Electronic Communications of the EASST
Volume 75 (2018)

43rd International Conference on Current Trends in Theory and Practice of Computer Science - Student Research Forum, 2017 (SOFSEM SRF 2017)

Model-to-Model Transformation in Meta-Modeled CINCO Domains

Dennis Kühn
Email: dennis.kuehn@lero.ie

12 pages

Guest Editors: Anila Mjeda
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Model-to-Model Transformation in Meta-Modeled CINCO Domains

Dennis Kühn
Email: dennis.kuehn@lero.ie

Lero - The Irish Software Research Centre
University of Limerick, Ireland

Abstract: In this paper we present an approach that describes model-to-model transformation from certain graph-based MDD (*Model-Driven Development*) domains into a specific target domain. We present the target domain; DIME, a graph-based MDD suite that is capable to deploy its models as a web application. We then introduce a source domain in which users define image-driven stories as graph models, called *Webstory*, and demonstrate the process for the exemplified model-to-model transformation. The approach allows graph models that are created in their own domain, possibly tailored to one specific purpose, to benefit from DIME's advantages such as, most significantly, deploying a web application with the modeled content.

Keywords: XMDD, Meta-Modeling, Model Transformation, CINCO, DIME, Webstory

1 Introduction

Model-Driven Development (MDD) [BCW12] has become an increasingly wide-spread form of software development that, in contrast to "traditional" programming, requires no deep expertise in programming languages. This makes MDD a valuable approach to bring domain experts and technical experts to the same table during the analysis and development phase of software engineering, bridging the semantic gap that divides the non-technical domain experts, specialized in the target domain (e.g. a hospital, research lab or industry partner), and the implementing technical experts who are unfamiliar with the domain. Harnessing the knowledge of the domain experts and at the same time allowing them to be included as developers is the *Extreme MDD* approach (XMDD) [KJMS09, MS12].

Since modeling tools are to be used by a broad audience with varying technical expertise, they benefit from being as intuitively usable as possible. One way to achieve this is to use a domain-specific editor with a feature set, model representation and views that are all tailored to the needs of the domain for which the software is modeled just as, e.g., a web developer would use an IDE that offers an integrated preview of the modeled website.

Developing a completely different IDE for every new domain at first appears challenging. In practice, utilizing some of the features provided by meta-modeling frameworks such as the *Eclipse Modeling Framework* (EMF) [SBPM08] allow a good part of the tools to be generated, for instance a file explorer, the menu bar and the general interface with modeling canvas and toolbox. Still, many steps remain before generating the target domain-specific editor such as developing the representation of target models and their persistence, devising and constructing

the basic model elements or specifying the scope for the target editor. Specializing on a certain type of editor (e.g. tools that only offer graph-based models) is a way to reduce the complexity of this task in exchange for focusing on a somewhat limited subset of target domains.

CINCO [NLKS17] is a meta-tooling suite developed by the Chair for Programming Systems of TU Dortmund University. It allows to specify a meta-model and generate domain-specific editors based on this description. These generated tools are referred to as "CINCO products" and focus on models as graph-like structures. Throughout this paper, two CINCO products will be used for the model transformation concept: *Webstory* and DIME (*DyWA Integrated Modeling Environment*) [BFK+16].

This approach targets the MDD model transformation into DIME, a tool suite focused on developing and deploying web applications. We chose DIME as the target domain due to its wide array of use cases and its rich feature set specialized in handling web applications without much configuration. The Webstory domain on the other hand is a simplistic editor to design image-driven stories and has the proper complexity to showcase the approach without being convoluted. While Webstory itself is a CINCO product just as DIME is, this does not need to be the case. For an automated transformation of models into a DIME representation, the source domain only needs an API to allow access to its models and their content elements. For this paper, choosing a CINCO product as the presented source domain eliminates the introduction to another API for the source domain.

In this paper, we illustrate the architectural background of CINCO and CINCO products that is relevant for model-to-model transformation into the DIME domain. We then demonstrate how the transformation procedure can be implemented using the example of transforming a webstory to a DIME application.

