecture (MDA), ontology modelling and logical analysis. The approach consists of several steps of modelling and finite scope logical analysis for automated translation of business processes into the sequence of service invocations. Formal language of relational logic is proposed as a key element of the proposed approach which is responsible for logical analysis and service workflow generation. We present a logical theory to automatically specialize generic orchestration templates which are close to semantic specification of abstract services in OWL-S. The developed logical theory is described formally in terms of Relational Logic. Our approach is implemented and tested using MIT Alloy Analyzer software.

---

H. Abdulrab  
INSA de Rouen, LITIS Laboratory, BP08 Avenue de l'Université, 76801  
Saint-Étienne-deRouvray, France  
e-mail: [abdulrab@insa-rouen.fr](mailto:abdulrab@insa-rouen.fr)

E. Babkin (✉)  
National Research University Higher School of Economics, B. Pechorskaya 25/12, 603155  
Nizhny Novgorod, Russia  
e-mail: [eababkin@hse.ru](mailto:eababkin@hse.ru)

J. Doucy  
EADS Defence & Security, Information Processing Control and Cognition, Parc d'Affaire  
des Portes BP 613, 27106 Val De Reuil, France  
e-mail: [jdoucy@gmail.com](mailto:jdoucy@gmail.com)

**Keywords** Information systems • Web-service • Composition • Formal analysis • Relational logic

## 1 Introduction

Almost all modern companies use distributed and heterogeneous information systems (IS). It implies a variety of various applications based on different information technologies (IT) which interact with each other using diverse interfaces. In addition, modern companies are much interested in IS integration with their business partners in order to speed up and improve operational business processes. All mentioned above set new requirements to IT integration which is currently one of the top priorities for modern software engineering.

However in practice traditional approaches to software engineering can hardly meet the requirements of dynamic business environment, such as flexibility and simplicity of IT operations. A significant shortcoming of traditional integration approaches is software redundancy and difficulty of software reuse. A concept of service-oriented architecture (SOA) offers new solutions to the mentioned problems.

SOA provides the opportunity of abstraction from software and hardware implementation [8]. This makes IS solutions much more flexible and capable of quick adaptation to business process changes. According to IBM, SOA can be defined as an application architecture in which functions are represented by independent and coarse-grained services with triggered interfaces. SOA enables business process automation in the form of workflow—a predefined sequence of business process activities [3]. Several machine languages were proposed for definition and enactment. Among them BPEL [4] and OWL-S [19] are mostly used in industry and research.

In SOA terminology a coordinated aggregate of services refers to service composition [9]. The following types of service composition are distinguished: choreography and orchestration. Orchestration is the management of services within a single run of a business process. Choreography is defined as the management of services during the asynchronous run of the business process working simultaneously with several data flows [8]. Our work deals with automation of service orchestration solely. Of course such a limitation does not allow modeling the complete life cycle of information systems; however, we believe that the proposed methodology can be further developed to automate service choreography as well.

However, in spite of SOA advantages and its popularity within business and IT communities, practical benefits of SOA are still intensively discussed. Many enterprises argue that SOA introduced more complexity into their information systems rather than resolving it. Heterogeneity of individual services and the need for multi-aspect modeling of complete distributed systems contribute to extreme complexity and high costs of SOA solutions. The solution for that problem has been looked for by means of enhancing semantic description of web services and

wide application of formal analysis and reasoning methods for semantic integration of web services. Among the recent projects and standards we need to mention SUPER project [17], OWL-S [20], and IRS-III framework [5]. Such works as [21, 22, 28, 34, 35] mainly focus on using planning algorithms to address the automatic services composition challenge. The work of Duan et al. [7] was one of the first to introduce a formal logic-based model for specification and refinement of abstract business processes using the concepts of BPEL and program logic. Recent researches use Semantic Web Service Ontology (OWL-S) [19] for description of static and dynamics properties of abstract and concrete services, as well as different formal methods [10, 13].

Our analysis shows that most of the existing reasoning algorithms are based on theorem proving which influences the scope of solvable tasks. They do not enable analysis based on counterexamples generation, and this significantly restricts the capabilities of analysis and practical impact. The first significant problem is the presence of a multi-layered hierarchy of services. Their intrinsic interdependences and complex connections of pre- and post-conditions require application of more sophisticated formalisms and logic analysis. The second problem arises from the practice of software engineering. Software engineers frequently require support for the iterative process of service development and orchestration of the group of the services in the customized end-user application. Connection of software engineering methods of program verification and formal principles of semantic service composition give us a hint to study opportunities of traditional model checking tools for application in the context of service-oriented software systems. Several research works provide us with the background. For example, the works [23, 32] investigate opportunities of Object-Z and Rewrite Logic to specify formally semantics of OWL-S. Such works as [31, 33] show great potential of relational logic and Alloy Analyzer for formal specification and refinement of business processes.

The objective of this research is to develop a new formal approach which uses benefits of finite model checking and automates service orchestration based on business-process logic taking into account existing constraints in the context of an enterprise. Such an approach makes it possible to skip the manual analytical step and to automate service orchestration process based on business requirements and changes of business processes. Our work proposes a solution based on such concepts as model-driven approach, ontology modeling, business process modeling, and relational logic. These concepts, belonging to different areas, allow the creation of an advantageous integrated approach.

The chapter has the following structure. In Sect. 2 we present the main foundations of the proposed approach in the realm of IT architecture and formal logic analysis. Section 3 offers an overview of used formal tools and the proposed approach. In Sect. 4 we give detailed explanation of our approach using two specific case studies. In Sect. 5 we discuss the achieved results, compare them with the related research works, and determine further research directions.

## 2 Foundations

Our research fuses the concepts of IT architecture and of particular methods of formal analysis. For better comprehension we provide for the key foundational elements of these two realms.

### 2.1 *IT Architecture*

Large-scale enterprises entail a great number of business processes and procedures linked with each other on different levels and evolving constantly in the turbulent business environment. The complexity of business processes manifests itself in the number of participants, documents, interactions, and different scenarios that may happen. In the result the business process analysis and modeling become very difficult. Additionally, there are quite a lot of formally fixed policies and rules, informal restrictions, and constraints that should be taken into account as well in order to automate the processes so that they comply with the business reality. Different factors are spread across various conceptual layers of organizational structure and cannot be analyzed within the single modeling practice. That is why design of multi-layered models of description and composition of services is the central task for development of service-oriented information systems. One widely accepted software engineering approach is model-driven architecture (MDA) introduced by Object Management Group (OMG) [24]. MDA combines the advantages of modeling and SOA. According to MDA there are several abstraction levels of modeling. Three layers of models are distinguished in MDA [25]:

- The computation independent model (CIM) describes a system from the computation-independent viewpoint, addressing structural aspects of the system. A CIM is often called a domain model.
- The platform independent model (PIM) can be seen as defining a system in terms of a technology-neutral virtual machine or a computational abstraction.
- The platform specific model (PSM) usually consists of a platform model that captures the technical concepts and services that make up the platform and an implementation-specific model geared towards the concrete implementation technique.

Large number of various enterprise models are used to capture different facets of knowledge about processes, their automation, and software implementations. The variation of notations leads to the great complexity of service composition process. Unified Modeling Language (UML) was proposed to offer the single common modeling notation. However, currently UML cannot fully support domain models; therefore, means for high level modeling are required.

The approach proposed in this work utilizes the ontology modeling concept [12]. It has proved itself as an efficient knowledge management approach intended to

simplify and structure the composition process. Ontologies were applied for web services description and integration by introducing the concept of Semantic Web Service (SWS). It stands for Web Service enhanced with semantic description. According to Bhiri [1] there are four main SWS initiatives, namely WSMO/L/X Framework [26], OWL-S, [20], IRS-III framework [5], and METEOR-S system [30].

Also ontology dialects for widely spread modeling methodologies EPC, BPMN, BPEL were created. sEPC, sBPMN, and sBPEL ontologies have been developed and proposed within the framework of SUPER project as part of Semantic Business Process Modeling methodology [17]. Such achievements provided a practical opportunity to link business process models to the domain ontology concept, to set users objectives and services capabilities in terms of the domain, and to automate service orchestration based on formal reasoning tools.

Ontology and MDA approaches are tightly coupled. As for description of SWSs, ontologies are mostly used for creation of PSMs (WSMO, OWL-S, etc.). However, ontologies can be successfully used for model creation on all MDA abstraction levels.

Application of ontologies and other conceptual models requires a well-developed language for ontology and models descriptions supported with formal reasoning. Syntax of such language should be intuitively clear for non-experts and compatible with existing SOA standards. Semantics of such language should be formally defined as it should provide consistent interpretability. In other words, no varied interpretations should exist. Expressive power of the ontology language should allow for fine-grained detailed description without overcomplications preventing a formal reasoning process [2].

Formal reasoning tools are required to control and support ontology quality. For example, such tools can be used during the ontology developing phase for testing model consistency and adequacy.

Formal semantics and meta-language are the key differences between formal logic languages and other languages. Main advantages of formal logic application for knowledge representation are the following:

- Consistency, lack of expression interpretation ambiguity
- Ability of distinguishing of logic expressions from the conclusions of their validity

Leading notations and methodologies for ontology development (OWL, WSMO) are based on Descriptive Logics (DLs) which are the formal extensions of first-order logic [2]. However, despite many important results achieved in using DLs in knowledge representation, there are still a number of principal scientific and engineering issues to be solved. For example, in reality many attractable DLs are ExpTime-complete (e.g., SHIQ) and only a few polynomial-complexity DL dialects for very strict domains have been developed so far. Additionally, in order to support effective practical engineering activities during design and evolution of service-oriented systems reasoning mechanisms should also provide model finding and counterexample generation capabilities, while DL-based reasoning does not allow them.

