Platform-independence in Model-based Multi-device UI Development

David Raneburger[1], Gerrit Meixner[2] and Marco Brambilla[3]

[1]Institute of Computer Technology, Vienna University of Technology, Gusshausstr. 27-29, 1040 Vienna, Austria

[2]Faculty of Computer Science, Heilbronn University, Max-Planck-Str. 39, 74081 Heilbronn, Germany

[3]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy

Keywords: Multi-Device, Platform, Model-driven Development, Model-based Development, User Interface, Model Driven Architecture, Cameleon Reference Framework.

Abstract: Platform-independence of a model clearly means that a certain model does not depend on characteristics of a certain platform. The crucial issue in this definition to make it concise is: what is a platform? The answer to this question is important as such a platform definition defines which characteristics must not be considered in platform-independent models. This paper compares the notion of platform and the corresponding implications in the Model Driven Architecture proposed by OMG and the Cameleon Reference Framework, a framework that has been developed to classify model-based user interface generation approaches. In particular we compare the implications of platform-independence in the context of different model-based user interface development approaches that support multi-device UI development.

1 INTRODUCTION

One of the main promises of model-based software development is to allow for building higher-level models that do not require the designer to specify all implementation details at the beginning of the development process. Subsequently, such higher-level models can be transformed to implementations for different platforms, with the main intention to save development time and effort. The term platform is overloaded with different meanings, which may lead to confusions if no concise specification is provided for a certain development context.

The platform definition provided by the Model-Driven Architecture (MDA) (Miller and Mukerjij, 2003) has a very wide scope, as it has been designed to support model-based application development for a wide range of different application domains. It does not require the explicit consideration of hardware characteristics. An MDA compliant platform model may therefore consider hardware and software characteristics, or software characteristics only.

If the MDA platform definition is applied in the context of Model-based User Interface Development (MBUID), a platform can be interpreted as user interface toolkit (Truyen, 2006). Hence, MDA compliant transformation approaches support multi-modal user interface (UI) development. Such multi-modal UIs typically combine different modalities (e.g., graphical, speech, or gesture UIs) to allow for a more natural and more robust interaction. Graphical User Interfaces (GUIs) are special in comparison to serial modalities like speech or gesture, as they allow for parallel information exchange. GUIs shall fit the screen of a certain device to achieve a good level of usability. If hardware characteristics like screen size are not considered in the platform model they have either to be considered by the transformations, or during the creation of the high-level model that specifies the flow of information (e.g., which information is exchanged in parallel).

Models that implicitly consider hardware features of the target platform are still platform-independent models according to the MDA definition, as they do not consider software features of the target device, but they do not support multi-device UI development. For supporting multi-device UI generation, platform-independence must include independence of a certain software *and* independence of a certain hardware. A *device* is thus specified through a platform definition that considers software and hardware characteristics. Such a platform definition is provided by the Cameleon Reference Framework (CRF) (Calvary et al., 2003) that support the classification of UI generation approaches and their models in the context of MBUID.

This paper recaptures the MDA and CRF platform definitions, showing that both definitions support the inclusion of hardware characteristics in addition to software characteristics in the platform model. We show that such a platform definition supports model-based multi-device UI development and illustrate its implications on the involved models and the transformation approach through testing a one-to-one correspondence between MDA models and CRF levels. To test this correspondence, we classify the models of six existing model-based UI generation approaches that support multi-device UI generation. In particular we compare the recently adopted Object Management Group standard IFML[1](Interaction Flow Modeling Language), four task-based approaches that are well established in the MBUID community (Meixner et al., 2011b) and contribute to a currently proposed W3C standard[2], and a Communication-Model-based transformation approach that supports automated GUI optimization for different devices (Raneburger et al., 2011).

2 BACKGROUND

This section presents the platform definitions of the MDA and the CRF together with the context in which they have been developed.

2.1 Model Driven Architecture

The Object Management Group's (OMG) MDA (Miller and Mukerjij, 2003) distinguishes three different types of models that reside on different levels of abstraction. These levels are (from abstract to concrete):

1. the Computation Independent Model (CIM),
2. the Platform Independent Model (PIM) and
3. the Platform Specific Model (PSM).

The *Computation Independent Model* is on the highest level of abstraction and therefore also platform independent. It describes the usage scenarios in which the system will be used, specifying exactly what the system is expected to do. These models are sometimes referred to as domain, business, or requirement models in the context of MDA.