The overall goal of the approach is to take advantage of already implemented features and established transformations (i.e. DIME to web application) and minimize overhead. As illustrated in more detail in section 2.2, DIME is capable of easily deploying the modeled application as a web application with an extensive tool stack, e.g. out-of-the-box data persistence. As depicted in figure 4, other domains such as Webstory can use this already defined DIME to web application generation by transforming their concrete domain models into DIME.

This way, instead of implementing a Webstory generator that converts and deploys Webstory models to web applications, the much easier route of transforming Webstory model to DIME application can be chosen, making the DIME generator an "Archimedean Point" [SN16] for multiple other domains.

In the following, the respective domains are presented, while section 3 depicts the technical aspects of the domains which are relevant to the transformation process and the implementation.

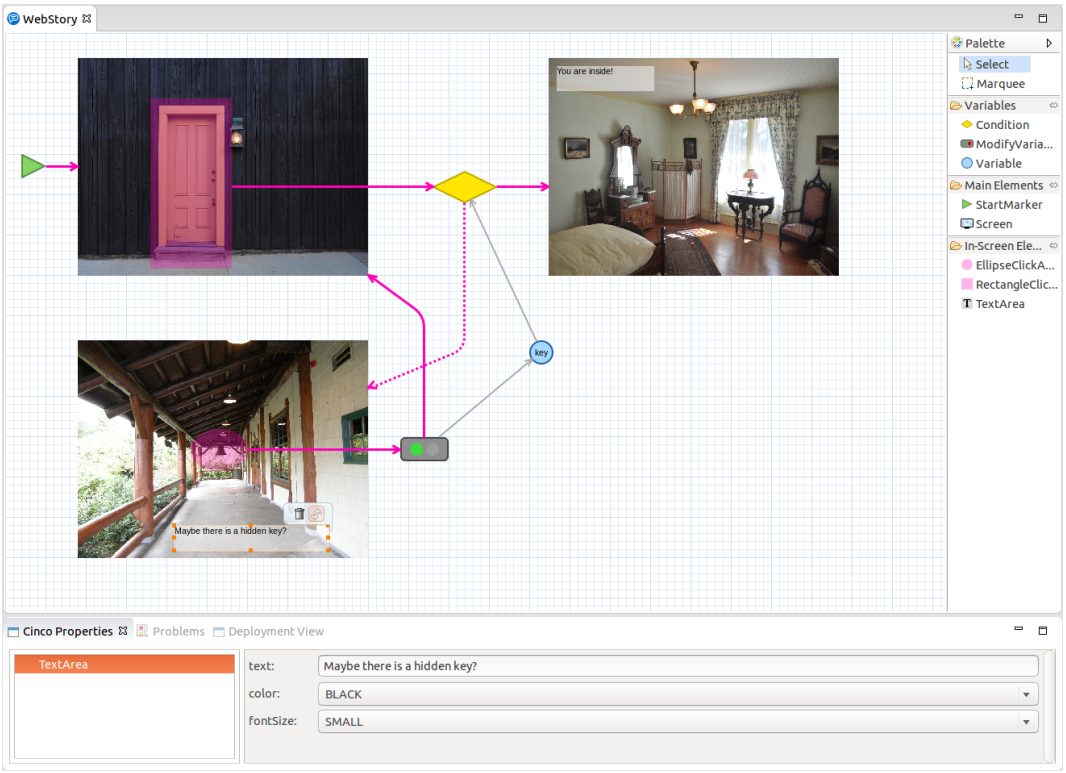

Figure 1: Exemplary webstory using the available basic features, modeled in the Webstory editor

2 The domains and CINCO

2.1 Webstory Editor

The Webstory editor[1] is an example of a tool that is specifically tailored to one domain in order to allow the participation of a broad audience. It is a CINCO product that was created by the CINCO community to be used as a simple example product in workshops on meta-modeling, as well as educational introduction to MDD in general. As such, the Webstory editor only has a limited set of features but in turn was designed to be easily extendable by customizing the meta-model definitions and still contains elements of data flow as well as process flow.

Figure 1 shows a screenshot of the Webstory editor's model canvas containing a modeled webstory, toolbar on the right and CINCO Properties view on the bottom. A webstory model defines a chain of different scenes, similar to a storyboard consisting of multiple storyboard elements. A valid webstory model can be generated to Javascript from within the Webstory editor and then played locally in the browser, allowing the user to navigate through the different scenes. In the Webstory editor, scenes are called *screens* and each screen defines an image that is the background of the correlating scene. Screens can also contain additional elements such as text and interactable areas which, when clicked on in the browser, navigate to a next screen.

