Mastering Model Driven Engineering complexity by interactive visualization

Illustration in Human Computer Interaction

Mir'atul Khusna Mufida , Sophie Dupuy-Chessa , Gaëlle Calvary

Univ. Grenoble Alpes, Grenoble INP, CNRS, LIG
38000 Grenoble, France
Firstname.Lastname@imag.fr

ABSTRACT. *Model Driven Engineering (MDE) can play an important role for the design of applications in many domains. Its principles that are separations of concerns, capitalisation of knowledge thanks to models, meta models, and transformations, are well accepted today. Then the approach becomes a good candidate for scaling up to more complex applications. However the use of model driven engineering puts people in front of a new problem, which is mastering the complexity of many and possibly big models. The paper presents MoVi (Model Visualization) an interactive environment as a proof-of-concept that investigates model exploration by processing models as data. It is illustrated with models taken from the Human Computer Interaction domain.*

RÉSUMÉ. *L'ingénierie dirigée par les modèles (IDM) peut jouer un rôle important pour le développement d'applications dans de nombreux domaines. Ses principes, qui sont la séparation des préoccupations et la capitalisation du savoir et du savoir-faire sous la forme de modèles, métamodèles et transformations, sont aujourd'hui bien admis. L'approche devient alors un bon candidat pour un passage à l'échelle sur des applications complexes. Cependant, l'utilisation de l'ingénierie dirigée par les modèles met les concepteurs face à de nouveaux problèmes comme la gestion de la complexité de modèles de plus en plus nombreux et potentiellement grands. Cet article prèsente MoVi (Model Visualization), un environnement interactif validant le concept de l'exploration de modèles en considérant les modèles comme des données. Il est illustré par des modèle du domaine de l'interaction Homme-Machine.*

KEYWORDS: *complexity, modeling, models exploration, visualization, HCI.*

MOTS-CLÉS : *complexité, modélisation, exploration de modèles, visualisation, IHM.*

DOI:10.3166/TSI.35.175-202

1. Introduction

Nowadays, Model Driven Engineering (MDE) is a well-known approach that aims at simplifying application development by abstracting through design. Abstractions are made by models and transformations. Each model represents one aspect of the system (Schmidt, 2006). So system viewpoints are depicted by different models. For example, in engineering of Human Computer Interaction (HCI), many languages have been proposed to describe User Interface (UIs) from several viewpoints (static and dynamic) with different concepts (tasks, users, widgets ...) and at several levels of abstractions (task, Abstract User Interface, Concrete User Interface, Final User Interface). At EICS'2010, 68 model based languages were denominated. The variety and the number of models and relationships between them cause a model complexity problem.

Model complexity raises new challenges for designers. Among many heterogeneous application concepts, designers need to have a global understanding about their model eco-system (Sottet *et al*., 2009). The model complexity factor does not only come from understanding one concept with its syntax, semantics, and relationships with other concepts, but it also comes from the holistic models perspective. By mastering the model complexity problem, designers should be able to understand the whole design perspective and the role of each model in the whole design.

This paper focuses on the problem of model proliferation i.e. the management of a lot of models and their relationships. Our approach considers model as data to take advantage of the data visualization research for managing the problem of model proliferation. The interactive visualization approach provides a large set of features like filter, sort, detail on demand, history and extract to explore large dimensional data. Applying such features to models can help to access and understand a large set of models. The issue is how to design these adequate features to access models. One well-known interactive visualization technique that could answer this question is the seeking mantra (Shneiderman, 1996). The seeking mantra is an interesting approach, thanks to its holistic perspective to design visualization tools. It is an important methodological contribution by providing visualization tools design guidelines (Craft, Cairns, 2005). To show that it is able to master the model complexity problem, we develop and experiment the MoVi (Model Visualization) environment following its principles.

MoVi is a proof-of-concept that supports model exploration by applying the seeking mantra to support multi models exploration. The mantra becomes our methodological approach that guides the design of the interactive visualization features. For implementation, we use Data Driven Document (D3) (Bostock *et al*., 2011), a library that provides an expressive and simple language and allows to generate interactive visualization tools based on data. In our case, models are considered as data, creating then models networks, which can be manipulated thanks to the seeking mantra. Some users' feedbacks confirmed the interest of the mantra and of its application into the MoVi environment for the management of model eco-systems.

The organisation of the paper is presented as follows: the background of this work is discussed in Section 2. In Section 3 we explain the state of the art. Our contribution that convers up the conceptual and technical solutions is explained in Section 4. Finally, Section 5 proposes a summary of our work and some perspectives.

2. Background

This section gives the background of this work: the illustration of the modelling complexity in HCI and some visualization foundations which can be useful for mastering model complexity.

2.1. Modelling complexity

To illustrate the problem of model proliferation, we consider models proposed in the Human Computer Interaction (HCI) community. There is no standard method to create good user interfaces (UIs). It depends on many aspects like the context of use, the look and feel, and ability to accomplish user task efficiently. Thus, the user interface modelling approach plays an important role to propose appropriate, accessible and understandable abstractions, for instance using declarative user interface model (UIMs) (Da Silva, 2001).

Every user interface modelling aspects such as domain, task, platform, presentation, and application have their own role in design processes. Some works aim at helping designers to understand how using models in HCI. For example, one of the well-known framework is CAMELEON (Calvary *et al*., 2003), which identifies and structures a set of models for adaptable UIs. It supports building multi-target user interfaces by presenting several levels of abstraction. It defines the structure of user interface design with four levels (fig.1):

1. The highest layer of abstraction is represented by the domain and task models that is related to the Computer and Platform Independent Model (CIM/ PIM) in MDE. Domain model describes the application concepts and their relationships. It can be represented by UML class diagrams. Task model considers the sequencing of tasks realized by users, the system or in interaction between them. It can be represented as CTT (Concur Task Tree) (Paterno *et al*., 1997), which associates tasks with temporal operators.

