Upgrading EasyTime: from a textual to a visual language

Iztok Fister Jr.,* Tomaž Kosar,† Marjan Mernik,‡ and Iztok Fister§

Abstract

Measuring time in mass sports competitions is usually performed using expensive measuring devices. Unfortunately, these solutions are not acceptable by many organizers of sporting competitions. In order to make the measuring time as cheap as possible, the domain-specific language (DSL) EasyTime was proposed. In practice, it has been proven to be universal, flexible, and efficient. It can even reduce the number of required measuring devices. On the other hand, programming in EasyTime is not easy, because it requires a domain-expert to program in a textual manner. In this paper, the domain-specific modeling language (DSML) EasyTime II is proposed, which simplifies the programming of the measuring system. First, the DSL EasyTime domain analysis is presented. Then, the development of DSML is described in detail. Finally, the DSML was tested by regular organizers of a sporting competition. This test showed that DSML can be used by end-users without any previous programming knowledge.

To cite paper as follows: I. Fister Jr., T. Kosar, M. Mernik, I. Fister, Upgrading EasyTime: from a textual to a visual language, In Proceedings of the 21st International Electrotechnical and Computer Science Conference*, Portorož, Slovenia, 2012.*

*University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: iztok.fister@guest.arnes.si

†University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: tomaz.kosar@uni-mb.si

‡University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: marjan.mernik@uni-mb.si

§University of Maribor, Faculty of electrical engineering and computer science Smetanova 17, 2000 Maribor;

Electronic address: iztok.fister@uni-mb.si

I. INTRODUCTION

The problem of measuring time in sporting competitions is relatively old. Many approaches have been developed to deal with this problem. One of the more efficient solutions was the domain-specific language (DSL) [1][2] EasyTime[6]. The development of EasyTime arose from the need to cover the results of double triathlon competition in 2009, but outgrown limits of this particular triathlon competition. After its first successful use in practice, two demands for the future development of EasyTime were revealed:

- how to satisfy the demands of various competitions,
- how to simplify the handling of the measuring system in order for it also be usable for regular organizers of sporting competitions.

The first demand was satisfied with EasyTime. EasyTime is a small and efficient DSL with high expressive-power. Until now, it has been applied to various sports competitions like triathlons, time-trials in bicycling, cyclo-cross, running, etc. Unfortunately, its widespread usage was limited because of a lack of measuring devices. Specifically, the equipment for measuring times in a swim course is very expensive. Therefore, we concentrated on measuring time in relatively small competitions. It is encouraging that many industries showed an interest in using EasyTime for their measuring systems.

Although the original EasyTime demonstrated itself to be robust and universal in the practice, it still required a domain-specific expert to program the measuring domain. Therefore, the development of a domain-specific modeling language (DSML) EasyTime II was proposed that could satisfy the second demand. Note that DSMLs are currently one of the most interesting research topics in the area of computer languages.

In this paper, the development of an EasyTime II DSML is presented that consists of four core stages: meta-model construction, the definition of a graphical model, obtaining the semantic model, and code generation. In the beginning, the meta-model for our language is defined. Then, the graphical elements are designed that represent the language concepts and are visible to a user on a panel. In the third stage, the semantic model is obtained. Finally, this model is transformed into an executable code (e.g., Java code) with a model-to-code transformation.

The structure of this paper is as follows: in Section 2 we describe the problem of measuring time in sports competitions. Section 3 presents DSML EasyTime II and its development stages from the meta-model to code generation. Section 4 describes the practical example using this DSML. The paper concludes with practical experience and directions for future development.

II. MEASURING TIME IN SPORTING COMPETITIONS

Not long ago, measuring time in sporting competitions was performed manually by people who wrote the results for each competitor on paper and at the end, put the competitors in order according to their achievements. This way of measuring time is impossible nowadays because of the large number of participants. On the other hand, there are many modern sports, e.g., the triathlon, aquathlon, etc. that require a very precise measurement of results. Therefore, there is a significant need for electronic measuring devices. These devices need to work precisely and securely in all weather conditions, i.e., in rain and snow.

"Multi-sport" refers to competitions, where more than one discipline is involved. The most popular multi-sport disciplines are:

- triathlon (consists of swimming, biking and running),
- duathlon (consists of running, biking and running),
- aquathlon (consists of swimming and running),
- winter triathlon (consists of running, mountain biking, cross-country skiing),
- etc...

Moreover, all these disciplines consist of courses with various distances. For example, a triathlon is divided according to distance into: Ironman, Half Ironman (also Ironman 70.3), Olympic triathlon, Sprint triathlon, etc. Measuring time for these multi-sport competitions is more complicated because of their long duration and a number of the competitors.