[1] Webstory website: https://cinco.scce.info/examples/webstory/

Section 3.1 elaborates more on the technical background of the meta-model of the Webstory editor.

As displayed in the webstory model depicted in figure 1, screens are represented by their chosen background image while the pink areas in the two left screens are clickable areas. The transitions leading from the areas to other model elements define the process flow and can point to other screens or process flow elements, in this case a condition (yellow diamond) and variable modification (green traffic light) nodes. A screen is not limited to one clickable area, and different areas may have different process flow targets.

Beside the process flow, webstories also provide basic data flow elements. Boolean variables (blue circles) can be connected with the modification nodes, which set the state of the variable to either `true` or `false`, or with condition nodes, which switch over the current state of a variable and define the process flow in both the `true` case (straight line) and `false` case (dotted line). With the capability to model decisions like this, webstories can go back to previous screens without resulting in an infinite loop, allowing to model anything from a quiz in the fashion of a multiple-choice gameshow to point-and-click adventure games.

2.2 DIME

DIME (*DyWA Integrated Modeling Environment*) [BFK+16] is a CINCO product that provides a framework for creating web applications. It follows the *One Thing Approach* (OTA) [MS09] in that it – unlike the Webstory editor which has one comprehensive model for each webstory – provides multiple model types for specialized purposes of one central web application. Of the following model types, the GUI and process models are the two relevant kinds for this paper since webstories only use primitive data types which are always included by default. More complex domains would utilize a data model for the definition of complex domain-specific data types and enumerations.

the data model defines the application-specific types and describes their attributes, relations (such as inheritance or uni-/bidirectional associations)

GUI models represent single web pages or page components

process models handle the process flow and business logic

GUI models describe the structure and appearance of individual pages inside the web application. They contain all elements that style or the associated page but can also describe only parts of a page and be used as components in other GUIs. GUI models also have a data binding mechanism and can display and manipulate primitive values and types defined in the application's data model in various ways.

Process models specify the business logic of the application, handling the transition from GUI to GUI – ergo the process flow – and the management of context data. Each process contains a number of activities (these are called *service independent building blocks*, or SIBs) which are connected via process flow transitions. A SIB is an encapsuled component that is a reference to a process model, a GUI model, or a code-level implementation of a service (called a *native SIB* in DIME).