2. The second layer is the Abstract User Interface (AUI), that is independent of any implementation and interaction modality (i.e. voice, graphic, text, etc.).

3. The third layer is the concrete user interface (CUI), that is independent of any implementation but dependent of interaction modality. For instance, UIs as the graphical widget representation (i.e. textfield, button, label, etc). It is Platform Specific Model (PSM) and it can be expressed by XIML (Puerta, Eisenstein, 2002).

4. The lowest layer is the final user interface (FUI), that is dependent to both the implementation and interaction modality. It represents concrete UIs that is implemented using a specific technology (SwingX, AWT, HTML, etc).

Based on different contexts of use (user, platform or environment), models can be adapted.

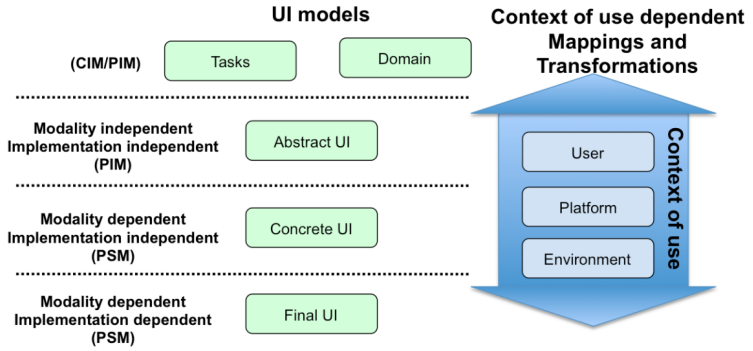

Figure 1. Cameleon Reference Framework (Calvary et al.*, 2003)*

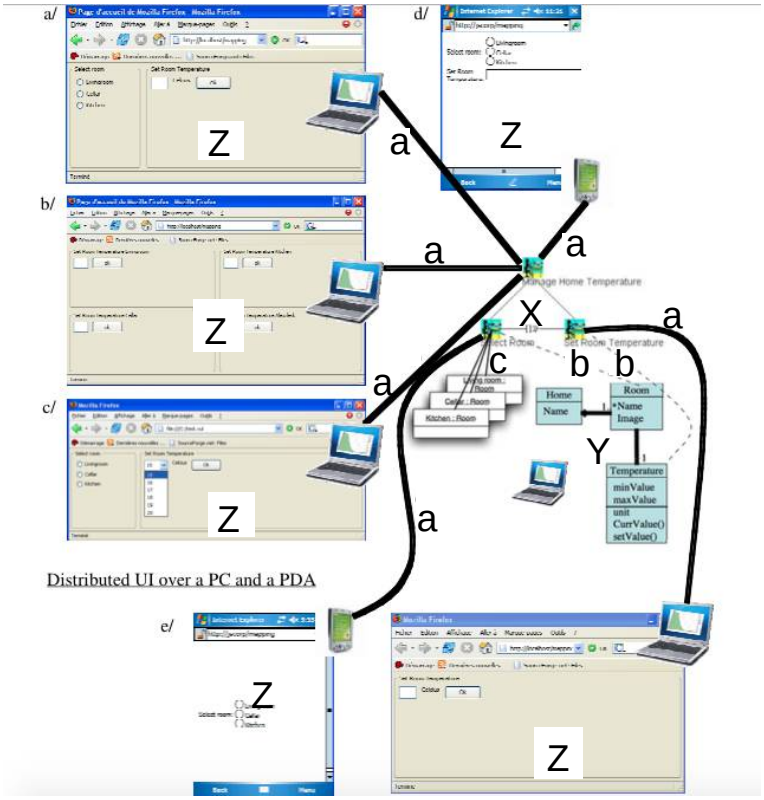

Figure 2. Multi models of a single system (Sottet et al.*, 2007)*

Figure 2 illustrates a system to manage home temperature. Several models represent different aspects. Firstly, a task model (fig-2-X) is expressed by CTT, which

represents the user's and system task and their sequencing. Here there is a task "manage home temperature" which is decomposed into two subtasks "Select room" and "Set room temperature". Secondly, the domain model (fig-2-Y) is expressed by a UML class diagram, which represents all concepts and how they are connected to each other. Here there are three classes "Home", "Room" and "Temperature". Some links can be made between the task and the domain models: in b, the link expresses that the task "Select room" (resp. "Select temperature") concerns the concept "Room" (resp. "Temperature") ; in c, the instances of rooms are linked to the task that manipulates them. Thirdly (in links labeled a), several FUIs (fig-2 a/, b/, c/, d/, e/) are proposed depending on the context of use i.e. the devices and their technologies.

Therefore, modelling UIs requires many models. We can see with a very simple example the number of models created and imagine the models proliferation for more realistic applications.

2.2. Overview of model editors

To manipulate models, designers generally use model editors. For instance, Eclipse and EMF can be used to edit UML, BPMN and OWL based models. Typically, specific models editor or Integrated Development Environment (IDE) helps to create and explore a single type of models. There is also research on UML analysis and design based on different point of views (Nassar, 2003) that guides designers in developing UML models. Openflexo (Guychard *et al*., 2013) provides some conceptual interoperability though model federation that captures model transformation and model development based on its federation.

