**1 Introduction and Motivation**

Developing dependable systems is one of the most important targets nowadays. Technologies are involved in our daily life. People use smartphones, laptops, and different vehicles to solve their everyday problems. They rely on their gadgets which sometimes can work improperly and cause troubles. In case when the systems should be reliable and should work stable all the time, software has to be verified. Verified software allows user to be sure that it will work accurately in any situation. There are some tools developed for this purpose. One of them is Rodin Platform, which provides effective support for refinement and mathematical proof [http://www.event-b.org]. Event-B is not the only one language which can be used in verification. This work is dedicated to comparison of two verifying languages – Event-B and Modular Event-B. Practical work based on creating a Plug-in which translates source code in Event-B into Modular Event-B (inverse?). The Rodin Platform is Eclipse-based, so the Plug-in is written in Java. The reason because these two languages have been chosen is that Event-B is quite popular language, but it doesn’t have well-developed modularization constructs and it is not easy to combine specifications in Event-B with those written in other formalisms [Marie’s paper].

**2 Event-B**

Event-B is a notation for formal modelling based around an abstract machine notation [Rodin User’s Handbook]. It allows the user to verify difficult real-life tasks. There are some examples of using the verification in daily life: smart-grid modelling and railway interlocking models. Verification of the systems is used to ensure the safety of people or to avoid costs caused by improper work of the system. The main advantage of using the Event-B is that all development errors in the model can be easily found since in incomplete and inaccurate model some proofs cannot be done.

**2.1 Contexts and machines**

Event-B models consist of two main parts: contexts and machines. Context shows all static parts of the model, while machine represents dynamic parts of the model. These two main parts allows creating efficient models and describing the behavior of the system. The key feature of the Event-B is that the primary model can be really simple, but with the help of the refinements, it could be improving gradually and become complicated enough. The term refinement applies to dynamic parts of the model, so-called machines. The most famous example of the Event-B model is the “Controlling Cars on a Bridge” model. It describes the traffic lights for cars crossing the bridge from mainland to island and vice versa. The first model developed for this study case was really simple, it had island and bridge joint together and only two colors in traffic lights: red and green, while the final model had not only traffic lights, but also car censors and all three parts of the study case – island, bridge and mainland. This example shows the idea of the refinement – gradual improvement of the model using the refinements.

**2.2 Events**

The main part of machines in Event-B is event. At the beginning of development there is just one event in each created machine – INITIALISATION event. This event is used for initializing actions (variables, invariants (?)). No one model can work properly without this event. While developing the final model, different events can be created to describe the model. Each event should describe the one action in real life. In given example “Controlling Cars on a Bridge” there are different events describing “A car is leaving the mainland and entering the Island-Bridge”, “A car leaving the Island-Bridge and re-entering the mainland”, etc. actions. The more precise model becomes the more events it usually includes. Events can have no guards, they can be also simple and guarded (keyword where) or parameterized and guarded (keywords any and where) [2].

**3 Theory of institutions and SBOs**

Despite the fact that Event-B is quite popular language and is used in industry, it has a great disadvantage - it doesn’t have well-developed modularization constructs and it is not easy to combine specifications in Event-B with those written in other formalisms [Marie’s paper]. Modularization constructs are the base of the general theory of institutions. What is institution? The concept of institution is introduced to formalize the informal notion of “logical system”. Institutions enable abstracting away from syntactic and semantic detail when working on language structure “in-the-large”; for example, the language features can be defined for building large structures from smaller ones, possibly involving parameters, without commitment to any particular logical system. This applies to both specification languages and programming languages. Institutions also have applications to such areas as database theory and the semantics of artificial and natural languages [1]. A specification is the main modelling unit in an institution, but specification language is not a programming language, this is a collection of sentences about programs [1]. For a (pure) logical programming language, the specification is also a program [1].

The key concepts in the theory of institutions are:

* A specification is the main modelling unit in an institution.

In terms of Event-B, a specification is referred to as a component: i.e. it is the description for either a machine or a context. In the theory of institutions, a specification consists of a signature along with a set of sentences over that signature.

* The signature of a specification is the set of names used in that specification.

For an Event-B machine this is the set of (global) variables and event names. For an Event-B context this is the set of constant and set names. We don't worry about the names of invariants, guards and axioms, since these are just labels for information and can't be seen by other specification.

* The sentences in a specification are just the predicates that define things.

In Event-B machines these are the invariants, guards and actions; in Event-B contexts they are the axioms.

**References**

[1] J. A. Goguen and R. M. Burstall. Institutions: Abstract Model Theory for Specification and Programming. Journal of the A.C.M., 39(1):95{146, January 1992

[2] J.-R. Abrial. The Event-B Modelling Notation, October 2007