Formal Approach use to Choose a Software Manufactoring Cell's SDLC

Denis Ávila Montini
Brazilian Aeronautics Institute of Tecnology – ITA

Danilo Douradinho Fernandes
Brazilian Aeronautics Institute of Tecnology - ITA

Francisco Supino Marcondes
Brazilian Aeronautics Institute of Tecnology - ITA

Paulo Marcelo Tasinaffo
Brazilian Aeronautics Institute of Tecnology – ITA

Italo Santiago Vega
Pontifícia Universidade Católica de São Paulo - PUC-SP

Luiz Alberto Vieira Dias
Brazilian Aeronautics Institute of Tecnology – ITA

Abstract

This paper shows how to use state machines and systematic approaches to software modeling to help modeler to improve, verify and validate a Domain Analysis and also refine and improve enterprise business processes. The main objective of this approach is how to systematic got a DSL from a Domain Analysis which can be used code system respecting, all business rules without complex definitions or documents. Many problems of Computer Software Systems (CSS) are derived from a lack of its behavior specification in order to solve that problem, but even with a well defined system behavior, many business rules are not properly treated since formalization becomes on the design phase. This paper shows an approach on how to systematically refine domain analysis to consider all business rules. It considers a state machine which represent all aspects of the domain choice. The state machines use is based on user friendliness and formality.

Keywords: Domain Analysis, Domain Specific Language; Formal Methods and State Machines.

1. Introduction

Too much formalization can bring problems to software development, increasing costs and time in an other hand, there is not a conclusive study that can prove that formal methods are less error prone than heuristic methods [1] [2]. Actually formal methods bring rigor to model leading the modeler to got well defined models and make him to feel secure with the system under development. Before making this study, the authors concluded that too much formalization can be a problem, lack of formalization also can be so.

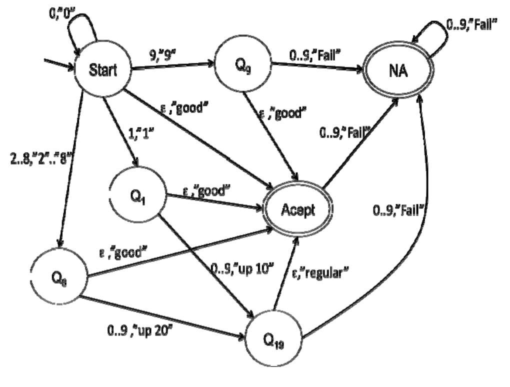
Model Driven Architecture (MDA) is composed by 3 viewpoints: a) Computer Independent Model (CIM); b) Platform Independent Model (PIM); and c) Platform. When business rules are properly defined 1/3 of the system is done [2].

This poster shows an approach on how to systematically refine domain analysis to consider all business rules. It is considered a state machine which represents all Domain aspects. The use of state machines are based on user friendliness and formality.

2. Case study

As Case Study it was developed an automata to sort a range of values aiming to categorize a received result by a numerical string. A main idea is that after the automata processing is finalized, it can decide whether a given range of values is accepted or rejected in accordance with a specific range of values determined by the domain.

Automata was modeled by a Mealy Machine [2] so that the computer can be monitored and so that it can known what should be done after the final state.



Mealy Machine was defined by M=(∑,Q,δ,q0,F,Δ) where:

∑={0,1,2,3,4,5,6,7,8,9}
Q={Start, Acept, NA, Q9, Q1, Q19, Q8}
δ= { (Qx∑) → (QxΔ*)}
q0={ Start }

F={ Accept, NA }
Δ={“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “regular”, “good”, “fail”, “up 20”, “up 10”}

The language part generated by the resulting machine is defined as follows below.

START → 0START | 9Q9 | □Accept | 1Q1 | 2Q8 | 3Q8 | 4Q8 | 5Q8 | 6Q8 | 7Q8 | 8Q8.

The language derived from a Mealy Machine is DSL. With this DSL, it is possible to create a compiler or interpreter of commands that converts into a programming language (general purpose), to ensure that all business rules will always be followed. If the code generated is not in accordance with the business rules described in the case study, the system will not succeed in compilation.

3. Pactical use for DSL teory

A practical experiment to DLS theory is to apply in a decision process. The experiment described in Figure 1 was chosen to be applied in one type of software development life cycle. - SDLC. In that case, a decision process consists in using Software Requirement Specification – SRS information.

The concept and plan test starts with project SRS. In this SRS it were analyzed two types of requirements: functional and no-functional [2]. To prove this hypothesis an algorithmic was identified to the decision process which is the main contribution of this paper.

The use of this model for a decision process was restricted to the algorithm, to choose a manufacture cell SDLC [3]. To understand and to visualize this behavior of this Mealy Machine the Table 1 shown this concepts.

Table 1 shows the decision parameters obtained during the quantitative and qualitative Mealy Machine grammar analysis which is the main contributions of this poster.

Table 1 – Formal approach use to choose software manufacturing cell SDLC.

Formal approach for Mealy machine automata	**Qualitative analysis**	**Quantitative analysis**	
DFA (Σ, Q, δ, q0, F, Δ)	Parameters Functional and non-Functional requirements	High Joining: SDLC Maintance	Low Joining: SDLC Development
Q - States	to alter the numbers of states.		**X**
Σ - Alphabet	to alter the symbols.	**X**	
δ - Transition Function	to alter them you screech.	**X**	
δ - Transition Function	to create new rules.		**X**
q0 - Initial State	to change the initial states.	**X**	
F - Final States Group	to change the numbers of final states.	**X**	
Δ - Exit symbol alphabet	to change the final symbols.	**X**	

4. Recommendations and suggestions

This research develop alternatives to improve aspects of Systematic and Formal Approach to a Domain Specific Language to building software.

The systematic use of the Formal Approach concept, on a large scale of software components is recommended. The extension of se proposals processes and methods are the criterion of interested groups of research in conception tests software. It is suggested that the use of formal tests concepts, in Lines of Production of Manufacture Cells [3] specialized.

A formal approach must be revised, refined, and be customized for each new Line of Production. It is suggested that the application of this Model in Software Houses [3], aiming profits of scale in production of SDLC.

5. Acknowledgments

The authors would like to thank ITA and GPES for support, partnerships with companies of productive sector, as TCS - Tata Consultancy Services, shown in this article, actually represented excellent exchanges of professional and academic experiences.

6. References

[1] OMG, “Model Driven Architecture”, 2003.

[2] Bjorner, D: Software Engineering 3, Domains Requirements and Software Design, Springer, 2006.

[3] Montini, D Á, Modelo de indicadores de risco para o orçamento de componentes de software para célula de manufatura, In Portuguese, UNIP, 2005.