In Figure 1, the Olympic triathlon is presented. The Olympic triathlon consists of 1.5 km of swimming, 40 km of biking and 10 km of running [4]. Additionally, competitors have to go through two transition areas. In the first transition area, the competitor leaves their

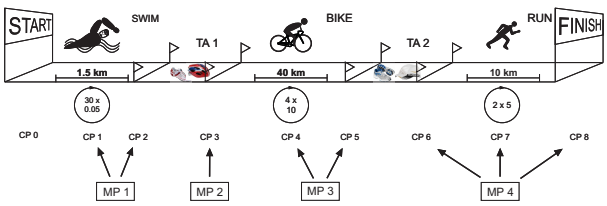

FIG. 1: Olympic triathlon

swimming equipment and prepares for biking. In the second transition area, the competitor has to drop their bike and prepares for running. Every discipline in this triathlon is split into laps. Therefore, in addition to measuring time, it is also necessary to count laps.

On the other hand, "single-sport" competitions consist of only one sport (e.g., running, swimming, cycling, etc.). Measuring time in these competitions is not as difficult as it is in multi-sports.

III. DESIGN AND IMPLEMENTATION OF EASYTIME II

DSMLs have been developed in a number of areas to facilitate the construction of models at a level closer to the conceptual model, thereby making model implementation more accessible to domain experts [5][9][12][13]. DSMLs are a special kind of languages, where the user does not need to write code. These languages go to the 4th generation of computer languages and are currently one of the most interesting topics of research in the area of computer languages. These languages have a bright future, because of their simple usage. The design and implementation of these languages is a bit more complicated and can be split into the following four core stages:

- meta-model construction,
- definition of a graphical model,
- obtaining the semantic model, and
- code generation.

The development of DSML EasyTime II grew out of DSL EasyTime. Note that EasyTime is a little textual language for measuring time in sports competitions, which is very efficient and assures the flexibility of a measuring system. The language is based on the

compiler/interpreter implementation approach [3]. For more information, the design and implementation of DSL EasyTime has been presented in more detail in [6][7][8].

However, each DSL development started with a *domain analysis*, in which the *concepts* of DSL are defined and represented within a *feature diagram*. The feature diagram describes the dependencies between these concepts. Furthermore, the concepts can be broken-down into *features* and *sub-features*. Let us remember the main concept of application domain measuring time in triathlon consists of the following features: *events*, *control points*, *measuring time*, *transition areas* and *agents*. Events arise via different disciplines, e.g., sub-features, *swimming*, *cycling*, and *running*. Each control point is described by its *start* and *finish* time together with the number of *laps* to go. The feature *transition area* can be calculated by the difference between the finish and start times, while the *measuring place* is determined by the sub-features *updating time* and *decreasing laps*. Finally, the feature *agent*, which is dedicated to processing events received from the measuring device, can act either *automatically* or *manually*.

The feature diagram served as a basis for EasyTime II development. Fortunately, this diagram can be incorporated into the Eclipse Modeling Framework tool (EMF) [16]. The feature diagram was used as a reference for the construction of a *meta-model*. Furthermore, the meta-model served as a basis for the Eclipse Graphical Modeling tool (GMF) [14][17], in which the used graphical interface (GUI) is defined. Then, the model transformations must be defined in order to call the domain framework, which is a platform that provides functions to implement the semantics of DSMLs in a specific environment. In order to obtain a *semantic model*, this GUI is mapped to the EMF concepts. Finally, this semantic model is translated into an executable code - Java code, which is executed on a Java Virtual Machine. In the rest of paper, the DSML development stages are described in more detail.

A. Meta-model construction

Meta-modeling is the construction of a collection of concepts within a certain domain presented in a context diagram. A model is an abstraction of phenomena in the real world. That is, the meta-model highlights the properties of the real processes. As a result, the model conforms to its meta-model in the way that a computer program conforms to the grammar of the programming language in which it is written [10][11][12]. In essence, with a

meta-model, a business logic of a process that performs the measuring time in triathlon is described.

EMF allows us to create good meta-models in a very simple way. The meta-model that performs the triathlon presented in Figure 1 is presented in Figure 2. The meta-model was developed in Ecore notation. In Figure 2, it can be seen that this meta-model is a conceptual class diagram (defines a set of concepts in the form of classes together with relations).

