Archetypes based process design methodology

Tallinn University of Technology, Institute of Informatics, Raja 15,
12618 Tallinn, Estonia
enn.ounapuu@ttu.ee

Abstract. In the dynamic world of today, Enterprises need to increasingly support their critical business operations with the Internet of Services. In this context, an appropriate Model Driven Development (MDD) methodology would be beneficial for them in terms of flexibility, reusability and wider applicability of the available services. But, traditionally, MDD methodologies have been developed for large companies and are currently too complex and too expensive for wide range of applications. In this document the concept of process archetype is developed. To make these archetypes executable MDD is advanced. Our final goal is to develop methodology and tool for enterprise information systems development based only on models - so named XNP (eXtreme Non Programming) approach.

Keywords: Business archetype, Model Driven Development, Process Design, Goal modeling, Cloud computing, Software as a Service -SaaS.

1 Introduction

Process design is the conscious evaluation and organization of the tasks that a business process is composed of. Designing a process that improves current performance and/or conformance is a challenging task that requires a cooperation of multiple specialists to define organizational strategies, goals, constraints and resources.

The intelligent solution is able to interact with its environment and change its behavior, structure, and strategy—behaving actually as an intelligent entity.

It is able to adapt to rapid changing situations, gradually change its model, and survive in the next usage cycle. The intelligent solution as we see it is characterized by its ability to learn from and adapt to changes in its environment and reinvent itself, sometimes with surprising results. We see future of the intelligent solutions deriving efficiencies through the automation of their core processes and the exploitation of knowledge inherent in their organization. Their ability to respond quickly to changes will improve significantly as the knowledge base and "intelligence density" within the enterprise grows and problem-solving capabilities improve dramatically.

First, it is important to have clear requirements of what they are trying to achieve. Often, separate parts of the organization have different goals that influence the scope and design of the solution. Gaps in requirements should be addressed early in the project to avoid larger problems later in the project.

Our final goal is to develop methodology and tool for cloud solutions development (Software as a Service –SaaS solutions) based **only on models** so named XNP (eXtreme Non Programming) approach [19]. To make clear, what we want to achieve, let's look to the most known Saas application – Salesforce [22]. For customizing Salesforce applications they created framework for this [23]. „The Application Framework lets you customize existing applications or build applications from scratch without writing any code. You can use the declarative power of the Application Framework to quickly create robust applications on Force.com. The Application Framework Builder gives you easy-to-use tools to modify characteristics of your data, as well as specify the scope of applications or the layout of data on a page. You can also define workflows based on user interaction with data, or create reports on the data. You can use buttons or custom links to extend the default capabilities of your Force.com application. You can create and modify tabs, which can be associated with a Force.com object, Visualforce page, s-control or any web page. You can give users access to tabs, and the user can customize the display of their own set of tabs within an application" [23].

What we try to achieve is to make this approach common for all Saas applications.

In this article we first analyze fundamental concepts of the approach and propose the conceptual model of this. Important part of conceptual model is Goal model what is integrated with process model. Later we propose Process Archetype pattern what is somehow central in our approach. The idea is close to the A *software product line* approach: "A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [18]. In last chapter we present the complex of models of the methodology.

2 Fundamental concepts

One of the central concepts of our approach is the process and process model. In this chapter we define this concept using the notion of domain state. Later we show how our definition extends traditional process notation BPMN.

In our approach we rely on Dines Bjorner a Triptych of Software Engineering approach [2]:

Dogma: *Before we can design software we must have a robust understanding of its requirements. And before we can prescribe requirements we must have a robust understanding of the environment, or, as we shall call it, the domain in which the software is to serve – and as it is at the time such software is first being contemplated.*

According to the ontological framework [20], the world is made of things that possess properties. Properties can be intrinsic (e.g. age) to things or mutual to several

things (e.g. a person works for a company). Things can compose to form a composite thing that has emergent properties, namely, properties not possessed by the individuals composing it. Properties (intrinsic or mutual) are perceived by humans in terms of attributes, which can be represented as functions in time. The state of a thing at a given time is the set of values its attribute functions (also termed state variables) attain at that time. When properties of things change, these changes are manifested as state changes or events. State changes can happen either due to internal transformations in things (self action of a thing) or due to interactions among things. Not all states are possible, and not all state changes can occur. The difference between internal transformations and interactions can be modeled by the distinction between unstable and stable states. If an event occurs in a thing due to an internal transformation, the state prior to the event is considered unstable. If there is no internal transformation that can change the state, it is considered stable and can only be changed via interactions with other things.

To provide a formal basis for expressing process-related concepts in ontological terms, we use definition of the domain concept proposed in [21].

Domain: A part of the world whose changes we want to model.