It is known that interactive visualization technique can simplify complex models (numerous models, with relationships) analysis (Bull, Favre, 2005). However, as the classical editors, the existing visualization tools (Lanza, Ducasse, 2005), (Blouin, Combemale *et al*., 2015), (Garcia *et al*., 2010), (Maletic *et al*., 2001) are designed for representing models in a fine-grained approach: they mainly create models editor by separating models.

However editing models in separated environments or panels is restrictive: for instance, using the CTT editor to describe a task model and a UML editor to specify a domain model does not allow designers to link a concept to the tasks that manipulate it (e.g. linking Temperature to "Set Room Temperature"). This situation emphasizes the need for an integrated environment that allows designers to create links and navigate among models and get a holistic perspective of design.

2.3. Interactive visualization

2.3.1. Overview

Interactive visualization is a technique to visually represent data to amplify cognition thanks to user-system interaction (Card *et al*., 1999). It is supported by two main

components, visual representation and interactive aspect. Visualization is all about representing data in a way that makes it simpler to understand and faster to grasp the data set phenomenon (Bihanic, Polacsek, 2012). It can be applied to large set of data. For instance, the InfoVis Toolkit (IVTK) (Telea, 2014) introduces visualisation analysis framework like Gephi (Bastian *et al.*, 2009) and Tulips (Auber, 2004) and provides a set of features to examine big data.

Interactive visualization has been used in many domains like computer science to help analysing data combined with statistic approach (Fayyad *et al.*, 2002). The medical domain uses it to simulate operation or visualise treatment for critical organ (brain, heart) (Ropinski, Preim, 2008), geology to simulate earth-quake (Yu *et al.*, 2004), industry to help in the automotive development (Engel *et al.*, 2000) and many other domains.

Visualization types not only differ from their implementation, but also from the visual design representations (fig-3). Figure 3 presents some examples of visualization diagrams (a) Force-Directed Graph, (b) Sunburst, (c) Chord Diagram [1]. Visualization diagram usage depends on the data display format and on the visualization design preferences.

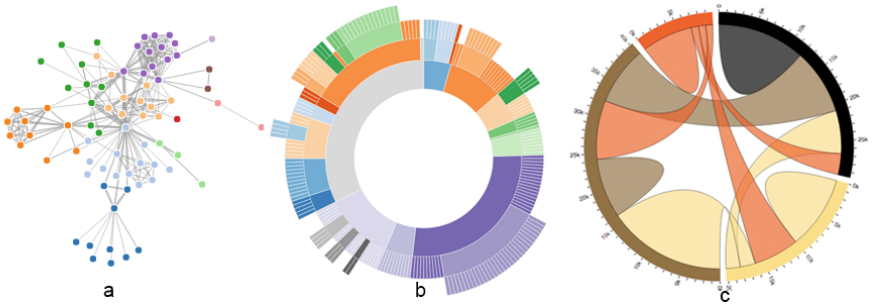

Figure 3. Different types of visualization

One of the common visualization advantage is to support data analysis by presenting a set of methods that allow to represent, visualize, categorise, and group numerous data based on their characteristics. It also provides fundamental techniques to make tremendous data accessible for users. It is widely used to express large data sets with high dimensional space and complex structure (Bull, Favre, 2005). For instance, ManyEyes (Viegas *et al.*, 2007) (fig-4)[2] is a visualization web tool that enables the creation of user based visualization and that fosters large collaborative development. It allows visualization to be implemented in many domains.

2.3.2. From data to visual representation

The visual representation process is defined by a pipeline from raw data to the final

1. visualization diagram https://github.com/mbostock/d3/wiki/Gallery
2. http://www-01.ibm.com/software/analytics/many-eyes/

Figure 4. Manyeyes : web visualization examples from left to right tree-maps, bubble chart, matrix bar chart, matrix pie chart, map, interactive scatter plot, word cloud, word tree, matrix chart, block histogram, phrase net, interactive bar chart

presentation (Card *et al.*, 1999; Chi, Riedl, 1998). (Santos, Brodlie, 2004) formalizes this pipeline as (fig-5):

– Data Analysis: techniques such as smoothing filter, interpolating missing values, or correcting erroneous measurements are used to prepare for visualization.

– Filtering: users can select data portions to be visualized .

– Mapping: focus data are mapped to geometric primitives (e.g., points, lines) and their attributes (e.g., color, position, size);

– Rendering: geometric data are transformed to image data.

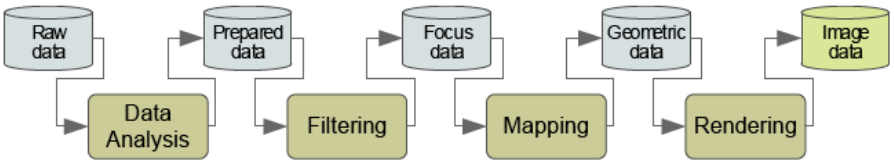

Figure 5. Visualization pipeline (Santos, Brodlie, 2004)

2.4. Actions on visualization

More than representation, interactive visualization concerns interaction. One seminal work in visualization is the information-seeking mantra (Shneiderman, 1996). It gives a "design pattern" to develop visualization tools by applying information visualization main tasks. The seeking mantra consists of some main functions:

– Overview: provides global context to understand dataset.

– Zoom and Filter: select and focus only on the interesting part by reducing the complexity of data representation.

– Details on demand: enables information appearance only when it is needed.

– View Relationships: displays relationships between visualization data items.

– History: provides feature to restore the previous events.