## 2.2 *Relational Logic and Alloy Analyzer*

In our research we use Alloy Analyzer [16] which gives an opportunity to enhance the logic-based method with the features of constraint-based approach. Theoretical foundations of Alloy Analyzer include a mathematical theory of relational logic and finite scope logical analysis. The first Alloy prototype came out in 1997 from MIT Software Design Group and to the moment it has evolved to the matured simulation and verification system. The formalism of Relational Logic is based on the first-order logical theory; it facilitates rigorous definition of the structure and constraints of data structures in the generic form of relations. Alloy Analyzer consists of the structural modeling language based on first-order logic and of a Java-based constraint solver for models analysis and verification. That modeling language is rooted in well-known formal language Z for program specifications, but it uses different modeling capabilities like inheritance and reuse of formulas, which facilitate declarative object-oriented description of the problem. The users of Alloy Analyzer can select the most appropriate formal approach to define the structure and behavior of the system among the following alternatives: predicate calculus, relational calculus, navigation expression style. Simple heuristic of incremental grow of the instances bound was used to find the minimal number of the instances which satisfy the specified theory.

In Alloy language all universe of discourse is modeled in terms of atoms and relations. Atoms model indivisible, constant entities, while relations with multiple arities represent meaningful relationships and dynamical aspects. Expressive means of logic include:

- Set constants (empty set, universal set, identity set)
- Commonly accepted set-theoretical operators (union, intersection, subset inclusion, etc.)
- Relational operators (product, join, transpose, transitive closure, etc.)

Non-trivial constraints against the relations can be made from usual logical operators, quantifiers, specific multiplicity constraints restricting the basic relational operators, and cardinality constraints.

Practice-oriented modeling language of Alloy facilitates declaration of logic sentences in the text object-oriented form and provides convenient means to organize large models into tractable components and to manage simulation or verification. In the Alloy modeling language the basic building block is referred to as signature. In principle each signature represents a set of atoms. Specific constraints, defined in the modeled domain, are expressed in terms of facts, predicates, and functions. Assertions denote studied properties of the domain, which are verified by the means of the model simulation. Finally, testing of the model is performed with the help of a pair of control commands: run and check. Run command starts Alloy analyzer in order to find a correspondent model for a given number of instances.

In Alloy the flexible separation of concerns principle is used for implementation of simulation and checking. To verify a model against constraints Alloy analyzer

translates the definition and constraints of the model into binary constraints and passes them to an external satisfiability solver in order to solve the classical finite domain constraint satisfaction problem via the search in the state space. Thus simulation of the model gives back such instances of states of executions that satisfy a given constraint (the model of the logical theory), and checking gives instances of the model, which violate the specified constraints. As an example of Alloy syntax, we consider the following definitions of signatures which describe basic constituents of service-oriented information systems:

```
sig Condition {}

sig WorkflowElem {}

sig ControlElem extends WorkflowElem {}

sig Service extends WorkflowElem {
  preconditions: set Condition,
  effects:      set Condition
}
```

In terms of relational logic these signatures define such mathematical structures as four sets (Condition, WorkflowElem, ControlElem and Service) and two binary relations between the set “Service” and the set “Condition” (the relations has name preconditions and effects correspondingly). An additional signature and more relations may be defined to model a generic linked structure of services in terms of relational logic.

```
sig WTemplate {
  elems : set WorkflowElem,
  first  : one elems,
  last   : one elems,
  transition: (elems - last) -> (elems - first)}
}
```

Relation “transition” demonstrates how ternary relations can be defined in Alloy language. In this case the relation is defined between WTemplate, and two subsets of WorkflowElem, thus determining the order of control flow transition from one service to another. The following fact sets a logical constraint on the transition tuples.

```
fact f1 {
  all p: WTemplate | let t = p.transition |
  all s1,s2: Service | s1->s2 in t =>
  s2.preconditions in s1.effects
}
```

According to that fact a pair of services may be linked in a workflow template only if the preconditions of the former service are in the set of effects of the former service. Later in the text of the chapter we describe in more detail how these and other similar definitions allow to facilitate service composition.

It is important to understand that Alloy logic is unresolvable [23]. Therefore, Alloy logical means for resolution and reasoning are based on automated generation of examples (for predicate validation) and counterexamples (for constraints assertion) within the finite scope. However, finite scope of Alloy logic does not lead to poor reliability of such analysis because Alloy allows modeling of an infinite number of objects and relations between them. In addition to the capability of simultaneous analysis of a great number of objects, Alloy provides opportunity to model and analyze such complex data structures as trees, which makes it convenient and applicable for real domain objects modeling. Moreover, the structure of the Alloy language allows integration of its models with the models described in other notations and languages such as UML.

### 3 Proposed Approach to Service Composition

In order to manage and orchestrate program services and ensure sufficient quality of SOA implementation business analysts should take the role of a handler and an integrator of all information gathered during the studies of the application domain and analysis of the business processes. Also for creation and further maintenance of information systems the majority of industrial developers perform system modeling in different styles, and finally execute manual or semi-automated composition of independent program services. Besides the costs associated with this kind of analytical work, such a huge amount of manual work is accompanied by high risk of losing or skipping relevant information that needed to be taken into account in the course of software implementation.

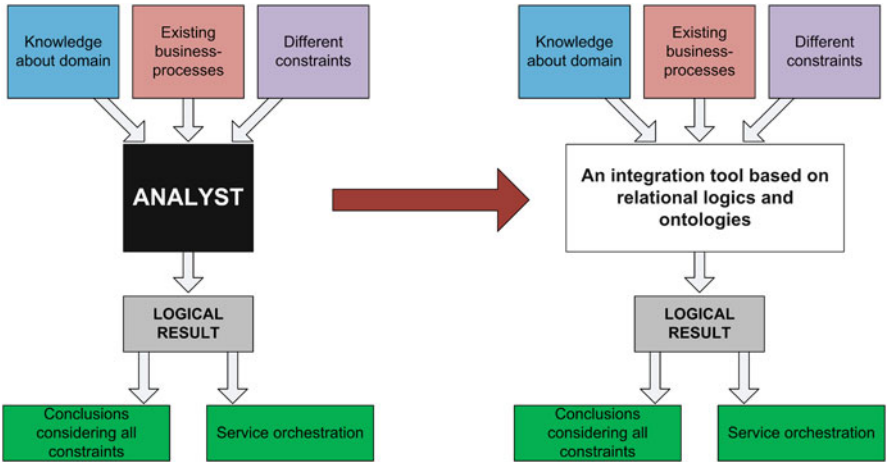
Changes in business processes require deep analytical work including interviews with involved employees, analysis of different models and restrictions. This does not allow the information system to be as flexible as business demands. Such a manual and loosely controlled process of service orchestration leads to inefficiency of information systems usage and high costs of SOA implementation.

#### 3.1 *New Basic Principles of Composition and Modeling*

Most of the SOA solution vendors define at least two levels of SOA: level of business process and level of services [9]. The correspondence between services and business operations is the core of SOA concept. Despite all the results in researches of using semantics to service composition, existing approaches are more focused on technical description of service-related data and neglect a considerable part of business requirements.

However, SOA does not cover only one business process, but the whole range of enterprise business processes in a given application domain. All business processes in one business environment have a common set of notions and features that are defined by functional and industrial specificity of the application domain. In order





**Fig. 1** Human factor elimination in the proposed approach to service composition

to achieve the common approach to modeling and consequently to automation of business processes in one domain, it is necessary to introduce concepts of a given application domain.

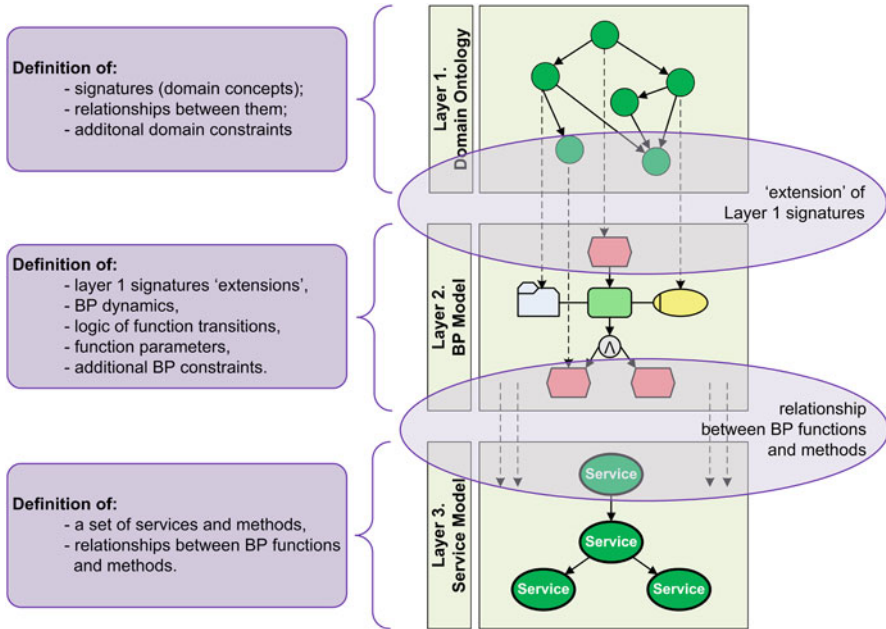
On the contrary, our approach to service composition is based not only on the service description but also on the description of the domain in the form of ontology, business process, and business requirements (see Fig. 1). Using ontology it is possible to define a thesaurus, which can function as a “translator” between the system and its users, between business representatives and software developers, and between the business process model and the model of technical realization. It makes possible to combine together different types of models created and analyzed during the implementation and change processes and to ensure consistency of all models.