The *Platform Independent Model* describes the system to be built, without specifying details on the implementation platform that will be used. It will be suited for a particular architectural style, but can be mapped onto different platforms. Requirements specified through a certain platform model must not be considered in a PIM.

The *Platform Specific Model* specifies how a system is implemented upon or uses a particular platform. This model needs to specify all details necessary to derive the *Implementation* of the system.

MDA applies *Model Transformations* to transform PIMs to PSMs. Such transformations need to provide the additional information required to produce the PSM from the PIM. Sometimes, no transformation between CIM and PIM is possible, as CIMs may not be specified in a formal way, or could describe completely manual behaviors, not implemented on any platforms.

Platform is probably the most fundamental concept, as the MDA promises of resilience to technology obsolescence, rapid portability, increased productivity, shorter time-to-market, consistency and reliability of produced artifacts (Truyen, 2006) are based on the abstraction from a certain platform.

According to MDA, *platform* is defined in the following way:

> *"A platform is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented (Miller and Mukerjij, 2003)."*

A platform is represented as a *platform model*, which is defined as:

> *"A platform model provides a set of technical concepts, representing the different kinds of parts that make up a platform and the services provided by that platform. It also provides, for use in a platform specific model, concepts representing the different kinds of elements to be used in specifying the use of the platform by an application. A platform model also specifies requirements on the connection and use of the parts of the platform, and the connections of an application to the platform. A generic platform model can amount to a specification of a particular architectural style (Miller and Mukerjij, 2003)."*

The MDA platform and the platform model definition do not distinguish hardware and software of a system explicitly. Thus, such models may consider hardware characteristics in addition to software char-

[1]http://www.omg.org/spec/IFML/

[2]http://www.w3.org/wiki/Model-Based_User_Interfaces

acteristics, but the corresponding definitions do not enforce their consideration.

2.2 Cameleon Reference Framework

Model-based UI Development uses models to specify all aspects that are involved in the development of user interfaces. MBUID approaches typically refine high-level interaction models over different levels of abstraction to source code that represents the UI. The Cameleon Reference Framework (CRF) supports the classification of UIs that support multiple targets, or multiple contexts of use in the field of context aware computing (Calvary et al., 2003). It additionally introduces the notion of plasticity. *Multi-targeting* focuses on the technical aspects of context adaptations, while *plasticity* provides a way to qualify the usability of an adapted UI. Thus, a plastic UI preserves its usability on all targets. The CRF is an established approach in the MBUID community and can be used to specify methods and tools (for run- and design-time UI generation), with the intention to ease understanding and comparing them.

In particular, the CRF introduces four levels of abstraction. These are (from abstract to concrete):

1. Tasks & Concepts,
2. Abstract User Interface (AUI),
3. Concrete User Interface (CUI) and
4. Final User Interface (FUI).

The *Tasks & Concepts* level contains interaction models, specifying the interaction between the user and the system to be built, and models of the domain of activity (Vanderdonckt, 2008).

The *Abstract User Interface* level typically contains a presentation and a dialog model that render the domain concepts into canonical expressions that are independent from any concrete interactors available on a certain platform.

The *Concrete User Interface* level contains models in which the canonical expressions have been replaced through concrete interactors that specify the look and feel of the user interface, but are still independent from a certain toolkit.

The *Final User Interface* represents the UI source code that can be compiled and run.

MBUI typically applies model-to-model transformations between the upper three levels of abstraction and model-to-code transformations for concrete to final UI transformations.

The term *platform* is here tailored to UI development and defined as part of the *context of use*, together with the *user* and the *environment*[3]. The platform consists of:

> *"a set of hardware (e.g., processor, screen, and mouse) and software resources (e.g., operating system, technological space) that function together to form a working computational unit whose state can be observed and/or modified by a human user. Single resources (processor, peripheral devices etc.) are unable, individually, to provide this functionality. A platform may be either elementary or form a cluster of platforms. Synonyms: target platform."*

The CRF platform definition explicitly distinguishes hardware and software of a computational unit. The consideration of both, hardware and software characteristics, is important in the context of UI development as it allows for tailoring a UI to a certain device, and thus to achieve a good level of usability.