In ontological terms, a domain consists of a set of things and their interactions. By defining a process over a domain, we set the scope (of control) of the process. This provides a clear distinction between what would be considered external events, which result from actions of things outside the domain and hence are outside of the process control, and internal events that may occur while the process is enacted and are governed by the process.

Our purpose is to define abstract process models that reflect the dynamics of processes**.** We therefore abstract the process domain in terms of its states:

State of a domain (at a given time): The values at the given time of a set of time-dependent attributes (state variables) that provide sufficient information about the domain for the purpose of modeling.

The state of the domain is determined by the states of the things included in it and by emergent state variables of composite things, or of the domain itself, which arise due to interactions. We view states as being discrete, meaning that any change from one state to another occurs at a certain moment in time.

Let's define the notions of stable and unstable states. We link these concepts to the notions of domain:

- Stable state domain: A state that can change only as a result of an action of a thing outside the domain.
- Unstable state domain: A state that must change due to internal events and interactions of things inside domain.

This distinction has a special role in our model, as we will later view the execution of a process in terms of state transitions.

A **business process** is a collection of related, structured activities that produce a specific service or product (serve a particular goal) for a particular customer or customers.

It is also possible to define the basic abstract notion of a process using state notion [21]:

A process: A sequence of unstable states leading to a stable state.

This definition of a process does not incorporate the origin of the initial unstable state. In particular, it can be the outcome of an interaction between (things in) the domain and things outside the domain.

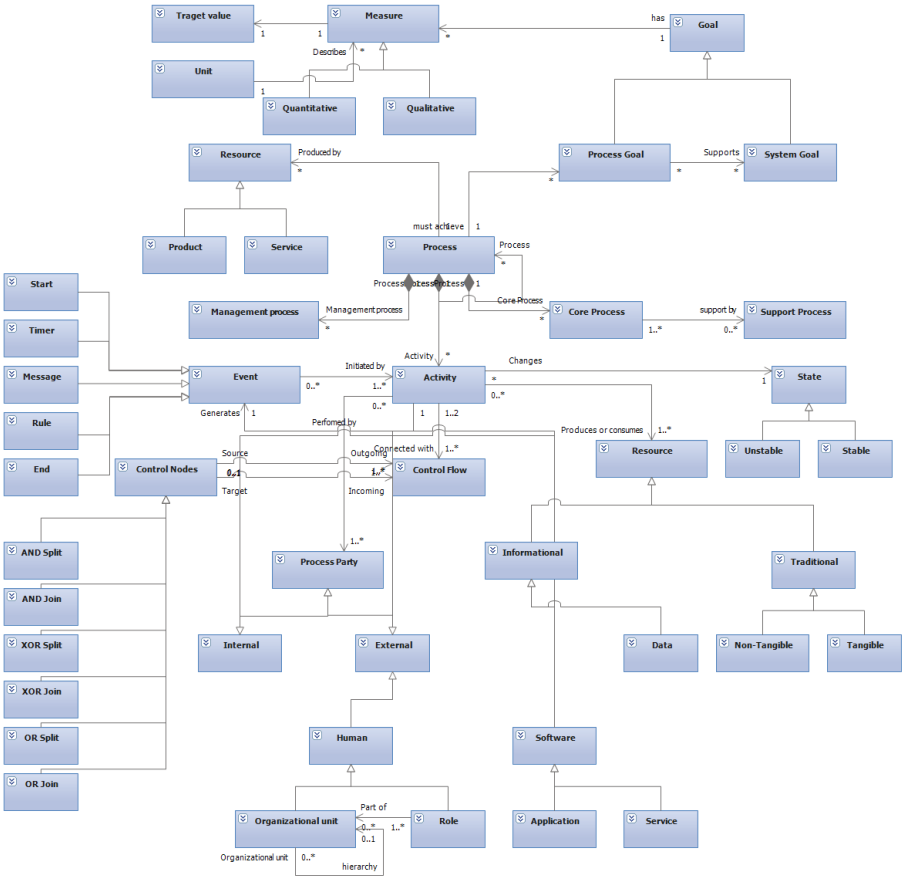

Fig. 1. Process conceptual model.

Thus, the process can only change state variables that are part of the domain definition.

It is the initial subset of unstable states (of the domain) on which the process begins. Considering the domain has been in a stable state prior to the initiation of the process, these states result from events external to the domain, and these events trigger the process. It is important to note that the starting point of the process is an unstable state. This is because it is the initial unstable state of the domain that determines how the process will proceed. This initial state is attained via various combinations of a prior (stable) state and an external event.

For practical reasons, we might not include all state variables of things in the process domain in the definition of domain state. Note, the definition of a process implies that to be a process, a set of ordered activities needs to lead to a defined goal.

