An extended configurable UML activity diagram and a transformation algorithm for business process reference modeling

Yosra Badr

2009 / 12

The American University in Cairo
School of Sciences and Engineering

An Extended Configurable UML Activity Diagram and a Transformation Algorithm for Business Process Reference Modeling

A Thesis Submitted to the
Department of Computer Science and Engineering
in partial fulfilment of the requirements for
the degree of Master of Computer Science

By: Yosra Osama Badr
B.Sc. Computer Science
Qatar University, 2004

Under the Supervision of:
Dr. Hoda Hosny
Dr. Sherif Aly

June 2009

The American University in Cairo

AN EXTENDED CONFIGURABLE UML ACTIVITY DIAGRAM AND TARNSFORMATION ALGORITHM FOR BUSINESS PROCESS REFERENCE MODELING

A Thesis Submitted by

Yosra Osama Badr

to Department of Computer Science & Engineering

July 2009

in partial fulfillment of the requirements for

The degree of Master of Science

has been approved by

Dr. Hoda Hosny
Thesis Committee Chair / Adviser H. Hosny
Affiliation: Professor, Department of Computer Science & Engineering, The American University in Cairo.

Dr. Sherif Aly
Thesis Committee Chair / Adviser Sherif Aly
Affiliation: Associate Professor, Department of Computer Science & Engineering, The American University in Cairo.

Dr. Sherif El-Kassas
Thesis Committee Reader / Examiner [illegible]
Affiliation: Associate Professor, Department of Computer Science & Engineering, The American University in Cairo.

Dr. Awad Khalil
Thesis Committee Reader / Examiner Khalil
Affiliation: Professor, Department of Computer Science & Engineering, The American University in Cairo.

Dr. Mohamed Fahmy Tolba
Thesis Committee Reader / External Examiner [illegible]
Affiliation: Professor, Faculty of Computer & Information, Ain Shams University.

[illegible]	July 26, 2009	[illegible]	July 27, 2009
Department Chair	Date	Dean	Date

ABSTRACT

University Name: **The American University in Cairo**

Thesis Title: **An Extended Configurable UML Activity Diagram and a Transformation Algorithm for Business Process Reference Modeling**

By: **Yosra Badr**

Supervisors: **Prof. Dr. Hoda M. Hosny, Dr. Sherif Aly**

Enterprise Resource Planning (ERP) solutions provide generic off-the-shelf reference models usually known as "best practices". The configuration /individualization of the reference model to meet specific requirements of business end users however, is a difficult task. The available modeling languages do not provide a complete configurable language that could be used to model configurable reference models. More specifically, there is no algorithm that monitors the transformation of configurable UML Activity Diagram (AD) models while preserving the syntactic correctness of the model. To fill these gaps we propose an extended UML AD modeling language which we named Configurable UML Activity Diagram (C-UML AD). The C-UML AD is used to represent a reference model while showing all the variation points and corresponding dependencies within the model. The C-UML AD covers the requirements and attributes of a configurable modeling language as prescribed by earlier researchers who developed Configurable EPC (C-EPC). We also propose a complete algorithm that transforms the C-UML AD business model to an individual consistent UML AD business model, where the end user's configuration values are consistent with the constraints of the model. Meanwhile, the syntactic correctness of the transformed model is preserved. We validated the Transformation Algorithm by showing how all the transformation steps of the algorithm preserve the syntactic correctness of any given configurable business model, as prescribed by

earlier researchers, and by running it on different sets of test scenarios to demonstrate its correctness. We developed a tool to apply the Transformation Algorithm and to demonstrate its validity on a set of test cases as well as a real case study that was used by earlier researchers who developed the C-EPC.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Business process modeling has recently gained a great deal of attention due to the rapid increase in the size and complexity of organizational practices which forced organizations to continuously improve their standards to meet the growing competition. One of the major stimulators behind this trend is the desire to automate and to enhance the current processes.

Many domains and business areas, such as procurement, material management and sales, implement similar processes across many organizations, with some specific modifications that meet the requirements of each individual organization. This fact has influenced the emergence of reference models. Reference models are "generic conceptual models that formalize recommended practices for a certain domain, often labeled with the term best practice" [15]. Reference models claim to capture reusable state-of-the-art practices. Reference models were promoted by Enterprise Resource Planning (ERP) vendors, who used them as a prescription for processes that should be adopted as part of the implementation of the ERP system [11]. Reference models exist in the form of functions, data, system organization, objects and business process models, although business process models are the most popular type [15].

The process of building accurate reference models is a major issue that has been discussed frequently [11]. Although the construction of reference models is a complicated and critical task, reusing these generic reference models and customizing them to meet an individual organization's requirements is a difficult and costly task that needs to be more carefully studied and managed. The configuration process of the generic reference model is very critical because it may either lead to the success or failure of the whole organization. "Consider the example of FoxMeyer, once a $5 billion wholesale drug distributor, which filed for bankruptcy in 1996 after Andersen Consulting concluded that the insufficient customization of its reference model crippled the firm's distribution” [15].

All the reference models that are available in industry today are built using standard available modeling languages that are designed for the modeling of specific individual models rather than generic reusable ones. Hence, these languages are not capable of expressing any variation points or options or even giving guidance and help at build time, i.e. time of reusing and individualizing a business model.

A number of methodologies were applied to reuse (customize) reference models to individual cases. One new and promising approach is to extend the modeling languages to make them more configurable in order to use them to model the reference models, giving them more guidance and help. This, in return, facilitates the reusability of the reference models and makes their adaptation almost error free.

The use of such extended modeling languages introduces an additional step whereby the “configurable” generic business model gets customized to a specific individual business model. Transformation algorithms are thus needed to ensure and monitor the syntactic correctness of the derived individual reference models from the generic configurable reference models.

1.1 Motivation

Since business process modeling became an essential and crucial tool for the success of an organization, greater attention has been given to all the possible approaches that could be applied to help organizations set their own successful business models. This fact motivated a lot of academic and practical work in this field, finally reaching the idea of reference modeling.

A major difficulty with reference modeling is the way a reference model is customized to each individual organization to meet its specific requirements. Hence, alternative approaches must be considered to make the reusability of the reference model more effective, in terms of cost and time. To do so, reference models are changed to include more guidance and clarification for the user about the set of dependencies at each possible variation point. These models, known as ‘configurable reference models’, need some new modeling languages other than the standard ones, since the standard languages are not capable of expressing such dependencies or providing such guidance.

It is also essential to define algorithms that transform such configurable reference models to syntactically correct, configured and instantiated individual models.

The above approach has been applied successfully on reference models constructed using the Event Process Chain (EPC) modeling language. The Unified Modeling Language (UML), a more widely known modeling language than EPC, (and specifically its activity diagrams) was not extended likewise, despite recommendations to apply the approach on any modeling language [28].

Recent research work conducted in this area was directed towards proving the perceived usefulness and ease of use of the configurable reference modeling technique as opposed to the standard reference modeling technique. But all this research targeted the EPC language. The researchers also proved, by experiments, that the new methodology C-EPC (Configurable-EPC) possesses more expressive power than EPC in terms of identifying configuration decisions. C-EPC also increases the clarity of the configuration process in terms of selecting alternative configuration decisions in a reference model. The research concluded that the configurable reference modeling technique is very promising and needs to be further researched [16].

UML is the de facto standard language used in practice. It is also known as the “swiss army knife” of systems modeling and design activities [29]. It includes a number of modeling possibilities that have broad applications in capturing both the static and dynamic aspects of software systems. Therefore, providing configurable reference models using configurable UML activity diagrams would be useful and essential to the business process modeling community. We were strongly motivated to join the research efforts in proposing the first configurable UML activity diagram (AD).

1.2 Background

The idea of business process modeling had a strong impact on the design of enterprise systems in the 19th century and resulted in the emergence of the business process management. The business modeling community looked for several solutions to facilitate the process of modeling the business processes of enterprises. Since many business scenarios of similar industries have similar and common business processes the idea of business process reference models was brought up, also known as ‘generic’, ‘best practice’ models or ‘reusable business process’ models.

One of the major problems that emerged with reference models is how to reuse and individualize the reference models. Aalst and Rosemann [28] were the first to introduce a new approach for the re-usability of reference models, also known as ‘configuring/customizing’ of the reference models. Since all the available methodologies do not really help in the configuration process, the new approach was to indicate all the possible variation points and their corresponding dependencies in the model. But the available modeling languages, such as EPC and UML AD and Petri net, were still not capable of demonstrating such variations and providing guidance to model such configurable business process models [16]. Aalst and Rosemann [28] therefore, started to put the foundation steps for any configurable business process modeling language. Then they extended the EPC language to C-EPC. The next step was transforming the new C-EPC business models back to EPC at build time while keeping the models syntactically and semantically correct [20], [23], and [24]. Their algorithm, introduced in [13] and [14], was the first algorithm provided to ensure that the transformation/configuration of C-EPC business models to individualized EPC is syntactically correct.

Other approaches that were applied on UML AD to express variation points of the model did not provide an algorithm that preserves the syntactic correctness of the derived individualized UML AD. A detailed comparison of the previous approaches are discusses in section 2.9.

1.3 Research Objective

The idea of setting a configurable business process modeling language was only considered for EPC and not for UML AD. Hence, our research work aims at extending the UML Activity Diagram to set a Configurable UML AD (C-UML AD) modeling language covering a wider set of the requirements than those set in [28].

We also set a complete algorithm that transforms a C-UML AD business model to an individual UML AD business model that is both consistent and syntactically correct.

1.4 Research Achievement

We successfully developed and implemented the Configurable UML-AD requirements in a similar manner to the work done on C-EPC. We developed graphical notations and attributes for our newly introduced constructs. Our work also includes a full algorithm that is responsible for transforming a configurable UML Activity Diagram (C-UML AD) business model to a standard UML AD at build time without causing any syntactic errors to the model.

1.5 Thesis Structure

The remaining chapters of this thesis are organized as follows: the literature survey is covered in Chapter 2; Chapter 3 introduces the extended parts of the C-UML AD modeling language; the transformation process that monitors the derivation of standard UML AD business models from C-UML AD business models is explained in Chapter 4; and the testing and validations are presented in Chapter 5. Finally, the research summary and conclusion are made in Chapter 6.

2. Literature Survey

2.1 Business Process Models

In today's world, companies of all sizes are getting more and more sophisticated. Accordingly, there is a growing need for in-house organizations that rise to streamline the internal business processes and allow the technical staff to focus on more important issues like the markets and contextual changes, and the business's strategies.

To facilitate this streamlining, global trends are moving towards embedding information technology into the heart of the business and its processes to the furthest extent possible. However, introducing information technology to an institution that lacks proper definition of its business processes will make matters even worse, and will only slow down the processes and consume more financial and non-financial resources. Hence, proper business process modeling is a mandatory prerequisite for introducing an effective IT system to a company, and using it to take the company to new horizons of efficiency and reduction in costs.

In 1934 Nordsieck [1] stated that the structure of a company should be process-oriented and he compared it to a stream. Based on this idea and many other similar ones, business process management became a popular approach. And only since then the business process orientation managed to significantly impact the Information Systems field. Later, the concept of Enterprise Resource Planning (ERP) systems was established. ERP projects may vary in size and structure, each requiring careful management decisions during implementation. Today, these ERP systems are known as enterprise systems. Nowadays, enterprise systems need to offer more complicated and interrelated business processes to meet the organization's requirements.

Before introducing business process models one needs to clarify what is meant by a model. A model "is a representation of a part of a function, structure and/or behavior of an application or system." Representation is then considered formal when it is based on a language that is well-defined, i.e. syntax, and has a meaning, i.e. semantics, and possibly defined rules and a proof for its constructs [18].

The syntax of a model could be expressed graphically or textually. The semantics of a model could be defined on different levels of formality, based on the things being observed in the world being described, or by translating higher level language constructs into other constructs that have a well-defined meaning [31].

A model is also defined in Wikipedia [38] as "a pattern, plan, representation (especially in miniature), or description designed to show the main object or workings of an object, system, or concept." Another definition for a model is that "A model is a set of statements about some system under study." [4]

Business Process was defined by the Workflow Reference Model [26] as a "set of one or more connected activities, which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional responsibilities and relationships." Davenport and Short [26] state that a business process is a set of logically related tasks performed to achieve a defined business outcome.

In Wikipedia [38] a business process model is referred to as an enterprise process model, where a process model is used to show how things must/should/could be done in contrast to the process itself which is really what happens. A process model is roughly an anticipation of what the process will look like.

The problem with business process modeling in enterprise systems is that it is a long and tedious process in itself, especially when the task owner has to conduct it for all the processes within a given organization. Consequently, it is essential to find off-the shelf generic business process modeling packages, known as Business Process Reference Models, which could be used and customized by the organizations to model their processes in an easy and quick manner.

2.2 Business Process Reference Models

In the 19th century the term "reference" was initially used in the business language to refer to a person or company who is capable of giving information concerning the trustworthiness of a business partner. The definition of a person or place to whom or where one could appeal for his or her (social) recommendation came later [32].

Towards the end of the 1980s, Scheer [32] was one of the first users of the technical term "reference model". In one of his books he referred to the model as a reference model. The acceptance of the model as a reference model in practice encouraged Scheer to give his book a different subtitle in the second edition: *Reference Models for Industrial Enterprises* [32].

A Business Process Reference Model, or simply a reference model, is defined as a "conceptual framework that can be used as a blueprint for information system construction" [33]. Reference models are also called universal models, generic models, or
model patterns.

"The main objective of reference models is to streamline the design of particular models by providing a generic solution. The application of reference models is motivated by the "Design by Reuse" paradigm "[15]. Reference models accelerate the modeling process by providing a repository of potentially relevant models. These models are ideally "plug and play" but often require some customization and configuration [15].

Reference models can be differentiated along the following main criteria as stated in [28]:

- Scope of the model (*e.g.*, functional areas covered)
- Granularity of the model (*e.g.*, number of levels of decomposition detail)
- Views (*e.g.*, process, data, objects, organization) that are depicted in the model
- Degree of integration between the views
- Purposes supported
- User groups addressed
- Internal or external (commercial) use
- Availability of the model (*e.g.*, paper, tool-based, Web-based)
- Availability of further textual explanation of the model
- Explicit inclusion of alternative business scenarios

• Existence of guidelines on how to use these models

• Availability of relevant quantitative benchmarking data

In [33] reference models are characterized according to a set of criteria, as shown in Figure 2.1. The Application category is an important and a crucial one, if not the most important. Therefore, looking at the application methods, reuse and customization of a reference model is a critical step in the success of any reference model.

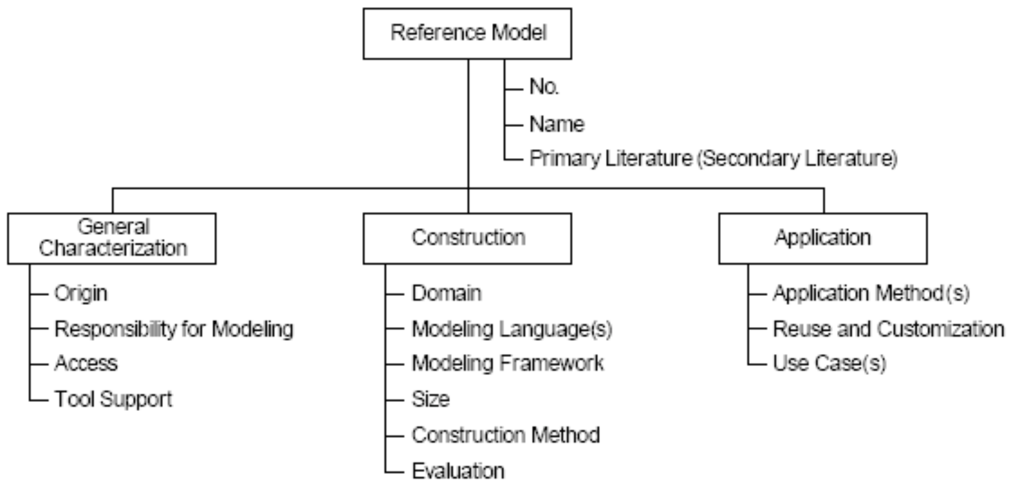


Figure 2.1. Criteria for Characterizing Business Process Reference Models [33]

The reuse and customization of the reference models are studied in more details and they are taken into a next step where they are classified into four categories according to the way the business processes within the reference models are reused: reuse by adoption, reuse by assembly, reuse by specialization, or reuse by customization [11].

1. Reuse by adoption: Reference models used by this approach are very detailed and they provide knowledge at the lowest level of abstraction. These models should be used as-is without any modifications. Most organizations do not use the full models; therefore they have to change the areas and parts of the model that do not suit them by modifying or building them from scratch. These

models do not provide any guidelines for such actions, such as SAP and Scheer reference models [32].

2. Reuse by assembly: Reference models used under this approach are also detailed and provide knowledge at the lowest level. However, they represent the processes as different building blocks that need to be selected and assembled to form a final customized model. This approach offers more flexibility than the above but keeping the whole model consistent is more difficult. The DEM reference model is an example.

3. Reuse by Specialization: Reference models used under this approach differ completely from the earlier ones; they provide knowledge at the highest levels of abstraction. Organizations using such models need to specialize these models to come up with lower level models that could be used. The problem is that no guidance is given when specializing these models. The Supply Chain Operations (SCOR) reference model falls under this category of reference models.

4. Reuse by customization: Reference models used under this approach are known as configurable reference models, and they are similar to the ones used in the reuse by adoption approach. This approach overcomes the limitations of the reuse by adoption approach by using a detailed low-level model that shows explicitly all the possible variation and configuration possibilities as well as their dependencies. This approach is a new one that has not been actually implemented on any model but has gained much interest. A configurable

reference model will show all different possible configurations of the generic model. Each organization could then easily choose and apply one of the possible given configurations. A configurable reference model must be able to provide a complete, integrated set of all possible process configurations. Only in this case can each individual model be derived from the model. In other words the configurable reference model can be described as the "least common multiple" of all process variations. The task of configuration is to create a new model by selecting that parts of the configurable model that are relevant to the user or – the other way around – by deselecting the irrelevant parts [7]. This approach is the one that we studied in depth and based our work on.

2.3 Business Process Modeling Languages

2.3.1 UML 2.0 Activity Diagrams

Many modeling languages have been set to model business processes. One of these modeling languages that were derived from the object-oriented paradigm is the Unified Modeling Language (UML) [30]. UML is a graphical language for visualizing, specifying, constructing and documenting the artifacts of software systems. UML was originally derived from the three leading object oriented methods: Booch, Object Modeling Technique (OMT) and Object Oriented Software Engineering (OOSE). Today, UML is a common standard for object-oriented modeling and is derived from a shared set of commonly accepted concepts, which have successfully been proven in the modeling of software systems. The UML is increasingly being seen as the de-facto standard for software modeling and design. [11]. However, the object-oriented methods were used to cover the implementation aspects, not the business processes. The developers of UML realized the need to extend UML's capability to model business processes. Hence, diagrams like use cases and activity diagrams are incorporated into the UML. Activity diagrams (AD) are capable of modeling business processes since they have a control and data-flow model. The Activity Diagram is represented as a graph made of nodes and edges. Data values and control are passed on from one node to another through the edges. The nodes operate on the received inputs and provide output that flows out onto the edges to be passed on to the following nodes of the diagram.

In the UML Activity Diagram the fundamental unit of behavior specification is the action. An action takes a set of inputs and converts them to a set of outputs. Actions may also modify the state of the system. In order to represent the overall behavior of a system, the concept of the activity is used. Activities are composed of actions and/or other activities and they define dependencies between their elements. Nodes represent Actions, Activities, Data Objects, or Control Nodes. The various types of Actions and Control Nodes are shown in Figure 2.2.

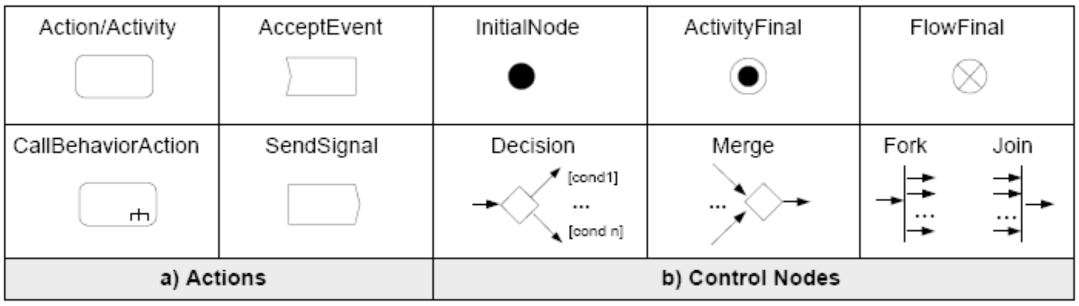


Figure 2.2. Basic Elements of the AD [12]

The diagram in Figure 2.3 shows an example of procurement logistics processes using an Activity Diagram (AD).

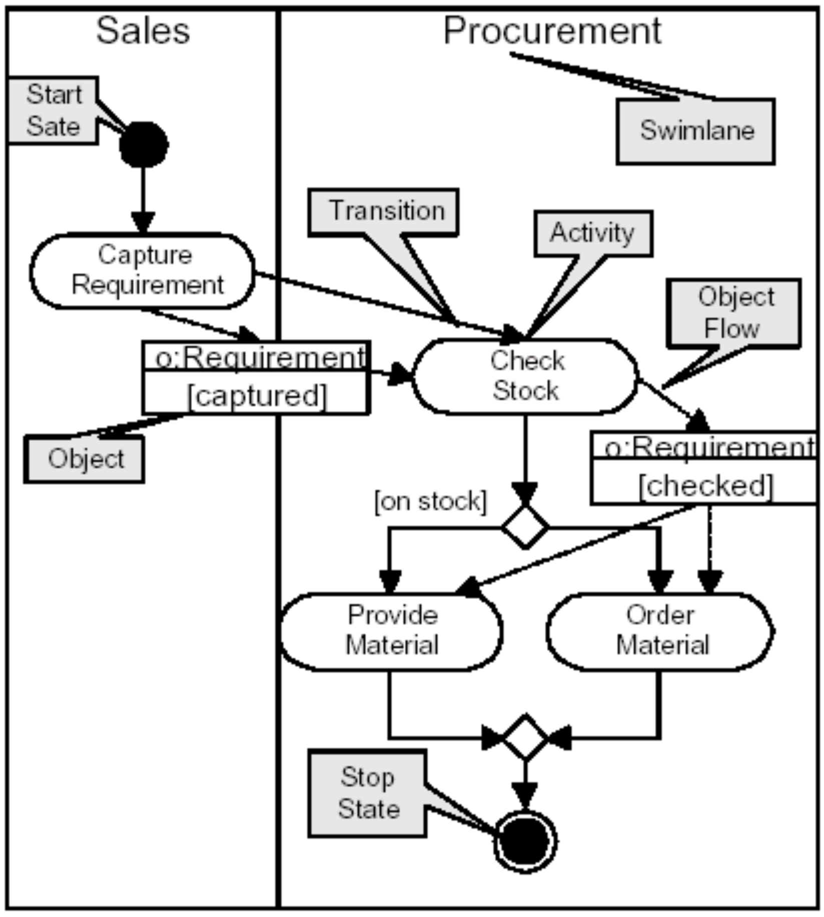


Figure 2.3. Procurement Logistic Processes Using AD [12]

2.3.2 Event Process Chain

Another modeling language that has been used by many reference models is the Event Process Chain (EPC), which was developed within the framework of Architecture of Integrated Information System (ARIS) by Scheer, Keller and Nüttgens [15]. ARIS is a standard framework for business process engineering that is accepted in research and in practice [28]. The strength of EPC lies in its easy-to-understand notation that is capable of portraying business information systems. EPC is used within many models; one of the most common reference models that uses EPC is the SAP reference model [15].