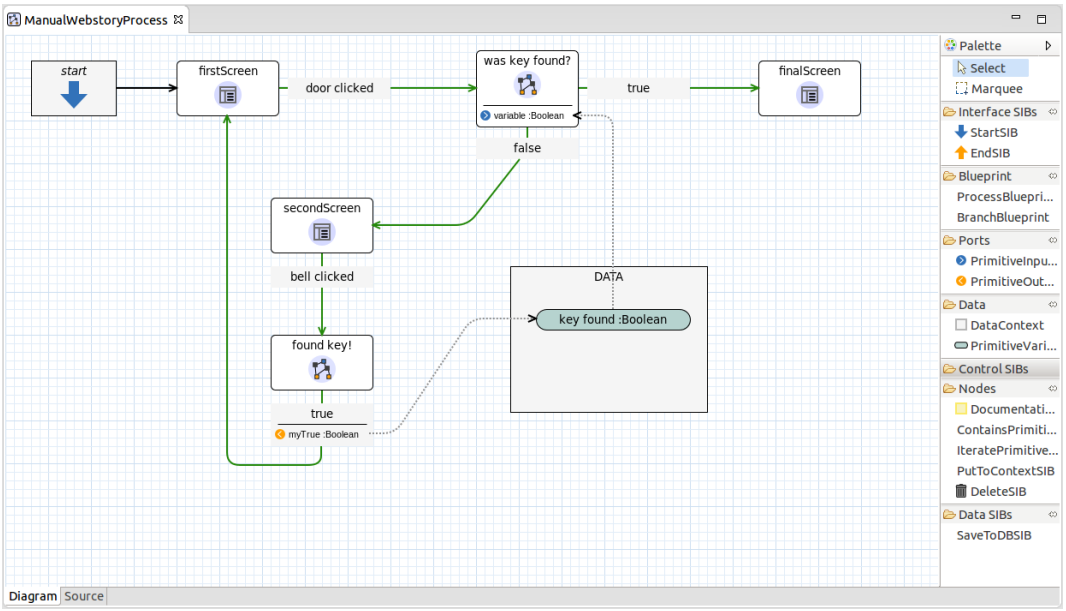

Figure 2: Exemplary DIME process, representing the webstory in figure 1

Figure 2 shows a DIME process model that is the equivalent of the webstory depicted in figure 1. The model contains three elements that each represent separate GUI models for the three screens of the original webstory. The DATA-container manages the variables, namely "key found" of type boolean. The process SIB "found key" sets the variable to true, while the evaluating SIB "was key found?" checks the variable content and defines the process flow for all cases, i.e. true and false. Further technical detail concerning GUI and process models is provided in [BFK+16].

Once the web application is modeled using the three model types, DIME is capable of generating the web application and deploying it using the integrated *Dynamic Web Application* (DyWA) [NFSM14] technology. With DyWA, data persistence and the runtime environment are provided for the target application and are set up automatically from the application specification without further need for configuration.

2.3 CINCO Architecture

CINCO products are specified on the meta-model level using two types of meta-level specifications, the *Meta Graph Language* (MGL) and the *Meta Style Language* (MSL). These two languages are specified in the *Eclipse Modeling Framework* (EMF) [SBPM08] core meta-model.

For each model type that will be available in the target CINCO product, an MGL and MSL must be defined. The Webstory domain, as an example, is specified by a `Webstory.mgl` and `Webstory.style`, while the DIME meta-model definition includes an `.mgl` and `.style` specification for Data, Process and GUI.

The MGL describes the graph structure for the respective model type in the target CINCO product. These graph models consist of elements that are either nodes or edges. The MGL

Figure 3: Excerpt of the MGL and MSL meta-model definition of the process type in DIME

specification defines the permitted node elements for the respective graph model, each of these elements' attributes and the possible outgoing edges.

The MSL specifies the appearance of different node- and edge styles. The MGL then links elements to their style description in the MSL by referring to the corresponding style using the `@style` annotation. Figure 3 depicts two examples of the MGL and MSL definition for a DIME-specific element called `GUISIB`, which is a type of node inside DIME process models. In this example, the MSL definition also requires one additional value, indicated by the `(1)` after the style name, and this value is provided in the MGL `@style` annotation with the `GUISIB` element's value for its `label` attribute.

The tool can be generated once every future model type of the target CINCO product is described by a respective MGL and MSL specification. This produces a meta-model which consists of specific generated implementations for the respective domain, based on the meta-model specification such as the Webstory- or DIME meta-model in figure 4. The generated meta-model can then be deployed to start the target editor suite which allows modeling inside the specified domain.

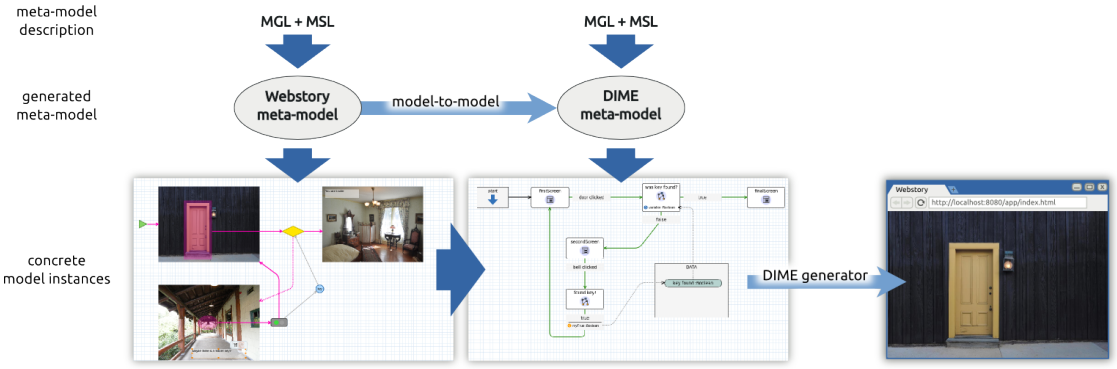

Figure 4: Architecture outline for meta-model and CINCO product level with focus on Webstory and DIME

3 Webstory-to-DIME Generator

The model transformation approach presented in this paper aims to transform models of a source domain into DIME models in order to deploy the resulting DIME project as a web application. To generate these DIME models, first the transformation rules need to be devised. This is done exemplary for Webstory models in section 3.1, since this task is individual to each source domain. The resulting conceptual rules describe how the source models can be translated into DIME models, either manually or when implemented as a generator that can be executed to automate the process.