Finally, we recognize that a process might progress through different sequences of states, depending on the initial state, and on state changes caused by events external to the process domain.

How you can see the notion of goal is incorporated into the abstract (state-based) view of a process described above.

Processes are executed in order to achieve some predefined outcomes. These outcomes are in harmony with some organizational objectives (system goal). We term the desired outcome of a process a **process goal**. Process goal is formed using state variables and must have exact measures. It is very important to have on place target values for process goals. This enables of self adaption of the system. In the case of different options for sub processes this enables make directed decisions.

3 Data and Process Archetypes

The Merriam- Webster dictionary [16] defines:

„Archetype - the original pattern or model of which all things of the same type are representations or copies".

In the field of informatics, an **archetype** is a formal re-usable model of a domain concept. Traditionally, the term *archetype* is used in psychology to mean an idealized model of a person, personality or behavior. The usage of the term in informatics is derived from this traditional meaning, but applied to domain modeling instead.

An archetype is defined by the OpenEHR foundation [4] for health informatics as follows:

> *An archetype is a computable expression of a domain content model in the form of structured constraint statements, based on some reference model. openEHR archetypes are based on the openEHR reference model. Archetypes are all expressed in the same formalism. In general, they are defined for wide re-use, however, they can be specialized to include local particularities. They can accommodate any number of natural languages and terminologies.*

The use of archetypes in health informatics was first documented by Thomas Beale, who stated the concept was coined by Derek Renouf. According to Beale, Renouf applied archetypes to configuring Smalltalk systems [5].

Business archetypes and archetype patterns are originally designed and introduced by Jim Arlow and Ila Neustad [3]. Recently in Tallinn University of Technology is defended thesis where as a result, the refined and enhanced version of business archetypes and archetype patterns is proposed. This refined and enhanced version of archetypes and archetype patterns were used for engineering of business domains,

requirements and software. The methodology is used for clinical laboratory domain model in real life LIMS (Laboratory Information Systems) development [7].

Beyond the state of the art

All archetypes referred are in the class of the object-oriented design patterns. Object-oriented design patterns typically show relationships and interactions between classes or objects, without specifying the final application classes or objects that are involved.

In our approach we use archetype principles in the business processes. Our approach is process aware. Process aware information system" (or PAIS) can be defined as a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models [9].

Most prominent author in this area is Wil van der Aalst. In process-aware information systems various perspectives can be distinguished. The control-flow perspective captures aspects related to control-flow dependencies between various tasks (e.g. Parallelism, choice, synchronization etc). Originally twenty patterns were proposed for this perspective, but in the latest iteration this has grown to over forty patterns. The data perspective deals with the passing of information , scoping of variables, etc, while the resource perspective deals with resource to task allocation, delegation, etc. Finally the patterns for the exception handling perspective deal with the various causes of exceptions and the various actions that need to be taken as a result of exceptions occurring [1].

We make one step forward to direction of the business. *Our Process archetypes* are an abstract notion of one or more service utilities, which are used in a business process along with various Models. Domain-specific multimodeling is a software development paradigm where each view is made explicit as a separate domain-specific language (DSL). Successful development of a modern enterprise system requires the convergence of multiple views. Business analysts, domain experts, interaction designers, database experts, and developers with different kinds of expertise all take part in the process of building such a system. Their different work products must be managed, aligned, and integrated to produce a running system. Every participant of the development process has a particular language tailored to solve problems specific to its view on the system. The challenge of integrating these different views and avoiding the potential cacophony of multiple different languages is the coordination problem [10, 11].

4 Archetypes based Process design

In our approach business process Archetype model, what is central is supported by additional domain-specific models, such as:

- Motivation Model - is used for measurement of effectivity of the business process;
- Decision Model - is used by both service providers and business people to define the inter-national, national, organizational and other business rules for the particular service;

- End User Interface Model - is used to define the needed interfaces for the end-user (e.g. what the end user must input or choose as information and what is the necessary service output);
- Integration Model - is used to define how other service utilities (systems) may be used from this service utility (e.g. integration with Enterprise Resource Planning systems);
- Organization Model - is used to define, who is responsible for what in the company;
- Security Model - is used to define the security infrastructure that should be put in place;
- Information and Data Model - is used to define the exchange of necessary information among the various actors (humans, companies, service utilities);

Model	Basic notation
Business process model	BPMN 2.0 [6]
Business Motivation Model	BMM [15]
Decision model	RIF
End User Interface Model	XForms 1.1 [13]
Integration Model	WS-I [14]
Organization model	
Information and Data Model	Relational data model
Security Model	

All named models are well known and have already community acceptance, but every of them is criticized from some point. We believe that our systematic integrated approach can fit them together to solve our challenging main XNP problem.