The EPC originated from the business prospective rather than from IT. It has many ambiguities and deadlocks that can cause many technical issues when checked or implemented as stated in [12]. EPC is basically denoted by directed graphs, which visualize the control flow and consist of events, functions and connectors [15]. Each EPC starts with at least one event and ends with at least one event. An event triggers a function, which leads to a new event. The basic elements used in EPC are shown in the Figure 2.4.

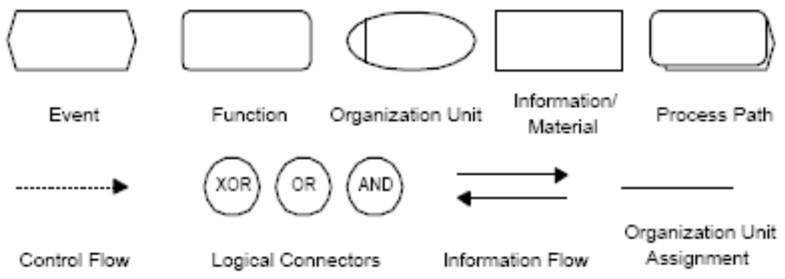


Figure 2.4. Basic Elements of the EPC [12]

Figure 2.5. shows the same procurement logistics example illustrated in Figure 2.3 but using EPC.

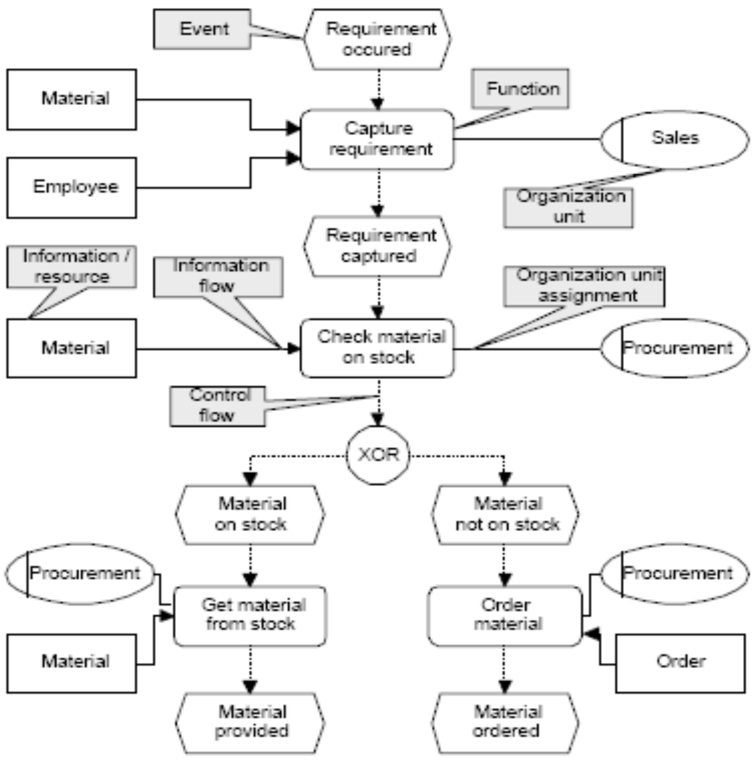
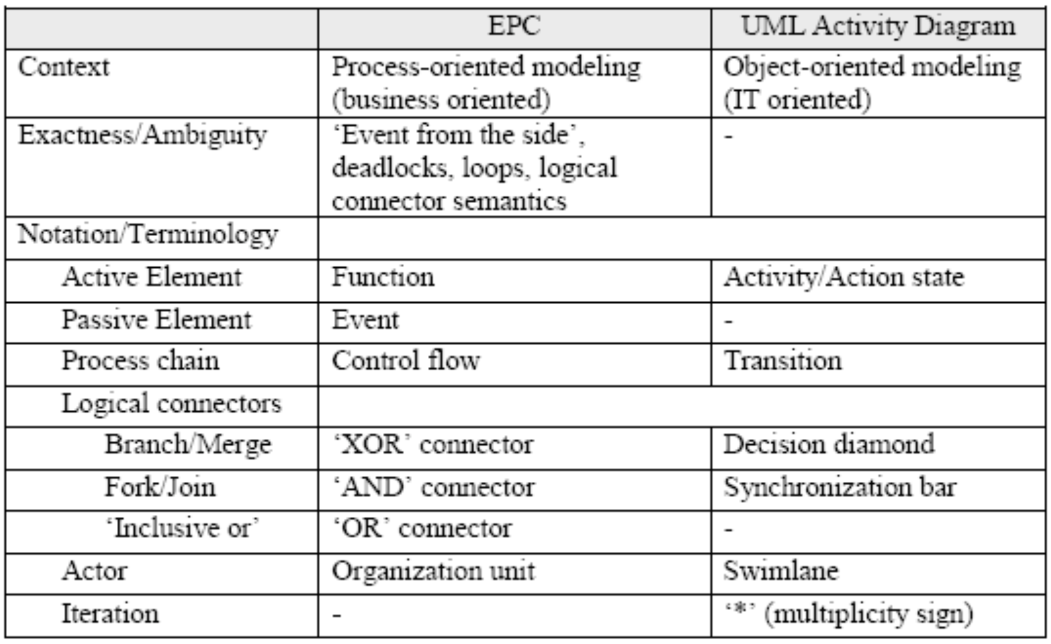


Figure 2.5. Procurement Logistic Processes Using EPC [12]

2.3.3 UML and EPC

It is important for us to study the similarities and differences between the two modeling languages to be able to convert from one form to another. Converting EPC to UML is our first step in this research since all the previous work done in the area of configurable reference modeling was done using EPC only. The comparison between them is summarized in Table 2.1.

Table 2.1. Comparison Between EPC and AD [12]



	EPC	UML Activity Diagram
Context	Process-oriented modeling (business oriented)	Object-oriented modeling (IT oriented)
Exactness/Ambiguity	'Event from the side', deadlocks, loops, logical connector semantics	-
Notation/Terminology		
Active Element	Function	Activity/Action state
Passive Element	Event	-
Process chain	Control flow	Transition
Logical connectors		
Branch/Merge	'XOR' connector	Decision diamond
Fork/Join	'AND' connector	Synchronization bar
'Inclusive or'	'OR' connector	-
Actor	Organization unit	Swimlane
Iteration	-	'*' (multiplicity sign)

The process of transforming EPC models to Activity Diagrams is almost straight forward with one major problem. The inclusive "OR" in EPC is not represented with a corresponding connector in Activity Diagrams; hence, it could be represented indirectly as shown in Figure 2.6:

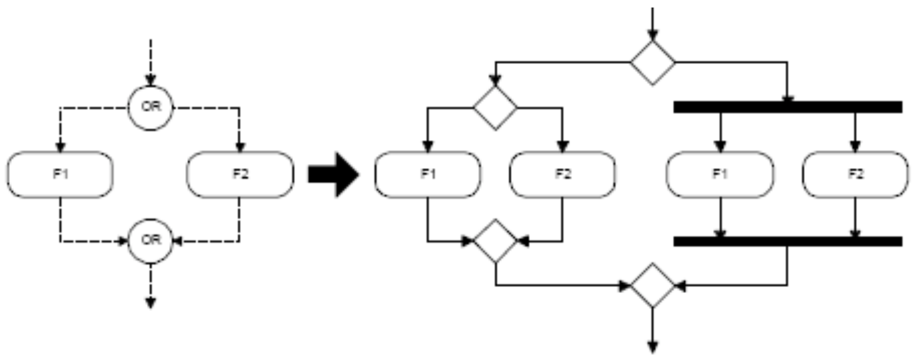


Figure 2.6. Expressing inclusive "OR" Using EPC and AD. [12]

In [10] a thorough comparison was made between the syntax, semantics, tool availability and software lifecycle availability of EPC and UML AD modeling languages. The following points were made to sum up the advantages of UML AD over EPC:

1. UML is a commonly accepted international standard while EPC is mainly used in Germany (specifically for organizations using SAP software).

2. There is a huge set of implementation tools available for UML AD with different ranges of prices than there is for EPC.

3. Since UML is a de facto standard of software engineering, all professional software engineers (and most computer scientists) will have had at least some exposure to UML, but frequently none to EPC. Therefore, UML AD is a more viable choice in a software development project than EPC.

4. More scientific books, papers and tools are available for UML AD than EPC

None of the above reasons is entirely compelling but together, however, it is likely that UML AD will stay dominant over EPC in the long run. "Even for specialties of EPC, where it currently still has an advantages over UML AD (such as certain tools, as SAP tools) UML AD will spread to become the standard notation. How long this process may take remains to be seen." [10]

2.4 SAP Reference Model

The SAP Reference Model is a set of information models that are utilized to guide the configuration of SAP systems. It is used to describe all areas of the system, from logistics to personnel time or compensation management. The SAP R/3 reference model has evolved in the middle of the 1990s in different versions to suit the implementation and configuration of various systems [6]. It is said to be the biggest enterprise system vendor worldwide with more than 100,000 installations [39]. It covers data and organization structures, but it is mainly known for its business processes.

The widespread and practical acceptance of SAP motivated our choice of using it in this research. As mentioned earlier, the SAP uses the EPC modeling language. In this research work we rely on parts of the SAP reference model to demonstrate and verify our results so that we can compare our work with the work done on EPC.

2.5 Configurable Business Process Reference Models

The configuration process is defined by Davenport [35] as an approach applied to balance the business IT functionalities with its requirements. It is also described by Soffer [2] as an alignment process of adapting the needs of the enterprise to its system.

Configuration and customization are very close and they are often used interchangeably. As stated in [1] and referred to by the Webster's Collegiate Dictionary configuration is defined as "relative arrangement of parts or elements" while customization is defined as "to build, fit, or alter according to individual specifications" [25].

Configuration of business reference models could be summarized as the process of adapting business processes of a given generic reference model to meet the requirements and needs of an enterprise. However, the main problem with the configuration of reference models is the lack of explicit support on how and where to apply such configuration in the reference model and the set of dependencies that will result due to such configuration. Accordingly, there is a shortage in the available standard reference modeling languages since they are not capable of highlighting and expressing the variation points and their corresponding dependencies. Consequently, the unguided configuration of reference models may lead to inappropriate semantic and syntactic reference models [3]. Hence, Aalst and Rossmann, as mentioned earlier, developed the concept of Configurable Reference Models using extended modeling languages that assist in the construction of such reference models.

The idea of configurable business process reference modeling is simply a model that represents a set of all possible variants of the model. This idea is consistent with the concept of Software Production Lines (SPLs), where all the possible alternatives are captured as variation points in the model or system. [17]

2.6 Configurable Business Process Modeling Languages

The construction of configurable reference models need to be supported by configurable modeling languages that are capable of expressing the variation points and their corresponding dependencies.

A configurable modeling language has to capture decisions at the type level as well as instance level. Decisions at type level, i.e. at build time, have an impact on the actual structure of the model unlike decisions at instance level, i.e. at run time. The build time decisions have to be clearly identified in a model and they are known as variation points.

There is also configuration time, which is defined as “the moment in time where configuration decisions need to be made” [1]. A model in this phase cannot necessarily be executed. It rather captures different alternatives for a domain and has to be configured before it can serve as the actual build time model for individual process instances.

All the above decision levels are summarized in Figure 2.7 using EPC notation [15].

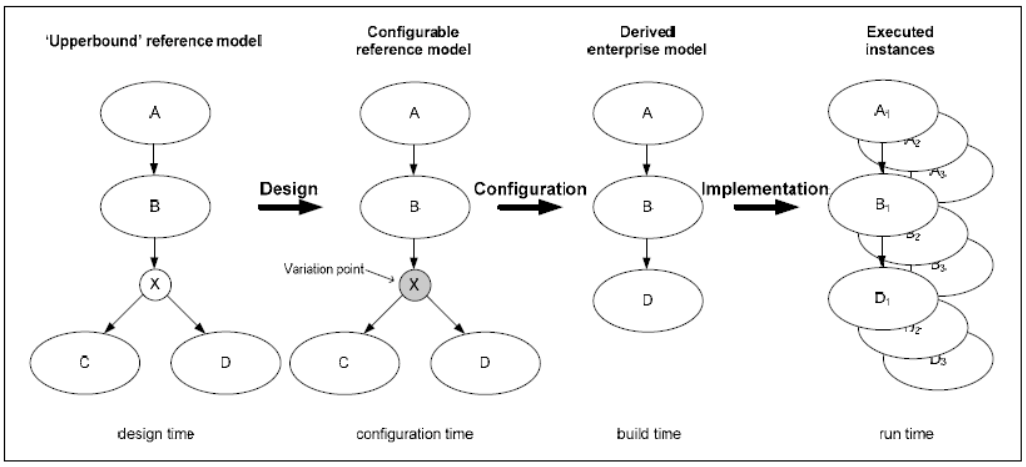


Figure 2.7. The Different Decision Levels [15]

2.6.1 Configuration Patterns

The first step towards establishing configurable modeling languages is to analyze the different configuration scenarios of business processes. To do so, configuration patterns were developed in [1].

Configuration patterns are defined as “patterns which depict a configuration scenario and highlight the potential implementation alternatives that are available. A configuration pattern shows the options that are available at configuration time.” [1].

These configuration patterns were used to construct configuration notations forming an extension to the standard EPC modeling language and thus leading to a new configurable modeling language; Configurable-EPC (C-EPC).

The configuration patterns according to Aalst [1] are summarized as follows:

1. Optionality: Functions within EPC that could be switched ON, OFF or OPTIONAL.

2. Parallel Split: This pattern signifies a point in the model where one path is split into multi paths all of which need to be executed in synchronization. In EPC this comprises the AND connector in a split.

3. Exclusive Choice: This pattern considers all alternative cases involving a configurable XOR connector in a split.

4. Multi Choice: This pattern considers all possible alternatives available at an OR connector split.

5. Synchronization: This pattern is similar to pattern 2 except that it considers the alternatives at an AND join, where at least two branches are being joined instead of one being split.

6. Simple Merge: This pattern is similar to pattern 3, except that again it considers the alternative cases where the paths are being joined by an XOR connector instead of being split.

7. Synchronizing Merge: This pattern is similar to pattern 4, except that it considers the alternatives of an OR merge and not a split.

8. Interleaved Parallel Routing: This pattern considers the case when the order of execution of a number of processes is configurable.

9. Sequence Inter-relationships: This pattern is based on the case where configuration of one process could depend on the configuration of another isolated one. This interdependency is described as a relationship.

2.6.2 Configuration Requirements

A configurable modeling language has also to demonstrate the following characteristics as stated in [28]:

1. The language has to support configurations regarding entire processes, functions, control flow and data.

2. It should be possible to differentiate configuration decisions into mandatory and optional decisions.

3. Configuration should be differentiated into global and local decisions.

4. Configuration decisions should also be differentiated into critical and non-critical decisions.

5. Configuration decisions can have interrelationships. Any pre-requisites for a configuration decision should be clearly highlighted.

6. Configuration decisions can be made on different levels.

7. Variation points should refer to further related information within the Enterprise System. This can include the system online help or the system implementation guide.

8. The entire configuration process should also be guided by recommendations or configuration guidelines.

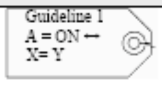
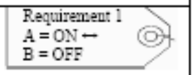
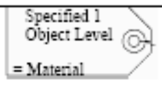
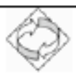
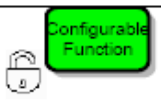
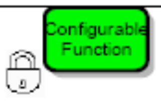
9. Enterprise System reference models are already very comprehensive. Any further extension of these modeling languages has to carefully consider the impact on the perceived model complexity.

2.6.3 Configuration Attributes and Configurable Nodes

Based on the above configuration patterns and configuration requirements, configuration attributes are constructed for describing the configurable nodes. In EPC Configurable nodes consist of configurable functions and configurable connectors.

The configurable nodes are denoted with a thicker border than non-configurable ones [1]. The configuration attributes are summarized in Table 2.2.

Table. 2.2. Configuration Attributes [1]

Name	Description	Notation	Configuration Pattern or Configuration Requirement [RoAa03]
Guideline**	Soft Recommendation: guides possible configuration decisions		Pattern 9 *Sequence Inter-relationships*
Requirement**	Hard Recommendation: used to describe a system constraint		
Specification Level	This notation element is used to specify the level at which a configurable node needs to be specified. This can be either at System, Object or Occurrence Level		
Routing Container	Order of configurable functions can be changed arbitrarily at configuration time. However, a decision can be made at runtime to specify which order the functions can be executed in		Pattern 8 *Interleaved Parallel Routing*
Non-Critical Configurable Function / Connector	The manner in which the Node is configured is a non-critical decision (by default)		Configuration Requirement: *Critical* and *Non-Critical Decisions*
Critical Configurable Function / Connector	The manner in which the Function is configured is a critical decision. The configuration decision is only hardly reversible		
Default	A configurable connector can have a default like a particular sequence	e.g., *default= ON* or *default= seq 1*	Configuration Requirement: *Mandatory* and *Optional Decisions*
Optional Node	If no explicit configuration decision was made a configurable function is switched on or off by by default. Connectors can have default configuration values as well.	e.g., *opt, default= ON* or *opt, default=seq1*	

The formal description and details of building a configurable-EPC can be found in [28]. The language is not fully configured since it does not cover all the aspects of a configurable modeling language yet.

The earlier work done in the area of constructing a configurable EPC (C-EPC) did not cover all the requirements discussed above in section 2.6.2. The following list reflects the parts that were covered by C-EPC in [28]:

1. The C-EPC language mainly focuses on the process and control-flow aspects. The data aspect and function aspect have not been addressed explicitly. Note that functions can be configured but this only refers to their presence rather than the functionality of these functions.

2. It does not distinguish between mandatory and optional decisions. To do so, an additional attribute has to be added to the configurable nodes identifying whether their configuration is mandatory or not.

3. It does not differentiate between global and local decisions. Again, another attribute has to be added to each configurable node. However, the real challenge is to get this information.

4. Similar remarks hold for the difference between critical and non-critical decisions.

5. Configuration decisions can have interrelationships. This is partly covered by the requirements and guidelines in a C-EPC. However, these are restricted to interrelationships within one model.

6. Configuration decisions can be made on different levels. This can be supported by the concept of partially configurable C-EPCs. A partially

configured C-EPC is simply a C-EPC with some nodes that have already been configured and others that are not. A partially configured C-EPC allows the configuration to take place at subsidiary levels.

For example, there could exist a top level C-EPC model like the SAP model, which indicates all the possible configurations with respect to a given process. This model could be partially configured for each industry, i.e. some nodes could be configured and others could be left out for configuration in the next level. Then, each industry could use this partially configured C-EPC as a starting point within a given organization. For large organizations there may be different versions of the same process for each country or region. Therefore, the industry specific C-EPC may be partially configured into an organization-specific C-EPC. Only the bottom organization-specific C-EPC has to be configured completely and not partially to support a concrete business process model.

This level specification requirement could also be added as an attribute to each configurable node indicating at which level its configuration should take place.

7. In C-EPC variation points do not refer to further related information within the enterprise system. However, this could be easily applied.

8. The entire configuration process should also be guided by recommendations or configuration guidelines. This is supported by the guidelines attribute.

9. The last requirement refers to the impact of configuration extensions on the perceived model complexity. The C-EPC is a natural extension of the standard EPC and should not cause any problems for the typical user of a reference model. The only addition to the language is the logical expressions used in describing the interrelationships used in the guideline and requirement attributes. To maintain the simplicity of the model further investigations have to be conducted to convert such logical expressions to some sort of graphical notations.

2.7 EPC Invoice Verification Business Example

The following example was used in [1] and [28] to show the extended C-EPC modeling language. Figure 2.8 shows part of the SAP reference model SAP R/3 Ver. 4.6c using the standard EPC language.

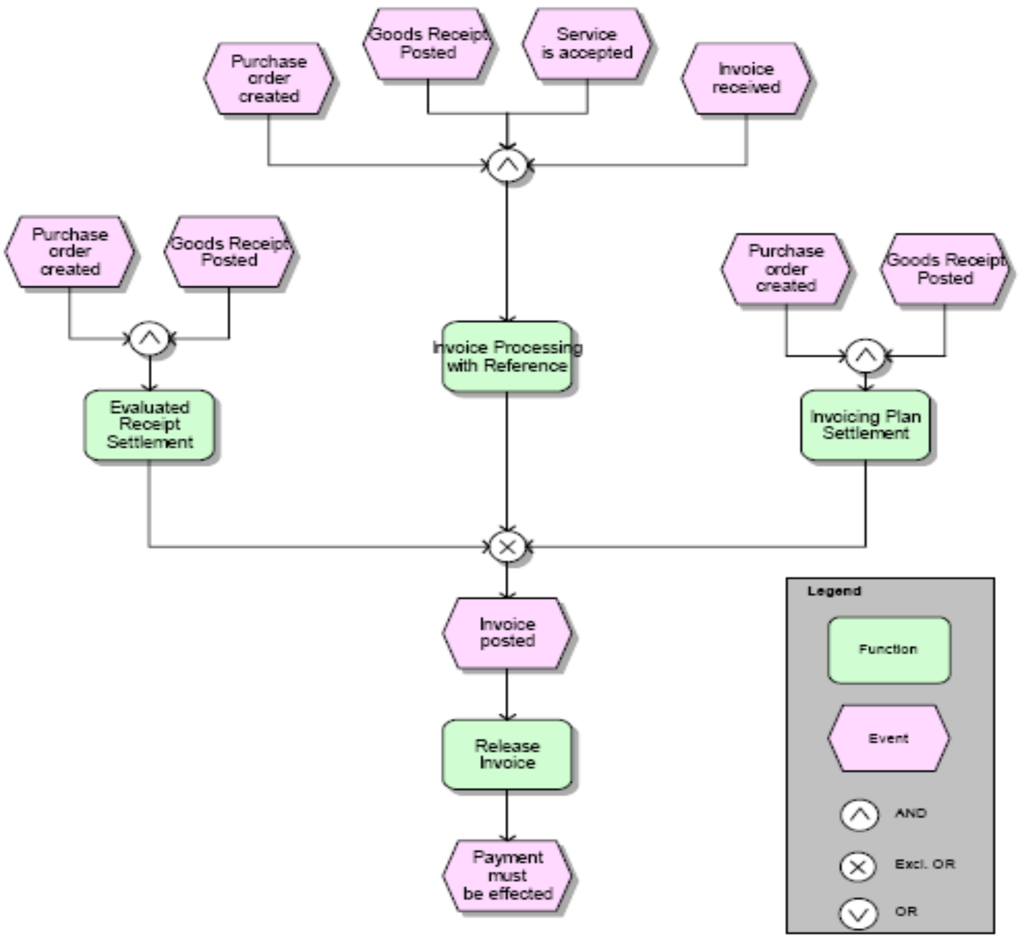


Figure 2.8. Invoice verification Example Using EPC ([1], [28])

Figure 2.9. shows the same SAP reference model section using C-EPC. This model shows three configurable connectors and two configurable functions (Evaluated Receipt Settlement (ERS) and Invoicing Plan Settlement).