3 COMPARISON OF MDA AND CRF ON PLATFORM-INDEPENDENCE IN MODEL-BASED UI DEVELOPMENT

A platform definition that shall support multi-device UI development needs to consider software and hardware characteristics of a device. Considering this information in the platform definition implies that it does not have to be considered in other high-level application models. Encapsulating this information in the platform model therefore supports the creation of platform-independent high-level models, as the platform information is added during their transformation to PSMs.

An MDA platform specifies a coherent set of functionalities through interfaces and usage patterns. Examples are operating systems, programming languages, databases, middleware solutions or user interfaces (Truyen, 2006). The MDA platform definition is very generic and does not inhibit its interpretation as depending on software alone or on software and hardware together (e.g., a smartphone with a certain operating system and physical characteristics like memory or screen size).

It is recommendable to refine the MDA platform definition for a certain application domain to facilitate its applicability and avoid misunderstandings.

[3]see also `http://www.w3.org/wiki/Model-Based_User_Interfaces`

The CRF platform definition can be seen as such a refined definition for the MBUID domain. A CRF platform consists of hardware (physical) properties (e.g., screen size and resolution, supported interaction modalities) and software properties, meaning toolkits that implement a certain modality (e.g., Java Swing or HTML). A platform definition that contains software and hardware characteristics supports multi-device UI development and allows to establish a clear one-to-one correspondence between MDA models and CRF levels. This one-to-one correspondence is shown in Table 1 and was most probably intended when the CRF was defined, as the MDA definition was already available and established then. Additionally, it has already been used during the initial development of the UsiXML approach that is also presented in the next section (Vanderdonckt, 2005).

Table 1: Correspondence Scheme.

MDA		**CRF**
CIM	↔	Tasks & Concepts Level
PIM	↔	AUI Level
PSM	↔	CUI Level
ISM	↔	FUI Level

High-level interaction models, typically used to specify the interaction between the user and a system, together with models that specify the concepts of a certain application domain (i.e., domain models) are CIMs. Such models reside on the tasks&concepts level and can be used as a basis for multi-device UI development if they do not consider any platform characteristics. There is no need to consider hardware and software characteristics of a certain platform in the models used on the tasks&concepts level if these characteristics are specified in the platform model.

A PIM that is derived from a CIM is still platform-independent. Thus AUI models derived from platform independent models that reside on the tasks&concepts level correspond to PIMs.

Platform specific information is added during the transformation of an AUI model to a certain CUI model, which means that PSMs reside on CUI level.

Finally the PSM is transformed to the source code that implements the UI. Thus, the implementation (or Implementation Specific Model (ISM)) corresponds to the FUI.

So, what does this assignment imply for models on Tasks&Concepts level and implicitly also for AUI models derived from them? Models on these levels must not restrict the rendering possibilities that are supported through the platform model.

The implication for the domain model is that it needs to define all concepts of the application domain, regardless whether they are used by a certain model (that may already be device-dependent).

The implication for the high-level interaction model is that it must not constrain the amount of exchanged information (e.g., based on the available screen space). This means that all information that can be exchanged at a certain point in time needs to be modeled as concurrently available.

The same implication as for the high-level interaction model is valid for the abstract user interface model. An AUI model still needs to specify all canonical expressions that render the information specified as concurrently available in the high-level interaction model, as part of the same presentation (i.e., presentation or dialog model unit).

In terms of GUI development this means that both, the high-level interaction models and AUI models such models, assume a "potentially infinite" screen. Specifying information as concurrently available on Tasks&Concept and on AUI level allows for splitting it to different (smaller) screens according to platform constraints on concrete user interface level. Tailoring the UI to fit a limited screen on CUI level allows for top-down multi-device UI generation (i.e., starting from the Tasks&Concept or AUI level). Doing it the other way round (i.e., combining information bottom up from the CUI level) is hard to achieve as it requires the analysis of dependencies between the exchanged information to detect which information can be exchanged in parallel on a device with a larger screen.

The remainder of this paper tests whether existing MBUID approaches support multi-device UI generation and discusses whether the meta-models of the corresponding models can be assigned according to the one-to-one correspondence scheme presented in Table 1.