Conclusions

The main features of our approach:

- It is fully model driven. The models of the approach are consistent and allow the system to create a working application. The modification of the application takes place through the change of the models.
- Project goal is to be achieved, since its main elements are already widely used in practice and new elements can be relatively painlessly integrate to the system. Our approach is deliberately attempting to use tried and tested standards of practice (BPMN 2.0, BMM, Xforms, Rule systems). It makes possible also to reuse the existing implementations of them.
- Our Process Archetype Pattern concept makes the design of processes more flexible, reusable and adaptable. Using rule systems you can adapt yours implementation to the different enterprises, countries, regions and users. You can adapt to the changes in economical situations.
- An expanded process state handling of our approach makes it possible to substantially strengthen the realization of BPMN. It makes possible to integrate system and process goals to the process execution. Provides a basis for decision-making models to integrate to the processes.

Acknowledgments.
This project is supported by Estonian science foundation Project SF0140013s10 "Model-based Creation and Management of Evolutionary Information Systems".

References

1. van der Aalst, W. M. P., ter Hofstede, A.H.M.,Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer. 694 p. (2009)
2. Bjorner, D.: Software Engineering 3. Domains, Requirements, and Software Design. Spinger-Verlag, Berlin Heidelberg (2006)
3. Arlow, J. and Neustadt, I.,: Enterprise Patterns and MDA: Building Better Software With Archetype Patterns and UML. s.l. : Addisson-Wesly. (2003)
4. S. Heard & T. Beale. (eds.): Archetype Definitions and Principles. openEHR. (2005) http://www.openehr.org/svn/specification/BRANCHES/Release-1.0.2-candidate/publishing/architecture/am/archetype_principles.pdf Retrieved 11 Jan 2012.
5. Beale, T.: Archetypes: Constraint-based Domain Models for Future-proof Information Systems. Proceedings of the 11th OOPSLA Workshop on Behavioural Semantics. (2002) http://www.openehr.org/publications/archetypes/archetypes_beale_oopsla_2002.pdf Retrieved 11 Jan 2012.
6. BPMN notation. http://www.omg.org/spec/BPMN/2.0/ Retrieved 11 Jan 2012.
7. Piho,G.: Archetypes Based Techniques for Development of Domains, Requirements and Software. Towards LIMS Software Factory. PHD THESIS, TUT PRESS.(2011)
8. Seshan, P.: Process-Centric Architecture for Enterprise Software Systems. Auerbach Publications. (2010)
9. Dumas, M., van der Aalst W.M.P., ter Hofstede A.H.M.: Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley-Interscience, Hoboken, NJ, USA. (2005)
10. Hessellund A.: Domain-Specific Multimodeling. Ph.D. Dissertation, Supervisors: Peter Sestoft and Kasper Østerbye. (2009)
11. Lochmann H., Hessellund A.: An Integrated View on Modeling with Multiple Domain-Specific Languages. Proceedings of the IASTED International Conference Software Engineering (SE 2009). pp. 1-10. (2009)
12. SBVR.: Semantics of Business Vocabulary and Business Rules, Version 1.0. http://www.omg.org/spec/SBVR/1.0/ Retrieved 11 Jan 2012.
13. XForms 1.1.: http://www.w3.org/TR/xforms/ Retrieved 11 Jan 2012.
14. WS-I.: Web Services Interoperability Organization. http://www.ws-i.org/ Retrieved 11 Jan 2012.
15. BMM.: Business Motivation Model. http://www.omg.org/spec/BMM/
16. http://www.merriam-webster.com/dictionary/archetype Retrieved 11 Jan 2012.
17. Borko, F., Armando E.: Handbook of Cloud Computing 1st Edition. 653p. Springer (2010)
18. Clements, P., Northrop L. Software Product Lines : Practices and Patterns. Addison Wesley. (2001)
19. Embley, D.W., Liddle S.W., Pastor O. Conceptual-Model Programming: A Manifesto. Theory, Practice, and Research Challenges. Pp. 3-16. Embley, D.W., Thalheim, B. (Eds.) Handbook of Conceptual Modeling. Springer (2011)

20. Etien A., Rolland c. A Process for Generating Fitness Measures. Advanced Information Systems Engineering. Lecture Notes in Computer Science, Volume 3520/2005, pp. 17-52. (2005)
21. Soffer P., Wand Y. Goal-Driven Multi-Process Analysis. Journal of the Association for Information Systems: Vol. 8: Iss. 3, Article 9. (2007)
22. Salesforce http://www.salesforce.com/ Retrieved 11 sept. 2012
23. Force framework http://wiki.developerforce.com/page/Application_Framework Retrieved 11 sept. 2012