The proposed approach unites the restrictions of all levels during the service orchestration. As a result, our approach features the modeling hierarchy with three layers: domain ontology, business processes, and program services (see Fig. 2). All layers and dependencies between them can be described in the logic language using Alloy Analyzer.

Domain ontology defines the basic notions of business environment and its restrictions. This layer corresponds to the CIM model in MDA. The analysis performed on Layer 1 results in the consistency checking of different restrictions.

Layer 2 models describe the dynamics and logic of business process based on the concepts defined on the Layer 1 model. This layer corresponds to the PIM model in MDA. The analysis performed on Layer 2 shows different scenarios of business process execution taking into account all the restrictions from the Layer 1 and Layer 2 models. Again during the analysis the consistence of the restrictions is checked.

Layer 3 model defines the core notions of the service environment and its restrictions. This layer corresponds to the PSM model in MDA. The analysis

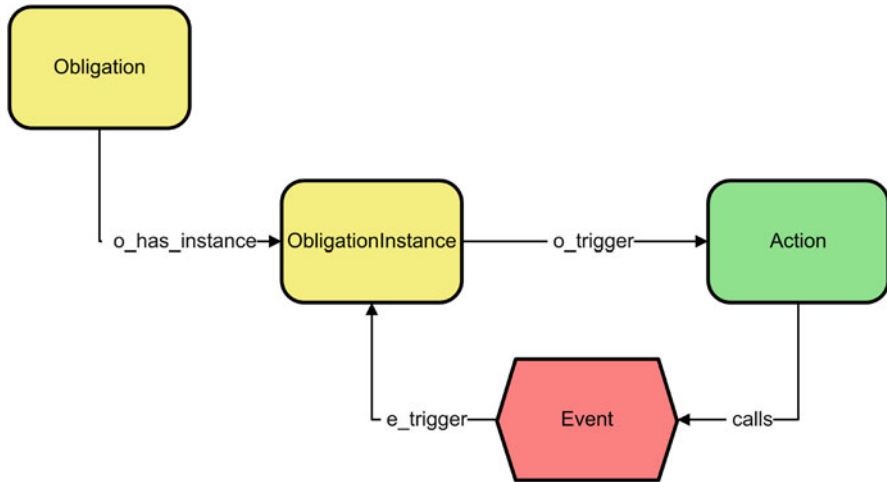


**Fig. 2** Three layers of modeling hierarchy in the proposed approach

performed on Layer 3 shows multiple scenarios of service invocations taking into account all the restrictions from the Layer, Layer 2, Layer 3 models. Also the analysis checks the consistence of the restrictions. Alloy allows for scenarios to be generated in XML format. This enables different scenarios to be “glued” and translated into BPEL executed code according to architecture described above.

### 3.2 Mapping to Existing Approaches in the Realm of Business Process Modeling

To show how mapping to existing approaches may be implemented, we utilize widely used event-driven process chain (EPC) notation. The objective of the first task in our approach is to define the way of translation from a business process model to a particular logic language of Alloy Analyzer. According to EPC, business processes consist of a sequence of events and actions. Therefore, we introduce two concepts of ontology: Event and Action. For modeling the transitions between these concepts, an additional object has been introduced—Obligation Instance which was taken from the work [27]. The transition between these three concepts is defined by the relations between appropriate signatures (see Fig. 3).



**Fig. 3** Business process logic. Relation between objects Event, Action, ObligationInstance

Proposed approach for business process description gives a possibility to describe real business processes with all branches of the scenarios. First of all, the model signatures have fields with quantitative keywords ensuring a possibility to define complex conditions for triggering events. Secondly, using classical logical operators such as OR and AND it is possible to describe the logic of complex process. For example, one Action can trigger several Events. That is modeled by Alloy relational logic language as follows:

```

abstract sig Action { // definition of signature Action
  a_pre, a_post: one Time,
  contains: Operation -> DocumentInstance,
  // one Action can trigger several Events
  calls: set Event,
  performed_by: one Role
}

```

An advantage of the proposed approach is modeling of the business process dynamics. For that an additional concept Time is introduced, which is linked with other business process notions according to the scheme in Fig. 4.

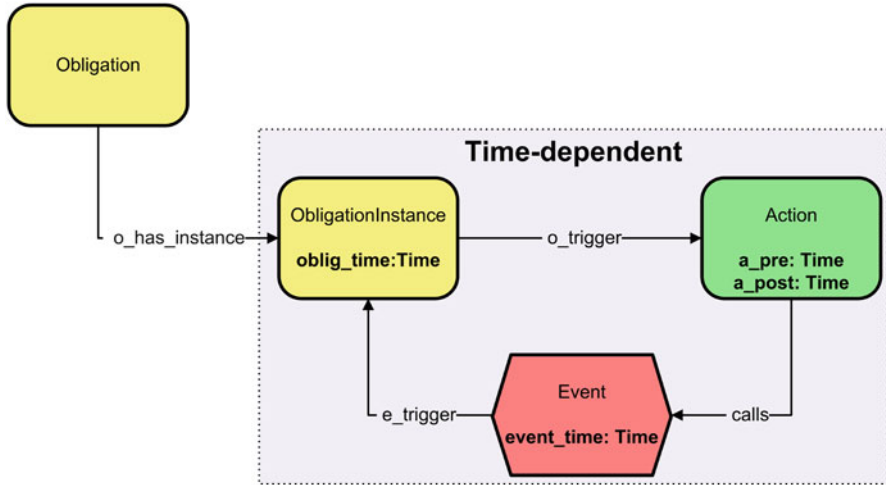
Using the following Alloy fact, the synchronization of all business process notions is achieved:

```

fact Times
{
  all a: Action | all e: Event | e=a.calls =>
    a.a_post=e.event_time

  all o: ObligationInstance | all e: Event | o in e.e_trigger =>
    e.event_tim = o.oblig_time
}

```



**Fig. 4** Business process logic. Dynamics modeling

```
all o: ObligationInstance | all a: Action | o.o_trigger=a =>
    a.a_pre=o.oblig_time
```

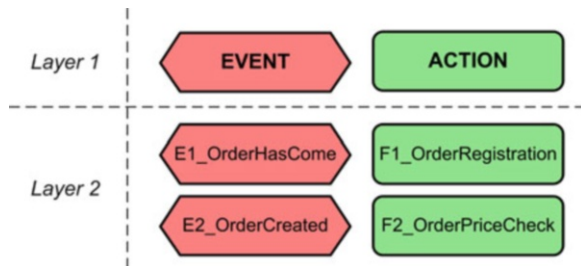
```
all a: Action | all t: Time | t=a.a_pre=>a.a_post=t.TO/next
}
```

All described methods of business process modeling are defined on Layer 1 of our modeling hierarchy. This ensures that the common rules for business processes are set on a general level and they are valid for all introduced business processes within the business environment. Modeling a typical business process and its analysis are carried out on Layer 2 of the modeling hierarchy. The main point is that the concepts of Layer 1 are defined as Abstract, which means that there is no real object related to them. Using the extensions of such an abstract signatures specific business process elements are defined on Layer 2. For example, on Layer 1 an action is defined as an abstract concept. Particularly in Alloy it is expressed as: `abstract sig Action`. At the same time on Layer 2 a specific concept is defined as an extension of the signature Action introduced on Layer 1: `sig F1.OrderRegistration extends Action`. Usage of extensions ensures the consistency between concepts and models of the two layers (see Fig. 5).

### 3.3 Program Services Modeling

One of the main objectives in our approach is to ensure relations between models of different abstraction levels and to automate service sequence generation based on the business process logic. Therefore, Layer 3 of the modeling hierarchy in

**Fig. 5** Correspondence of Layer 1 and Layer 2 signatures



our approach, which is represented by services, is strictly linked with the business process layer and depends on the business process flow. As already mentioned above, a particular service represents the technical implementation of a particular business function. Service-oriented architecture implies existence of a number of services and technical methods which can be reused during the business function run. So, it is logical to introduce concepts Service and Method as an element of a given Service. In Alloy language, Methods are linked with Services using the field `consists_of` in Service signature, and Methods are linked with the corresponding business function from Layer 2 using field `corresponds` in Method signature. Based on such a simple approach a link between Layer 2 and Layer 3 of the modeling hierarchy is implemented (Fig. 6).

In addition to the introduced relation between the business function and the implementation method, there is also a fact which defines the logic of methods triggering in real time:

```
fact Synchronisation
{
  all s:Service |all m:Method| m in s.consists_of =>
    m.(s.current_method_pre) = (m.corresponds).a_pre

  all s:Service |all m: Method | m in s.consists_of =>
    m.(s.current_method_post) = (m.corresponds).a_post
}
```

Such an approach ensures the link between the technical and the business parts of SOA. This link enables automatic service triggering in the sequence defined by the business process flow.