4 CLASSIFYING MBUID APPROACHES AND THEIR MODELS

Model-based UI development approaches typically support multi-platform development. Their presentation, however, frequently focuses on the involved models and does not provide a clear definition of the platform model used, which has already been raised as an issue at the Model-Driven Development of Advanced User Interfaces Workshop (Van den Bergh et al., 2010). In this section we classify the models of six MBUID approaches that support multi-platform development to test the correspondence scheme introduced in Table 1. Table 2 lists the approaches and assigns their models to MDA levels using the CRF platform definition. The remainder of this section summarizes each approach and discusses the assignment of Table 2 in detail.

4.1 Interaction Flow Modeling Language (IFML)

The Interaction Flow Modeling Language[4] (IFML) supports the creation of visual models of user interactions and front-end behavior in software systems, independently of a certain execution platform and has been adopted as a standard by the OMG[5] in March 2013. IFML is a PIM-level language in MDA parlance, and it perfectly fits into the AUI level of the CRF. The Business Process Model and Notation (BPMN) language may be used in the context of IFML to provide CIM models. The Web Extension of IFML, called WebML[6] (Ceri et al., 2009) extends the general purpose IFML concepts with some more precise UI characteristics, considering the peculiarity of the user interaction on the Web, while still keeping a platform-independent vision (both in terms of independence from software and hardware features). WebML also includes a presentation model that covers the PSM level. The WebRatio tool-suite automatically generates industrial-strength running applications (Acerbis et al., 2008) from WebML/IFML models, exploiting the presentation model at the PSM level. The IFML standardization document includes a set of guidelines for the mapping of IFML models to PSMs, namely software platform models, such as Java, .Net WPF, and HTML. IFML supports multi-device UI generation through the definition of rules at the PIM level for self-adaptation of user interfaces depending on the device, screen size, or location. It also allows to incorporate platform-specific aspects in the transformation towards the PSM level. IFML therefore satisfies the correspondence scheme shown in Table 2.

4.2 TERESA

TERESA (Transformation Environment for inteRactivE Systems representAtions) uses a so-called "One Model, Many Interfaces" approach to support model-based GUI development for multiple devices from the same ConcurTaskTree (CTT) model (Mori et al., 2004). This approach defines a platform as *"a class of systems that share the same characteristics in terms of interaction resources (e.g., the graphical desktop, PDAs, mobile phones, vocal systems). Their range varies from small devices such as interactive watches to very large flat displays (Mori et al., 2004)"*. The corresponding transformation method requires the manual refinement of the platform-independent CTT model to different platform-dependent System Task Models (e.g., a desktop, a cellphone or a voice System Task model, still specified in CTT). This already platform-dependent CTT model is subsequently refined to an AUI, a CUI and finally the FUI for the corresponding platform.

The System Task Models that are derived from the platform-independent task model are platform dependent, but are still assigned to the tasks&concepts level of the CRF. This means that the correspondence scheme defined in Table1 is not valid for this approach, but its models are rather assigned to the MDA models as specified in Table 2. The reason is that the transformation approach requires the System Task Models already to take platform specific information into account. This allows for a more straight forward transformation approach, but reduces the re-usability of involved models to the topmost CTT models.

TERESA also shows that models and transformation method are closely intertwined. In general, it is not feasible to assign a certain meta-model to a certain MDA model type or MDA models to a certain CRF level. TERESA uses CTT instances on CIM, PIM and PSM level for example and several PIMs that reside on different CRF levels (see Table 2).

4.3 MARIA

A more recent approach based on CTT models uses the MARIA language to specify the AUI and CUI model (Paternò et al., 2009). This approach specifies platform according to the CRF definition and the corresponding transformation method supports the generation of multi-device UIs based on annotated Web-services. Platform-specific information does not have to be provided in the AUI model, but is encapsulated in the Web-service annotations. This means that the AUI and the corresponding CTT model are platform-independent and can be used for multi-device UI generation. Table 2 shows a one-to-one correspondence between MDA and CRF levels.

It is noteworthy that MARIA does not annotate the high-level interaction model (i.e., the CTT model), but rather the Web-service specification that provides the functional description of the UI. The reason is that the such an annotated Web-service specification provides the basis for multi-modal multi-device UI generation in the context of MARIA and not the CTT

[4]`http://www.ifml.org/`

[5]`http://www.omg.org/spec/IFML/`

[6]`http://www.webml.org`

Table 2: Classification of UI generation approaches based on the CRF platform definition (i.e., device).