Generators are defined as part of the meta-level description and use the C-API (CINCO *transformation API*) that is described in section 3.2. To implement a generator, meta-modeler specifies the model-to-model transformation on the generated meta-models of the source domain and DIME (see figure 4), based on the devised transformation rules, as depicted in section 3.3. The domain-specific generator, in this case the `WebstoryToDIMEGenerator`, can then be executed on instances of Webstory models in order to create instances of DIME applications.

3.1 Generator Concept

This paper focuses on the translation from Webstory to DIME to show an exemplary transformation of models from one CINCO product to another. For a generic transformation approach, the key aspects that need to be addressed to transform a source domain model into a DIME application are the **process flow** and a **visual representation** or feedback.

Without any kind of process flow defining an executable sequence, models from the source domain would only be descriptive, for example character relation diagrams or building layouts. These could theoretically become interactive web applications by adding process flow that allows to navigate between entries of a relationship diagram or by defining the transition between certain floors in a building layout, but unless specified in the source domain's representation the resulting DIME application would only be static with no way to switch perspectives or access information other than the default representation.

The need for a visual representation of the central aspects of the models is trivial; it must be possible to display information that the models hold. Even if the models do not inherently require any graphical representation – as the Webstory domain does – but serve to solve a problem like Dijkstra's algorithm for shortest path or other graph-theoretical questions, some form of feedback must be delivered to the user.

In the example of the Webstory domain, models have both characteristics as distinct features: the graphical representation is clearly recognizable as each screen symbolizes and describes the appearance of one scene, while the process flow is defined by the clickable areas on said screens and defines the transitions between the screens. An additional aspect of Webstory is the manipulation of data which happens on primitive boolean values but shows how data flow must be addressed during the transformation as well.

To prepare the model transformation the primary consideration is how the information will be presented in the target domain models. This was done for Webstory by taking into consideration all of the possible node and edge types of a model and matching the behaviour to DIME

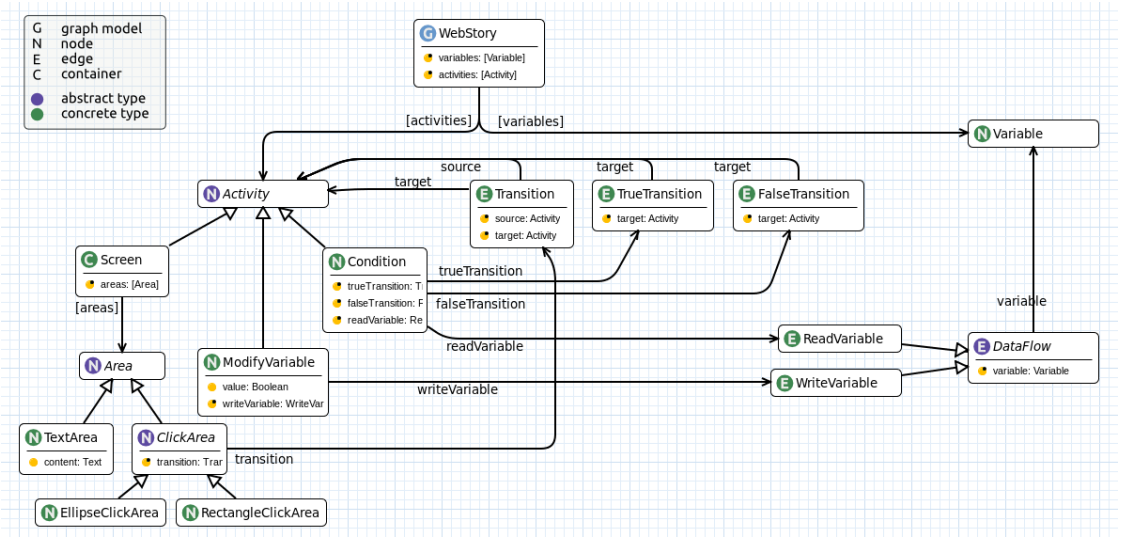

Figure 5: Relation diagram of webstory types in the meta-model

components. Figures 5 and 6 correlate the relevant node and edge types for this transformation.