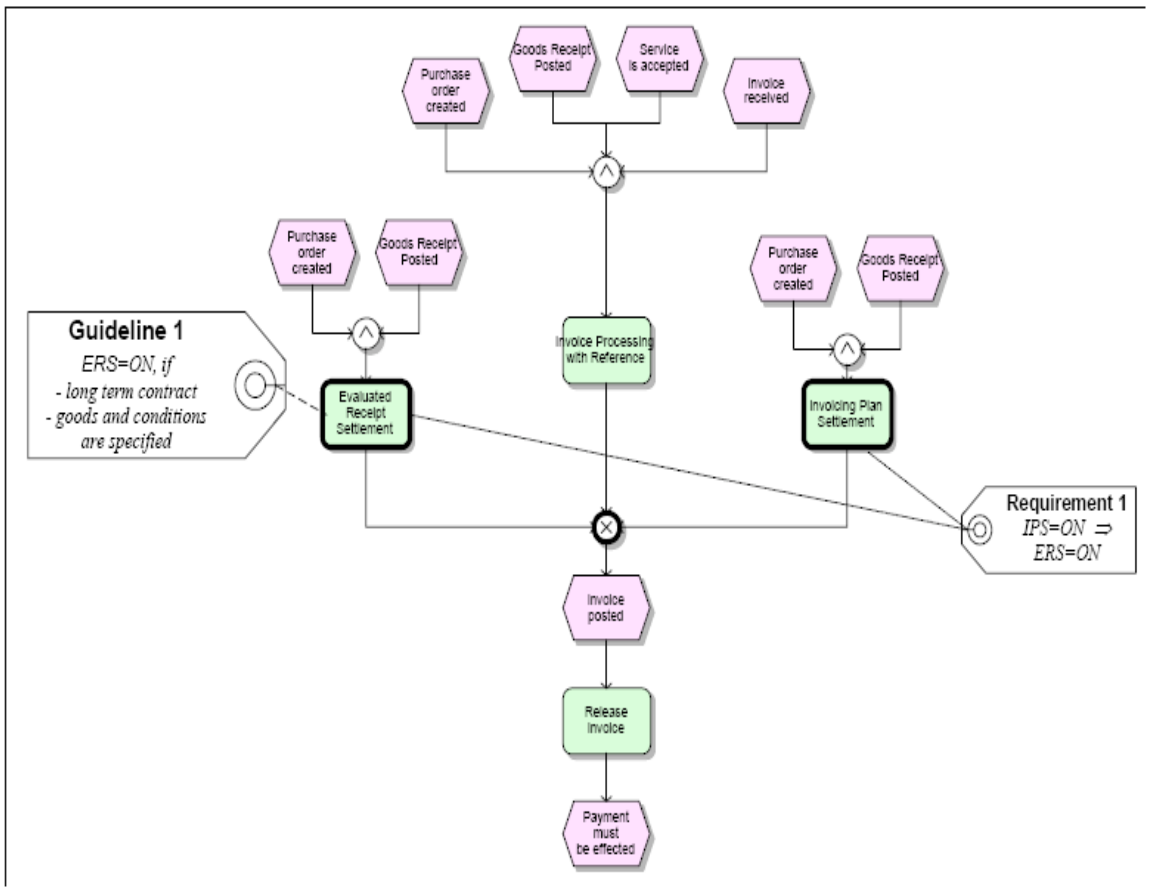


Figure 2.9. Invoice verification Example Using C-EPC ((1], [28])

Based on the organization's requirements and needs in a given scenario the above model was configured by turning the Evaluated Receipt Settlement (ERS) ON. The transformation process is discussed in more details in the next section.

2.8 The Transformation Process

The transformation process of a configurable model to a configured model is the process of deriving a configured build time model according to the organization's requirements and needs from the configurable reference model. The terms configuration, individualization and derivation are used interchangeably in the literature with the term transformation.

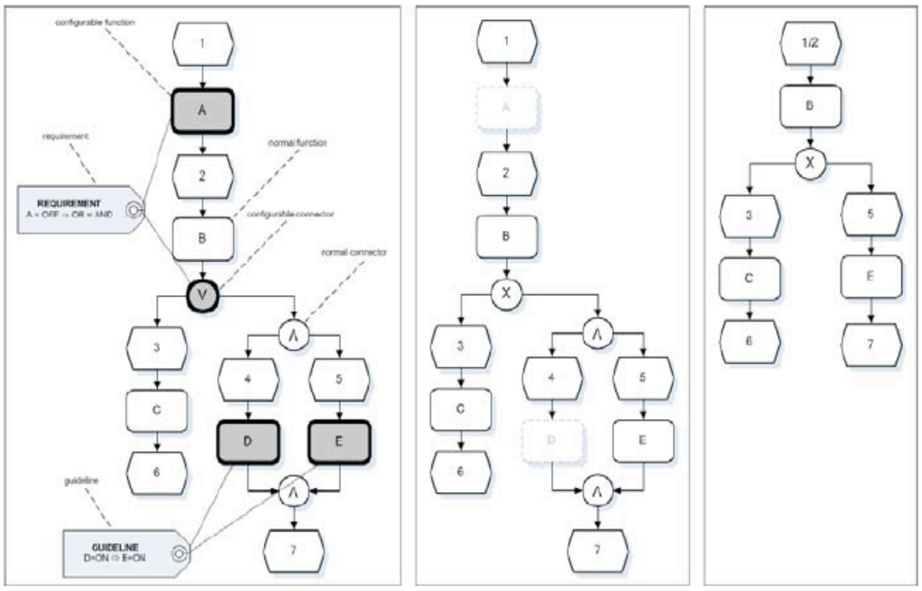


Figure 2.10. An Example with a Syntactic Error [14]

This process may sometimes result in a syntactically and semantically incorrect model. This issue has been ignored for a long while, although several of these problems were mentioned in [14].

Aalst and Rosemann in [28] tried to establish a systematic approach for ensuring the syntactic correctness of the derived build time model using the C-EPC and EPC modeling language. Let us consider the example in Figure 2.10 which shows one of those syntactic problems. The rightmost part of Figure 2.10 shows a syntactically incorrect model, where two events follow each other directly, after function A has been turned off and removed (Event 1 and Event 2). Therefore, establishing a transformation algorithm that ensures the syntactic correctness of the derived build time model is essential for the success of the whole configurable reference modeling methodology. The next section illustrates the steps that were applied in [13] to ensure such correctness, using the C-EPC modeling language. The transformation process constitutes three major steps:

1. Deriving Configured Functions
2. Deriving Configured Connectors
3. Deriving the complete model by excluding all unnecessary paths.

The calculation of deriving a correct configured EPC is based on the minimality criterion: “if elements have to be added by a configuration, add as few elements as possible; if elements have to be removed by a configuration, remove as many as possible and optimize the graph so as to include no unnecessary paths.” [13]

More recently, Aalst et al. [40], [41] addressed the semantic correctness of the derived individualized model. For example, a semantic error could be a model with a deadlock. As semantic correctness is beyond the scope of our work, it is not covered in this survey.

2.8.1 Deriving Configured Functions

According to C-EPC [13], the four cases where a configurable function may appear in a C-EPC are:

1. Between two events
2. Between a connector and an event
3. Between an event and a connector
4. Between two connectors

Figure 2.11. illustrates the derivation rules for these four cases:

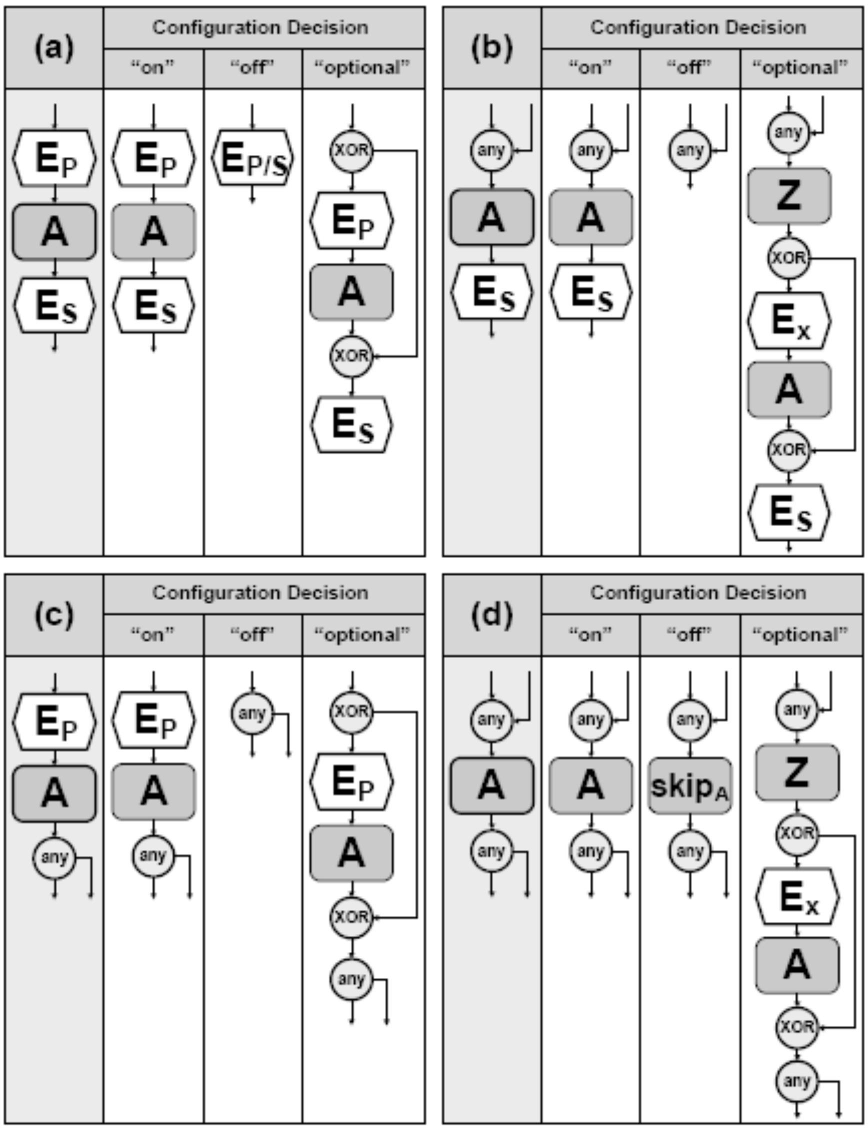


Figure 2.11. Derivation Rules for Configurable Functions [13]

2.8.2 Deriving Configured Connectors

As stated in [13]: "Configuring connectors is quite straight forward. The connector's label is only changed unless it is configured to the sequence "Seq n" type. If a connector is configured to a sequence, the succeeding paths that are not to be included in the build time model have to be eliminated. This means that all subsequent elements are to be excluded from the model until a join connector is reached. If there are no more paths to be eliminated, it must be further checked to determine whether there are join connectors in the model that do not link to any incoming arcs. Paths starting with these joins have to be eliminated, too, and the check must be repeated. This procedure is iterated until there are no more connectors without incoming arcs." Figure 2.12 shows a corresponding example.

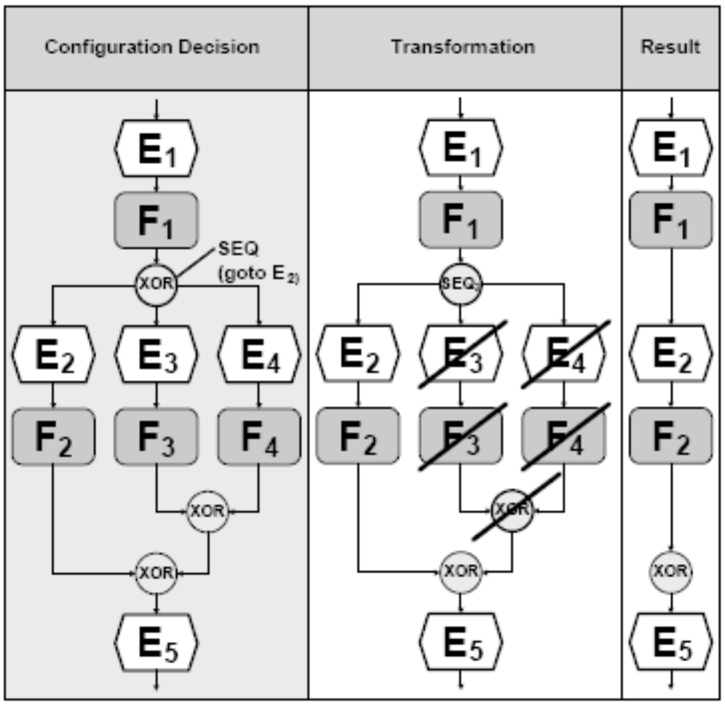


Figure 2.12. A Possible Derivation of a Configured Connector [13]

2.8.3 Deriving the Complete Model

After the derivation of configured functions and connectors the model may end up with some unnecessary paths or connectors that need to be removed. The model is visualized as a graph. Thus, some reduction rules need to be applied to recalculate the final graph. The reduction rules that were derived in [13] are illustrated in Figure 2.13. Figure 2.13 (a) eliminates any empty arc that goes from an AND connector to another join connector. Figure 2.13 (b) eliminates an arc that goes from an AND connector to another join connector if it only contains an event and no functions. Figure 2.13 (c) deletes any connector having one incoming arc and one outgoing arc.

Figure 2.13 (d) shows that if more than one empty arc are found between an OR/XOR connector and a join connector then all the empty arcs are eliminated leaving only one arc. Figure 2.13 (e) merges more than one event if and only if they are the successors of an OR/XOR connector and the predecessors of the same join connector.

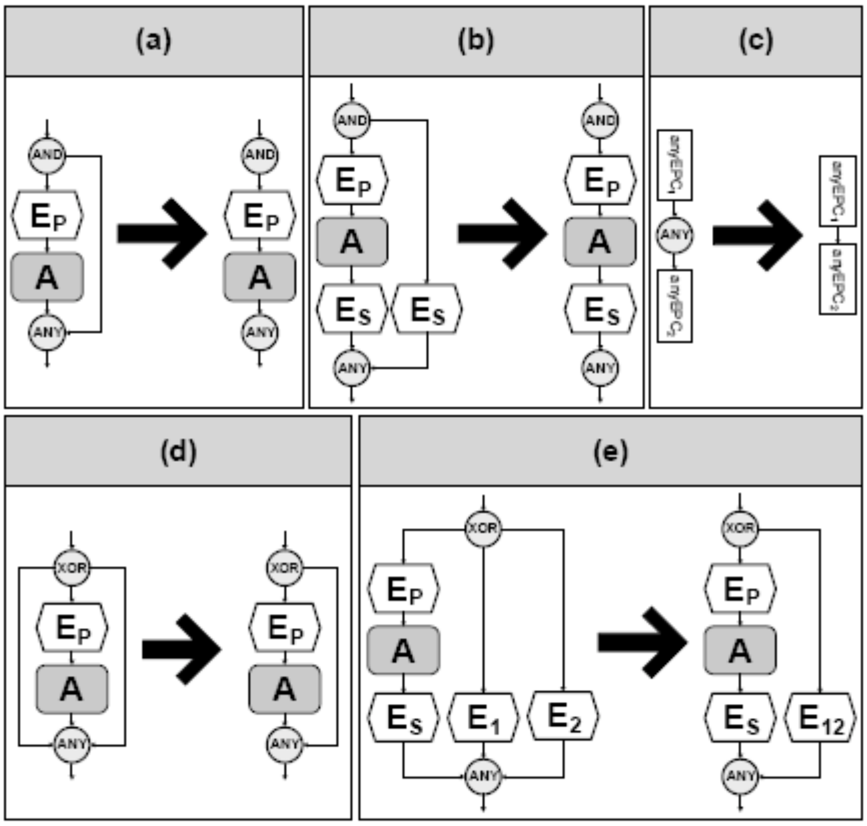


Figure 2.13. The Derived Reduction Rules [13]

2.8.4 Deriving the complete Algorithm

Eventually, all the above cases and their corresponding decisions were summarized and put together in the following algorithm [13]:

1. Change the connector type of configured connectors to their configuration value.

2. If the configuration value is "Seq n" eliminate paths (including all nodes), i < > n, with n being specified in the go to attribute, until a join connector or an end node is reached.

3. Check whether there is a connector c without any incoming arcs. If yes, go to 4. If no, go to 5.

4. Eliminate all paths starting with connector c until a join connector or an end node is reached. Go to 3.

5. Check whether one of the reduction rules (as in section 2.8.3 above) is applicable. If yes, go to 6, if no, go to 7.

6. Apply one of the reduction rules and go to 5.

7. Configure functions according to the rules/cases mentioned above (in section 2.8.1).

8. Check again whether one of the reduction rules is applicable. If yes, go to 9. If no, end.

9. Apply one reduction rule and go to 7.

The above algorithm was proven to be syntactically correct and its termination was also proven in [13]. Semantic correctness however, was beyond the scope of the research work in [13] and [14].

2.9 Discussion of Related Work

The more recent work on configurable process modeling was implemented on top of C-EPC to aid in the transformation process, taking into consideration the “requirement” attributes and constraints of the model. Introducing such constraints together with the concept of preserving the semantic correctness of the derived model, induced from the transformation process, increased the complexity of the transformation process considerably.

Only a few researches addressed the constraints issue. One of the approaches discussed in [40] and [41] used prepositional logic to express the constraints of the model. Other researches (such as [21], [23] and [24]) discussed the use of a questionnaire-based interface to configure and individualize a configurable business process reference model while preserving the semantic and syntactic correctness of the model. Ensuring the semantic correctness of the transformed model was only recently considered by large-scale research efforts such as [20].

Since all the previous work was applied on the EPC modeling language only, it was strongly recommended (as in [28] for example) to extend such approach to other modeling languages such as UML AD, BPMN and Petri nets.

The concept of expressing variations points and their variants using the UML modeling language was discussed in [19], [34], [36], [42] but the work applied in all of these papers was aiming to support and assist in the building of Software Production Lines (SPLs). None of these researches aimed at building a complete configurable modeling language that could be dedicated for building configurable business process models.

Some other approaches were proposed (such as [8], [17] and [27]) to extend the UML AD for modeling business models. These approaches, however, used the annotation technique to identify the variation points within a model, which complicates the model and makes it less understandable, especially to the business end users who are going to use the configurable reference models. Usually the business users of such models are not IT experts. Besides they are not really aiming at building a completely configurable modeling language that covers all the requirements discussed in section 2.6.3.

Finally, these techniques did not provide a full algorithm for transforming and individualizing the configurable reference model while ensuring the syntactic correctness of the model [20].

A summary and comparison of the recent approaches applied in the area of configurable modeling languages is shown in Table 2.3. Based on the authors' reports, the table indicates the language that each approach used, the variation mechanism, the presence/absence (+/-) of each of syntactic correctness, semantic correctness, transformation algorithms and in the last column our judgment of the approach's effect on complexity.

Table 2.3. Comparison of the Recent Approaches in Configurable Business Process Modeling

The Author(s) of the Approach	Target Language	Variation Mechanism	Syntactic correctness	Semantic correctness	Transformation algorithm	Complexity
Aalst, Dumas, Gottschalk, ter Hofstede, La Rosa and Mendling [20]	EPC	Configurable nodes	+	+	+	Not increased
Puhlmann,Schnieders, Weiland and Weske [8]	BPMN, UML AD	Annotation	-	-	-	Increased
Razavian and Khosravi [27]	UML AD	Annotation	-	-	-	Increased
Czarnecki and Antkiewicz [17]	UML AD	Annotation	-	-	+	Increased

The approach that we introduce in this research aims at filling the missing gaps of the previous approaches that were used to model configurable business process models using UML AD. Therefore, we extended the standard UML AD using configurable notations and attributes that focus on fulfilling the requirements of a configurable modeling language as discussed in section 2.6.2 and 2.6.3. We called this extended language C-UML AD. We also developed a complete algorithm that transforms the C-UML AD back to individual specific UML AD that is both consistent and syntactically correct.

3. The Configurable UML Activity Diagram

In this chapter we explain, in details, all the extensions that we applied to the UML Activity Diagram (UML AD) modeling language to make it a configurable one. We called this extended language Configurable-UML Activity Diagram (C-UML AD) modeling language. Following a similar approach to that applied in C-EPC, the C-UML AD is developed in such a way so as to allow the user to express the variation points of a business model and to give him/her more guidance and help in the process of reusing and configuring the business model.

The first part of this chapter explains the configurable elements of the C-UML AD. The second part shows how the set of requirements that need to be covered by any configurable modeling language, discussed in section 2.6.2, were fulfilled by the C-UML AD.

3.1 The Configurable Elements

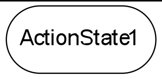
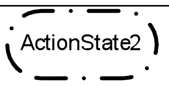
C-UML AD is a configurable language and hence some configurable nodes had to be introduced to it. A configurable node is a point in the business model that represents a variation point. At any configurable node a decision has to be taken by the business model end user. We referred to such decisions in section 2.6 as build time decisions.
In C-UML AD a configurable node could be either a configurable action or a configurable connector.

3.1.1 Configurable Actions

The first step in extending the UML AD is to set the new notations for the extended elements of the language. This section covers the standard action. The configurable actions are represented in a similar way as the standard actions but configurable actions use dashed borders to distinguish them from standard actions.

Table 3.1. shows the notations of a standard and a configurable action.

Table 3.1. Notations of a Standard and a Configurable Action in C-UML AD

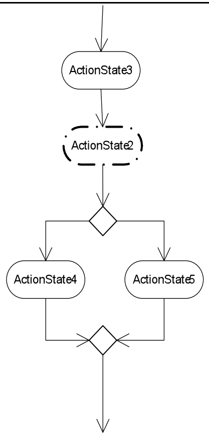
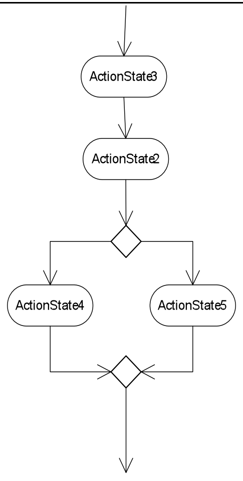
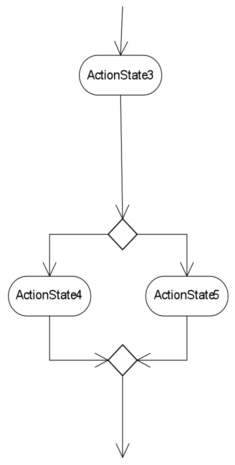
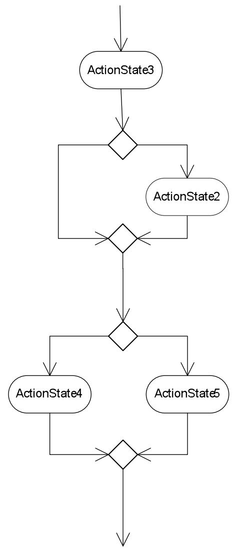
Standard Action	**Configurable Action**
	

Possible Configurations:

Our main objective here is to configure the presence of the action rather then its functionality. The configurable actions could be configured to ON, OFF or OPTIONAL. The ON option indicates that the user of the business model always needs this action. OFF means that this action is never needed, while OPTIONAL indicates that the decision of choosing this action will be made during run time and therefore, this action in the business model will be given two alternative paths: one with the action and one without it. Since a decision node will be introduced to the model in the OPTIONAL case then a set of appropriate conditions should be associated with the decision node to identify when each path will be taken.