	IFML	TERESA	MARIA	MBUE	UsiXML	UCP
CIM	BPMN	CTT	CTT	UseML	Task Model	Communication Model
PIM	IFML (and WebML)	CTT	MARIA	DISL	AUI	UI Behavior Model
PSM	Presentation + SW Model	CTT/AUI/CUI	MARIA	UIML	CUI	Screen Model
ISM	FUI	FUI	FUI	FUI	FUI	FUI

model. The CTT model for a certain device can be derived automatically from this specification with the corresponding tool support (MARIAE [7]).

4.4 Model Based Useware Engineering (MBUE)

The Model Based Useware Engineering (MBUE) architecture has been derived and refined on the basis of the (meta-)architecture of the CRF (Meixner et al., 2011a). Therefore, the levels of the MBUE correspond to the levels of the CRF as shown in Table 2.

The MBUE relies on the CRF platform definition and its transformation approach ensures that the models can be assigned according to the correspondence scheme defined in Table 1. It is noteworthy that MBUE applies task annotations in its useML task model that assign tasks to a certain context of use (i.e., platform and/or user and/or environment). This way MBUE supports the automated tailoring of the UIs to a certain context of use. Such annotations are similar to the Web-service annotations used in MARIA. Similar to the platform-independent AUI model in MARIA, there is only one platform-independent (annotated) task (i.e., useML) model in MBUE which can be used for multi-device UI generation.

4.5 USer Interface eXtended Markup Language (UsiXML)

The USer Interface eXtended Markup Language[8] (UsiXML) specifies different meta-models that support MDA compliant multi-platform UI development (Vanderdonckt, 2005). UsiXML support the specification of UIs in a way that remains independent of a certain interaction technique (e.g., mouse, touch, keyboard or voice recognition) and a certain computing platform (e.g.,. mobile phone, Tablet PC, kiosk, laptop or wall screen). UsiXML also supports the inclusion of references to a certain platform in its models similar to the annotations used by MARIA and MBUE. The development of UsiXML started with the definition of the CRF and it was designed to support the one-to-one correspondence scheme shown in Table 1.

UsiXML is currently available in Version 2.0 and its meta-models are applied in various other approaches too (e.g., (García Frey et al., 2012)). Please note that the classification shown in Table 2 is not necessarily valid for those approaches. The way how the instances of the UsiXML meta-models are applied in a specific UI development approach depends on if and how multi-device UI generation is achieved by the approach's transformation method and may still violate the one-to-one correspondence.

4.6 Unified Communication Platform (UCP)

The Unified Communication Platform[9] (UCP) supports the automated generation of GUIs that are optimized for different devices (e.g., smartphone or tablet PC) (Raneburger et al., 2011). This approach relies on Discourse-based Communication Models (Falb et al., 2006) to model the high-level interaction between a user and the system platform-independently. A platform is defined through software (e.g., supported GUI toolkits) and hardware characteristics (e.g., screen size) of a certain device. Communication Models are CIMs and can be transformed automatically to a UI Behavior model that resides on AUI level (Popp et al., 2009) and is still platform-independent, and to a Screen Model that is tailored to a certain device and thus already platform-dependent. The Screen Model is finally transformed to HTML code that represents the FUI/ISM. Table 2 shows that a one-to-one correspondence can be established for UCP.

5 DISCUSSION

The classification of the MBUID approaches presented above reveals that no simple one-to-one correspondence as in Table 1 can be established in general, but that the used meta-models and how their instances are applied depend on the platform definition and are characteristic for a certain transformation approach. For example, a transformation approach that