The structure that allows to progress through a webstory is represented by a DIME process containing the business logic as well as the (in Webstory very limited) data flow of the webstory model. This global process will be referred to as "interaction process", which is the process type of the overarching process model. Every other transformed element of the webstory will be added to the interaction process.

Webstory screens represent individual "pages" of the webstory end product, which correlates with GUI models in DIME. For every screen, a new GUI model needs to be created, yielding the background image and all areas contained in a screen. Areas will be represented by GUIElements; TextAreas simply display text content on a GUI model, while ClickAreas can be represented by Buttons with various CSS attributes for elliptic appearance.

Variables in the Webstory domain are only primitive booleans. For this reason, the target web application does not need to define complex types to support the use of variables, and DIME's default data model is sufficient, but for more complex source domains the data model would be enriched by relevant types with their respective attributes. Primitive boolean variables can be added to a data context in the interaction process.

Conditions and variable modification nodes are not directly correlating to any type in DIME but can be represented by sub-processes. These process SIBs must be predefined as process models and stored to a resources folder for the Webstory meta-model. This allows the generator to access the SIBs as assets and copy the process models into the CINCO products target directory when needed, i.e. when generating a webstory to DIME.

3.2 CINCO transformation API

Generating the meta-model of a specific CINCO product creates implementations of the previously specified meta-model definitions for for every model type, including the nested node- and

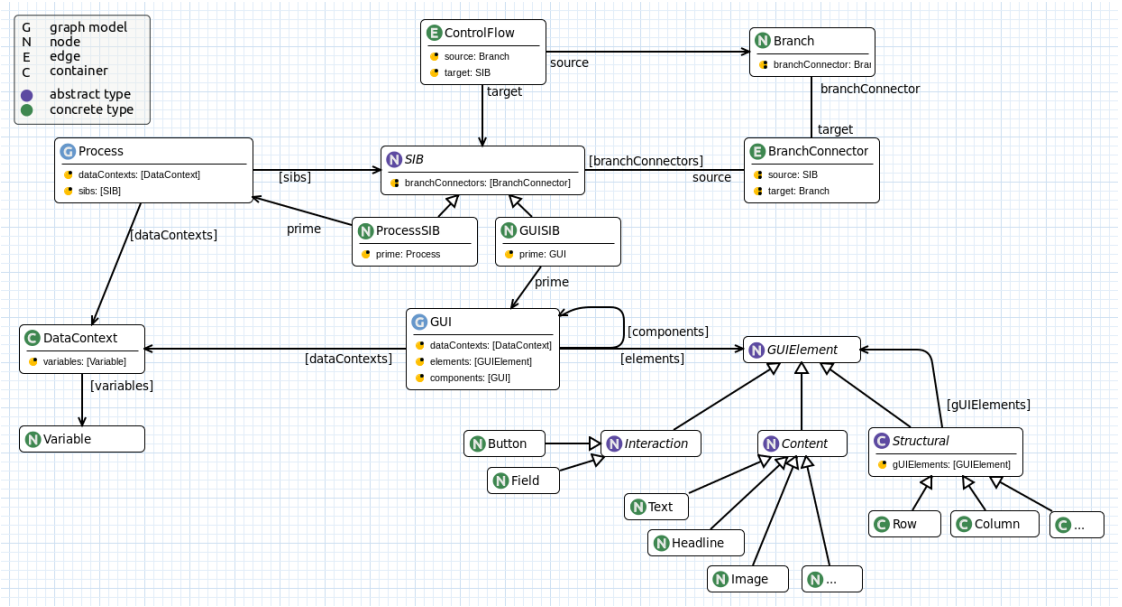

Figure 6: Simplified relation diagram of DIME types in the meta-model

edge types they define. The generated output from the MSL and MGL represent the information given in the meta-model description in separate classes. For DIME the generated meta-model has the implementations `ProcessDiagram` which is generated from the MSL and `Process`, constructed from the MGL to hold the business object information (see figure 7).