Table 3.2. illustrates the possible configurations of an action.

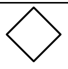
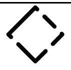
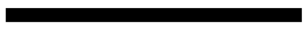
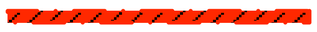
Table 3.2. Possible Configurations of a Configurable Action

C-UML AD business model	ON	OFF	OPTIONAL
			

3.1.2 Configurable Connectors

This section shows the configurable connectors that were added to the C-UML AD. In C-UML AD we extended the basic connectors, i.e. control nodes, which include the Decision/Merge and Fork/Join. Unlike configurable actions, the functionality of the configurable connectors as well as their presence could be configured. A configurable connector could be converted to another connector type or to one specific path out of the available paths. Table 3.3 shows the notations of the configurable connectors against the standard ones:

Table 3.3. Notations of the Standard and Configurable Connectors in C-UML AD

Connector Type	Standard Notation	Configurable Notation
Decision/Merge		
Fork/Join		

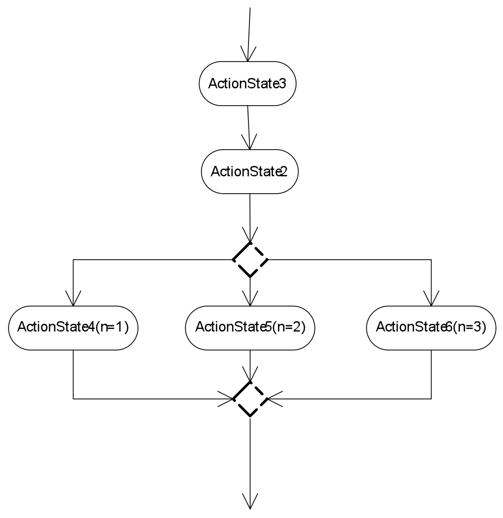
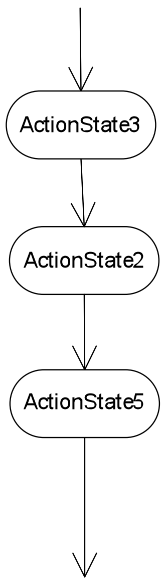
Possible Configurations:

The researches that developed the configurable EPC allowed the configuration of the configurable connectors to other connectors that only restrict their behavior. For example an "AND" connector cannot be transformed to any other connectors but an "Or" connector could be transformed to any other connector. In our research the configuration of connectors is wider, aiming to make the configuration more useful. Therefore, a configurable connector could be transformed from one connector type to another or could be configured to a specific path (called a sequence).

A sequence is a new terminology that we introduce in our C-UML AD language. A sequence "Seq n" represents a specific path where "n" is one of the available paths. Table 3.4 shows a sequence "Seq 2" where n=2.

Table 3.4. Configuring a Configurable Decision to a Seq n

C-UML AD business model	**Configured UML AD with configuration decision: "seqn" and n=2**

 ActionState3
ActionState2
ActionState4(n=1)
ActionState5(n=2)
ActionState6(n=3)
	

We summarize all the possible configurations that we introduced so far in Table 3.5. below.

Table 3.5. Summary of Possible Connector Configurations

	Sequence n	**Decision/Merge**	**Fork/Join**
Configurable Decision/Merge	Yes	Yes	Yes
Configurable Fork/Join	Yes	Yes	Yes

To apply the above configurations to a configurable connector in a C-UML AD business model, a couple of prerequisite decisions have to be made:

1. When configuring a configurable Fork/Join to a Decision/Merge a set of conditions have to be imposed and added to the new business model.

2. When configuring a configurable Decision/Merge to Fork/Join all the conditions that were available on the alternative paths of the decision have to be removed.

3.2 Configuration Requirements and Attributes

Any configurable modeling language has to meet a set of configuration requirements and characteristics as mentioned earlier in section 2.6.2. Although in section 2.6.3 many guidelines were given to apply such requirements, yet most of them were not actually implemented in C-EPC. In this work we managed to satisfy most of these requirements. The following list includes the set of requirements that are covered by the proposed C-UML AD modeling language:

1. Configuration coverage requirement: The configurable language should support configurations regarding entire processes, functions, control flow and data. However, C-UML AD mainly focuses on the process and control-flow aspects, noting that the presence of any action could be configured and not its functionality. The data aspect and function aspect have not been addressed explicitly. We emphasize that C-EPC [] covers this requirement in the same manner.

2. Mandatory/optional requirement: The configurable language should distinguish between mandatory and optional decisions. We fulfilled this requirement by adding an attribute with each configurable node that identifies such input. C-EPC does not provide an explicit implementation for such configuration attribute.

If the user of the model chooses an "optional" value for an attribute then a default value must be given as well. This default value will be used when the end user chooses not to give an explicit configuration value for the associated configurable node. Therefore, the default values must be associated with "optional" decisions. If "mandatory" is used for the associated attribute then an explicit decision must be taken from the end user.

For example, assume that a given business model has one of the attributes of its configurable actions set to optional. Then a default value has to be set for this configurable action, for example let us assume the default value is OFF. Then, this action will be turned OFF if the end user does not give any other explicit configuration value at build time.

3. Global/Local requirement: The configurable language should differentiate between global and local decisions. Again we introduce another attribute with each

configurable node to indicate such a decision. This attribute was also not explicitly covered by C-EPC but it was recommended and discussed.

4. Critical/non-critical requirement: The configurable language should also differentiate between critical and non-critical decisions. We implement this requirement by introducing another attribute associated with each configurable node to hold such a value. Again this attribute was not explicitly implemented by C-EPC.

5. Interrelationship requirement: The configurable language should be able to demonstrate interrelationships. This requirement is mainly covered by introducing the requirements and guidelines attributes associated with each configurable node to our C-UML AD. These attributes were implemented by C-EPC and we adapted them from C-EPC. However, these attributes in both C-EPC and C-UML AD are restricted to interrelationships within one business model.

6. Specification Level requirement: The configurable language should allow configuration decisions to be made at different organizational levels. This can be supported by the concept of partially configurable C-UML AD. The concept of partially configured C-UML AD is similar to that of partially configured C-EPC []. A partially configured C-UML AD is simply a C-UML AD with some nodes that have already been configured and others that have not. A partially configured C-UML AD allows the configuration to take place at subsidiary levels. To identify at which level a configurable node must be configured we add another attribute, called specification level attribute, to each configurable node stating the specific level at which the configuration should take place. The specification level attribute is not explicitly implemented in C-EPC.

7. Extended related information requirement: The variation points of a configurable language should refer to further related information within the enterprise system. We did not implement this requirement but to do so, C-UML AD may later on include a system online help or system implementation guide as proposed in [28].

8. Recommendation Requirement The configurable language should be guided by recommendations or configuration guidelines. This requirement is satisfied through the guideline attribute associated with each configurable node, if required. This attribute is provided by C-EPC as well.

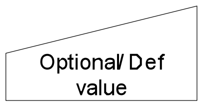
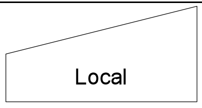
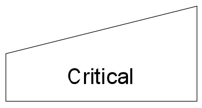
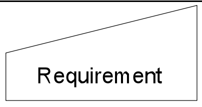
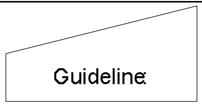
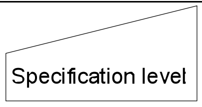
9. Complexity requirement: The last requirement of a configurable language refers to the impact of configuration extensions on the perceived model complexity. The only addition that we applied on the model for this requirement is the use of logical expressions when describing the requirements and guidelines attributes. However, the C-UML AD is a natural extension to the standard UML AD modeling language and the addition of such logical expressions should not cause a problem to the typical user of the language.

Based on Aalst et al. [28] configurable language requirements the set of configuration attributes discussed above had to be introduced to the C-UML AD modeling language. Any configuration attribute has to be associated with a configurable node in any given C-UML AD business model. The notations of the configuration attributes are shown in Table 3.6. In Table 3.7 we compare between the configuration attributes available in C-UML AD and those in C-EPC.

3.3 Implementation of the Attributes

We developed a C-UML AD Transformation Tool to help us implement the new notation. Our C-UML AD Transformation Tool includes the standard UML AD constructs as well as the new configurable actions, configurable connectors and configuration attributes introduced in Table 3.6.

Table 3.6. Notations of the Configuration Attributes in C-UML AD

Introduced Attribute	**Corresponding Requirement**	**Introduced Notation**
Mandatory/Optional *Default=Mandatory*	Mandatory/Optional	If optional: 
Local/Global *Default=Global*	Local/Global	
Critical/Non critical *Default= Non critical*	Critical/Non critical *Default= Non critical*	If critical (hardly reversible): 
* Requirement	Interrelationship	
*Guideline	Recommendation	
*Specification Level	Specification Level	

*If attribute is not provided then no checks or interrelationships are applied.

Table 3.7. Comparison Between C-UML AD and C-EPC

Attributes	**C-UML AD**	**C-EPC**
Mandatory/Optional	Yes	No
Local/Global	Yes	No
Critical/Non critical	Yes	No
Requirement	Yes	Yes
Guideline	Yes	Yes
Specification Level	Yes	No

The Local/global, critical/non-critical, guideline and specification level attributes were all implemented in a similar manner. These attributes can be added to the model to display some specific features. For example, if the configuration of a configurable node in the model has a critical effect then this configurable node will have the critical notation drawn associated to this node in the model.

The requirement attribute holds logical expressions that are used to express any interrelationships between the associated configurable node and other configurable nodes in the business model. The requirement attribute is handled in a different manner than all the other attributes because the logical expression that is stated in this attribute must be enforced on the model and therefore this logical expression must be acquired by the C-UML AD Transformation Tool. The logical expressions stated in the requirement attribute are not left to the user to enter them freely, since these expressions have to be specific and correct. However, we applied another technique to acquire these logical expressions. We captured these logical expressions through our own implemented graphical interface with the user. Hence, we are always in control of the requirements that the user enters. The only limitation with this technique is that each configurable node is limited to one type of requirements.

The mandatory/optional, local/global and critical/non-critical attributes have all been given default values. These default values are used since not every configurable node in a business model must have explicit values for all the attributes. If one of these attributes is not explicitly associated with a configurable node then the default values mentioned in Table 3.6 will be applied.

The requirement, guideline and specification level attributes are associated to configurable nodes only if interrelationships, guidelines or organizational levels need to be identified respectively, otherwise, these attributes are not used and no default values are needed.

3.4Invoice Verification Business Example

In Figure 3.1. we show the same example used in section 2.7 for Invoice Verification but we use C-UML AD instead of C-EPC:

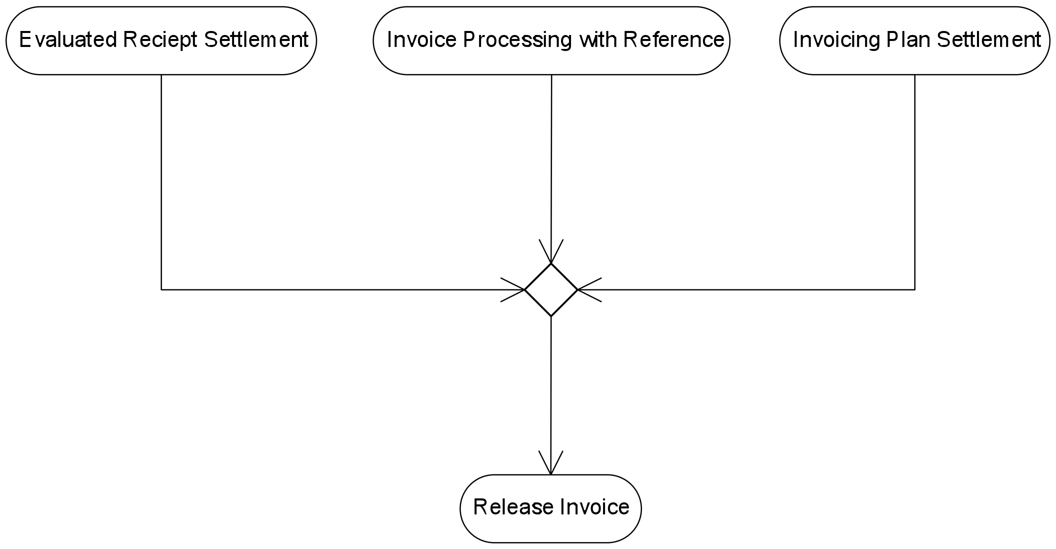

Evaluated Reciept Settlement
Invoice Processing with Reference
Invoicing Plan Settlement
Release Invoice

Figure 3.1. Invoice Verification Example Using Standard UML AD

The Configurable version of the above business model, Figure 3.1, is shown in Figure 3.2 using C-UML AD. The example shows three configurable nodes: two configurable actions, one configurable connector, and three configuration attributes associated with two configurable nodes: guideline, requirement and optional/mandatory attributes. The requirements attribute indicates that if Invoicing Plan Settlement (IPS) is left ON then Evaluated Receipt Settlement (ERS) *must* be included in the model as well, i.e. turned ON. The guideline attribute gives some guidance and help as soft constraints, i.e. suggestions. The third attribute is the optional/ mandatory attribute, which indicates that the configuration of the IPS is optional not mandatory, and if not configured explicitly then its default configuration will be ON, and as a result of turning IPS to ON, based on the requirement associated with it, ERS must be turned ON as well. The attributes are always associated with the configurable nodes using dashed connections.

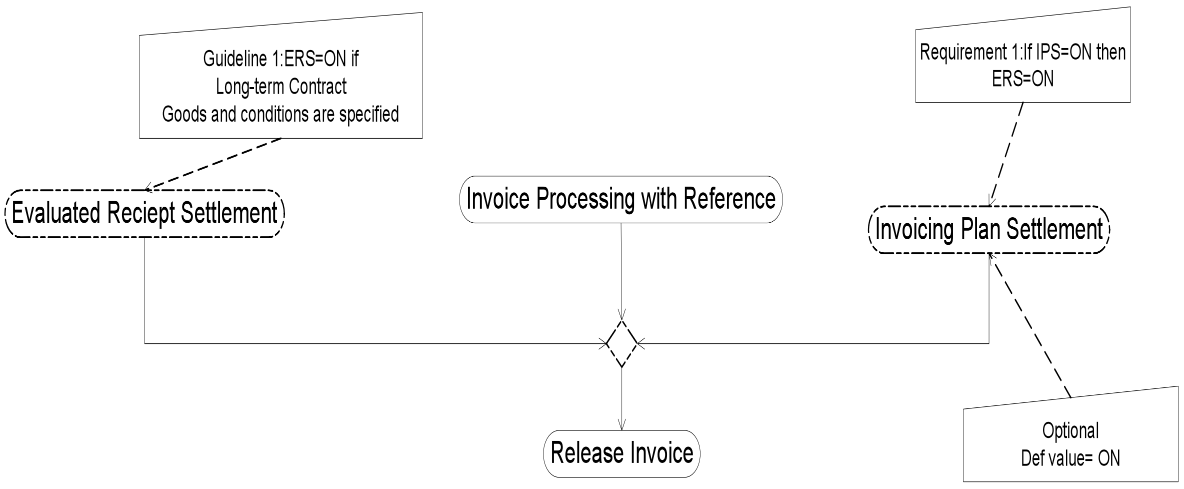

Guideline 1:ERS=ON if
Long-term Contract
Goods and conditions are specified
Requirement 1:If IPS=ON then
ERS=ON
Evaluated Reciept Settlement
Invoice Processing with Reference
Invoicing Plan Settlement
Release Invoice
Optional
Def value= ON

Figure 3.2. Invoice Verification Example Using C-UML AD

Figure 3.3 shows a possible configuration for the above model, where IPS= off and ERS=off and thus the configurable connector = seq1 (since it will be removed and one sequence will be followed always).

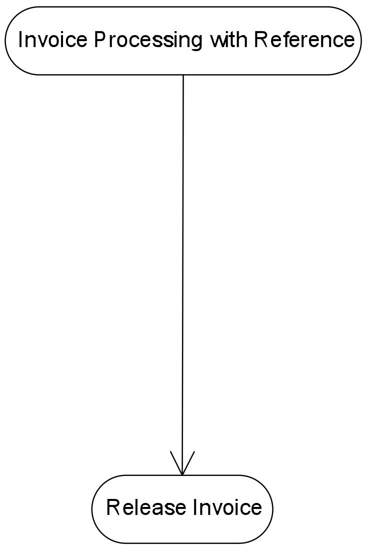


Figure 3.3. A Possible Configuration of the Invoice Verification Example

Another alternative configuration decision could be reached by keeping IPS= ON and since there is a requirement controlling the configuration decision on ERS then ERS *must be* = ON and as a result the connector will be kept as is. The result of such decisions will keep the business model as is without any changes i.e. as shown in Figure 3.1.

4. The Transformation Process

After establishing the C-UML AD modeling language notation (in Chapter 3), our next step is to ensure that the transformation process from the configured C-UML AD business model back to the standard UML AD at build time is syntactically correct. This chapter focuses on the algorithm that should be applied during the transformation process, sometimes called the configuration process. This transformation process explains how the end users can reuse (configure) the generic business models that are designed using the C-UML AD modeling language to suit their specific requirements, thus going from generic C-UML AD business models to specific UML AD business models.

We now show how the end user of the business model can configure the C-UML AD business model back to a syntactically correct UML AD. At run time all the business models have to be syntactically correct UML AD models and not C-UML AD. Therefore, all the configurable nodes and their associated attributes that were introduced in the C-UML AD modeling language, have to be exchanged with standard UML AD constructs based on the end user's decision.

It is only natural to expect that the configuration of C-UML AD models results in models that may have syntactic errors, such as unnecessary paths or nodes. We followed the approach suggested by [13] to develop a full algorithm that monitors the whole transformation process which would lead to a syntactically correct UML AD model. We note that in our algorithm we considered the configuration attributes associated with the configurable nodes, which is a significant difference from the approach followed by the C-EPC in section 2.8 [13].

4.1 Transformation of Configurable Actions

As mentioned in chapter 3, the three alternatives for a configurable action are ON, OFF or OPTIONAL. In each case a different transformation procedure takes place based on the position of the configurable action in the business model.

A configurable action may appear in four different positions within the C-UML AD business model, which are as follows:

Case 1: A configurable action can lie between two actions.

Case 2: A configurable action can lie between two connectors.

Case 3: A configurable action can lie between an action and a connector.

Case 4: A configurable action can lie between a connector and an action.

We analyzed each of the above cases to reach the correct transformation shown in Tables 4.1, 4.2, 4.3 and 4.4. respectively.

Table 4.1 shows a configurable action, indicated by the dashed borders, that lies between two actions. In case the end user's decision is ON then the configurable action is changed to a standard one, but if the end user's decision is OFF then the configurable action is removed. In case the end user's decision is OPTIONAL then two paths are given to the end user; one with the action and one without it. Hence, the end user's decision in this case will be left for run time.

Table 4.1. Configuration Alternatives of a Configurable Action Between Two Actions (Case 1)

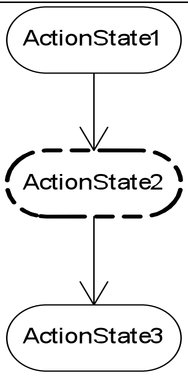
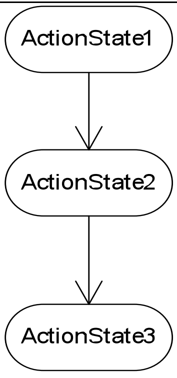
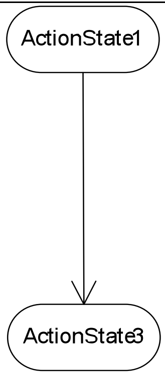
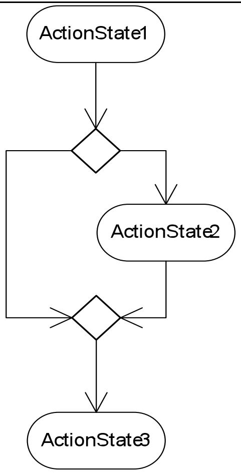
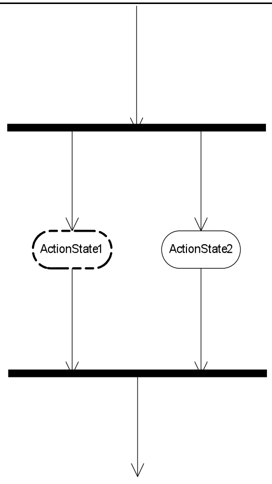
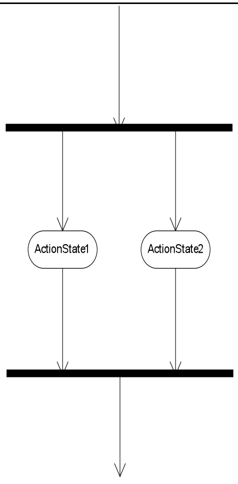
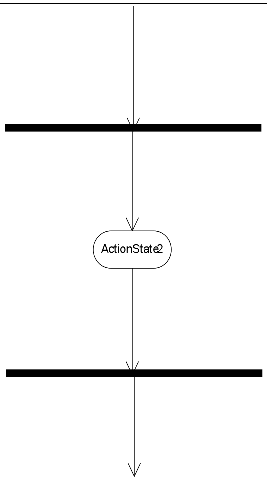
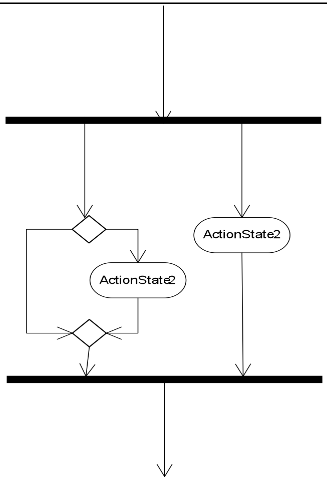
C- UML AD business model	ON	OFF	OPTIONAL
			

Table 4.2 shows an example of the second case, where a configurable action lies between two connectors. The same approach discussed in Table 1 applies here: if the end user's decision is ON then the action is kept and if the end user's decision is OFF then the action is removed. When the action is removed an empty path will appear between the fork and the join which is useless and has to be removed, this case is discussed in the next section, section 4.3, but if the end user's decision is OPTIONAL then two paths are given; one with the action and one without it.

Table 4.2. Configuration Alternatives of a Configurable Action Between Two Connectors(Case2)

C-UML AD business model	ON	OFF	OPTIONAL
			

The third case follows the same approach as the first two cases. However, when an end user's decision is OPTIONAL a model might end up with two consecutive connectors. This does not violate the syntactic correctness of the model. This is shown in Table 4.3.