– Extract: extracts important information for dataset featured visualization tools.

With such interactive visualization techniques, we expect to be able to master the model proliferation problem. As the next section will show it, other works already explored this solution.

3. State Of the Art

This section presents the state of the art of using visualization techniques for model proliferation. We classified four majors related work categories including: (3.1) Visualization techniques for complex software code; (3.2) tools for managing very large models; (3.3) applications to navigate between HCI models to support understanding and (3.4) visualization approach that is used to help in modelling.

3.1. Visualization technique for application code

Visualization technique is well-known to handle complex data analysis problem. Thus, several visualization based projects propose to visualize the implementation code in order to support programming tasks (fig-6).

For instance, (Ball, Eick, 1996) (fig-6-a) augments software code by presenting global to details views of software program with different colours based on the code type and manipulation history. With this representation, programmers can easily verify which part of the code is recently modified. Besides, it provides three different sizes of code windows that allow programmers to rapidly look at through the long line codes considering different colours code notation.

CodeCrawler (Lanza, Ducasse, 2005) is a visualization tool which allows to visualise object-oriented software in the fine-grained poly-metric views. It is used to visualize object-oriented code. It implements lightweight 2D-3D representation augmented with semantic informations extracted from miscellaneous code analyzers. It presents prominent class blueprint view as a hierarchical classes representations. It also uses colours to determine class categories and bar sizes to determine number of elements. Coloured ingoing and outgoing edges represent classes method invocations. More ingoing and outgoing edges in a class determines the important role of the class because it invokes several methods (outgoing links) and it is invoked by many methods (ingoing links). Programmers are also able to interact with the environment by select-

ing (right click) a particular class to examine its detailed elements. The objective is to help in the code maintenance.

The visualization techniques used to support programmers are interesting and could contribute to model exploration, especially the way to provide details and critical informations. It gives basic idea to links different levels of abstraction without sacrificing any data.

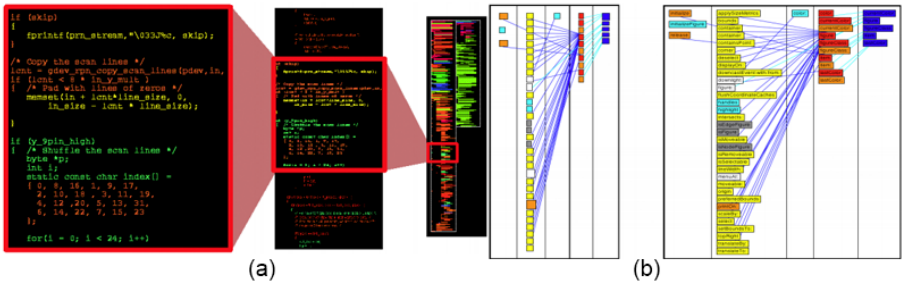

Figure 6. View of visualization code tools for (a) software visualization in the large (Ball, Eick, 1996) and (b) CodeCrawler (Lanza, Ducasse, 2005)

Following the idea of visualization of using colours or more generally visual clues to help in understanding one model, Moody proposes the physic of notation (Moody, 2009) which gathers principles to design the graphical notation of a language. From these principles, some authors like (Le Pallec, Dupuy-Chessa, 2012) try to define properties or metrics for the visual quality of models. If this approach is interesting, it is restricted to one model and does not consider the model ecosystem.

3.2. Managing very large models

Very large models require specific tools to deal with model complexity. Some researches about managing large models have been done like Map/Reduce on EMF models (Scheidgen, Zubow, 2012) or No4EMF (Benelallam *et al.*, 2014). (Scheidgen, Zubow, 2012) enables the processing of complex data structure using EMF programming modelling cloud computing. No4EMF (Benelallam *et al.*, 2014) is a scalable persistence layer for EMF models. It provides software development methods that are able to handle large-size artefacts by adding a new scalable persistence layer that exploits the efficiency of graph databases to store and access EMF models.

Nowadays, the cloud provides unlimited storage to support ubiquitous computing. Cloud computing is also used to facilitates very large model storage and model transformation executions. For example, (Clasen *et al.*, 2012) uses a cloud base architecture, with parallel processing to couple model transformations and models that are processed. They address two phases of model storage and model transformation execution in the cloud. Another scenario to work with very large models is using map-based transparent persistence (Gomez *et al.*, 2015). It builds an additional transparent

persistence layer using map-based model or database engine for modelling tools like Eclipse Modeling Framework (EMF).

Hence, solutions to deal with very large models is discussed and solved in many ways. They can contribute to manage and describe solutions to transfer models properties in details using cloud based technology. They tend to solve model scalability storage problem using cloud. However our problem is not limited to storage, it is also related to the way of understanding the global structure of a model ecosystem.

3.3. Navigation between models in the HCI domain

Model proliferation issues have been studied by different approaches. The first one is limited to models and does not consider meta-model and transformation like IdealXML (Montero, Lopez-Jaquero, 2007), Mega-UI (Sottet *et al.*, 2009), Genius (Sottet, Vagner, 2013), AME (Garcia Frey *et al.*, 2014), Model voyager and Quill (Genaro Motti *et al.*, 2013).

IdealXML is an model based application that allows to create fast prototypes that can be modified easily at the abstract level. It focuses on a high level of abstraction to reduce model complexity, so it considers only one level of abstraction of the HCI design.