The CINCO *transformation API* (C-API) provides an interface to modify and access the essential information of the Diagram and Business model, and concentrates the two representations into one object. This enhances procedures such as adding new elements to the existing process, as the `CProcess` offers methods to directly add elements that the `Process` is capable of holding. To access the C-API, each model type of the target domain has a wrapper class that is also generated during the meta-model generation. This way, the `ProcessWrapper` encapsulates the `Process` and `ProcessDiagram` to provide utility methods for working with DIME processes.

The `WebstoryToDIME` generator creates and manipulates concrete instances of DIME models using the C-API. These models are accessed via the wrapped C-API objects but exist as actual domain-models in the CINCO product project.

3.3 Implementation

When the `WebstoryToDIMEGenerator` is executed for any Webstory model, it is called with the instance of that `WebStory`, the source business model[2] for the specific webstory, and

[2] Note that the graphical information concerning the webstory components are irrelevant, which is why the business model suffices. Otherwise, the `WebStoryWrapper` could be utilized to access the C-API for the webstory

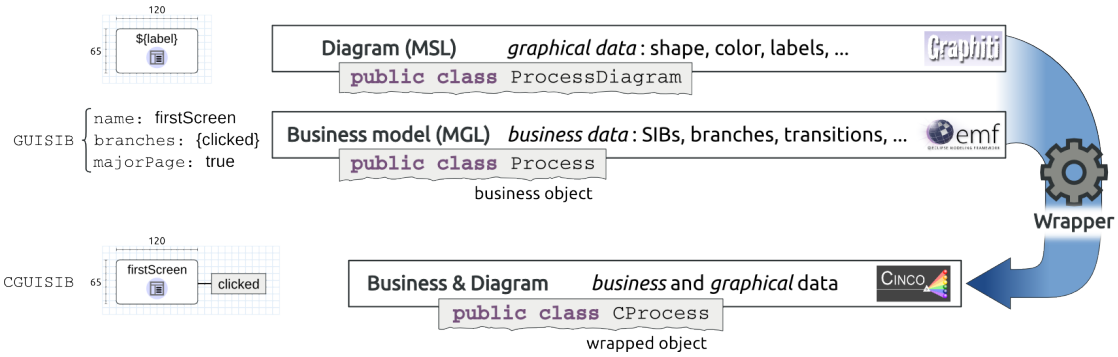

Figure 7: Architecture of the CINCO meta-model that encapsulates business model and diagram of a domain-specific type (here: DIME process) using the C-API

a reference to the resource that represents the Eclipse project in which the model is located.

The structure of the generator can be summarized by these steps:

1. create the main interaction process
2. copy resources from the meta-level to the CINCO product's project location
3. for every node element in the source webstory create the corresponding element in the interaction process
4. create transitions between the newly created elements based on edges in the source webstory
5. save interaction process to project location

Creating the interaction process in step 1 uses the C-API by invoking the `ProcessWrapper`, referencing the target project and specifying the process model name:

```
CProcess interactionProcess = ProcessWrapper.newProcess(
    location, "WebstoryMainInteractionProcess.process");
```

Step 3 transforms every node element of the webstory model based on the approach presented in section 3.1. As one example, the following excerpt transforms a screen into a `CGUISIB`:

```
String guiModelName = convertToName(screen.getId())+".gui";
CGUI guiModel = GUIWrapper.newGUI(location, guiModelName);
guiModel.setGeneralStyle(setBackgroundImage(screen.getBackgroundImage
    ()));

for (EObject subElement : screen.eAllContents) {
        addSubElement(guiModel, subElement);
}

CGUISIB guiSIB = process.newCGUISIB(guiModel.getModelElement(),
    getXCoord(screen), getYCoord(screen));
```

```
process.save();
return guiSIB;
```

This first creates a new GUI model, adds elements for the areas of the screen and then creates the `CGUISIB` that is directly added to the interaction process using the C-API.

4 Conclusion and Outlook

In this paper we presented an approach to transform models of MDD domains into DIME web applications by introducing the specification of cross-product generators to the meta-model definition of the source domain. Meta-modelers are able to automate model transformation of the specific domain to the DIME domain using the C-API to create and manipulate model instances in the target domain.