Table 4.3. Configuration Alternatives of a Configurable Action Between an Action and a Connector (case 3)

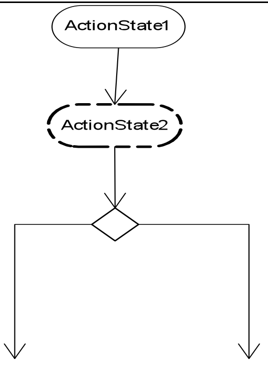
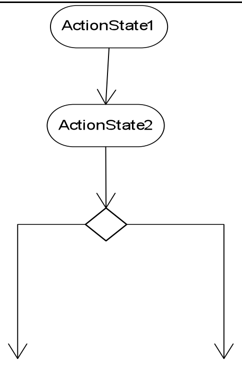
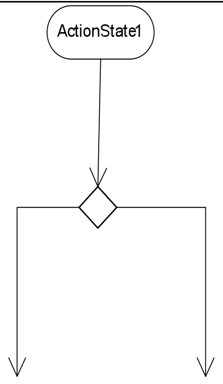
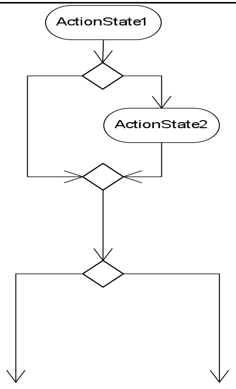
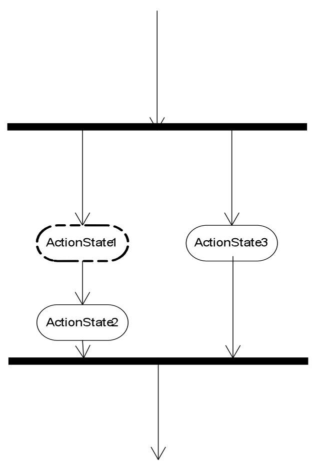
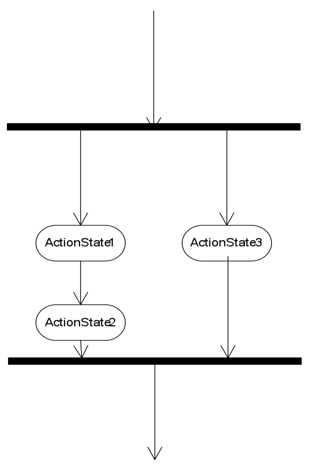
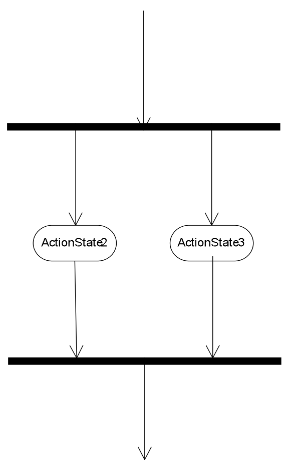
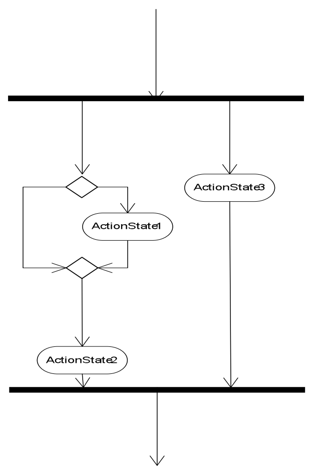
C-UML AD business model	ON	OFF	OPTIONAL
			

Table 4.4 shows the alternatives for a configurable action that lies between a connector and an action.

Table 4.4. Configuration Alternatives of a Configurable Action Between a Connector and an Action (case 4)

C-UML AD business model	ON	OFF	OPTIONAL
			

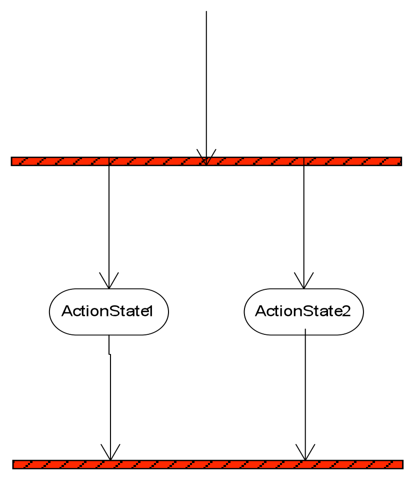
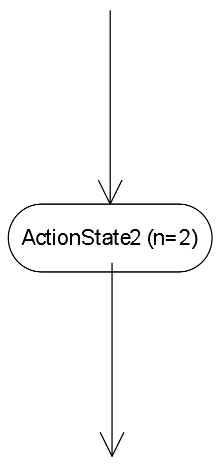
4.2 Transformation of Configurable Connectors

A configurable connector could be configured to stay as is or it could be configured to the other connector or to one of the available sequences (paths). Table 4.5 shows an example where a fork/join connector is configured to a sequence, Seq n. Table 4.6 shows an example where the fork/join connector is configured to a Decision/merge.

In the example shown in Table 4.5, a configurable Fork/Join is transformed to a seq n where n=2. Since the end user's decision was n=2 then all other paths between this configurable fork and its corresponding join are removed leaving only path 2.

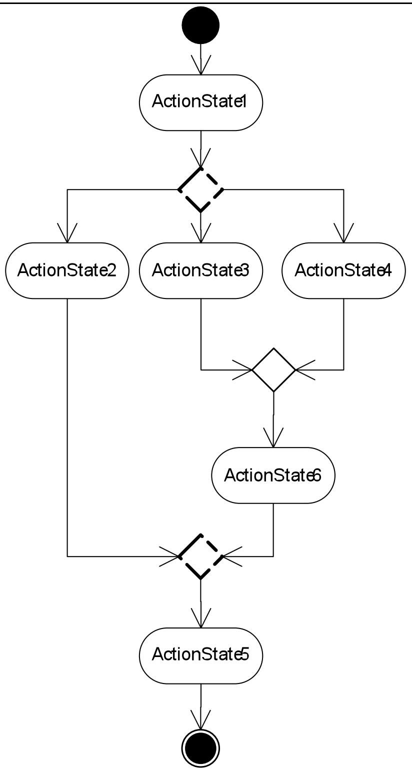
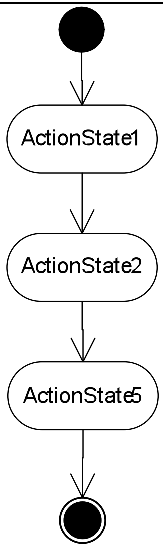
All the actions and connectors on all the other paths have to be removed until a connector with an incoming arc is reached. At this point no more actions or connectors should be removed. This case is shown in the example illustrated in Table 4.6.

Table 4.5. Example of Configuring a Configurable Fork/Join

Fork/Join	**Sequence n=2**
	

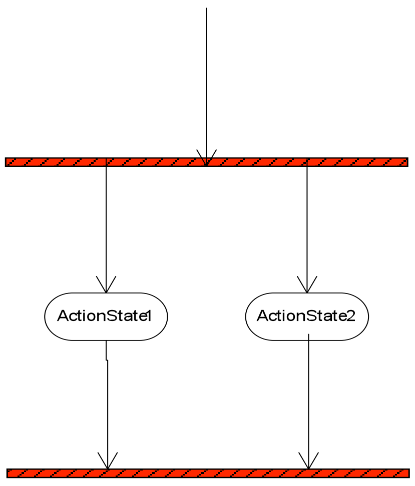
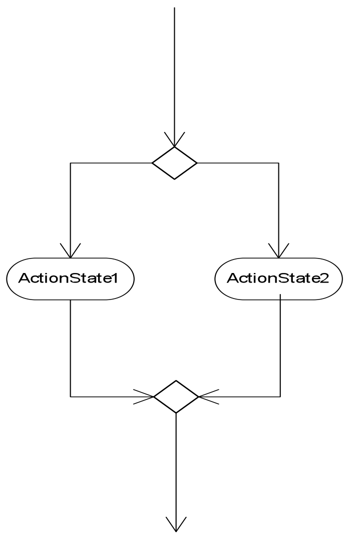
In Table 4.6 the configurable decision is configured to seq n, n=1. Therefore, Action State 3 was removed and the second decision node was not removed because there was still an incoming arc from Action State 4. Then Action state 4 was removed, and since the second decision node now has no other incoming arcs it was also removed. After that, Action State 6 was removed. When the last decision node was reached it was not removed at this point because it had an incoming arc from Action State 2. This last decision node was removed from the model in the next step based on the reduction rules discussed in the next section, section 4.3.

Table 4.6. Example of Configuring a Configurable Decision/ Merge

Decision/merge	Sequence n=1
	

In Table 4.7. the end user chooses to transform the configurable fork/ join connector to a decision/merge connector.

Table 4.7. Example of Configuring a Fork/Join

Fork/Join	Decision/Merge
	

4.3 Optimizing the Transformed Model

In some cases the transformation of the configurable nodes may lead to some unnecessary nodes and/or paths, which in turn, may lead to an incorrect UML AD business model. To overcome this problem we analyzed all the conditions that may cause such problems and followed a similar approach to that used in section 2.8.3 [13] by C-EPC. The configured business models were realized as a graph as stated in [13]. Therefore, all unnecessary paths have to be removed to meet the minimality criteria of a correct model [13].

Our analysis resulted in that the configuration of C-UML AD may lead to three cases where unnecessary paths/nodes are likely to appear. The three cases are as follows:

1. Having more than one empty path between a decision and a merge connector.
2. Having an empty path(s) between a fork and a join connector.
3. Having one incoming arc and one outgoing arc out of the same connector.

To remove the unnecessary paths/nodes the following three corresponding reduction rules were applied:

1. Remove all the empty paths between a decision node and a merge node, leaving only one empty path.
2. Remove all empty path(s) between a fork node and a join node.
3. Remove the connector that has one outgoing and one incoming arc.

In Table 4.8 (column 2) Action State 2 and Action State 4 are configurable actions. If the end user's configuration decision is ON for any of the two actions then this action will be changed to a standard action and the resulting model will be a correct model. If the end user's configuration decision for any of the two actions is OPTIONAL then this action will be exchanged with two paths; one with this action and one without it, and the model will still be correct. A problem will only appear if the end user's decision is OFF for both actions. The OFF value will cause the removal of both actions and in return the model will end up with two empty paths between the decision and its corresponding merge. Table 4.8 (column 3) shows this case where the first reduction rule was applied, i.e. all empty paths were removed leaving only one empty path between the decision and its corresponding merge.

Table 4.8. Example of Applying the 1st Reduction Rule

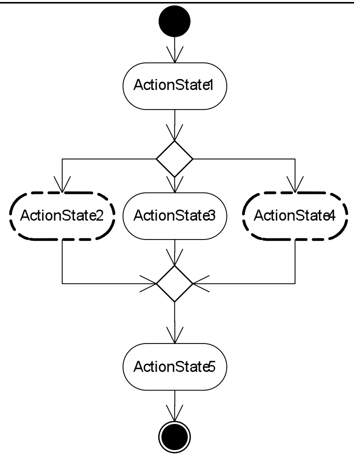
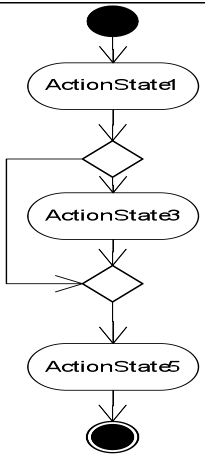
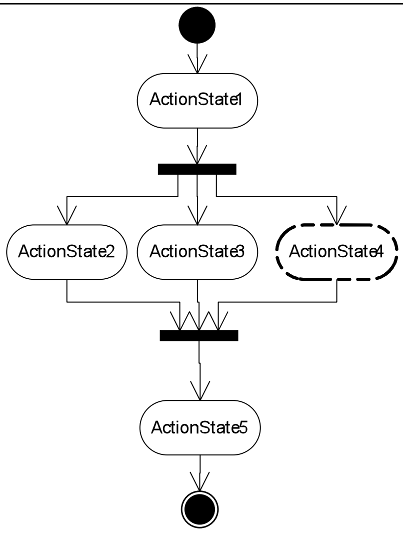
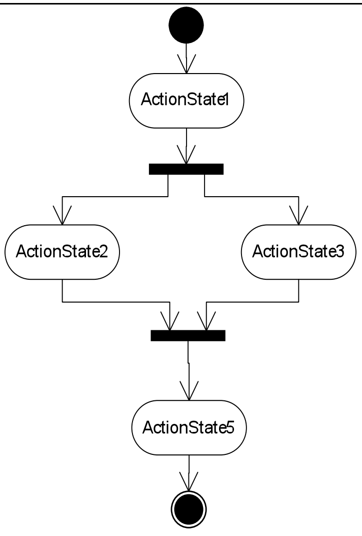
	C-UML AD model	**UML AD model after applying first reduction rule**
1st Reduction Rule	 ActionState1 ActionState2 ActionState3 ActionState4 ActionState5	

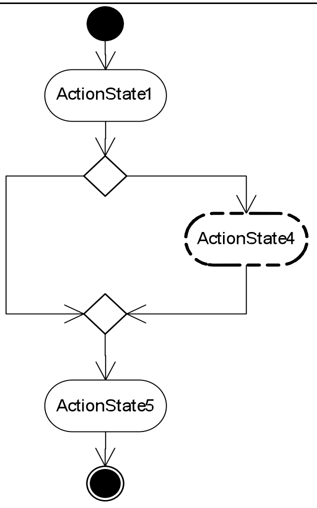
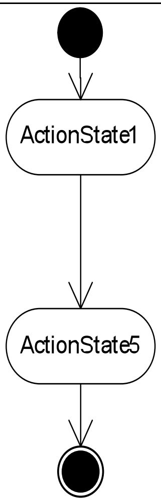
Table 4.9 shows an example where the second reduction rule was applied. The example demonstrates a scenario where the end user's configuration value for action state 4 is OFF, thus leading to an empty path between a fork and its corresponding join. In this case the empty path is removed.

Table 4.9. Example of Applying the 2nd Reduction Rule

	C-UML AD model	**UML AD model after applying 2nd reduction rule**
2nd reduction rule	 ActionState1 ActionState2 ActionState3 ActionState4 ActionState5	 ActionState1 ActionState2 ActionState3 ActionState5

The third reduction rule applies when you have any connector, whether a fork/join or decision/merge, with one incoming arc and one outgoing arc. The connector in this case is useless and therefore it should be removed. Usually, this case occurs when a configurable action is turned OFF and then its path is removed due to one of the other two previous reduction rules, ending up with one incoming path and also one outgoing path of the same connector. Table 4.10 shows an example where the end user's decision was to turn action state 4 OFF. Based on this decision Action state 4 was removed leading to two empty paths between a decision and its corresponding merge. Thus, the first reduction rule was applied leading to only one empty path between the decision and its corresponding merge. Finally, the third reduction rule was applied because of having one empty path going into the decision and one outgoing path from it. The third reduction rule removed this decision/merge connector leading to a simple path between action state 1 and action state 5.

Table 4.10. Example of Applying the 3rd Reduction Rule

	C-UML AD model	**UML AD model after applying 3rd reduction rule**
3 rd reduction rule		

4.4 The Transformation Algorithm

The transformation of the C-UML AD to UML AD based on the end user decisions has to be monitored to make sure that the configured derived model is syntactically correct. To achieve this goal, the rules applied in section 4.1, 4.2 and 4.3 above had to be integrated and applied together with the associated configuration attributes of each configurable node.

Not all the configuration attributes have a great influence on the transformation process. However, the “requirement” attribute and the “mandatory/optional” attribute will have the major influence, since they both impose some conditions on the transformation of the model. For example, applying the transformation algorithm on a configurable model having a configurable node with a mandatory value must force the end user to provide an explicit configuration value for such a node. Another configurable connector could have a requirement attribute. When configuring such a connector, its associated requirement has to be applied on the model automatically and the effect of a requirement and the explicit configuration value of the user should be compatible otherwise a contradiction/inconsistency problem will appear and the end user is alerted and must resolve the issue. All the other attributes listed in section 3.2 are just used to enhance and help the end user reuse and configure the C-UML AD business models by providing essential information to the end user, such as the guideline attribute. For example, the guideline attribute is used to display to the end user a recommended condition only. Hence, this attribute does not influence the transformation process directly and thus is not considered in the transformation algorithm.

The algorithm starts out by stopping at each configurable node and checking whether the end user would provide a configuration decision value or not. If the associated mandatory/optional attribute is set to mandatory then the end user is forced to enter a configuration value. In case the end user does not enter a value then the default value associated with the optional attribute will be used.

In any case, the configuration value is checked against any violation to a requirement that was applied in an earlier step. If a violation occurs then the inconsistency has to be resolved first. Then the associated new requirement is also checked to see if it is going to contradict any configuration value entered by the end user in an earlier step. If no inconsistency problems occur then the next step is to start applying the appropriate transformation.

If the configurable node is a configurable action then the appropriate rule (from section 4.1) is applied. If the configurable node is a connector then the appropriate rule (from section 4.2) is applied. After that, the model is checked to see whether a reduction rule needs to be applied or not, if a reduction rule is needed then it is applied until no more reduction rules are needed.

Finally, if a requirement attribute is associated with the node then a specific constraint needs to be applied on the model. The complete algorithm is expressed in the activity diagram in Figure 4.1.

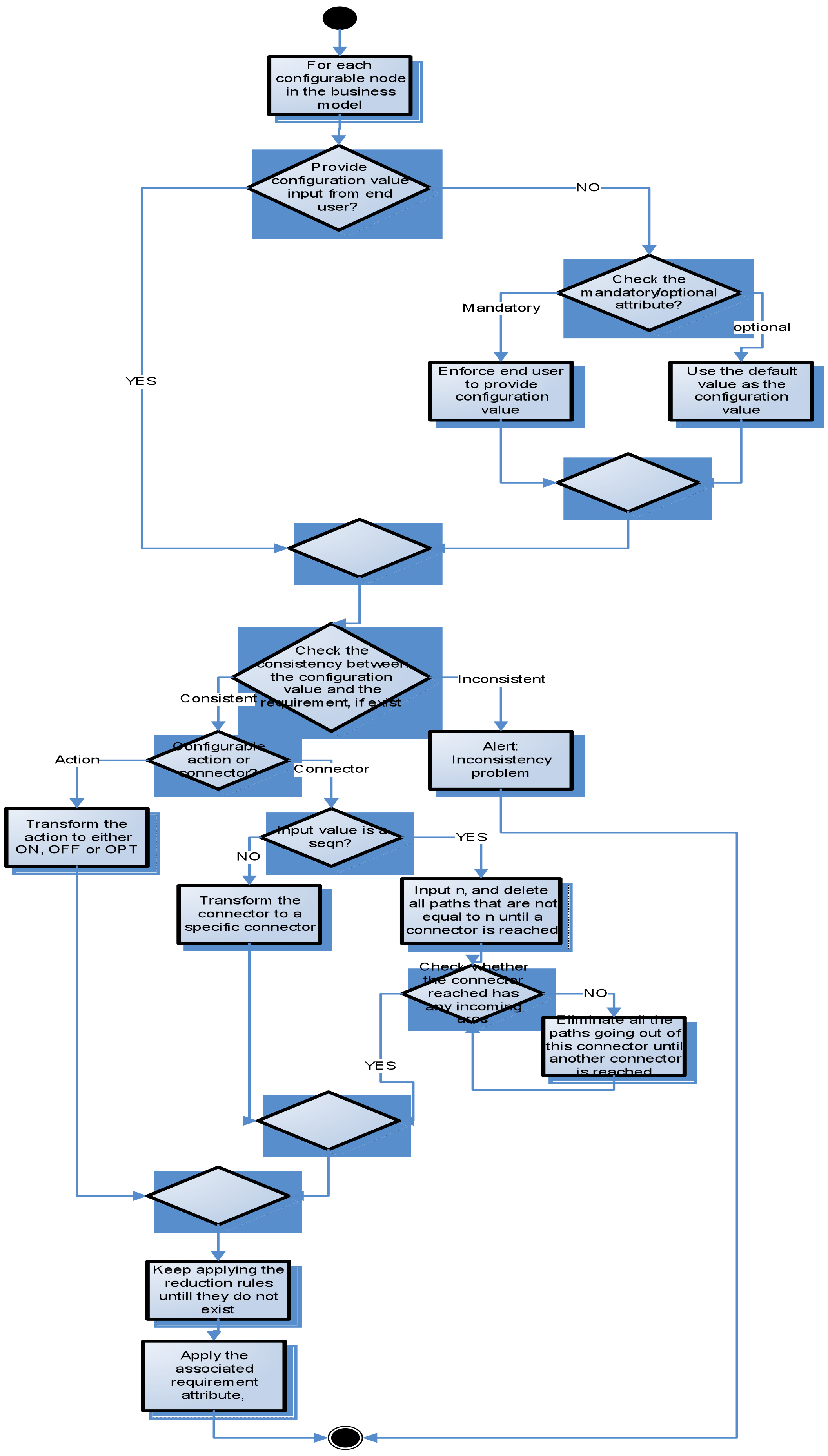

For each configurable node in the business model
Provide configuration value input from end user?
NO
YES
Check the mandatory/optional attribute?
Mandatory
optional
Enforce end user to provide configuration value
Use the default value as the configuration value
Check the consistency between the configuration value and the requirement, if exist
Consistent
Inconsistent
Alert: Inconsistency problem
Configurable action or connector?
Action
Connector
Transform the action to either ON, OFF or OPT
Input value is a seqn?
NO
YES
Transform the connector to a specific connector
Input n, and delete all paths that are not equal to n until a connector is reached
Check whether the connector reached has any incoming arcs
NO
YES
Eliminate all the paths going out of this connector until another connector is reached
Keep applying the reduction rules untill they do not exist
Apply the associated requirement attribute,

Figure 4.1.The Transformation Algorithm

4.5 Example

In this example we demonstrate how the full Transformation Algorithm is applied on a full business model. We chose an example with some nested connectors to illustrate how the algorithm works successfully on more complex C-UML AD models.

Figure 4.2 shows the configurable reference model with C-UML AD. Figure 4.3 shows the configured reference model after applying the end user's configuration values. The configuration value for the first decision node was seq n, n=2, and the configuration values for the second and third decision nodes were fork/join.

When the first decision node was configured to seq n, n=2 all other paths were removed, as mentioned in section 4.2, leaving the last decision node with one incoming arc and one outgoing arc. This decision node was removed when the reduction rules were applied. Then the second and third decision nodes were converted to fork/join. The resulting transformed model after applying the Transformation Algorithm is shown in Figure 4.3.