Mega-UI is an environment to manage MDE structures at various levels. It allows to navigate between components in the same model and navigate from one model to another model in the same system. It focusses on the usability aspect design instead of functional factors adaptation. It introduces UI design from different point of views. It demonstrates the importance of the possibility to navigate between models. However, it only considers different models in the same system and it does not present the external related models from another system that can contribute to the whole design.

Model voyager allows users to navigate inside models at the different levels of abstraction of the CAMELEON reference framework. It presents the various UI models using semantic tree representation. It supports users in understanding MDE approach especially in HCI domain. Nevertheless, it currently does not present many applications designs at the same time. Additionally, it only considers models but it does not consider meta-models and transformations.

Quill is a web based development environment that enables various stakeholders to collaboratively adopt a HCI design using model-based approach. It generates adaptable a user interface based on the input model. It compares visual representation and textual description to ensure application interoperability. It presents very interesting model design decision containing ten requirements namely flexibility, portability, context awareness, usability, scalability, consistency, persistence, functionality, collaboration and efficiency. Quill focuses on providing several adaptive User Interfaces depending on the given platform settings. Moreover, it does not consider meta-model representation and existing models as a reusable resources.

Only Mega-UI considers all model perspectives (model, meta- model, transformation) in order to help designers to understand the whole design picture. Accordingly, none of the studied approaches presents various model categories and several system designs at one time to give holistic design perspective and to promote model reusability.

3.4. Visualization for models

The visualization approach also proposes good opportunities to support model understanding by providing adequate features to navigate inside an ecosystem of models. Some visualization tools were designed to visualize some model complexity aspects such as (Bihanic, Polacsek, 2012), UML tree-map visualization (Garcia *et al*., 2010), "Moose" (Ducasse *et al*., 2000), "Imsovision" (Maletic *et al*., 2001) and "Explen" (Blouin, Moha *et al*., 2015). The visualization techniques used are based on graphs. There are three major visualizations in the graph representations (Bihanic, Polacsek, 2012): trees (fig-7-a), maps (fig-7-b), and landscapes (fig-7-c). Trees provide hierarchical representation of the information (inheritance trees). Maps are appropriate to illustrate complex data representation, for instance representing voronoi diagrams or networks. Landscapes provide multi-scale representations of planar shape with contextual and logical information flow. Each node represents a single data and a relationship illustrates the link between two data. For example in an air traffic control application, the nodes can represent airports and links plane routes.

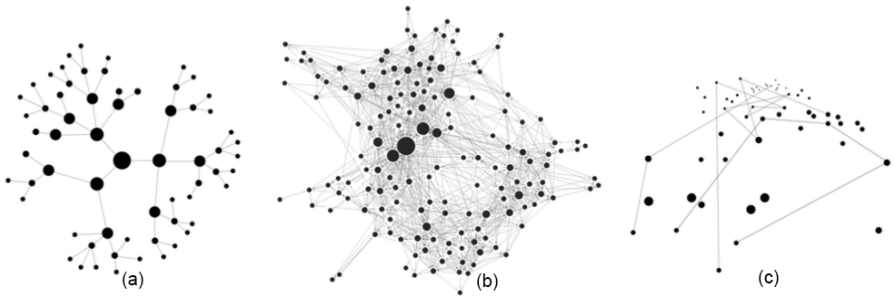

Figure 7. Three visual representations for complex information system (a) trees (b) maps (c) landscapes (Bihanic, Polacsek, 2012)

(Garcia *et al*., 2010) proposes a visualization tool that describes a large software thanks to a treemap hierarchy and a table lens representation. It considers UML class diagram models used for the reverse engineering of large scale softwares. As example, many open source codes are available on internet in code repository sourceforges[3]. In order to be able to modify these codes, it is necessary to understand the code structure. Reverse engineering extracts source code to the more abstract perspectives (models).

3. An open source software repository available in site : http://sourceforge.net/

The tool allows users to have a hierarchical representation of the models. However, it considers a single type of model (UML class diagram).

"Moose" (Ducasse *et al*., 2000) is an extensible language-independent environment that provides a meta-model system representation to support CodeCrawler. It allows to explore the meta-models of the software models that are represented by CodeCrawler. It aims to provide navigation and manipulation of models considering their different versions in order to support system reengineering. Together with CodeCrawler, they provide some interesting features such as model and meta model layers focussing on detailed design representation.

"Explen" applies slice-based visualization to support meta-model exploration. Slicing method mainly aims at providing an extraction feature to select only a subset of a meta-model. The features of "Explen" are developed based on "Kompren" (Blouin, Combemale *et al*., 2015), a domain specific language to define model slicers. Basically it provides a method to develop different kinds of filters to navigate and explore large meta-models. So it supports large meta-model exploration and helps in their understanding.

"Imsovision" builds task analysis based on the seeking mantra. It focuses on using 3D and Virtual Reality as the default technology to visualize UML models. It is an interesting approach, although it only considers one level of abstraction. So it potentially sacrifices many information like meta-models or models relationships especially when dealing with very large software projects.

To summarize, the existing model editors and visualization tools mostly concern a particular model type (e.g. UML, class diagram, meta-model) whereas having an holistic view of the design is an important capability for designers. Accordingly, it is important to create an integrated tool to support multi model exploration tasks that allow to map different types of models (Kent, 2002). It can be realized by applying visualization techniques. One seminal works that guides the design of visualization features is the seeking mantra (Shneiderman, 1996). But, except "Imnovision", none of the existing tools for model exploration fully supports the mantra and its main features (overview, zoom and filter, details-on-demand, view relationship, extract).

4. From Interactive Visualization to MDE in MoVi

This section discusses our solution to solve the model proliferation problem. We elicit model complexity factors like multi models, various types of model and different relationships and propose to overcome these factors by applying visualisation techniques.