[7]http://giove.isti.cnr.it/tools/Mariae/

[8]http://www.usixml.org

[9]http://ucp.ict.tuwien.ac.at

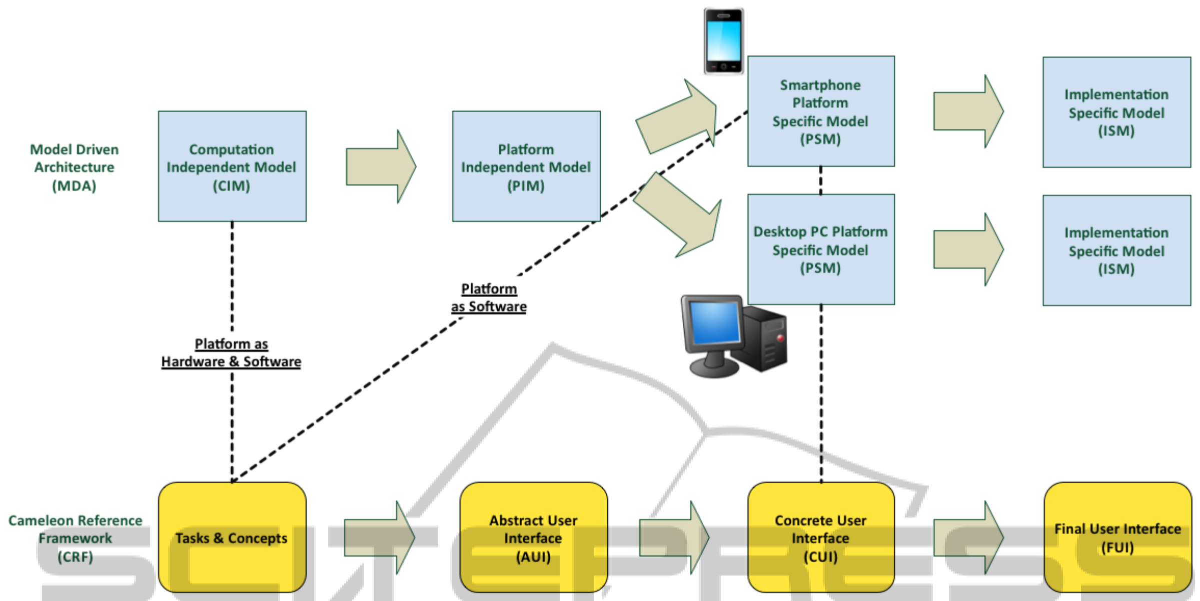
Model Driven Architecture (MDA)
Computation Independent Model (CIM)
Platform Independent Model (PIM)
Smartphone Platform Specific Model (PSM)
Implementation Specific Model (ISM)
Desktop PC Platform Specific Model (PSM)
Implementation Specific Model (ISM)
Platform as Software
Platform as Hardware & Software
Cameleon Reference Framework (CRF)
Tasks & Concepts
Abstract User Interface (AUI)
Concrete User Interface (CUI)
Final User Interface (FUI)

Figure 1: Correspondence between MDA and CRF.

does not explicitly consider hardware characteristics in the platform model requires their implicit consideration in the high-level interaction model. However, the same high-level interaction meta-model can be used by another approach to model interaction independently of hardware constraints if they are considered in the corresponding platform definition.

Figure 1 illustrates the implications of these two different platform definition on the correspondence between the MDA models and the CRF levels. The upper part of Figure 1 shows how multi-device UI generation is supported in MDA. The lower part shows the corresponding CRF levels.

A one-to-one correspondence can be established if the platform model considers software and hardware characteristics and if the high-level interaction models (e.g., task or communication models) specify all information that can be exchanged in parallel as concurrently available. This is indicated by the dotted line labeled "Platform as Hardware & Software" in Figure 1. All approaches whose classification shows a one-to-one correspondence support multi-device UI generation, which can thus be used as a criteria to detect such approaches.

No one-to-one correspondence can be established if the platform definition considers only software characteristics explicitly. Hardware characteristics like available screen space may already have to be considered in high-level interaction models that reside on Tasks & Concepts level in this case. Such models are already device specific just like the AUI and CUI models that are derived from them and therefore correspond to PSMs. This is indicated by the dotted line labeled "Platform as Software" and the unlabeled dotted line in Figure 1, that relate the Tasks & Concepts, the AUI and the CUI level to the PSMs. A platform model that considers software characteristics only does not support multi-device UI generation but it is still suitable to support multi-modal UI generation. The CRF levels can be used to classify different PSMs in this case.

We conjecture that a clear definition of the notion platform is also important in other application domains of model-based software development than MBUID to support multi-device development. The encapsulation of platform characteristics in the corresponding platform model together with the use of annotations as used in MBUID to avoid dependencies between a high-level model and a certain platform, is a generally applicable way to separate platform specific information from the model itself. Alternatively the platform-tailoring can be automated according to given optimization objectives, as shown for GUI optimization in (Raneburger et al., 2011). This saves the time and effort required for creating annotations manually, but may not produce the exact result expected by the designer.