### 3.4 Template-Based Service Orchestration

Proposed approach also allows for developing a specific method to automatically specialize generic web-service orchestration templates which are close to semantic specification of abstract services in OWL-S. In our case two main principles shape the proposed method for template-based service orchestration. First, abstract composite business processes take the role of a workflow template, thus reducing

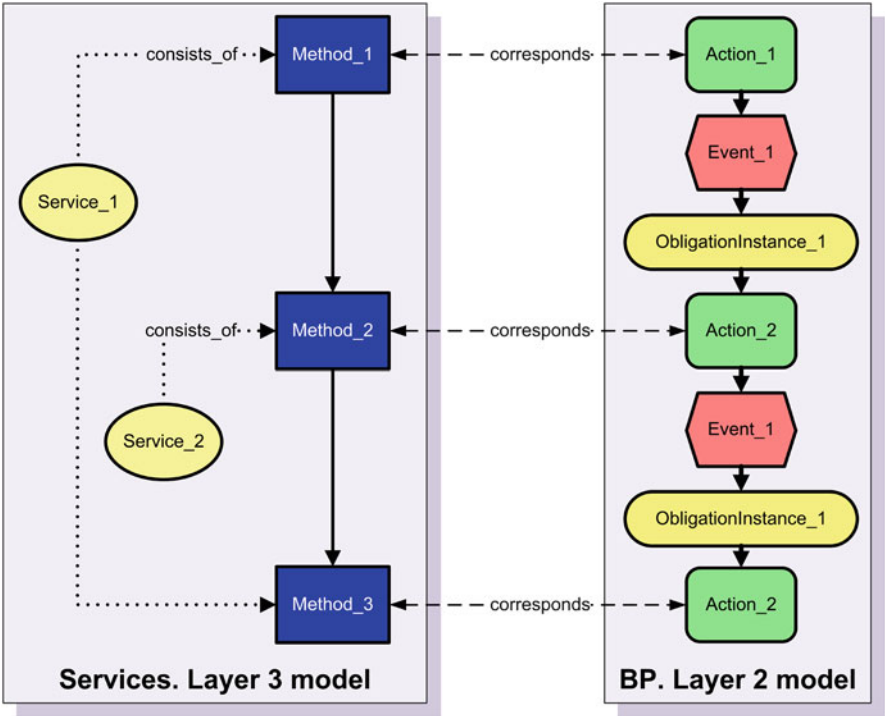


Fig. 6 The link between Layer 2 and Layer 3

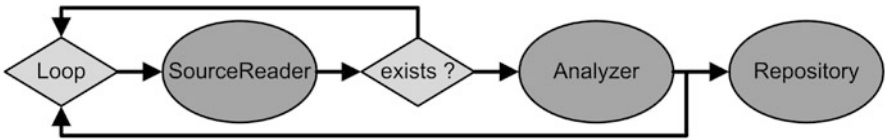


Fig. 7 One example of a generic workflow template

complexity and representing a repeating service workflow. Mainly inspired by the OWL-S definition of a composite process, for definition of workflow templates we use the limited set of such Control Constructs as: Sequence, Split, Split + Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, and Repeat-Until. Using these constructs business analysts and software architects may design general workflow templates on the basis of commonly used practices and existing processes of enterprise information systems. In most practical cases each workflow

template consists of four to five building elements with a clear control structure. Figure 7 exemplifies such a template. Diamonds define control flow elements and ovals represent abstract services.

Second, the logical theory and the principles of model finding facilitate specialization and validation of previously defined workflow templates which fit within the existing service infrastructure.

According to the stated principles we create a logical model which is capable of formal description of all needed abstract or specific service specifications. The model may conceptually be separated in four parts. The first, generic, part of our model describes such cornerstone concepts of our approach as composite processes (workflow templates), services, and control elements. This part of the model remains the same for all particular patterns and orchestration scenarios. The following Alloy statements represent key generic concepts: Condition, WorkflowElem, ControlElem, and Service.

```
abstract sig Condition { }
abstract sig WorkflowElem { }
abstract sig ControlElem extends WorkflowElem { }
abstract sig Service extends WorkflowElem {
  preconditions: set Condition,
  effects:      set Condition
}
```

Service and ControlElem are defined as extensions of WorkflowElem. In other words, services and control elements are defined as workflow elements, so they could be included into the workflow template as follows:

```
abstract sig WTemplate {
  elems : set WorkflowElem,
  first  : one elems,
  last   : one elems,
  transition: (elems - last) -> (elems - first) }

```

This formal description represents the workflow template mainly as a set of links between workflow elements, in other words, as a Directed Graph. Two other Alloy statements are needed for the purpose of services chaining in the workflow template.

```
fact f1 {
  all p: WTemplate | let t = p.transition |
  all s1,s2: Service | s1->s2 in t =>
    s2.preconditions in s1.effects
}

fact f2 {
  all p: WTemplate | all s1, s2: (Service-p.last) |
  all c: ControlElem | let t = p.transition |
  (s1->c in t) and (c->s2 in t) =>
    s2.preconditions in s1.effects
}
```

Fact f1 states that two services are linked if the preconditions of the former are included in the effects of the later. Fact f2 is quite similar, but it enables this chain through one control element. In other words, if we add a control element between two services, this control element becomes transparent in terms of service chaining.

The second part of the logical theory is domain-specific. It specifies existing services in accordance with the current configuration of the service infrastructure. One practical example of such specification is given in Sect. 4. The third part of the logical theory consists of formal specification of available abstract composite services, or workflow templates.

The following Alloy statements describe a frequently used workflow template with six nodes (including mandatory Start and Finish) and seven transitions. Further down, Start, Finish, SourceReader, and Analyser are defined as services, whereas Loop and IfCond are defined as control elements.

```

one sig P1 extends WTemplate { } {
  one Start
  one Loop
  one Finish
  one SourceReader
  one Analyzer
  one IfCond

  transition =
    Start->Loop +
    Loop->Finish +
    Loop->SourceReader +
    SourceReader -> IfCond +
    IfCond -> Analyzer +
    IfCond -> Loop +
    Analyzer ->Loop }

```

The final fourth part of the logical theory defines specific requirements for particular refinement of the workflow template and the implementation details of the service orchestration. Users will have to specify some parameters to enable selection of the relevant concrete services. These parameters are translated directly to the formal language interpretable by Alloy Analyzer. The following constraints define how to express the condition ContentExtracted.

```

pred CyclicProcess {
  one p: WPattern | p.first = Start
  one p: WPattern | p.last  = Finish
  Start.effects = none
  Start.preconditions = none

  Finish.effects = none
  Finish.preconditions = ContentExtracted
  one ContentExtracted
}

```



```

one p: WPattern | some s:SourceReader |
  s->s in ^(p.transition) }

```

## 4 The Illustrative Examples

The practical application of the proposed approach is demonstrated by the creation of described models for two particular business cases. The first business case shows the practical mapping of EPC diagrams to the relational logic and further logical analyses in order to verify information security constraints. In the second case we show another application of our template-based method of service orchestration to a particular service-oriented information system in the domain of multi-media processing.

### 4.1 *EPC Mapping and Security Constraints Checking*

In this case we examine business process of sales order creation in a large sales company. Employees of sales department register a sales order in the system-based on an MS Excel request sent by email. Once an order is saved the system automatically checks if the price was lower than minimal acceptable one, taking into account information on discounts. Minimal acceptable prices entered in the system by Finance controller are used for order check. If price of at least one item from the order violates the norms, the order is blocked. In this case the information is sent to the employee of the Finance controlling department. The employee can approve the existing price or change it to the proper one. As the price is approved by the Finance controlling department, the order is sent to the Credit controlling department. The described business process can be seen in Fig. 8.

First, the domain ontology was created and described in the Alloy language. The main entities and relationships between them were defined. Second, the business process was modeled in terms defined at Layer 1 of the modeling hierarchy, in other words, the instance of Layer 2 Model was created. Then, the program services model (Layer 3 Model) was created and mapped with the business process model. The methods for automated orchestration of services were defined and the workflow order was automatically generated. Moreover, information security restrictions were checked both on the level of business process and on the level of program services.

According to the proposed approach, the Layer 1 Model was created. Its concepts and relations are shown in Fig. 9.

For each general concept from the domain ontology the successor was defined in Layer 2 Model representing the specific process function, event, principal, and document. For example, specific functional roles were defined. Order Desk specialist was represented by the role `R1_OrderDeskClerk`,