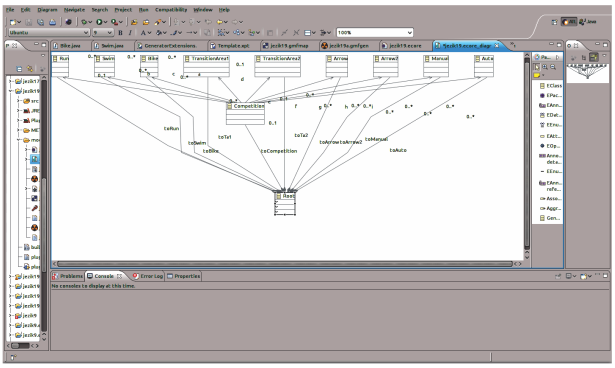

FIG. 2: Meta-model example

This meta-model consists of two main features: root, and competition. These main features are connected with sub features: swim, bike, run, transition area 1, transition area 2, arrow, arrow2, auto and manual. These sub-features are linked with features by a connection. Usually, cardinality 0..* is used. Cardinality 0..* means that we might use this feature exactly 0 or more times in our model.

B. Definition of the graphical model

EasyTime II was modeled using the Eclipse Graphical Modeling Framework (GMF). GMF allow us to create visual aspects of a generated graphical editor [15]. These visual aspects consist of the following definitions:

- graphical definition,
- tooling definition,
- mapping definition.

In a graphical definition, we choose the elements that will be shown to the user. Then, a tooling definition is performed, in which a visual model is defined, i.e., the palette and

toolbox. Finally, the meta-model (business logic) is mapped into a visual model (graphical and tooling definition).

A sample of the graphical model definition for measuring time in sporting competitions is illustrated in Figure 3.

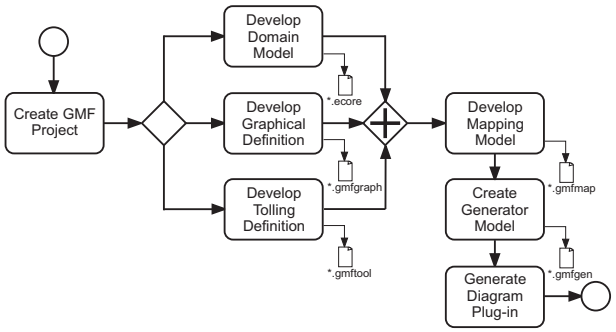

FIG. 3: GMF

In GMF, images are embedded into a modeling environment. These images are created by GIMP and Inkscape editors, and are shown in the model editor. For tooling definition model, the pictures that are shown in the toolbox are created.

C. Obtaining the semantic model

It is not sufficient to complete a DSML definition by only specifying the notions and their representations. The complete definition of DSML requires the semantics of language concepts. Therefore, the abstract syntax defined with Ecore is mapped into the function calls from the measuring time environment. These mappings lead to model transformations that are applied on EasyTime model instances at runtime in order to obtain their counterparts in real EasyTime infrastructures. The model-to-code transformations can be written in MOFScript, where rules contain calls to the domain framework of the EasyTime system.

D. Code generation

Currently, the definition of the model-to-code transformation is still in progress.

IV. WORKING WITH EASYTIME II

In Figure 4, the measuring time for the Olympic triathlon is presented. Users control the measuring with a simple toolbox, where they can choose between the following sports elements:

- swim (the graphical representation of swimming is a swimmer),
- bike (the graphical representation of cycling is a cyclist),
- run (the graphical representation of running is a runner).

Furthermore, users can select between elements that symbolize:

- transition area 1, and
- transition area 2.

For agents, users can select between:

- a manual agent (the graphical representation of the manual agent is a simple clock), and
- an automatic agent (the graphical representation of the automatic agent is a computer).

The arrows symbolize the order in which the sports are performed, and connect the particular sport with the agents.

V. CONCLUSION

In this paper, we presented the design and implementation of DSML EasyTime II. EasyTime II is an extension of DSL EasyTime and was developed to simplify the measuring of time in real sport competitions. In contrast to its predecessor, which required a domain expert to program the measuring system, EasyTime II is dedicated to ordinary users that