4.1. Overview

Our solution is composed of two complementary parts for users (fig-8):

– a (meta)model editor allows designers to create their models and links between them (e.g. version, abstraction).

– a model visualization tool for the exploration and visualization of (meta)models

Both parts rely on common functionalities to store and access to models, metamodels and transformations.

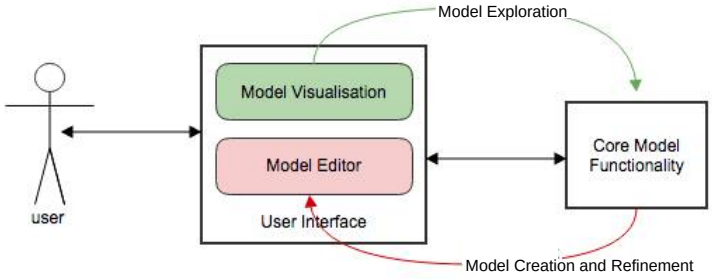

Figure 8. Overview of the solution

This paper focuses on model visualisation, which is the most original part of our work. It presents MoVi (Model Visualization), an environment for models exploration. In its current version, (meta)models are collected manually and entered in the system as images that are referenced in JSON files; their links are declared as properties. From this information, MoVi takes benefit from interactive visualization features inspired from the seeking mantra to permit model navigation. The next subsections describe the MoVi principles and implementation.

4.2. Unification principles

The key principle of our solution is to set higher level model exploration for supporting holistic design. Interactive visualization provides major features that are required to explore models like provide model representation, search particular model, filter only desired models, view only important model properties, or select related models. According to the taxonomy of interactive dynamics for visual analysis (Heer, Shneiderman, 2012), we map some visualization analysis features to model exploration tasks (Table 1).

Based on this mapping, we propose the MoVi prototype to support the seeking mantra.

4.3. Holistic design of MoVi

A model exploration scenario can follow a top-down approach by examining general to detailed models. Accordingly, it gives a general perspective of all models from

Table 1. Mapping visualisation analysis to model exploration task

Visualization Analysis	Model Exploration
Choose visual encoding	Design model representation
Derived value or models from source data	Implement model representation
Sort item to expose pattern	Search particular model(s)
Filter out data to focus on relevant item	Filter only desired models
Select important items	View only important model properties
Navigate to examine related items	nagivate among models

an abstract point of view. Additionally, relationships between models help to understand the model position in the whole model ecosystem.

MoVi (Model Visualization) presents various kinds of models/metamodels from the HCI domain. Of course other kinds of models can be considered, HCI models are only illustrative. MoVi features also manage meta-models and relationships between models and/or meta-models. In MoVi, a model is represented as a node (fig-9-a-1) with a specific colour identifying a model type. This representation aims to handle various model types. Model relationships are also presented as various coloured and distance edges between nodes (fig-9-a-2). Colour represents the relationship type and distances displays models similarity. Similarity is currently created manually by considering three types of model relationships (dissimilar, weakly similar and very similar). This encoding with colours and distances aims to tackle various types of model relationships by showing relationship diversity in the design space. It also complies to the "view relationship" of the mantra principle.

Displaying different kinds of models and metamodels gives an holistic perspective about models and their relationships to improve model understanding. The "Overview" feature (fig-9-a-8) also guides users while they are exploring models by providing a selection feature. It determines which set of models are selected amongst the model ecosystem. This feature mainly proposes a solution to present many models at the same time.

Regarding to the model levels of abstraction, we adapt seeking mantra "zoom and filter" feature. MoVi provides geometric zoom (fig-9-b) as well as filtering features to concentrate on some models and to exclude uninteresting ones. There are two main filter types: a) generic filters that allow designers to filter models based on their relationships (i.e. abstraction, generalization, composition, instantiation, and association) (fig-9-a-4), applicable to both the model and metamodel (fig-9-a-5) levels, and b) domain-dependent filters (e.g. domain, task model, Abstract UI, Concrete UI, Final UI in HCI) (fig-9-a-6).

Considering model exploration point of view, it is important to navigate inside models and to focus only on one part of the model subset. The mantra presents the "extract" principle that we use to extract important information from models. So there is a search based on the model name and category (fig-9-a-3) and a selection (fig-9-

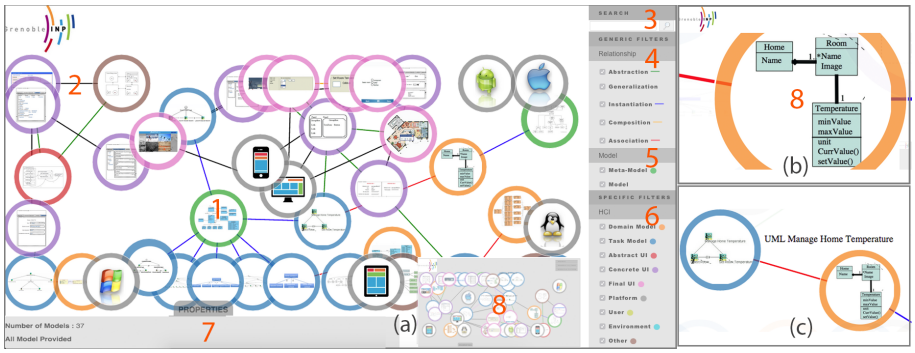

Figure 9. Holistic view of MoVi features from left to right (a) Main page of MoVi, (b) Zoom in feature, (c) Selection feature

c) for choosing target models. It is based on model neighbourhood (models that are directly related to the target model). It helps designers in examining a set of desired nodes that represent model mapping.