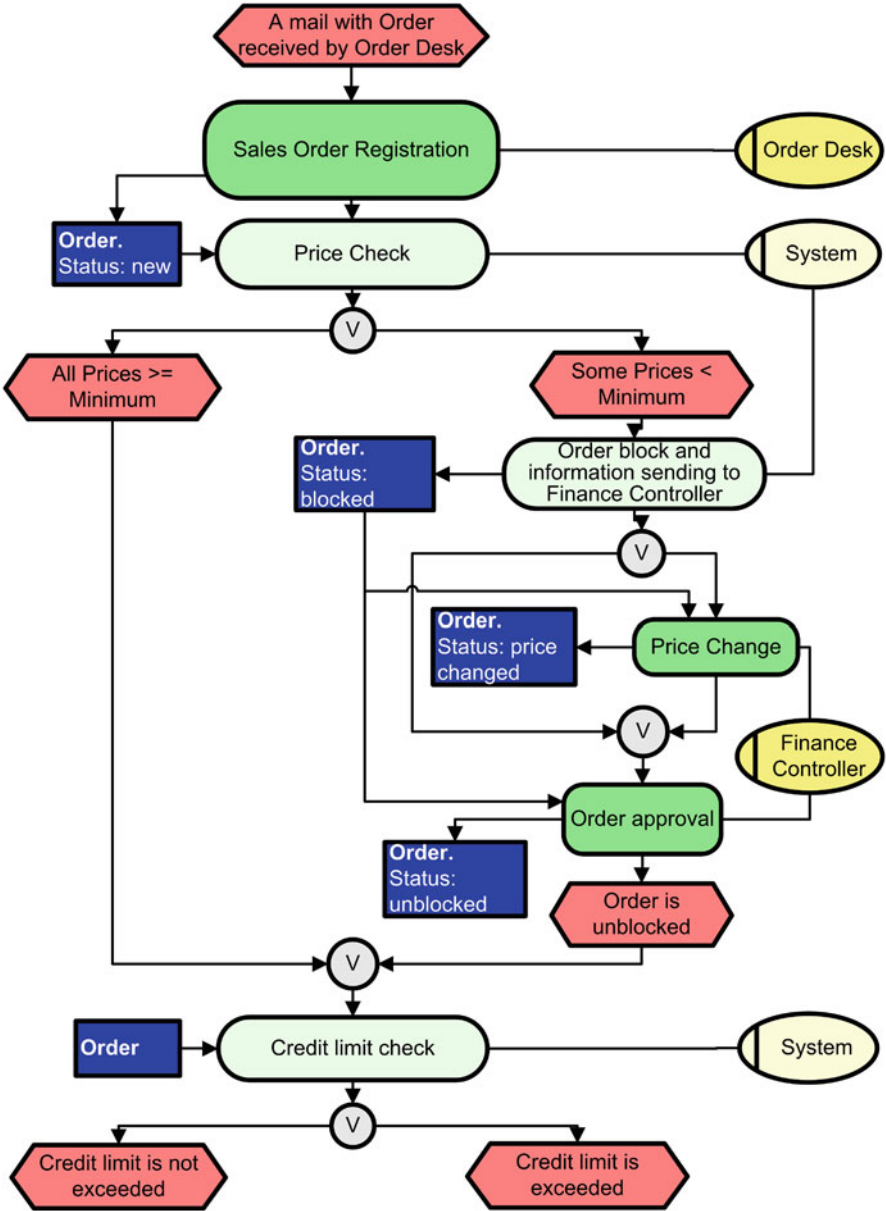


Fig. 8 The considered business process

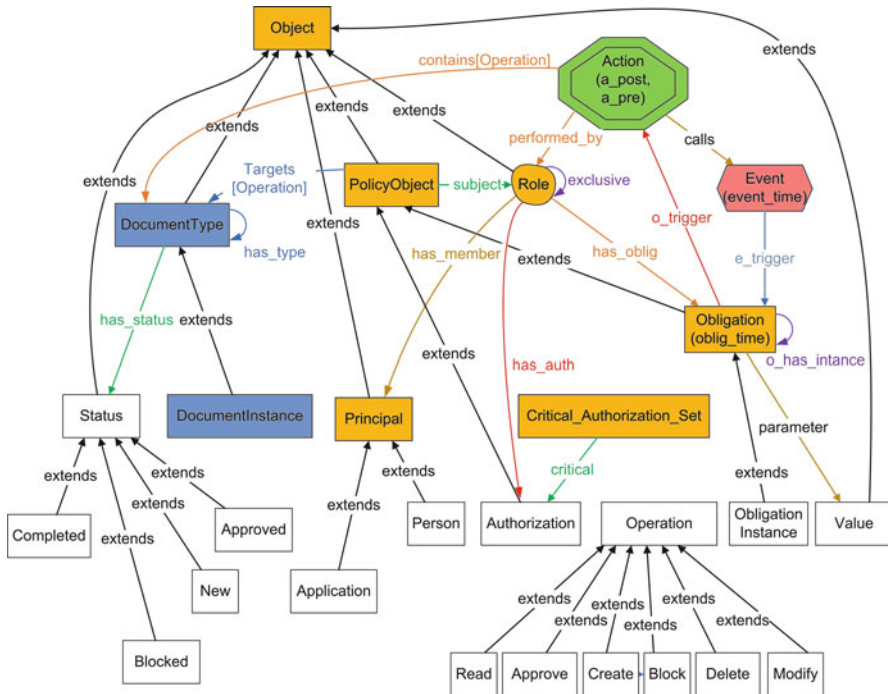


Fig. 9 Domain ontology

Finance controller—R2.FinanceController, Credit controller—R3.CreditController. Similarly, specific Authorizations, Obligations, and Documents were defined and relationships between them were set based on the ontology relations. For example, the role R1.OrderDeskClerk was set as exclusive to the roles R2.FinanceController and R3.CreditController using the relationship “exclusive” defined in Role signature:

```
one sig R1_OrderDeskClerk extends Role{
{
exclusive = R2_FinanceController+R3_CreditController
has_oblig = O1_ToRegisterOrder
has_auth = A1_ToRegisterOrder
has_member = J.Johnson
}
```

According to the EPC model, the process starts with the reception of the client’s order. The signature E1.OrderHasCome which is a successor of the Event concept was defined. The time of event was set as the very first moment TO/first represented by the successor of the Time signature. E1.OrderHasCome calls Obligation to register the order.

```

one sig E1_OrderHasCome extends Event {}
{
  o_has_instance.(e_trigger) = O1_ToRegisterOrder
  event_time=TO/first
}

```

O1.ToRegisterOrder is a successor of the Obligation signature; however, according to the modeling methodology, the event calls not Obligation but ObligationInstance that refers to a more general Obligation. Therefore, to link the current Event with Obligation the relationship o\_has\_instance which connects signatures Obligation and ObligationInstance is used. A special fact is defined to link obligation with the ObligationInstance events.

```

fact ObligationInstanceFromO1
{
  all oi: ObligationInstance |
    o_has_instance.oi = O1_ToRegisterOrder =>
    oi.o_trigger = F1_OrderRegistration
}

```

If the event triggers the obligation to register the order it means the execution of the function F1.OrderRegistration. Current model defines a strict logic of the business process, thus a possibility of accidental events and actions is not taken into account. This significantly restricts expressiveness of modeling; however, this aspect can be resolved while further model development.

Inside the Action of order registration several parameters are defined including Event that will follow Action execution, Role, which will execute Action and Document which will be the object of Action. The result of Action execution is a change of Document status.

```

one sig F1_OrderRegistration extends Action {}
{
  calls=E2_OrderCreated
  performed_by=R1_OrderDeskClerk
  contains.DocumentInstance =
  Create ((Create.contains).has_status).a_post = New
}

```

The current description corresponds to the business process part from Fig. 10. The same business process with all the connections is presented in Fig. 11. Finally, the result generated by Alloy Analyzer can be seen in Fig. 12.

In addition to the static modeling, the modeling of dynamics using Alloy may be demonstrated. A particular model is developed to show changes of the business process flow in the different moments of time. The concept of modeling dynamics using Alloy was initially described by Daniel Jackson in [16]. His concept with some modifications was used for the developed dynamics modeling algorithm. This algorithm is defined on the ontology level; however, as all the entities of the second level are successors of the first level signatures, all rules and restrictions are applied to them and dynamics on the business process level is also demonstrated.

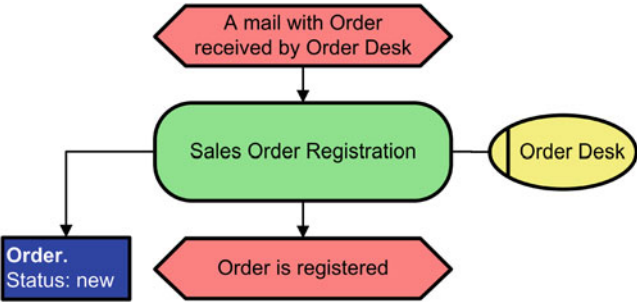


Fig. 10 The original EPC Diagram

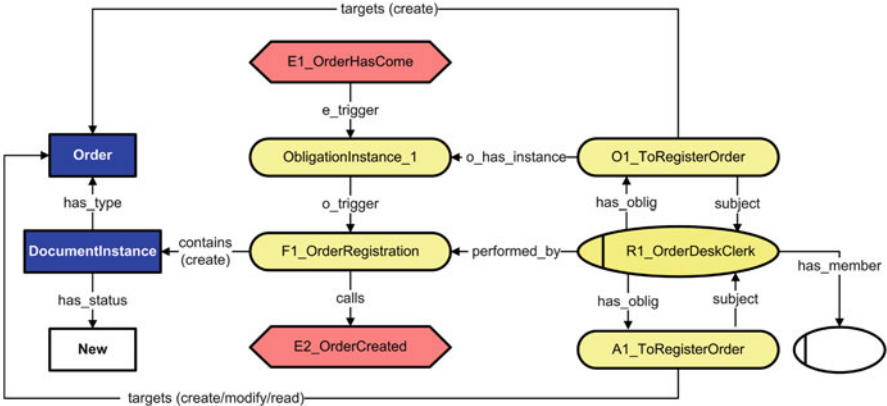


Fig. 11 Detailed information in EPC notation

MIT Alloy Analyzer generates the set of all possible scenarios of the business process flow that meet defined restrictions. The results of the particular scenario generation can be represented in several ways, including a graphical preview. This kind of representation shows graphical figures for all the instances entities described in the model. When the object (such as Event, ObligationInstance, and Action) occurs/starts or finishes executing, the corresponding time indicators appear in the figures. For example, at the moment Time0 the model shows which instances of Event, ObligationInstance, and Function occur/start executing at the first moment of time. In Fig. 10 E1.OrderHasCome has an indicator “event.time”, ObligationInstance2 has an indicator “oblig.time”, and F1.OrderRegistration has an indicator “a\_pre”. The indicators show instantiation of these objects at the current moment of time (Time0) (Fig. 13).