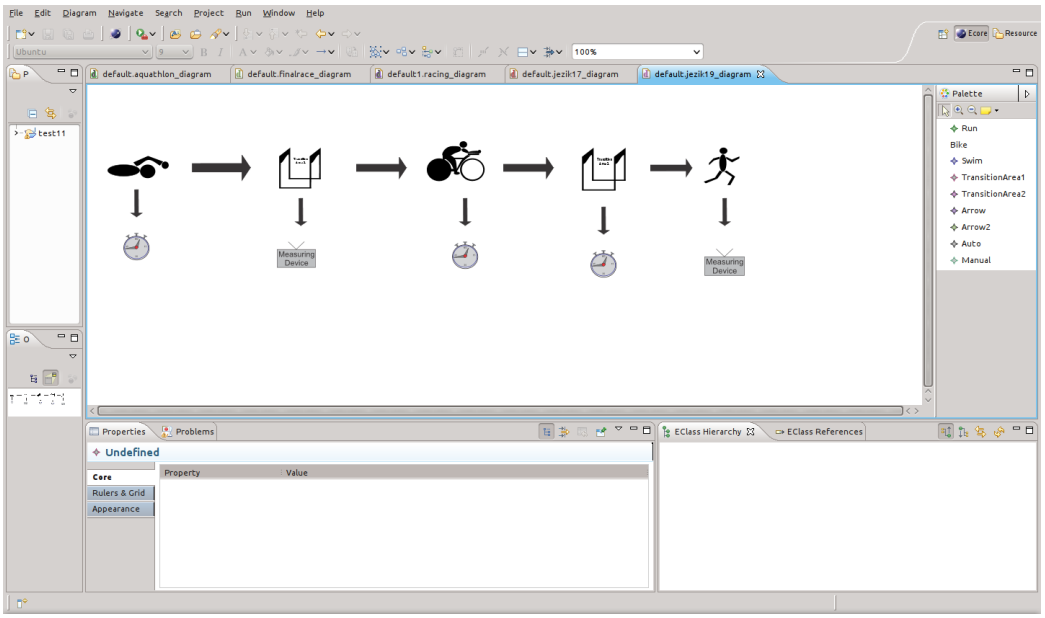

FIG. 4: EasyTime II in action

can control the complicated tasks through easy to use graphical elements. In the future, we intent to measure time in a real-world competition.

---

[1] Mernik, M. and Heering, J. and Sloane, A., *When and how to develop domain-specific languages.* ACM computing surveys, 37(4):316-344, 2005.

[2] Deursen van, A. and Klint, P. and Visser, J., *Domain-specific languages: An annotated bibliography.* ACM Sigplan Notices, 35(6):26-36, 2000.

[3] Kosar, T. and Martinez Lopez, P.E. and Barrientos, P.A. and Mernik, M., *A preliminary study on various implementation approaches of domain-specific language.* Information and Software Technology, 50(5):390-405, 2008.

[4] Petschnig, S., *10 Jahre IRONMAN Triathlon Austria.* Meyer & Meyer Verlag, 2007

[5] Fall, A. and Fall, J. *A domain-specific language for models of landscape dynamics.* Ecological Modeling, 141(1-3):1-18, 2001.

[6] Fister, Jr. I. and Fister, I. and Mernik, M. and Brest, J. Design and implementation of domain-specific language easytime. *Computer Languages, Systems & Structures*, 37(4):151-167, 2011.

[7] Fister, I. Jr. and Mernik, M. and Fister I. and Hrnčič, D. Implementation of the domain-specific language easy time using a lisa compiler generator. In *FedCSIS : proceedings of the*

Federated Conference on Computer Science and Information Systems, pages 809–816, Szczecin, Poland, 2011. Los Alamitos: IEEE Computer Society Press.

[8] Fister I. Jr., Mernik, M., Fister, I., Hrnčič, D., Implementation of EasyTime Formal Semantics using a LISA Compiler Generator. Computer Science and Information Systems, Article in press.

[9] Sprinkle, J., Rumpe, B., Vangheluwe, H. , Karsai, G., *Meta-modeling: state of the art and research challenges.* Proceedings of the 2007 International Dagstuhl conference on Model-based engineering of embedded real-time systems, November 04-09, 2007, Dagstuhl Castle, Germany

[10] Tolvanen, J.-P., Rossi, M., *MetaEdit+: defining and using domain-specific modeling languages and code generators.* Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, October 26-30, 2003, Anaheim, CA, USA

[11] Krahn, H., Rumpe, B., Volkel, S., *MontiCore: Modular development of textual domain specific languages.* In: Paige, R.F., Meyer, B. (eds.) Proceedings of the 46th International Conference Objects, Models, Components, Patterns (TOOLS-Europe), pp. 297-315. Springer, Heidelberg (2008).

[12] Gray, J., Tolvanen, J.P., Kelly, S., Gokhale, A., Neema, S., Sprinkle, J., *Domain-specific modeling.* In: Fishwick, P.A. (ed.) Handbook of Dynamic System Modeling. Chapman & Hall/CRC, Boca Raton (2007).

[13] Czarnecki, K., Eisenecker, U., W.,*Generative programming: methods, tools, and applications*, ACM Press/Addison-Wesley Publishing Co., New York, NY, 2000.

[14] Eclipse GMF, `http://wiki.eclipse.org/Graphical_Modeling_Framework`.

[15] Learn Eclipse GMF in 15 minutes, `http://www.ibm.com/developerworks/opensource/library/os-ecl-gmf`.

[16] Eclipse EMF, `http://www.eclipse.org/modeling/emf/`.

[17] Eclipse GMF, `http://www.eclipse.org/modeling/gmp/`.

Updated 20 August 2012.