Moreover, MoVi provides details on demand to show models label (fig-9-c) and properties (fig-9-a-7) only when it is needed otherwise it is hidden. This feature is directly referring to the mantra "details-on-demand" principle.

4.3.1. Model, meta-model and relationship visual encoding

In this section, we describe each solution by explaining model exploration tasks using MoVi. MoVi propose to manage models, metamodels and relationships between models and between models and metmodels. Model relationship expresses link between a model to another model. There are many types of model relationships. For instance, a model must be an instance of a metamodel or in HCI models, there is abstraction relationship between a task model and its corresponding abstract user interfaces.

In MoVi, model inputs are an image of the model and some properties like its type or its metamodel. As mentioned in Section 4.2, in this environment, models are represented as a node and relationship as an edge. Every node contains an image to determine original model-like views and colour to determine its types (fig-10-a). Links also have different colours to represent different relationship types (fig-10-b). Relationship types that are currently considered in MoVi are abstraction, generalisation, instantiation, composition, and association.

– Abstraction (fig-10-b.1) represents a relationship between different level of models, for instance relationship between abstract user interface (AUI) and task/domain model.

– Generalisation (fig-10-b.2) expresses generalisation-specialisation between models. In such manner, it uses to relate "manage digital home" task model as a generalisation of "manage home temperature" task model.

– Instantiation indicates that one model is an instance of another model. This relationship is mostly used to express that a model is conform to a meta-model. For instance, there is an "instantitation" relationship between the CTT meta model and a task model.

– Composition indicates that one model is composed of other models. For example "manage home temperature" AUI is made of two AUIs: one for "select room" and one for "set temperature"

– Association (fig-10-b.3) represents a link between models. For instance there is an association between the task model of "manage identity" and the domain model where the identity is defined.

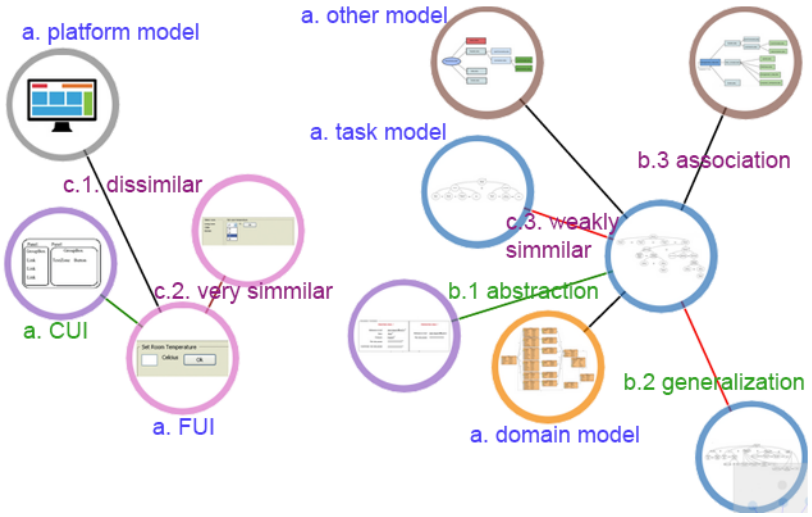

Figure 10. Visual encoding

In addition to models and relationships, one of the model exploration parameter is the distance between models. It applies different links length considering different similarity model distances. Similarity could be computed by machine learning, graph matching (graph theory), etc , but this is out of scope of this paper. In MoVi, model similarity is created manually by considering three types of model relationships (fig-10-c)(dissimilar (fig-10-c.1), weakly similar (fig-10-c.3), and very similar (fig-10-c.2)).

We also provide a legend (fig-11) to help designers in understanding the visual encoding in MoVi.

4.3.2. Overview

The concept of overview (Craft, Cairns, 2005) aims at giving a general context for understanding the whole model ecosystem. It presents an holistic view of all existing models (fig-12) with feedback to the user. It uses different opacity and sizes to highlight the model subset that was selected by the user in the model space.

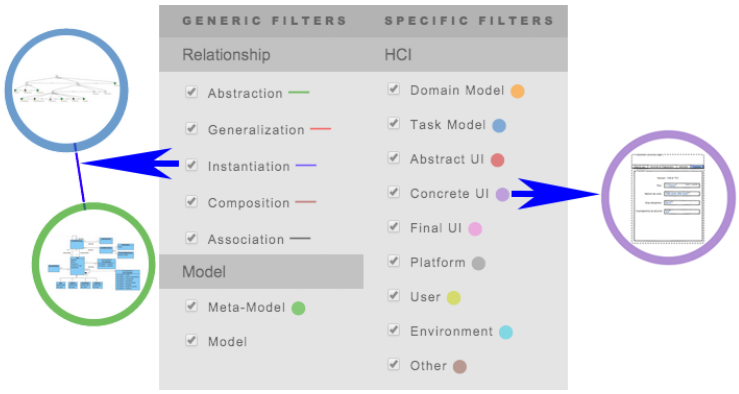

Figure 11. MoVi legend

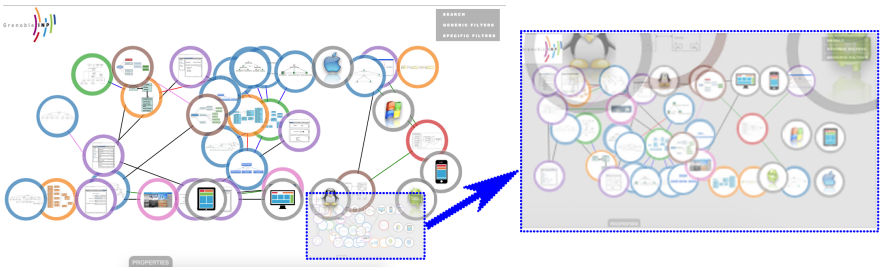

Figure 12. Overview

4.3.3. Details-on-demand

The details-on-demand feature (fig-13) helps users focusing on the particular part of the models or items. It displays a model label (fig-13-1), a relationship label (fig-13-2), and model properties (fig-13-3). These details appear only on demand. Model and relationship labels appear when user hovers the model of interest. Model properties are activated by clicking on the properties tab.