At the next moment Time1 function F1.OrderRegistration finishes its work (it has an indicator “a\_pre”). The function calls the next Event: E2.OrderCreated. The Event triggers ObligationInstance

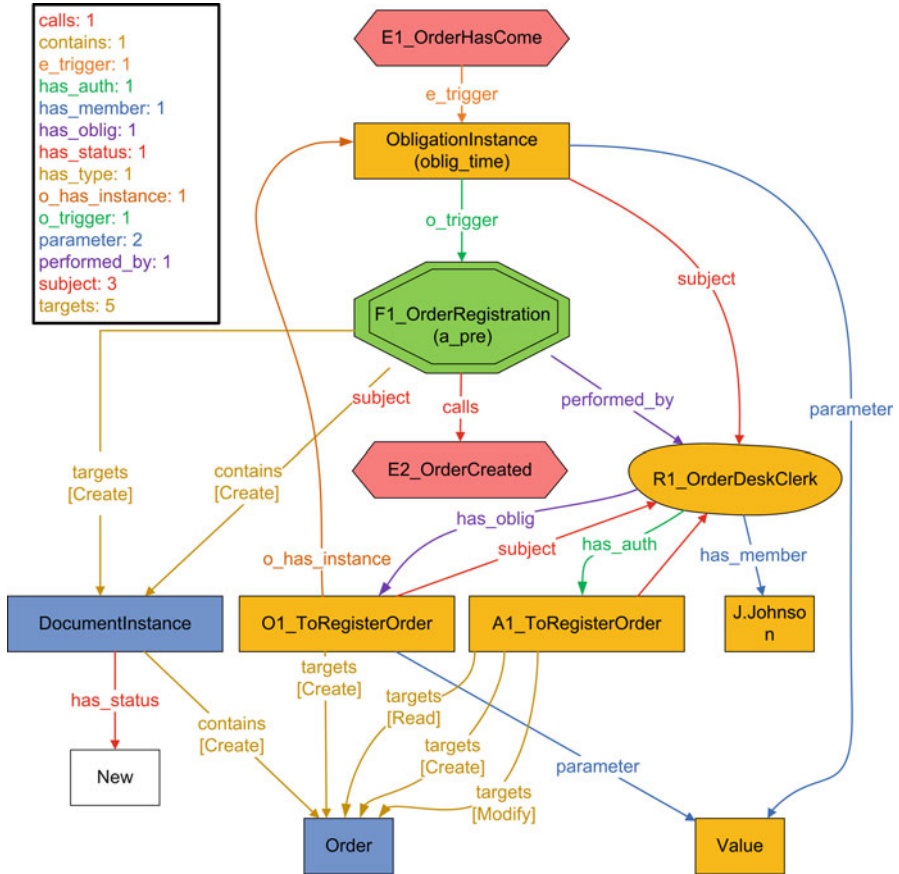


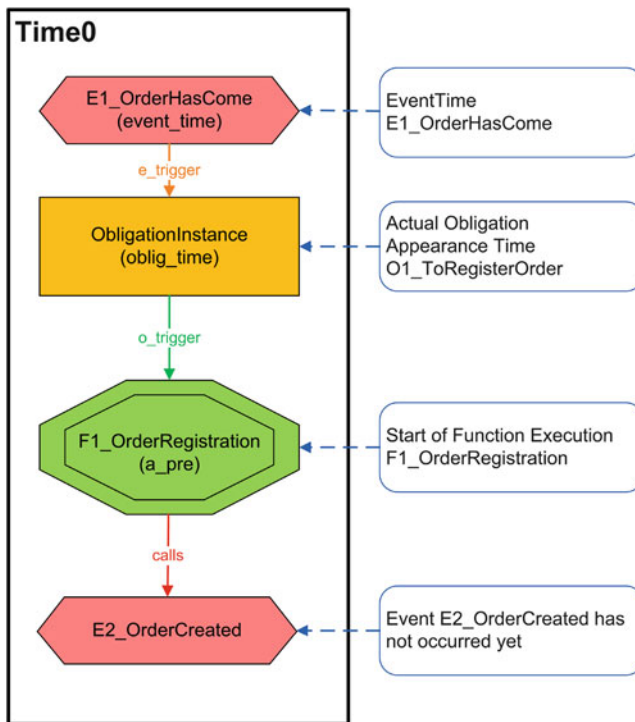
Fig. 12 The generated Alloy diagram

of O2.ToCheckPrices, which in turn initiates the start of function F2.OrderPriceCheck. (see Fig. 14).

The next step of the work is to model an algorithm which will automatically define the service orchestration scenarios using the approach to service modeling described above. Each function of the business process is associated with some program methods. Also the mechanism of synchronization of a service and a business function is defined enabling the automated service orchestration.

Figure 15 shows how at Time0 moment the function F1.OrderRegistration and the corresponding method M1.OrderRegistration (Service1) start execution.

At Time1 moment M1.OrderRegistration finishes its work and according to the business process the function F2.OrderPriceCheck should start execution. That, in turn, causes the method M2.OrderPriceCheck (Service2) to start execution (see Fig. 16).



**Fig. 13** The business process state at Time0

As a result of modeling at Layer 3, the sequence of services is defined for each scenario of the business process. In order to get the complete BPEL code those scenarios should be compiled together in one common XML code. The compilation of XML code and translation into BPEL is beyond the scope of the current case study.

Nevertheless, applicability of the developed method for automated orchestration of services was proved. Alloy assertions enable check of information security controls. These controls are thoroughly described in [27]. For example, the created model enables verification of the following assumptions:

- One principal does not have two exclusive roles;
- One principal does not have a set of critical authorizations;
- One role does not have a set of critical authorizations;
- No role can execute all the actions in the same document;
- Exclusive roles have different authorization sets;
- One service cannot execute all the actions in the document.

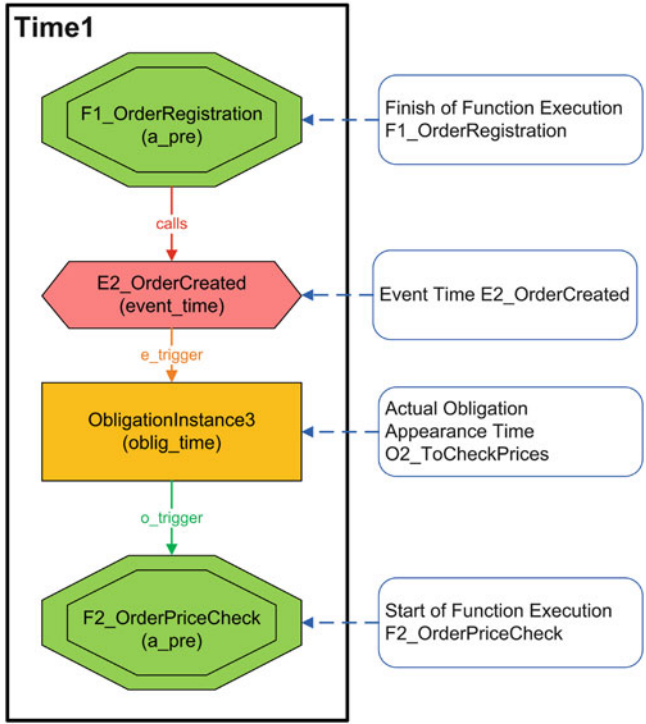


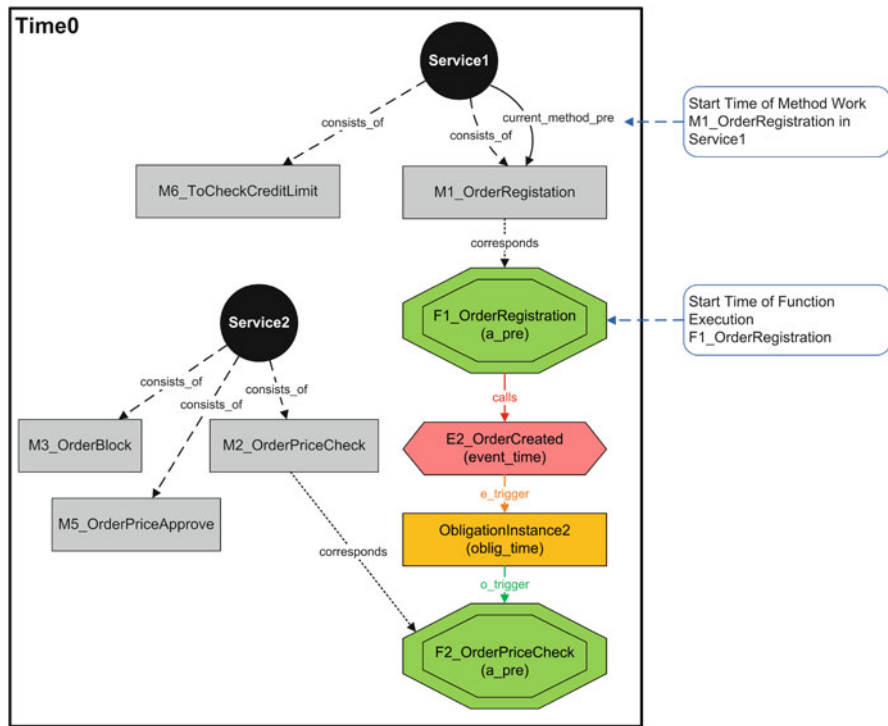
Fig. 14 The business process state at Time1

## 4.2 Service Template Specialization for WebLab Platform

In this sub-section we will focus on exploring our method of template-based orchestration (see Sect. 3.4) in the particular context of the WebLab platform providing a real-life set of abstract and concrete services. The WebLab platform [11] facilitates the development of multimedia projects leveraging a rich set of tools to create complex processing chains and dynamic graphical front-ends for domain users. The platform has different groups of users interested in design and using complex service chains of multimedia processing services. Each group of the users has own preferences and domains of discourse.