6 CONCLUSIONS

Platform-independence for a model requires that this information has to to be encapsulated in the corresponding platform model and *must not* be considered in the CIM and PIM model. Thus, the first implication in the context of model-based multi-device UI development is that a platform must specify software and hardware characteristics, so that such characteristics do not have to be considered in the high-level interaction models. The information required to transform such PIMs to PSMs is typically either provided in form of model annotations or platform-specific transformations. One further implication is that high-level interaction models need to specify all information that can be exchanged in parallel as concurrently available. This allows for splitting the information according to certain platform characteristics while the PSM is derived and thus for tailoring a GUI for a certain screen space.

We showed that the CRF platform definition refines the MDA platform definition. So, both definitions support multi-device UI development, but no general one-to-one correspondence between the meta-models involved in a certain MBUID approach classified as MDA models, and the CRF levels can be established. The reason is that the use of a certain meta-model instance strongly depends on the transformation approach. For example, task models are platform-independent if they use annotations to specify which task is available on a certain device, or platform-dependent if they consider hardware characteristics like screen-size when specifying which information is concurrently available. However, a one-to-one correspondence indicates that a certain approach supports multi-device UI generation.

REFERENCES

Acerbis, R., Bongio, A., Brambilla, M., Butti, S., Ceri, S., and Fraternali, P. (2008). Web applications design and development with webml and webratio 5.0. In Paige, R. and Meyer, B., editors, *Objects, Components, Models and Patterns*, volume 11 of *Lecture Notes in Business Information Processing*, pages 392–411. Springer Berlin Heidelberg.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308.

Ceri, S., Brambilla, M., and Fraternali, P. (2009). The history of webml lessons learned from 10 years of model-driven development of web applications. In Borgida, A., Chaudhri, V., Giorgini, P., and Yu, E., editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 273–292. Springer Berlin Heidelberg.

Falb, J., Kaindl, H., Horacek, H., Bogdan, C., Popp, R., and Arnautovic, E. (2006). A discourse model for interaction design based on theories of human communication. In *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*, pages 754–759. ACM Press: New York, NY.

García Frey, A., Céret, E., Dupuy-Chessa, S., Calvary, G., and Gabillon, Y. (2012). Usicomp: an extensible model-driven composer. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '12, pages 263–268, New York, NY, USA. ACM.

Meixner, G., Breiner, K., and Seissler, M. (2011a). *Model-Driven Useware Engineering*, chapter 1, pages 1–26. Studies in Computational Intelligence, SCI. Springer, Heidelberg.

Meixner, G., Paternò, F., and Vanderdonckt, J. (2011b). Past, present, and future of model-based user interface development. *i-com*, 10(3):2–10.

Miller, E. J. and Mukerjij, J. (2003). MDA guide version 1.0.1. Technical report, Object Management Group (OMG).

Mori, G., Paternò, F., and Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8):507–520.

Paternò, F., Santoro, C., and Spano, L. D. (2009). Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16:19:1–19:30.

Popp, R., Falb, J., Arnautovic, E., Kaindl, H., Kavaldjian, S., Ertl, D., Horacek, H., and Bogdan, C. (2009). Automatic generation of the behavior of a user interface from a high-level discourse model. In *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS-42)*, Piscataway, NJ, USA. IEEE Computer Society Press.

Raneburger, D., Popp, R., Kavaldjian, S., Kaindl, H., and Falb, J. (2011). Optimized GUI generation for small screens. In Hussmann, H., Meixner, G., and Zuehlke, D., editors, *Model-Driven Development of Advanced User Interfaces*, volume 340 of *Studies in Computational Intelligence*, pages 107–122. Springer Berlin / Heidelberg.

Truyen, F. (2006). The Fast Guide to Model Driven Architecture - The basics of Model Driven Architecture.

Van den Bergh, J., Meixner, G., and Sauer, S. (2010). MDDAUI 2010 workshop report. In *Proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010)*.

Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. In *CAiSE*, pages 16–31.

Vanderdonckt, J. M. (2008). Model-driven engineering of user interfaces: Promises, successes, and failures. In *Proceedings of 5th Annual Romanian Conf. on Human-Computer Interaction*, pages 1–10. Matrix ROM.