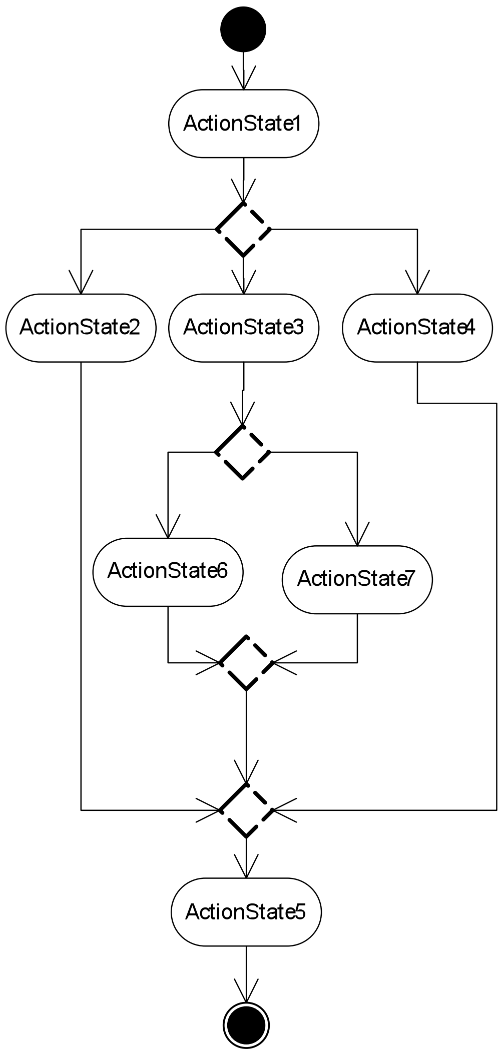


Figure 4.2. A Configurable Business Model

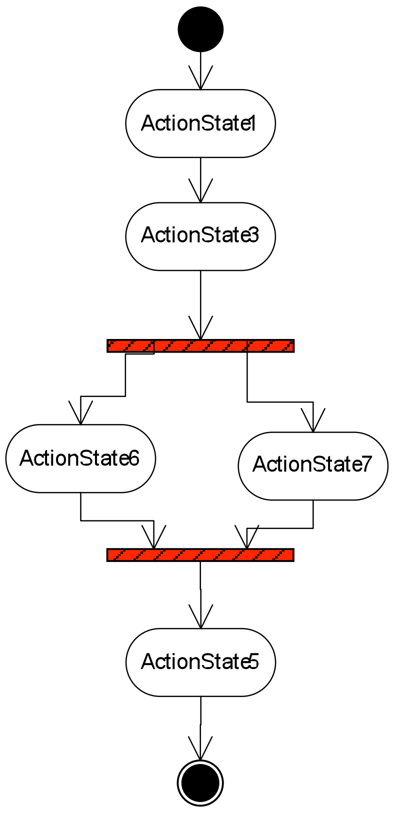


Figure 4.3. The Resulting Transformed Business Model from Figure 4.2.

4.6 Case Study

In this section we apply our configurable approach on a real life case to test the applicability of our approach. This case was used by Aalst et al. in [40] to demonstrate the applicability of C-EPC modeling in real life situations.

Figure 4.4 represents a travel form approval business reference model using C-EPC. The model in Figure 4.4 starts by giving two alternative paths one for domestic 'simple procedure' travel and another for complicated international travels. Some of the actions could be applied by the secretary and others by the employees. At the end, the travel form is accepted, rejected or changed.

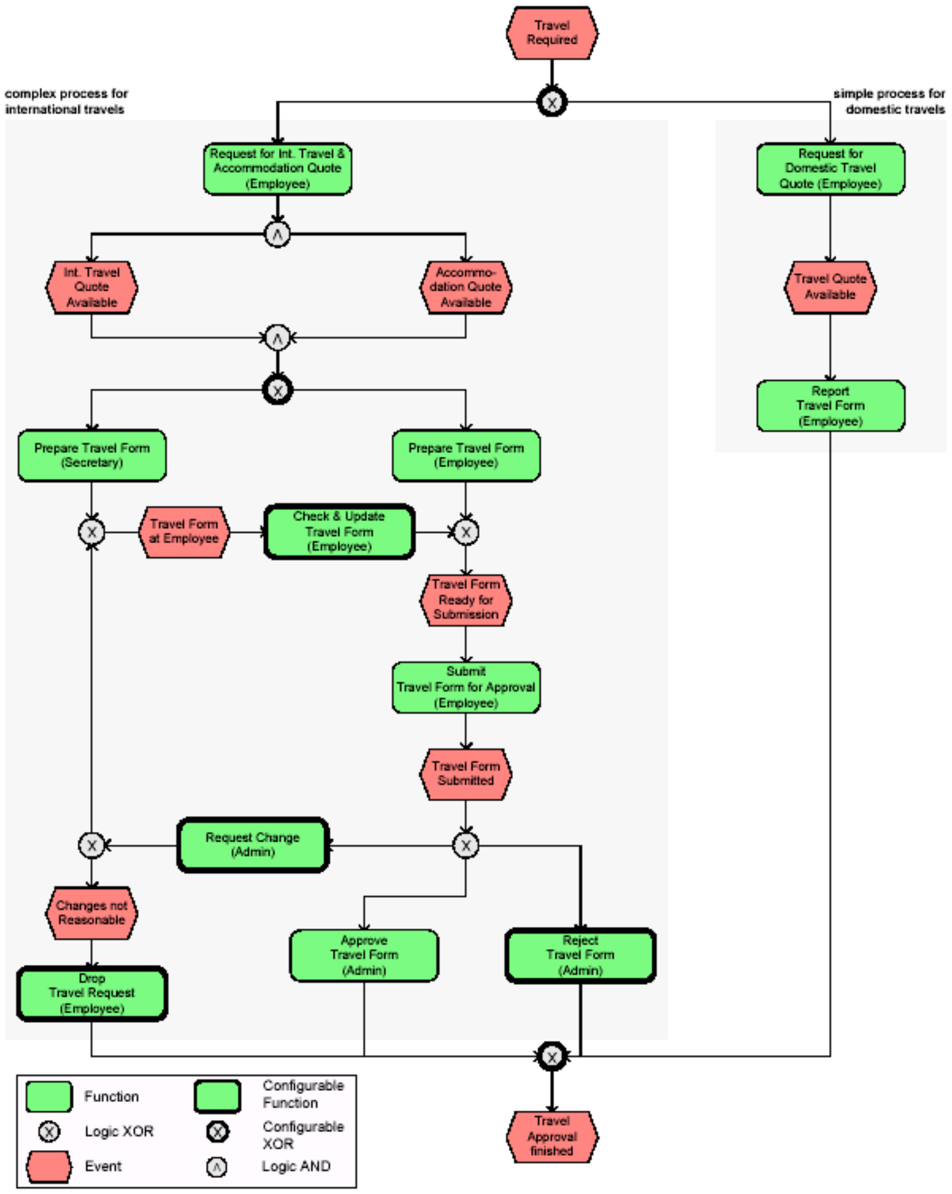

Travel Required
complex process for international travels
simple process for domestic travels
Request for Int. Travel & Accommodation Quote (Employee)
Request for Domestic Travel Quote (Employee)
Int. Travel Quote Available
Accommo-dation Quote Available
Travel Quote Available
Report Travel Form (Employee)
Prepare Travel Form (Secretary)
Prepare Travel Form (Employee)
Travel Form at Employee
Check & Update Travel Form (Employee)
Travel Form Ready for Submission
Submit Travel Form for Approval (Employee)
Travel Form Submitted
Request Change (Admin)
Changes not Reasonable
Drop Travel Request (Employee)
Approve Travel Form (Admin)
Reject Travel Form (Admin)
Travel Approval finished
Function
Configurable Function
Logic XOR
Configurable XOR
Event
Logic AND

Figure 4.4. Travel Form Approval Business Process Reference Model Using C-EPC [40]

First Step: Transforming the C-EPC model to C-UML AD

According to the formal rules stated in section 2.3.3 we were able to convert the above C-EPC business model to C-UML AD business model. Figure 4.5 shows the induced C-UML AD business model. To test the applicability of our approach we introduced some attributes to the model. First, we introduced a mandatory/optional attribute on one of the configurable nodes. Then we introduced a requirement attribute on another configurable node. All the other attributes stated in section 3.2 could also be added but we just focused on those attributes that directly affect the transformation process.

Second Step: Transforming the C-UML AD model to a consistent and syntactically correct configured UML AD model by applying the Transformation Algorithm outlined in section 4.4.

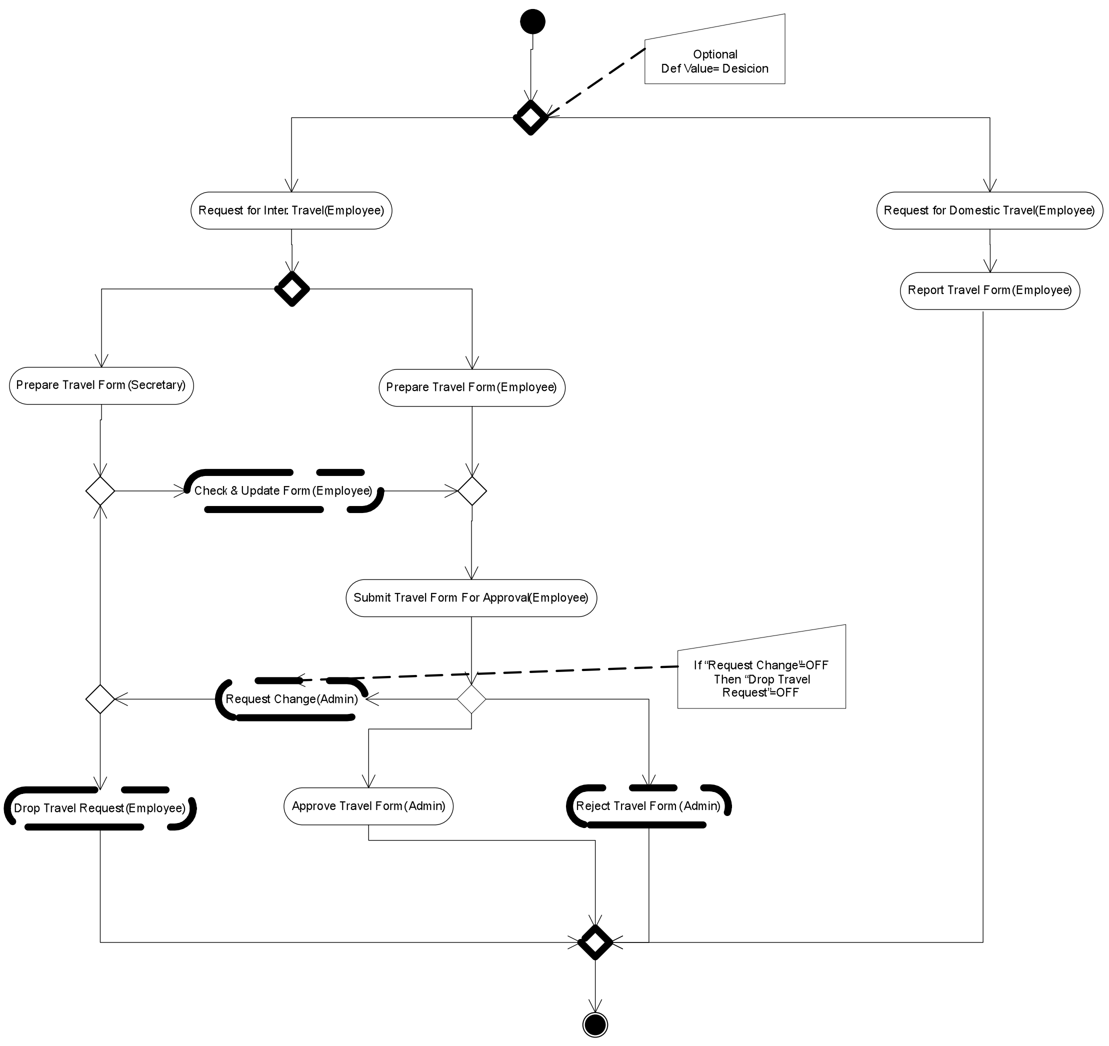
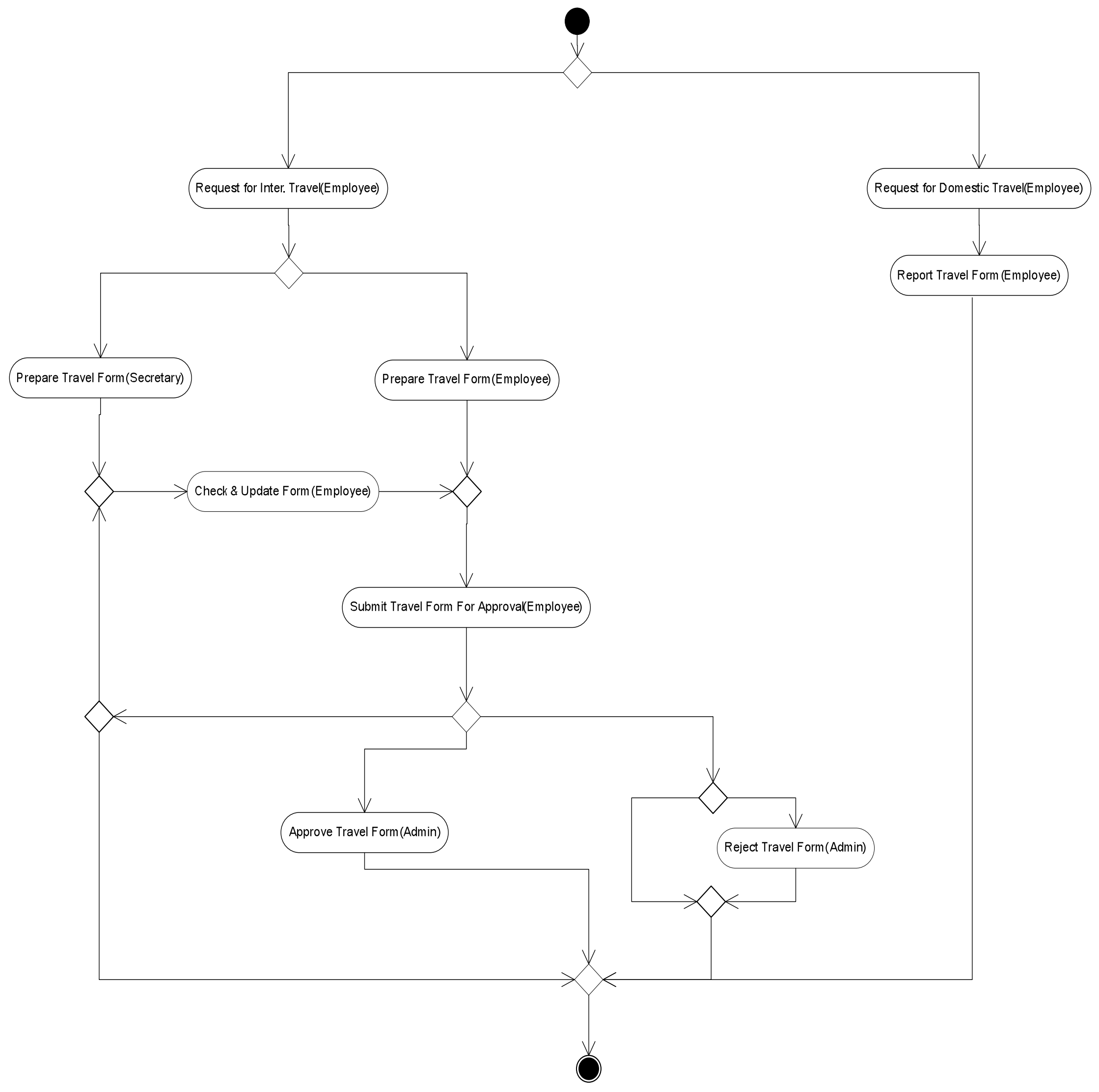

Optional
Def Value= Desicion
Request for Inter. Travel(Employee)
Request for Domestic Travel(Employee)
Report Travel Form(Employee)
Prepare Travel Form (Secretary)
Prepare Travel Form(Employee)
Check & Update Form (Employee)
Submit Travel Form For Approval(Employee)
If "Request Change"=OFF Then "Drop Travel Request"=OFF
Request Change(Admin)
Drop Travel Request(Employee)
Approve Travel Form(Admin)
Reject Travel Form (Admin)

Figure 4.5. Induced C-UML AD Business Process Reference Model

We transformed the configurable models to two different individual models based on a set of different configuration values for the configurable nodes of the model. Figure 4.6. and Figure 4.7. show two different individualized and configured UML AD models.

Figure 4.6 shows an individualized UML AD model where both domestic and international travels are supported. Both options for “Prepare Travel Form” by secretary and employee are also supported. Then the “Check & Update Form” is turned ON. Hence, it was simply kept as is while the “Request Change” and “Drop Travel Request” were turned OFF and thus were removed from the model. Finally, the action that was turned to OPTIONAL, “Reject Travel Form”, introduced two alternative paths to the model. The last remark that we need to make is that when “Request Change” action was turned OFF the “Drop Travel Request” action was forced to be turned OFF due to the requirement attribute associated with the “Request Change” action.



Request for Inter. Travel(Employee)
Request for Domestic Travel(Employee)
Report Travel Form (Employee)
Prepare Travel Form(Secretary)
Prepare Travel Form(Employee)
Check & Update Form(Employee)
Submit Travel Form For Approval(Employee)
Approve Travel Form(Admin)
Reject Travel Form(Admin)

Figure 4.6. A Configured UML AD Business Process Reference Model

In Figure 4.7 the individualized model is applied by a business end user who only uses the international travels. Therefore, the first decision is turned to a specific path (sequence). The second decision was also turned to a specific sequence since the "prepare Travel Form (secretary)" path is also never used by the business end user. The "Check & Update Form", "Request Change" and "Drop Travel Request" actions were all turned OFF.

By looking at the model in Figure 4.7 we note some logical and semantic errors but not syntactic errors. To avoid such errors, several prepared and detailed requirements/constraints, should be added to the model. The concept of applying requirements/constraints on the model needs more research to be able to identify and apply a whole set of constraints that force the model to be semantically correct. Very recent papers [20], [40], [41] have started researching and studying how to achieve such goals, i.e. ensuring that the transformed individualized model is syntactically and semantically correct.

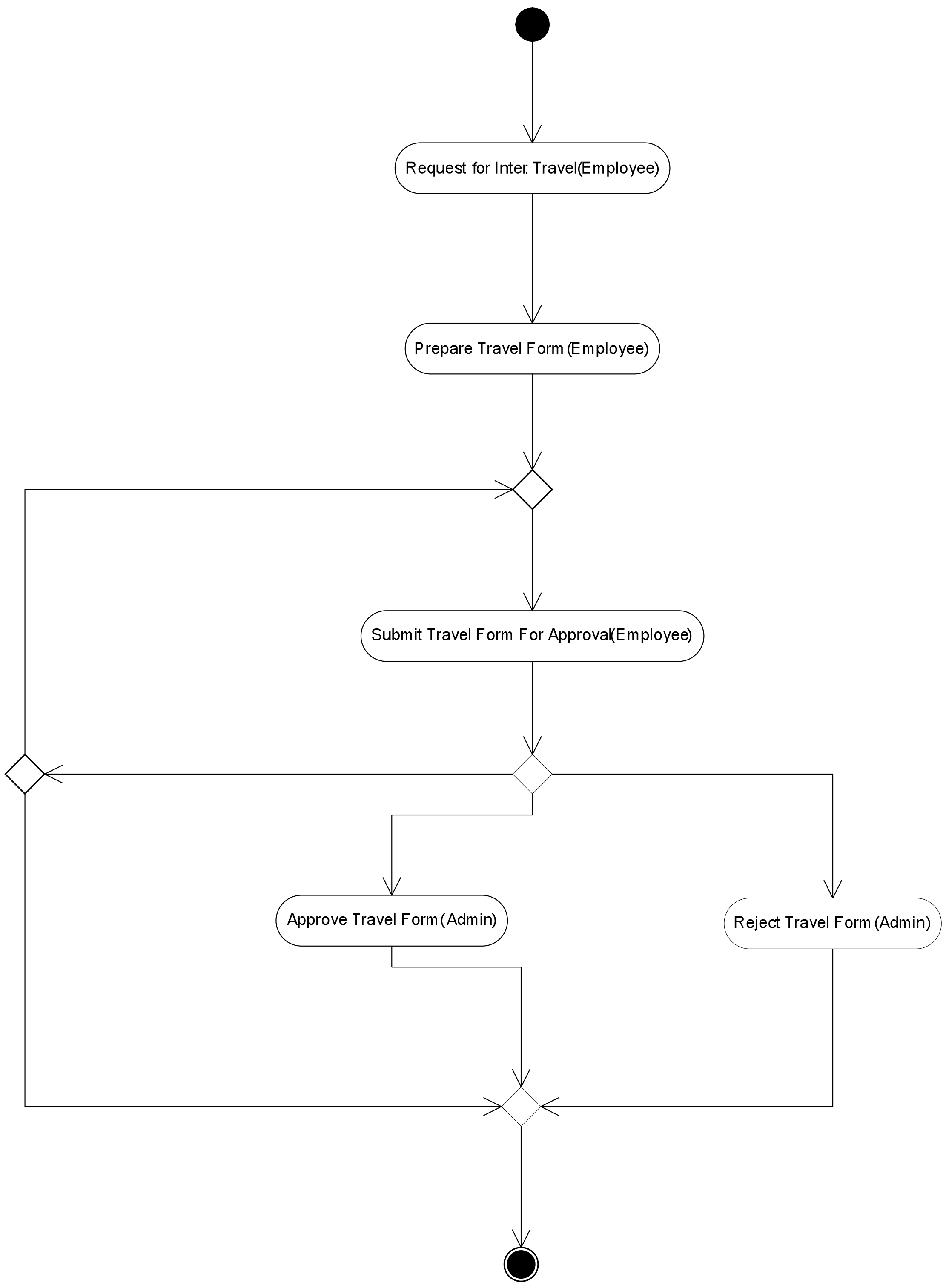

Request for Inter. Travel(Employee)
Prepare Travel Form (Employee)
Submit Travel Form For Approval(Employee)
Approve Travel Form (Admin)
Reject Travel Form (Admin)

Figure 4.7. Another configured UML AD Business Process Reference Model

5. Validation of the Transformation Algorithm

This chapter is dedicated to prove the validity of our Transformation Algorithm (as described in Chapter 4). We validate the Transformation Algorithm by showing how all the transformation steps of the algorithm preserve the syntactic correctness of any given configurable business model, as prescribed by Aalst et al. [14]. Semantic correctness however, was not considered as it falls beyond the scope of this work. We further validate our algorithm by running it on different sets of test scenarios to demonstrate the correctness of our approach.

To show the applicability of the algorithm and demonstrate its correctness, we devised a transformation C-UML AD Transformation Tool, which allows the end user to model a C-UML AD business model and then transforms it back to a standard UML AD based on the end user's configuration values/ decisions.

The C-UML AD Transformation Tool that we implemented converts the user's business model to XML files that are used for controlling the transformation process. The C-UML AD Transformation Tool also extends the UML AD standard constructs to include the newly introduced configurable nodes and their associated attributes. For details about the C-UML AD Transformation Tool description and use, please refer to Appendix A.

5.1 Validating the Transformation Algorithm

In this section we will prove that the algorithm for transforming C-UML AD back to UML AD will preserve the syntactic correctness of the model, following the approach suggested by Aalst et al. in [14].

The Transformation Algorithm will either transform a configurable action or a configurable connector. In case of transforming a configurable action, as discussed in section 4.1, the configuration alternatives (whether ON or OPTIONAL) will not introduce any syntactic errors to the model regardless of the action's location. However, when an action is turned OFF some unnecessary paths may result in the model, as discussed in section 4.3 and these unnecessary constructs will be removed by applying the three reduction rules of section 4.3

In case of transforming a configurable connector, as discussed in section 4.2, the transformation alternatives (except for the seq n) will not introduce any syntactic errors to the model since nothing will be introduced to the model that will violate any of the syntax of the UML AD models. However, the problem with the seq n is that you have to remove all the other paths and unnecessary nodes from the model. This case is also explained in section 4.2 and avoided by the reduction rules of section 4.3.

Eventually, after transforming both actions and connectors according to the rules of section 4.1 and 4.2 and applying the reduction rules of section 4.3, the derived model will always preserve the syntactic correctness of the initial model as was done by Aalst et al in [14].