The WebLab platform has a service-oriented architecture and uses a common data model and generic interfaces. However the design of customized applications and sustainable management of individual services or composite processes in the WebLab platform still require considerable intellectual efforts and time. Firstly, the WebLab platform contains a large and frequently changing set of heterogeneous multimedia components which are represented as services and may be integrated in processing chains in different ways. Secondly, different groups of the WebLab users prefer different approaches to description of service functionality.





**Fig. 15** Service launch at Time0

The semantically enriched service registry of the WebLab Platform [6] facilitates integration of the different viewpoints and maintains links between OWL-S and the domain-oriented WebLab vocabulary. An upper level ontology and specific inference rules help to seek a particular service and automatically infer the set of implementation-specific characteristics of that service (in particular, Input, Output, Preconditions, and Effects of the service—IOPE) on the basis of domain-oriented initial information. In turn, WebLab experts (like automated agents) are now able to search this registry for the services which fit with the needed IOPE (even indirectly).

However, currently the semantically enriched service registry of WebLab platform still does not have an opportunity to populate abstract OWL-S processes with particular instances of appropriate simple processes, thus the registry cannot implement service orchestration tasks. To do this we propose using formal methods of Relational Logic and specific tools of formal analysis such as Alloy Analyzer.

At first, we need to express the second part of the logical model in the particular terms of the existing WebLab services. For this purpose we use information from the semantically enriched WebLab service registry which stores information about service taxonomy and IOPE for each available service. An example given below shows the Alloy declaration of some known IOPEs of WebLab services.

```
sig AsNativeContent, ContentExtracted extends Condition{}
```

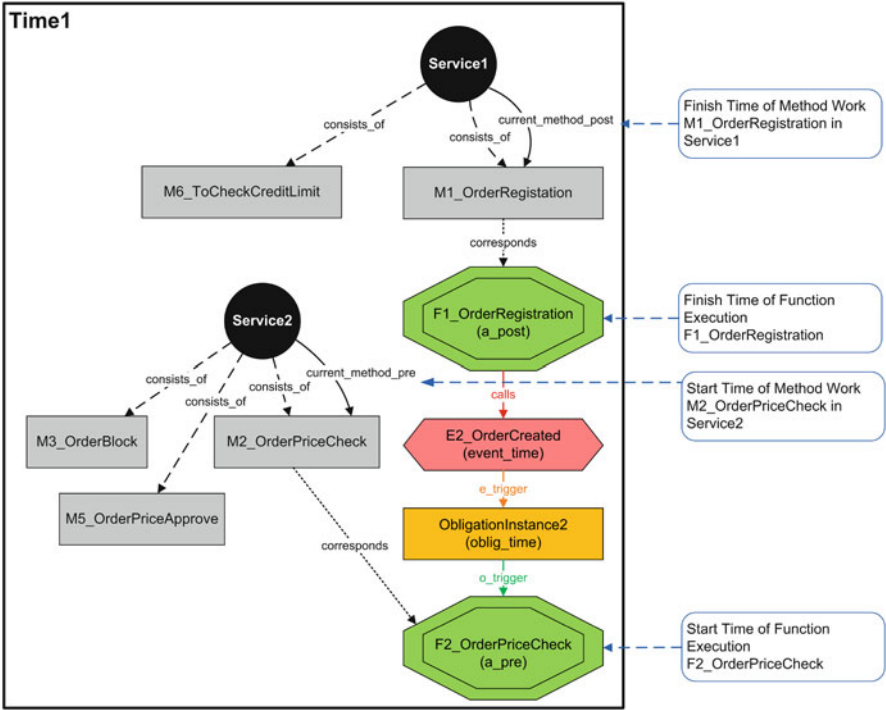


Fig. 16 Service launch at Time1

We also have to define service specification for each item in the registry. Such specifications are done in accordance with the following contents of the registry.

Service name	Preconditions	Effects
WebCrawler		asNativeContent
RSSCrawler		asNativeContent
Normaliser	asNativeContent	contentExtracted
EventExtractor	geoExtracted neExtracted	eventsExtracted
NEExtractor	contentExtracted	neExtracted
GeoExtactor	contentExtracted	geoExtracted
FullTextIndexer	uniqueURI	searchable
FileRepository		uniqueURI
TextCleaner	contentExtracted	textCleaned

WebCrawler and RSSCrawler are two robots which are able to download content, respectively, from web sites and RSS feeds. Normaliser is able to extract content from an original file. In other words, it extracts a normalized text from web pages, pdf files, doc files, etc. EventExtractor, NEExtractor, and GeoExtactor are text

parsers which are able to extract information from this text. NEEExtractor service focuses on the named entities part. The GeoExtractor service is able to extract a location (city, country, etc.) and to add its geographical position. EventExtractor processes geographical information and named entities in order to notify events in the text. FullTextIndexer is a classical text indexer which enables document retrieving from a text request. FileRepository manages the resources persistence, guarantees an unique URI on each resource it saves, and delivers the previously saved resource using its URI. TextCleaner is used to remove unused empty text (multiple new lines, etc.).

Each registry definition of the service including effects and preconditions is translated into corresponding logical statements. A simple extract from the complete formal specification of the services is given below.

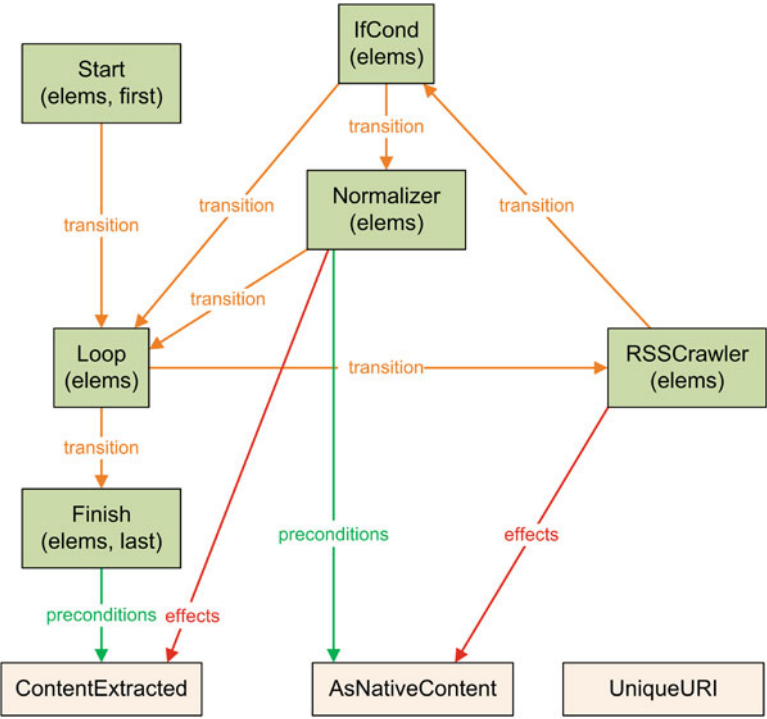
```
sig Normalizer extends Service {} {
  preconditions = AsNativeContent
  effects = ContentExtracted
  one AsNativeContent
  one ContentExtracted
}
```

The specification of all services defines the second part of the logical theory in our approach. Specification of available workflow templates is the next step in our method. In the explored use case we limited available templates to only one, already presented in Sect. 3.4. That template represents a classic processing workflow which satisfies multiple user needs. It is composed of a loop which could be interpreted as “while there is some resource to compute, compute it.” So the first treatment gets a resource from a reader, the second checks if this resource is already present in our platform. The third one analyzes this resource in order to extract certain information and the fourth one stores this analyzed resource.

Finally we need to specify and formalize requirements which will define the fourth part of the logical model. In our use case a user has needs in a service-oriented application which is able to make a continuous event extraction from different web sources about disasters on the world. The user also needs to search this collected data. Using this description, we can find high level preconditions and effects: “continuous” means cycle, “event extraction” means eventExtracted, and “need to search” means searchable. The specification of that requirement may be translated in terms of Relational Logic as it was shown in Sect. 3.4.

Having the defined logical theory we may execute logical analysis using Alloy Analyzer modeling tool: run cyclicProcess for 10. In the result we get several variants of instantiation of the signatures and relations defined in our logical theory. The collection of instantiated signatures and relations describes each possible transition between workflow elements (control elements and concrete services).

For visual overview the same result of logical analysis can be represented in the form of the graph. For example, in Fig. 17 one possible variant of instantiation is represented. Projection over instances of P1 signatures allows for intuitively



**Fig. 17** The required specification of the condition for service orchestration

clear representing the results. There are two types of nodes in the graph: workflow elements and Condition elements. The start of workflow is indicated by including the signature instance to the relation “first” (in our case it is the instance of the Start signature). The end of workflow is indicated accordingly, by including the instance to the relation “last” (it is the instance of the Finish signature). Tracing over the relation “transition” determines the control flow of the obtained solution. In our case it corresponds to the control structure of the orchestration pattern. Relations “preconditions” and “effects” show which particular preconditions are required for service execution, and which effects are present after service execution.

## 5 Conclusion

Our work introduces new approach for service orchestration. The developed approach covers main conceptual organizational levels required for business process automation: the level of enterprise application domain, the business process level, and the program service level. A formal logic language is used for modeling structural properties and dynamics of business processes.

An algorithm based on finite scope logical analysis and relational logic is developed to ensure consistency and relations between different level models. To implement the algorithm MIT Alloy Analyzer is chosen.

Finally, the application of proposed approach for two concrete cases was demonstrated. Usage of proposed hierarchical modeling, domain ontologies, and Alloy relational logics as its key element were justified.

