Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 104 (2017) 329 – 337

Procedia Computer Science

ICTE 2016, December 2016, Riga, Latvia

# Simplified Lisp Code Generation from the Two-Hemisphere Model

Konstantins Gusarovs[a,*], Oksana Nikiforova[a], Adrian Giurca[b]

[a]*Riga Technical University, Kalku str. 1, Riga, LV-1658, Latvia*
[b] *Brandenburg Technical University, Cottbus-Senftenberg, Postfach 101344, 03013 Cottbus, Germany*

**Abstract**

Model Driven Software Development (MDSD) is one of the trends for software development that has been widely advertised in the last years proposing automation in the code generation from the system requirements represented in a model form. Several methods involving code generation to some degree of completeness have been proposed, however these methods usually focus on a static aspect of the system being built leaving the dynamics out of the transformation scope. In this paper authors propose an approach to generate simplified LISP code from the two-hemisphere model with the main attention to a dynamic aspect of the system.


Peer-review under responsibility of organizing committee of the scientific committee of the international conference; ICTE 2016

*Keywords:* Two-Hemisphere Model; Model Transformations; Code Generation; Extended Backus–Naur Form

## 1. Introduction

Model-Driven Software Development[1] is a paradigm of software development being focused on the modelling of the problem domain in order to produce an abstract representation of the system (usually in a form of a model of some kind, e.g. business process diagram) that could be later reused[2]. MDSD lifecycle should start with the development of the Computation Independent Model (CIM) that could be also called business or domain model because it uses concepts that subject matter experts are using. CIM should be understandable by the business representatives and should describe what the system is expected to do hiding the implementation-related information.

* Corresponding author. Tel.: +371-29588419.
*E-mail address:* konstantins.gusarovs@rtu.lv


Peer-review under responsibility of organizing committee of the scientific committee of the international conference; ICTE 2016
doi:10.1016/j.procs.2017.01.142

Later CIM is being used to produce the Platform Independent Model (PIM) which in turn exhibits a sufficient degree of the technical information. However, PIM should be usable when developing for the different platforms, which in turn means that the PIM should be abstract enough. To achieve this, PIM is usually being defined in a form of the abstract service set, where each service could be "something" – a class in an object-oriented language, a subsystem, a server in a distributed environment etc.

PIM in turn is being used to produce the Platform Specific Model (PSM) which is a combination of the specification presented in a PIM with the specific platform details. PSM could cover only part of the necessary details leaving others to be covered during the actual development of the software system. In this case PSM is being called abstract.

The main idea about the usage of three different kinds of models is reusability of once produced artifacts, i.e. single CIM model could be used to produce several PIMs which in turn can be mapped to a set of PSMs allowing the wide range of the platforms to run the developed system on.

It is possible to define a single question each kind of MDSD models should answer to:

- CIM – What system should do?
- PIM – How system should achieve its goals?
- PSM – How system should be implemented on a given platform?

MDSD proposes the transformation between the different kinds of the system models which means lesser human interaction in the development process. Transformations could be fully automated, manual or semi-automatic. Lastly, MDSD process should be able to produce the actual code from the system model that could be later completed by the system developers in order to produce the software system. In this paper authors are focusing on the transformation allowing to generate a code from the two-hemisphere model that is being developed in a Riga Technical University since 2004 by the research group lead by Oksana Nikiforova. This paper focuses on a brief explanation of the techniques being used.

This paper is structured as follows. In the section 2 an information about the two-hemisphere model driven approach is given, the section 3 contains a brief description of the related work focusing on the one of the researches that serves as a basis for the described transformation. Section 4 describes the necessary preprocessing to be done with the regular expression serving as an intermediate result before the code generation can be done. Section 5 shortly describes the simplified LISP dialect used in this work while section 6 gives an insight on the developed transformation and marks the main directions for the future work. Finally, conclusion is given in the section 7.

## 2. Two-Hemisphere model driven approach

In 2004 a first version of the two-hemisphere model driven approach was proposed[3]. This approach focuses on a representation of both static and dynamic aspects of a software system. This is being achieved by describing the system using the two models that are being linked together:

- Concept model describes the concepts of the system which are the objects or data types being used. Each concept has a set of attributes that are describing its internal structure. Each of the attributes has its type which in turn could be primitive data type, such as an integer or string as well as another concept or the collection of several same type objects
- Process model is used to define interactions happening in the system. The process model is a set of the processes that are being interconnected with the data flows. Each data flow carries one of the concepts defined in the concept model thus linking two parts of a model together. A process in the process model can be defined either as an internal or external. External processes are defining the points where the software system interacts with its environment and should only produce or consume data flows whilst internal processes are defining the activities inside the system and should simultaneously produce and consume data flows

A valid two-hemisphere model consists of a single concept model and several process models. Each of the process models should describe a single use case or a single activity the system performs. An example of a two-