We demonstrated the transformation of models from one domain into another for the specific domains of Webstory and DIME. For this purpose we elaborated on the generator approach in general as well as the transfer of Webstory-specific model elements into a DIME web application.

While in this paper we showcased the transformation from a CINCO product to DIME to demonstrate that the C-API can be used to access DIME model creation, this approach can also be applied on non-CINCO transformation cases. This includes domains with more complex data types than the primitive boolean values, which can be handled by utilizing the C-API to access DIME's Data model. Source domains are however required to specify a process flow that can be recreated in the target DIME model as well as a visual representation to define the appearance of the modeled content. For an automated transformation it is further required that the source domain offers an API to access the information its models hold.

With the transformation of domain-specific MDD models to a DIME application, we facilitate the possibility to benefit from all the features that DIME has to offer: the data management inherent through DyWA as well as the feature support for more complex background services, responsive behaviour of the web application and the easy deployment.

For the example of Webstory, there are also vast opportunities to enhance the transformed DIME representation of the webstory by features such as savegames (becoming an option through data management as well), allowing access from different devices since the web application can easily be hosted anywhere and not only deployed locally, or multiplayer gaming. In fact, these features could become future elements of the Webstory editor as well and could be automatically set up in DIME through the generator. Another possibility is to allow for templates to be created so that – for example – the content of screens can be modeled to follow themes that the user can define. This would allow to more easily specify and use a unified appearance for the entire webstory.

In conclusion, the model-transformation approach offers a wide range of opportunities and can be used to benefit from the inter-domain advantages to reuse existing tool stacks.

Bibliography

[BCW12] M. Brambilla, J. Cabot, M. Wimmer. *Model-Driven Software Engineering in Practice*. Morgan & Claypool, 2012.
doi:10.2200/S00441ED1V01Y201208SWE001

[BFK+16] S. Boelmann, M. Frohme, D. Kopetzki, M. Lybecait, S. Naujokat, J. Neubauer, D. Wirkner, P. Zweihoff, B. Steffen. DIME: A Programming-Less Modeling Environment for Web Applications. In *Proc. of the 7th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation, Part II (ISoLA 2016)*. LNCS 9953, pp. 809–832. Springer, 2016.
doi:10.1007/978-3-319-47169-3$_6$0

[KJMS09] C. Kubczak, S. Jörges, T. Margaria, B. Steffen. eXtreme Model-Driven Design with jABC. In *CTIT Proc. of the Tools and Consultancy Track of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA)*. Volume WP09-12, pp. 78–99. 2009.

[MS09] T. Margaria, B. Steffen. Business Process Modelling in the jABC: The One-Thing-Approach. In Cardoso and Aalst (eds.), *Handbook of Research on Business Process Modeling*. IGI Global, 2009.

[MS12] T. Margaria, B. Steffen. Service-Orientation: Conquering Complexity with XMDD. In Hinchey and Coyle (eds.), *Conquering Complexity*. Pp. 217–236. Springer London, 2012.
doi:10.1007/978-1-4471-2297-5$_1$0
http://dx.doi.org/10.1007/978-1-4471-2297-5_10

[NFSM14] J. Neubauer, M. Frohme, B. Steffen, T. Margaria. Prototype-Driven Development of Web Applications with DyWA. In *Proc. of the 6th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation, Part I (ISoLA 2014)*. LNCS 8802, pp. 56–72. Springer, 2014.
doi:10.1007/978-3-662-45234-9$_5$

[NLKS17] S. Naujokat, M. Lybecait, D. Kopetzki, B. Steffen. CINCO: A Simplicity-Driven Approach to Full Generation of Domain-Specific Graphical Modeling Tools. *Software Tools for Technology Transfer*, 2017. to appear.
doi:10.1007/s10009-017-0453-6

[SBPM08] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks. *EMF: Eclipse Modeling Framework (2nd Edition)*. Addison-Wesley, Boston, MA, USA, 2008.

[SN16] B. Steffen, S. Naujokat. Archimedean Points: The Essence for Mastering Change. *LNCS Transactions on Foundations for Mastering Change (FoMaC)* 1(1):22–46, 2016.
doi:10.1007/978-3-319-46508-1$_3$