One of the most important tasks of IT and business societies is to establish reliable knowledge management processes including usage of gathered knowledge. The described approach proposes the practical and structural way of knowledge gathering, analysis, and further using business automation based on SOA. Proposed approach provides for an opportunity to integrate different level models (enterprise, BP, service levels) and to take into account all constraints that can affect the IT solution in an automated and semi-automated way. Moreover, counterexamples showing violation of constraints can be found and demonstrated. The main advantage of the proposed approach is increased reusability of expert knowledge of domain experts and IT experts expressed through the workflow templates and an ability to use information about complex interdependences within the existing hierarchy of services. Because the requirements to the service orchestration may be redefined in an iterative manner the proposed approach satisfies the needs of software engineers who design customized service-oriented applications.

The risk of human mistakes and poor analysis inefficiency can be significantly mitigated due to decreasing the number of operations performed by people at the modeling and analysis stage during IT solutions implementation or change. In addition, time for development and implementation of new processes can be reduced. All listed benefits give an opportunity to make IT solutions based on SOA much more flexible, easier, and cheaper to implement and maintain.

Our work is significantly influenced by such SWS initiatives as OWL-S and WSMO. MDA approach [24] served as a basis for multi-layer methodology development. SUPER project [17], which had similar objectives and tasks, also significantly contributed to our work. However, our approach combines the means of MIT Alloy Analyzer with the advantages brought by joining different layer models into single conceptual model.

A correspondence to some other researches may be found, mainly to [7, 13, 29] which describe how to generate executable processes from automatic semantic services composition. However our approach has several differences. At first, our method of template-based orchestration is based on combination of a taxonomy of concrete and abstract services, and reusable domain expert knowledge in the form of workflow templates. Other approaches use only a flat set of services which is named library. Second, we propose to express requirements for service orchestration and the structure of reusable workflow templates directly using an end-user vocabulary for translation to the statements of the Alloy Analyzer language, whereas previously cited research works choose more complex dialects of logic.

As for the modeling methodology, concepts of policy objects, such as authorization, obligation, and obligation instance are taken from [27]. Also the organizational controls to be checked were worked out by Kuhn [18]. However, while the emphasis

in [27] was made on organizational control principles, our work is mainly aimed at developing architecture and a methodology for service composition. As for the modeling methods, including dynamics modeling, our work is contributed by the works of Daniel Jackson, creator of MIT Alloy Analyzer [14–16].

The SUPER project mentioned above also aims at developing a semantic-based and context-aware framework based on SWSs technology. It is intended to make companies more adaptive by management of business processes knowledge embedded in within IT systems and employees' heads [17]. However, our approach has several distinctive features. Means of MIT Alloy Analyzer are applied and different layer models are united in the single conceptual model. This enables:

- Ability of automated expansion of restrictions between all modeling levels
- Reduction of ontology transformation procedures
- Generation of examples and counterexamples for processes execution and for sequence of service invocation
- Simplification of reasoning

This work determines the key aspects of practical implementation of the proposed approach, which is the main task for further work in the current direction. In our further research we will work on developing original software architecture for a special middleware component which uses main principles of our approach for real-time composition of web services in complex distributed environments. We plan to evaluate the created component in several practical cases that will give us extra information about practical applicability of our approach to service composition.

## References

1. Bhiri, S., Gaaloul, W., Rouached, M., Hauswirth, M.: Semantic Web Services for Satisfying SOA Requirements. In: *Advances in Web Semantics I*, LNCS **4891**, 374–395, Springer, Heidelberg (2008)
2. Borgida, A., Brachman, R.J.: Conceptual Modeling with Description Logics. In: Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) *The description logic handbook: theory, implementation, and applications*, pp. 349–372. Cambridge University Press (2003)
3. Channabasavaiah, K., Holley, K., Tuggle, E.M.: *Migrating to a service-oriented architecture*. IBM DeveloperWorks (2003)
4. Curbera, F., Golan, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., Weerawarana, S.: *Business Process Execution Language for Web Services*, Version 1.1. <http://www-106.ibm.com/developerworks/library/ws-bpel/> Cited 01 May 2012
5. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Carlos, P.: IRS-III: A broker-based approach to semantic Web services. *J. Web Sem.* **6**(2), 109–132, (2008)
6. Doucy, J., Abdulrab, H., Giroux, P., Kotowicz, J.: A new approach to populate a semantic service registry. LNCS, **6724**, pp. 112–125, Springer, Heidelberg (2011)
7. Duan, Z., Bernstein, A., Lewis, P., Lu, S.: A model for abstract process specification, verification and composition. In: *ICSOC '04: Proceedings of the Second International Conference on Service Oriented Computing*, pp. 232–241. (2004)
8. Emig, C., Langer, K.: *The SOAs Layers, Cooperation and Management*, Universitt Karlsruhe (TH) (2006)

9. Erl, T.: What is SOA – Service-Oriented Architecture <http://www.whatissoa.com/p10.asp>. Cited 01 May 2012
10. Eshuis, R., Grefen, P., Till,: Structured service composition. Vol. 4102 LNCS. (2006).
11. Giroux, P., Brunessaux, S., Brunessaux, S., Doucy, J., Dupont, G., Grilheres, B., Mombrun, Y., Saval, A.: Weblab – An integration infrastructure to ease the development of multimedia processing applications. In: International Conference on Software and System Engineering and their Applications (ICSSEA) (2008)
12. Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, **5**(2), 199–220 (1993)
13. Halle, S., Villemaire, R., Cherkaoui, O., Ghandour, B.: Model-checking data-aware temporal workflow properties with CTL-FO+. In: Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC. pp. 267–278. (2007)
14. Jackson, D.: Automating First-Order Relational Logic, ACM SIGSOFT Software Engineering Notes, Volume 25, Issue 6, 130–139, November (2000)
15. Jackson, D., Shlyakhter, I., Sridharan, M.: A Micromodularity Mechanism, In Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 62–73. (2001)
16. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press Cambridge, Massachusetts (2006)
17. Karastoyanova, D., Lessen, T. van, Leymann, F., Ma, Zh., Nitzsche, J., Wetzstein, B., Bhiri, S., Hauswirth, M., Zaremba, M.: A Reference Architecture for Semantic Business Process Management Systems. In: Multikonferenz Wirtschaftsinformatik (2008)
18. Kuhn, D.R.: Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In: Proceedings of the second ACM workshop on Role-based access control, pp. 23–30. (1997)
19. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing semantics to web services: The OWL-S approach. LNCS **3387**, pp. 26–42, Springer, Heidelberg (2005)
20. Martin, D., Burstein, M., McDermott, D., et al.: OWL-S 1.2, <http://www.daml.org/services/owl-s/1.2/>. Cited 4 May 2012
21. McIlraith, S., Son, S., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems, **16**(2):46–53, (2001)
22. McIlraith, S., Son, S.: Adapting Golog for composition of semantic web Services. In: Proc. 8th International Conference on Principles of Knowledge Representation and Reasoning (2002)
23. Ning, H., Yongyi, P., Camilo, R.: Extend OWL-S dynamic semantics with rewrite logic. In: Proc. International Conference on Computer Science and Software Engineering, CSSE 2008, Vol. 2, pp. 346–349. (2008)
24. Object Management Group. MDA Guide V1.0.1. OMG <http://www.ultradark.com/01mda13userguide.htm>. Cited 4 May 2012
25. Pahl, C.: Ontology Transformation and Reasoning for Model-Driven Architecture. In: Meersman, R., Tari, Z. (eds.) CoopIS/DOA/ODBASE 2005, LNCS, **3761**, pp. 1170–1187, Springer, Heidelberg (2005)
26. Roman, D., de Bruijn, J., Mocan, A., Lausen, H., Bussler, C., Fensel, D.: WWW: WSMO, WSMML, and WSMX in a nutshell. In: 1st Asian Semantic Web Conference, pp. 516–522. Springer, Beijing (2006)
27. Schaad, A.: A Framework for Organizational Control Principles, PhD Thesis, Department of Computer Science. University of York (2003)
28. Sheshagiri, M., des Jardins, M., Finin, T.: A Planner for Composing Services Described in DAML-S. In: Proc. of Workshop on Web Services and Agent-based Engineering - AAMAS03, (2003)
29. Traverso, P., Pistore, M.: Automated composition of semantic web services into executable processes. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: International Semantic Web Conference. LNCS, **3298**, pp. 380–394, Springer, Heidelberg (2004)

30. Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The METEOR-S Approach for Configuring and Executing Dynamic Web Processes. LSDSIS technical report <http://lsdis.cs.uga.edu/projects/meteor-s/>. Cited 10 May 2012 (2005)
31. Wallace, C.: Using Alloy in process modelling. *Information and Software Technology*, **45**(15), pp. 1031–1043. (2003)
32. Wang, H. H., Saleh, A., Payne, T., Gibbins, N.: Formal specification of OWL-S with object-Z: The static aspect. In: *IEEE/WIC/ACM International Conference on Web Intelligence, WI 2007*, pp. 431–434. (2007)
33. Wegmann, A., Li, L. -, De La Cruz, J. D., Rychkova, I., Regev, G.: An example of a hierarchical system model using SEAM and its formalization in Alloy. In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*. (2007)
34. Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D.: Automating DAML-S Web Services Composition using SHOP2. In: *Proc. of the Second International Semantic Web Conference (ISWC2003)*, 2003
35. Yang, B., Qin, Z.: Composing semantic web services with PDDL. *Information Technology Journal* 9 (1), 48–54. (2010)



Dynamics of Information Systems: Algorithmic  
Approaches

Sorokin, A.; Pardalos, P.M. (Eds.)

2013, VIII, 344 p., Hardcover

ISBN: 978-1-4614-7581-1