5.2 Validation by Testing

After proving the correctness of our approach we extended our work to validate the applicability of the Transformation Algorithm by developing test scenarios and testing them on our developed C-UML AD Transformation Tool. Validation by testing is a commonly and widely accepted technique used in industry [5]. It is based on providing a large set of testing scenarios covering a wide range of possible test cases and observing their results. The output is compared with the expected test case output to ensure that the algorithm applied is correct. An additional step that we introduced is to convert the generated UML AD to java code to ensure that the model is syntactically correct.

The first step in this technique was to identify the set of test scenarios. We categorized the test scenarios into two main categories: Unit testing and Integration testing. The unit testing was used to ensure that every transformation rule is working properly on its own, without applying any of the reduction rules and without considering the effect of the associated attributes. Hence, the unit testing covered the transformation of configurable actions, in each of the four cases discussed in section 4.1, as well as the transformation of the configurable connectors, discussed in section 4.2. However, the integrated testing was applied to ensure that the whole algorithm is correct. The tested examples are presented and discussed below in section 5.2.

Whenever a test scenario was applied on our C-UML AD Transformation Tool, the generated output, i.e. the UML AD business model, was converted to java code to ensure that this output model is syntactically correct. The FUJABA CASE tool [9] was used in the conversion, since it is one of the few tools that are capable of transforming dynamic models, such as activity diagrams, to code [37]. Most of the available tools are capable of transforming the static diagrams only, such as the class diagram, to code. The FUJABA served as a very suitable CASE tool for generating code for the UML AD business models except that it was not capable of generating code for the fork/join constructs and therefore these few cases were tested manually without having code generated for them. Also in the current version of the C-UML AD Transformation Tool, when injecting any new conditions within a decision node on the business model one would have to set them manually.

5.2.1 Unit Testing

This section illustrates the steps that were followed to ensure that the transformation rules of configurable actions and connectors (as stated in sections 4.1 and 4.2) will result in a syntactically correct UML AD business model.

5.2.1.1 Transforming a Configurable Action

Tables 5.1, 5.2, 5.3 and 5.4 show the four cases where a configurable action might appear in a C-UML AD and the corresponding UML AD after applying the transformation rules using the OPTIONAL value as the configuration value (decision) of the end user. The OPTIONAL value was used for testing since it always results in the most complicated UML AD business model (more than ON and OFF values). It also implicitly covers the ON and OFF values.

Table 5.1 shows the case where a configurable action lies between two actions. The configuration value is OPTIONAL and therefore the output model after the transformation should have two alternative paths: one with the action and one without it.

Table 5.1. Test Case for Case 1 of Section 4.1

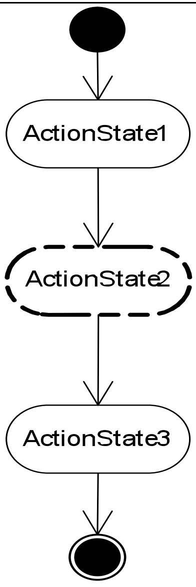
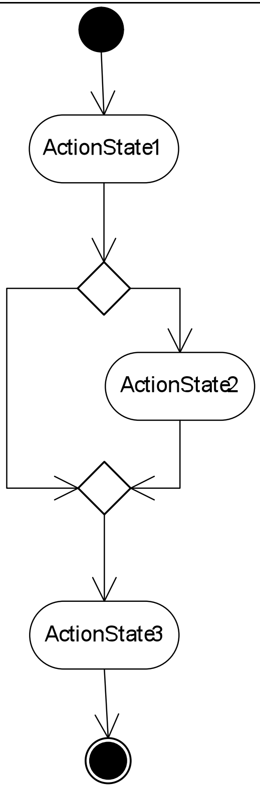
Section being tested	**Input Model C-UML AD business model**	**Generated Output Model Transformed UML AD business model**
Configurable action between two actions		

Table 5.2. shows another case where the configurable action lies between two connectors. The configuration value is OPTIONAL and therefore two alternative paths are provided in the output model.

Table 5.2. Test Case for Case 2 of Section 4.1

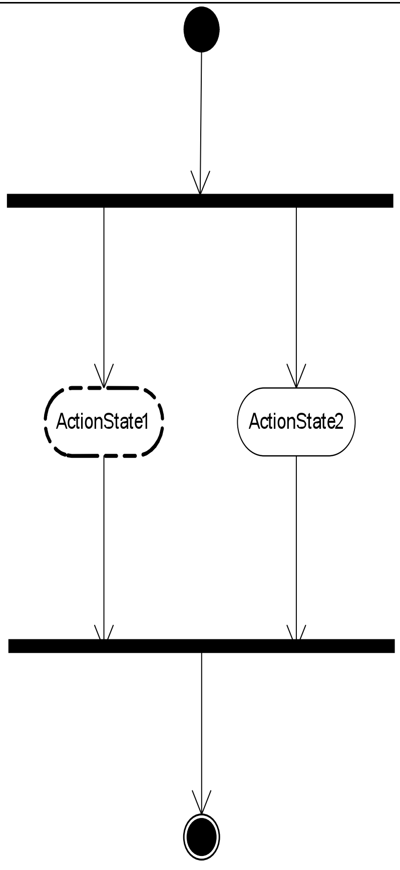
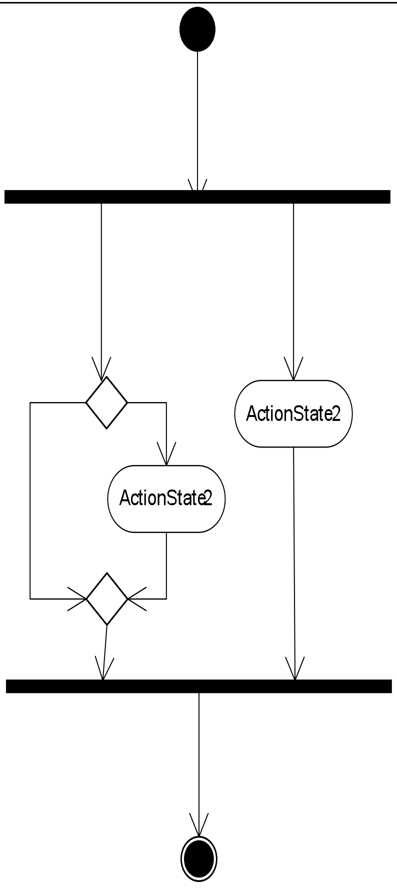
Section being tested	**Input Model** **C-UML AD business model**	**Generated Output Model** **Transformed UML AD business model**
Configurable action between two connectors		

Table 5.3. shows an example where the configurable action lies between an action and a connector. The generated output using the OPTIONAL value is shown in the third column.

Table 5.3. Test Case for Case 3 of Section 4.3

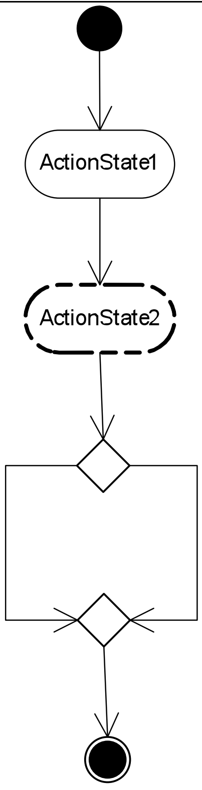
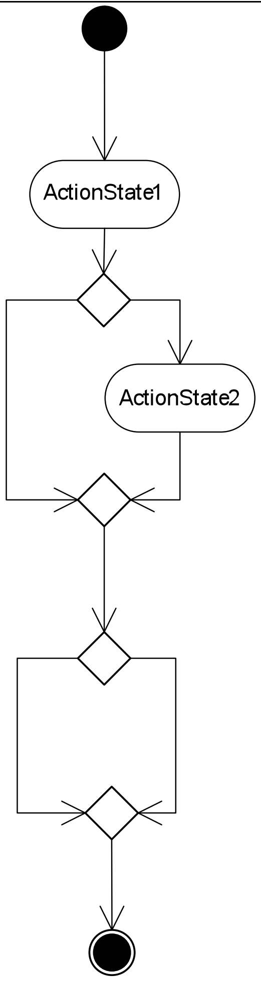
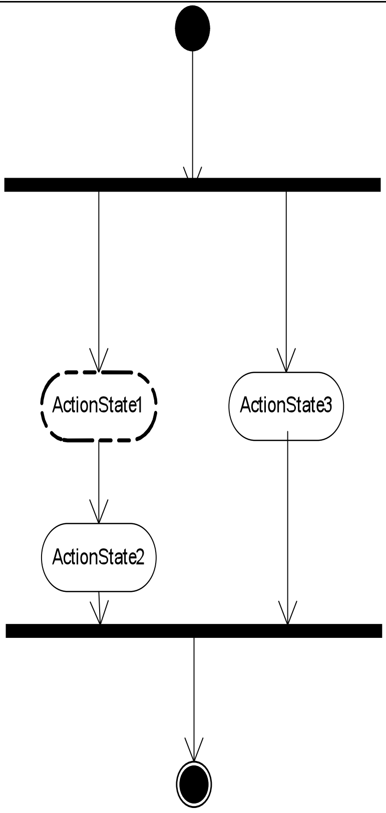
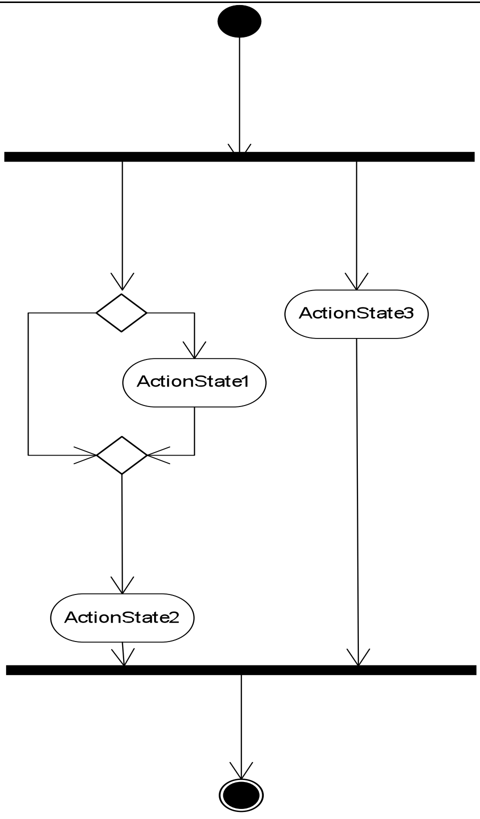
Section being tested	**Input Model** **C-UML AD business model**	**Generated Output Model** **Transformed UML AD business model**
Configurable action between an action and a connector		

Table 5.4. shows the fourth case where a configurable action lies between a connector and an action using the OPTIONAL value as well.

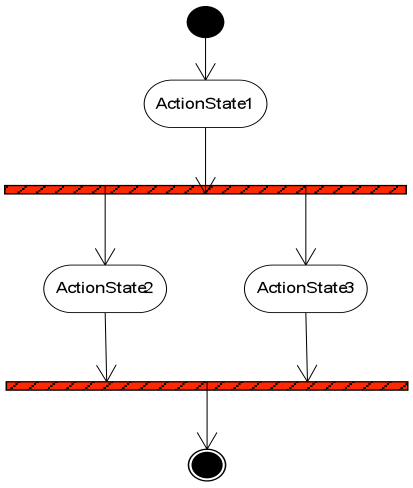
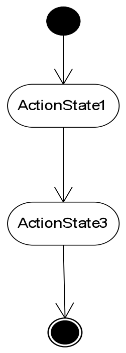
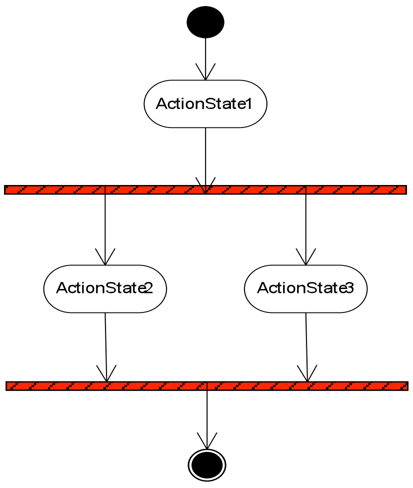
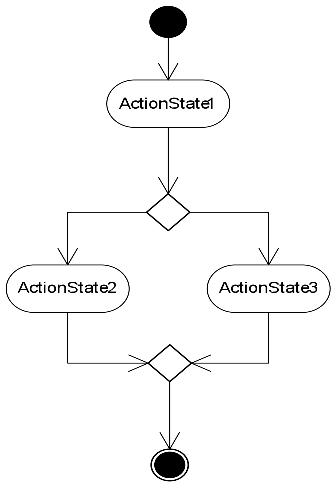
Table 5.4. Test Case for Case 4 of Section 4.1

Section being tested	Input Model C-UML AD business model	Generated Output Model Transformed UML AD business model
Configurable action between a connector and an action		

5.2.1.2 Transforming a Configurable Connector

Table 5.5 shows a subset of the test scenarios that were applied to validate the transformation rules of the fork/join and the decision/merge constructs discussed in section 4.2. The first case in Table 5.5 transforms the configurable fork/join to Seq2, i.e. path n= 2. Therefore, the generated output model removes all other paths leaving only path 2. The other case transforms the configurable fork/join to a decision/merge.

Table 5.5. Test Cases for Testing the Rules of Section 4.2

Section being tested	**Input Model C-UML AD business model**	**Generated Output Model Transformed UML AD business model**
Configurable fork/join transformed to sequence n=2		
Configurable fork/join transformed to decision/ merge		

Similar test scenarios were applied to test the transformation of the decision/ merge to fork/join and to a specific sequence.

5.2.2 Integration Testing

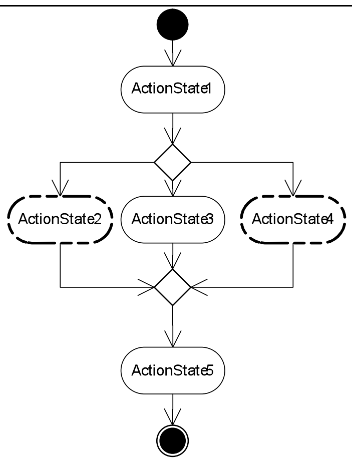
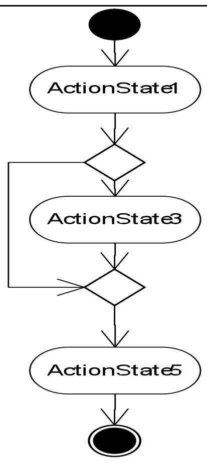
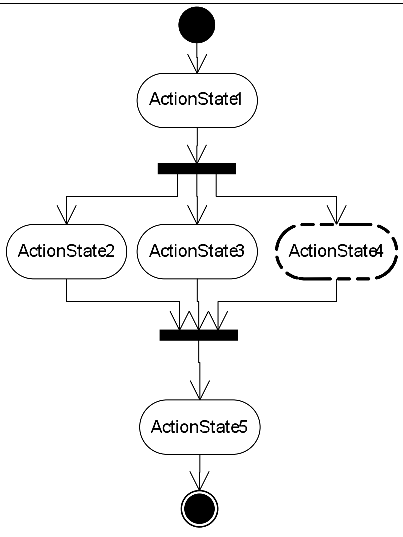
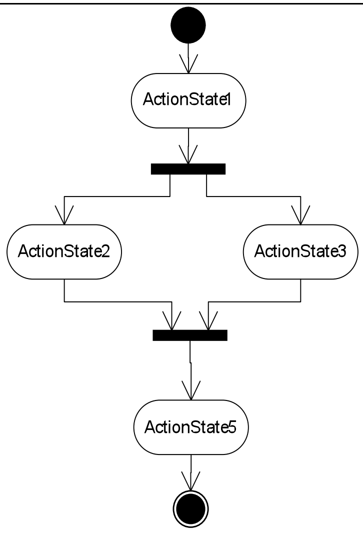
This section covers the test scenarios that were applied to ensure that the complete algorithm is integrated correctly. This includes testing the effect of the configuration attributes as well as the reduction rules with each other. Some test scenarios were used to show positive results and others were used to show negative results, such as an inconsistency problem.

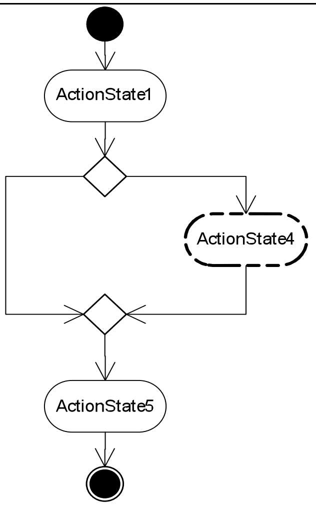
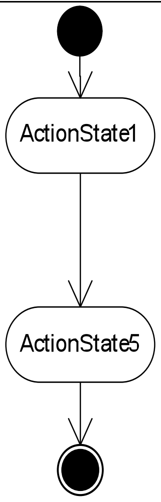
1. First, we tested the algorithm by using different models that do not have any explicit attributes. These test cases illustrate one or more of the reduction rules stated in section 4.3.

   In the first case, shown in Table 5.6, the 1st t reduction rule was applied when the configuration values for both Action State 2 and Action State 4 were OFF, since these values will cause the model to have two empty paths between the decision and the merge. Therefore, one path had to be removed.
   The second case was used to test the 2nd reduction rule. Therefore the configuration value for Action State 4 was OFF, leading to an empty path between a fork and a join.

   The third case was applied to test the 3rd reduction rule. Thus, the configuration value for Action State 4 was OFF leading to two empty paths between the decision and the merge. This forced the 1st reduction rule to be applied removing one path and leaving one. This resulted in a decision/merge having one incoming arc and one outgoing arc. The 3rd reduction rule was successfully applied in that case and the decision/merge connector was removed.

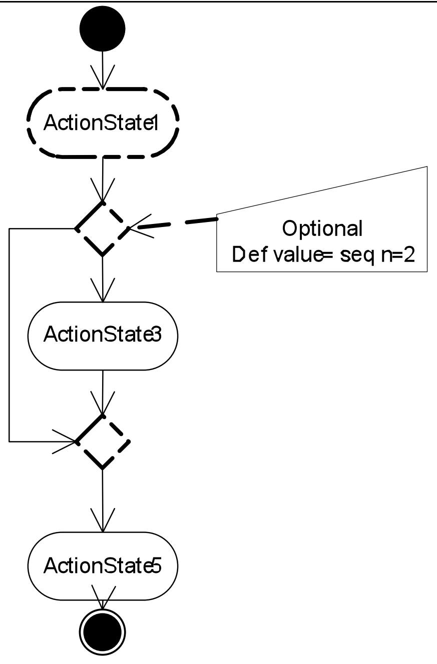
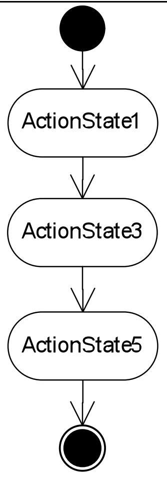
Table 5.6. Test Cases for Testing the Reduction Rules of Section 4.3

Section being tested	**Input Model** **C-UML AD business model**	**Generated Output Model** **Transformed UML AD business model**
1st Reduction Rule	 ActionState1 ActionState2 ActionState3 ActionState4 ActionState5	
2nd Reduction Rule	 ActionState1 ActionState2 ActionState3 ActionState4 ActionState5	 ActionState1 ActionState2 ActionState3 ActionState5

3rd Reduction Rule		

2. Second, we tested the algorithm by using a model having an explicit "mandatory/optional" attribute only. The optional/default value attribute shown in Table 5.7 is applied whenever the end user chooses not to give an explicit configuration value for such a node. The model output in Table 5.7 was generated when no explicit configuration value was given for the transformation and therefore the default value was used. The default value is Seq n=2. Hence, path 2 was kept and all the other paths were removed from the model.

Table 5.7. Test Case for Testing the Model with the "Mandatory/Optional" Attribute

	Input Model C-UML AD business model	**Generated Output Model Transformed UML AD business model**
	 ActionState1 Optional Def value= seq n=2 ActionState3 ActionState5	

3. Third, we tested the algorithm by using a model having both a “mandatory/optional” and “requirement” attributes together, but without having any inconsistencies between the end user’s configuration values and the requirements imposed from the “requirement” attribute.

Figure 5.1 shows a scenario where Action State 4 has a mandatory attribute and another requirement attribute. We applied the transformation using a configuration value for Action State 2 = ON and a configuration value for Action State 4 = ON and a configuration value for the configurable decision connector = standard decision connector. At Action State 4 the end user is forced to provide an explicit configuration value (because of the mandatory attribute) then the requirement attribute is applied. The requirement attribute in this case does not contradict the previous configuration value for action state 2. Therefore, no inconsistency problems occurred and the output resulting from the test is as shown in Figure 5.2.

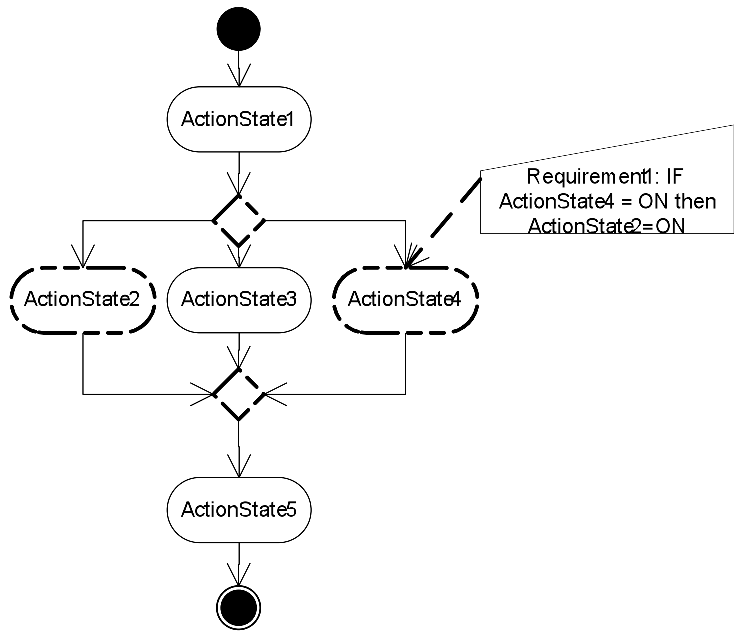


Figure 5.1. Input: C-UML AD Business Model

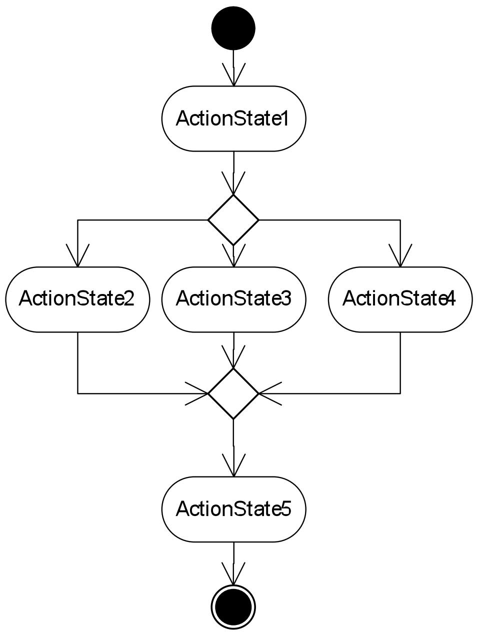


Figure 5.2. Output: Resulting Transformed Business Model from Figure 5.1.

4. The above test in Figure 5.1, was repeated but after imposing some inconsistencies between the end user's configuration value and the requirements imposed from the "requirement" attribute of the model. The same model used in Figure 5.1 was tested but with different configuration values.

   In this test we considered that the end user's configuration value for Action State 2 is OFF and his/her configuration value for Action State 4 is ON. An inconsistency occurred at this point and the configuration was not completed. In this case the end user should either change the configuration value for Action State 2 or for Action State 4. The end user could change the configuration value for Action State 2 to ON instead of OFF and Action State 4 stays ON, in this case no inconsistency problems will occur and the output model will be again as the model shown in Figure 5.2.

Another alternative to resolve the inconsistency problem is to change the configuration value for Action State 4 and make it OFF while keeping Action State 2 OFF then this will not cause any inconsistency problems as shown in Figure 5.3. From the same figure we note that there is only one empty path in the model because of the 1st reduction rule being automatically applied on the model.

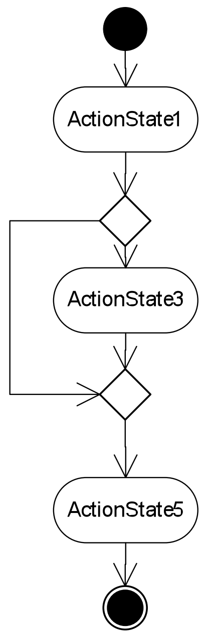


Figure 5.3. Output: Resulting Transformed Business Model from Figure 5.1.

The above scenarios were a sample of the test cases applied to test the correctness of the transformation process and hence to check the syntactic correctness and consistency of the derived UML AD business models. Since all the test scenarios discussed in this chapter and the case study discussed in Chapter 4 were successful, then we have reason to believe that our approach could be the first step towards applying C-UML AD in real life business cases.

6. Summary and Conclusion

Configurable business process reference models are simply a combination of all the possible variations of a specific business model within a given domain. For example, a model may include all the possible payments and invoice verification procedures that could take place in the logistics domain. Such models, known also as best practice models, are configured to meet the business end user’s specific individual requirements without having to design the models from scratch, and thereby allows him/her to reuse proven business practices. Configurable business process reference models provide the basic step for allowing the reusability of reference models in a guided and enhanced manner.

The available standard modeling languages, such as EPC and UML AD, were not initially designed for expressing variation points nor for building configurable reference models. They lack essential characteristics that should be covered by any configurable modeling language. Aalst and Rosemann [28] were the first to introduce a new approach for the re-usability of EPC reference models.

According to Recker et al. [15]: “Configurable reference models may be used to facilitate a model-driven implementation process of business systems. The usage of configurable reference models can lead to the cross-organizational consolidation of previous process configurations, thereby accumulating an evidence-based body of knowledge as to the configuration and enactment of business processes across multiple industry sectors, regions and cultures. These are just a few ideas but they already indicate that reference modeling and model configurability continue to emerge as a vibrant and influential research discipline in the future."

In this work we presented an extension to the UML AD modeling language which we named Configurable UML AD, C-UML AD for short. This language represents the initial steps towards having a complete Configurable UML AD modeling language.
We established the basic algorithm for transforming/configuring the C-UML AD business model to a specific individual model while making sure that the derived UML AD model is consistent and syntactically correct.

6.1 Research Contributions

While many approaches were previously applied on UML AD to express variation points for the Software Production Line engineering concepts only a few were dedicated for modeling configurable business process models. These approaches were not aiming at building a full and a complete configurable modeling language. The C-UML AD modeling language that we introduce is a basic language that covers the major attributes of a configurable modeling language. The notations that we adopted ensure that the complexity of the model does not increase. We were fully aware that these notations should be easy and understandable by business end users who are not IT experts.

We also developed a complete algorithm that monitors the transformation of the C-UML AD business model to a specific individual UML AD model. The algorithm satisfies all the associated constraints ("requirement attribute") of the model thus ensuring that the derived model is consistent. At the same time it ensures that the derived model is syntactically correct. None of the previous approaches provided such an algorithm.

We used a real case study to test the applicability of our introduced language and to validate the transformation algorithm. The case study used in this work was applied by Aalst et al. [40] to demonstrate the applicability of C-EPC.

We made our algorithm visible by embedding it within a C-UML AD Transformation Tool that we developed. The C-UML AD Transformation Tool is used to model C-UML AD business reference models and monitors their transformation/configuration to UML AD business models which are both consistent and syntactically correct.

6.2 Limitations and Future Work

Semantic correctness of the configured model was beyond the scope of our research work and hence was not covered. Also, the transformation of actions/functions was only limited to their existence not their functionality. However, further research can extend on our work and we expect future efforts that build on our work to move in one or more of the following directions:

1. Extending our approach to ensure the semantic correctness of the derived model.

2. Extending our approach so that the transformation of actions/functions goes beyond their existence and includes their functionality as well.

3. Checking the consistency of the constraints before applying them on the model, since the applied constraints could be contradicting among themselves. Different techniques should also be studied to acquire the business constraints from business experts in a language independent manner.

4. Extending BPMN and other standard business modeling languages in a similar way.

5. Suggesting extensions of UML CASE Tools to open the door for reconfigurable business modeling on such tools.

6. Enhancing the C-UML AD Transformation Tool that was developed in this work to include more features, such as automating the injection of new conditions (associated with the decision node).

7. References

1. A. Dreiling, M. Rosemann, W.M.PV.D. Aalst, W. Sadiq and S. Khan, “Model driven process configuration of enterprise systems.” In *18th Conference on Advanced Information Systems Engineering,* June, 2006, Luxembourg, Germany, 2005.

2. D. Dori, B. Golany and P. Soffer, “ERP modeling: a comprehensive approach.” *Information Systems*, 2003.

3. E. Kindler, *“*On the Semantics of EPCs: A Framework for Resolving the Vicious Circle.” In *Proceedings of the Business Process Management: Second International Conference,* Vol. 3080, pages. 82-97, Potsdam, Germany, 2004.

4. E. Seidewitz, “What models mean.” *IEEE Software,* Vol. 20, pages. 26-32, 2003.

5. F. Fleurey, J. Steel and B.Baudry, “Model-Driven Engineering and Validation: Testing Model Transformations.” In *Proceedings SIVOES-MoDeVa Workshop,* 2004.

6. F. Gottschalk, W.M.P.V.D. Aalst and M.H.J. Jansen-Vullers, “Configurable Process Models: A Foundational Approach.” *Reference Modeling, Efficient Information Systems Design Through Reuse of Information Models*, pages. 59-78, July 2007.

7. F. Gottschalk, W.M.P.V.D. Aalst and M.H.J. Jansen-Vullers, “SAP WebFlow Made Configurable: Unifying Workflow Templates into a Configurable Model.” *BPM 2007*, Vol. 4714, pages. 262-270, September 2007.

8. F. Puhlmann, A. Schnieders, J. Weiland and M. Weske, “Variability Mechanisms for Process Models.” *PESOA-Report TR 17/2005, Process Family Engineering in Service-Oriented Applications (PESOA),* Hasso Plattner Institut, Postdam, Germany, 2005.

9. Fujaba CASE Tool Website: http://www.cs.unipaderborn.de/cs/fujaba/projects/eclipse/index.html. [Accessed: 15/6/2009.]

10. H. Storrle, “A Comparison of (e)EPCs and UML 2 Activity Diagrams.” *Tagungsband des 5. Workshops des GI-Arbeitskreises "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)",* Vol. 224, 2006 Available: http://CEUR-WS.org/Vol-224/

11. I. Reinhartz-Berger, P. Soffer and A. Sturm, “A Domain Engineering Approach to Specifying and Applying Reference Models.” In *Proceedings of the Workshop Enterprise Modelling and Information Systems Architectures*, Vol. 75, pages. 50–63, Klagenfurt, Austria, October 2005.

12. J. Ferdian and W. Schmidt, "A Comparison of Event-driven Process Chains and UML Activity Diagram for Denoting Business Processes," M.S.thesis, Technische Universität Hamburg, Harburg, Germany, 2001.

13. J. Mendling, J. Recker, M. Rosemann and W.M.P.V.D. Aalst, "Generating Correct EPCs from Configured CEPCs." *ACM,* Dijon, France, 2006.

14. J. Recker, M. Rosemann, W.M.P.V.D. Aalst and J. Mendling, "On the Syntax of Reference Model Configuration. Transforming the C-EPC into Lawful EPC Models." In *Business Process Management Workshops,* Vol.3812, pages. 497-511, Berlin, Germany, 2006.

15. J. Recker, M. Rosemann,W.M.P.V.D. Aalst, M. Jansen-Vullers and A. Dreiling, "Configurable Reference Modeling Languages." In *Reference Modeling for Business Systems Analysis*, USA, 2006 http://eprints.qut.edu.au/10617/

16. J. Recker, M. Rosemann and W.M.P.V.D. Aalst, "On the User Perception of Configurable Reference Process Models – Initial Insights." In *Proceedings 16th Australasian Conference on Information System,* Sydney, Australia, 2005.

17. K. Czarnecki and M. Antkiewicz, "Mapping Features to Models: A Template Approach Based on Superimposed Variants." In *GPCE,* Vol. 3676 of Lecture Notes in Computer Science, 2005.

18. M.A. Mansour, "Business Process Normalization using Model Transformation," M.S. thesis, American University in Cairo, Cairo, Egypt, 2006.

19. M. Clauß, "Generic Modeling using UML extensions for variability." In *Proceedings of OOPSLA 2001 Workshop,* USA, 2001.

20. M. La Rosa, "Managing Variability in Process-Aware Information Systems." PH.D. dissertation, Queensland University of Technology, Brisbane, Australia, March 2009.

21. M. La Rosa, F. Gottschalk, M. Dumas and W.M.P.V.D. Aalst, "Linking Domain Models and Process Models for Reference Model Configuration." In *Proceedings 10th International Workshop on Reference Modeling*, pages. 417-430, Brisbane, Australia, 2007.

22. M. La Rosa, A.H.M. ter Hofstede, M. Rosemann and K. Shortland, "Bringing Process to Post Production." In *Proceedings International Conference Creating Value: Between Commerce and Commons*, Brisbane, Australia, 2008.

23. M. La Rosa and M. Dumas, "Configurable Process Models: How to Adopt Standard Practices in Your Own Way?" *BP Trends,* November, 2008.

24. M. La Rosa, W.M.P.V.D. Aalst, M. Dumas, M. and A.H.M. ter Hofstede, "Questionnaire-based Variability Modeling for System Configuration." *Software and Systems Modeling*,2008.

25. M. Merriam-Webster. *Merriam-Webster's Collegiate Dictionary*: 11 th ed. USA: Springfield, 2003.[E-book] Available:http://www.mw.com. [Accessed: 16/06/2009]

26. M. Modarres, "Predicting and improving complex business processes: values and limitations of modelling and simulation technologies." *Proceedings of the IEEE 2006 Winter Simulation Conference,* 2006.

27. M. Razavian and R. Khosravi, "Modeling Variability in Business Process Models Using UML." In *Proceedings of the 5th International Conference on Information Technology: New Generations (ITGN'08),* pages 82–87, 2008.

28. M. Rosemann and W.M.P.V.D. Aalst, "A Configurable Reference Modelling Language." Technical Report, Queensland University of Technology, Brisbane, Australia, 2003.

29. N. Russell, W.M.P.V.D. Aalst, A.H.M. Hofstede and P. Wohed, "On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling." *Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006)*, Hobart, Australia, 2006.

30. OMG. MOF. 2.0 Query/ Views/ Transformations RFP, OMG document ad/2002-04-10, 2002 Available: http://www.omg.org/docs/ad/02-04-10.pdf. [Accessed 14/02/2009]

31. OMG Unified Modeling Language (UML) Specification: Infrastructure, version 2.0

32. O. Thomas, "Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation." *Third International Conference on Business Process Management (BPM),* Nancy, France, September, 2005.

33. P. Fettke, P. Loos and J. Zwicker, "Business Process Reference Models: Survey and Classification." *Third International Conference on Business Process Management (BPM)*, Nancy, France, September, 2005.

34. R. Tawhid and C.D. Petriu, "Towards Automatic Derivation of a Product Performance Model from a UML Software Product Line Model." *ACM (WOSP'08),* USA, 2008.

35. S. Cantrell, T. Davenport and J. Harris, "Putting the Enterprise into the Enterprise System." *Harvard Business Review*, USA, 1998.

36. Schnieders and F. Puhlmann, "Variability Mechanisms in E-Business Process Families." In *Proceedings of the 9th International Conference on Business Information Systems (BIS'06)*, pages 583–601, Klagenfurt, Austria, 2006.

37. T. Dinh-Trong, “Rules for Generating Code from UML Collaboration Diagrams and Activity Diagrams.” M.S. thesis, Colorado State University, USA, 2003.

38. Wikipedia, http://www.wikipedia.org [Accessed: 18/06/2009]

39. W.M.P.V.D. Aalst, B.V. Dongen, J. Mendling and E. Verbeek, “Errors in the SAP Reference Model.” BPTrends, June 2006

40. W.M.P.V.D. Aalst, M. Dumas, F. Gottschalk, A.H.M, ter Hofstede, M. La Rosa and J. Mendling, “Preserving correctness during business process model configuration.” *Queensland University*, 2008. [Online]. Available: http://eprints.qut.edu.au/15717/. [Accessed: 17/06/2009].

41. W.M.P.V.D. Aalst, M. Dumas, F. Gottschalk, A.H.M. ter Hofstede, M. LaRosa and J. Mendling, “Correctness-Preserving Configuration of Business Process Models.” In *Proceedings Fundamental Approaches to Software Engineering (FASE 2008)*, Budapest, Hungary, 2008.

42. Y. Choi, G. Shin, G., Y. Yang, and C. Park, “An Approach to Extension of UML 2.0 for Representing Variabilities.” In *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS’05),* 2005.

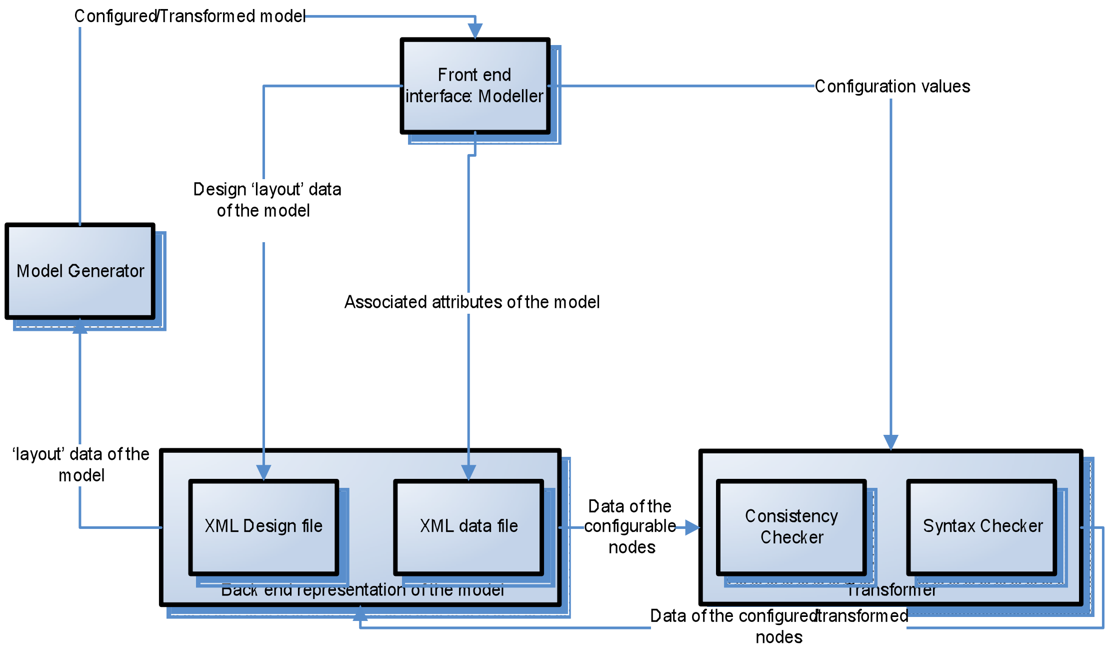
Appendix A. The C-UML AD Transformation Tool

This section is dedicated for describing and explaining the C-UML AD Transformation Tool that was implemented to support the work done in this research. This tool provides a graphical user interface to both the business process engineer who is responsible for building the configurable business process reference model and the business end user who is responsible for transforming and individualizing the configurable business process reference model to suit his/her specifications.

Figure A.1. shows the system architecture of the C-UML AD Transformation Tool. The "Modeler" is the front end interface that is used by the process engineer to build the configurable business process model. This interface provides all the elements (both standard and configurable elements) needed to build the business model.

Once the model is drawn on the Modeler the model is automatically converted to two XML files: one file stores the layout data of the model, such as the size of the elements, the color, border style etc. The other XML file stores the associated attributes data with each configurable node, such as the default value or the specification level, guideline information or the requirement attribute.

The business end user could retrieve the desired configurable business process model. Then the business end user starts the transformation process via the "Transformer" as shown in Figure A.1. The "Transformer" applies the Transformation Algorithm discussed in Chapter 4. At each configurable node the possible configuration values are displayed to the end user to choose one of the available options. Once a configuration value is chosen, the "Consistency Checker" is invoked to check for inconsistency problems. In case there is a problem an alert message is displayed to the end user. Otherwise, the transformation step is applied on the model and the "syntax checker" is invoked to ensure that the derived model at each transformation step stays syntactically correct. Each transformation step affects the associated XML files. After all the configurable nodes are transformed, the new transformed/configured business model is generated by the "Model Generator" and displayed to the end user on the "Modeler" interface.



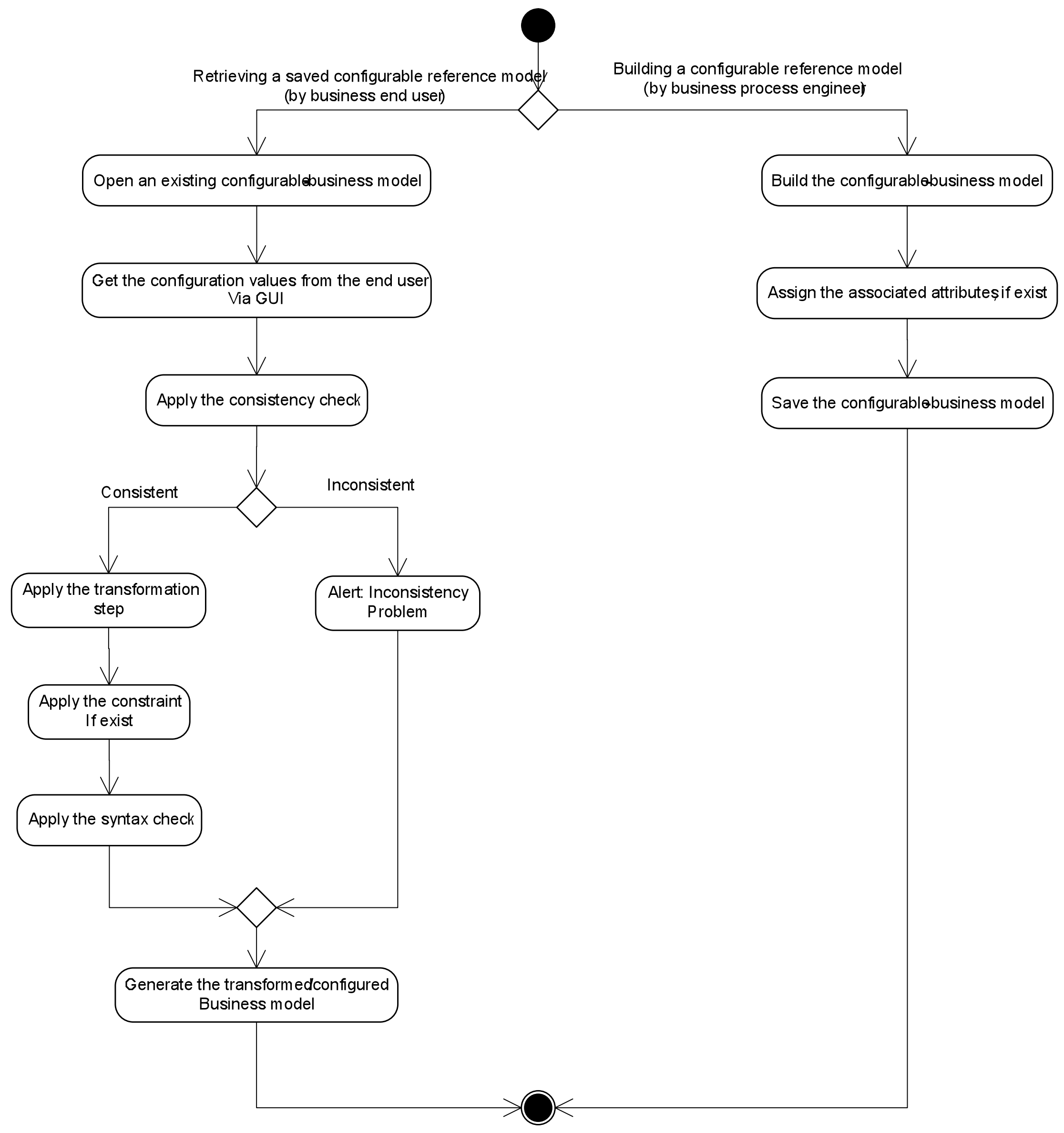
Configured/Transformed model
Front end interface: Modeller
Configuration values
Design 'layout' data of the model
Model Generator
Associated attributes of the model
'layout' data of the model
XML Design file
XML data file
Back end representation of the model
Data of the configurable nodes
Consistency Checker
Syntax Checker
Transformer
Data of the configured/transformed nodes

Figure A.1. System Architecture

Figure A.2. shows the basic steps of the system flow. The C-UML AD Transformation Tool could be either used by the business process engineer to build a configurable reference model or by the business end user to retrieve and transform a saved configurable reference model according to his/her specific business situations.

The business process engineer starts by building the configurable business model using the standard and configurable elements of the C-UML AD. Then the attributes, such as the "requirement attribute" are entered into the system via our implemented front end interface. Finally, the configurable reference model is saved.

At any other time, the business end user who is interested in one of the saved configurable reference model could retrieve this model and starts the transformation process. In this process the business end user will start giving the configuration values (according to his/her business requirements) to all the configurable actions and connectors in the model. At the back end, the Transformation Algorithm discussed in Chapter 4 will be applied, where both the consistency of the model (i.e. consistency between the configuration values entered by the business end user and any constraints "requirement attribute" applied on the model) and the syntax of the model are checked. If an inconsistency problem occurred then the business end user is alerted and the configuration value should be adjusted to avoid such problem. Eventually, after applying the transformation process the new transformed/ derived model will be displayed on the modeler to the business end user.



Retrieving a saved configurable reference model
(by business end user)
Building a configurable reference model
(by business process engineer)
Open an existing configurable business model
Build the configurable business model
Get the configuration values from the end user Via GUI
Assign the associated attributes, if exist
Apply the consistency check
Save the configurable business model
Consistent
Inconsistent
Apply the transformation step
Alert: Inconsistency Problem
Apply the constraint If exist
Apply the syntax check
Generate the transformed/configured Business model

Figure A.2. System Flow