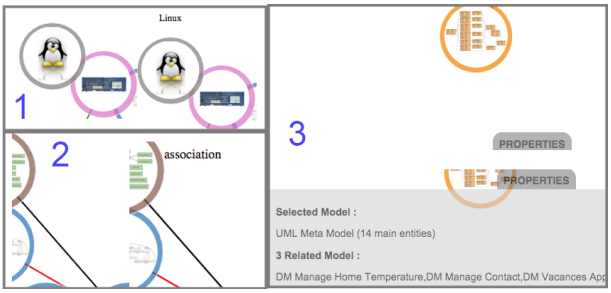

Figure 13. Details on demand

4.3.4. Extract

Model extraction is an important feature. It also helps users finding an interesting set of models. User double clicks on the model of interest and MoVi removes all the models that are not related to this model (fig-14-a). Overview is updated accordingly: it highlights the selected models by displaying them differently (fig-14-b). The selected model properties tab shows the selected model, its properties, and its related models (fig-14-c).

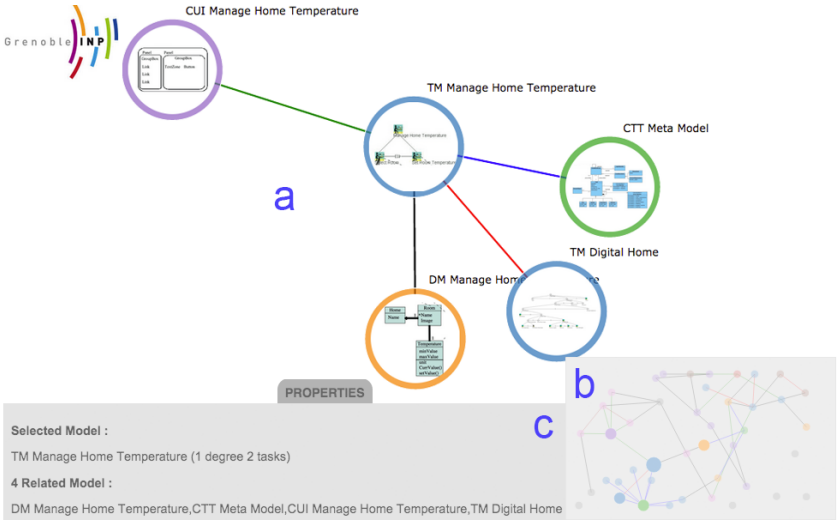

Figure 14. Extraction

4.3.5. Zoom

Zoom is a crucial feature to examine models into details. There are some types of zoom like the semantic and geometric zooms. In MoVi we only use the geometric zoom as a proof-of-concept. It shows the image at a bigger scale (fig-15). This function is activated by scrolling up or double clicking on the model of interest. Zoom out is activated by scrolling down.

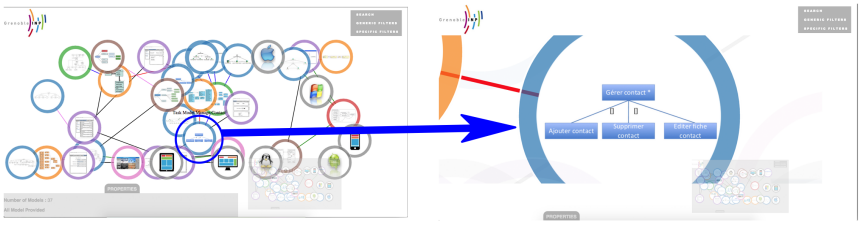

Figure 15. Zoom

4.3.6. Filter

As we mentionned in the holistic view of MoVi (section 4.3), there are two types of filters:

– Generic filters (i.e. abstraction, generalization, composition, instantiation, and association) for filtering models based on their relationships (fig-16-1). For instance, fig-16-1-a highlights a filter on the instantitation relationship. This means that only these relationships will be displayed in the MoVi main page. The filters are applicable to both models and metamodels (fig-16-2).

– Specific filters that are domain dependent. For instance, in our case, filters are specific to the HCI domain; so designers can filter models related to the domain, the tasks, the Abstract UIs, the Concrete UIs, Final UIs (fig-16-3). Here MoVi will display only the domain models in the main page.

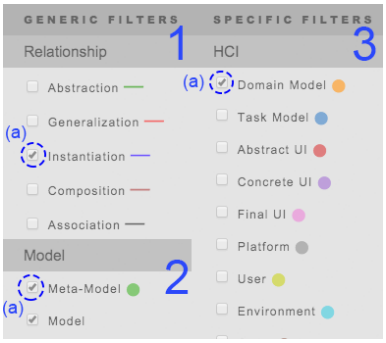

Figure 16. Filter parameter

One possible scenario is to filter models by combining instantiation, metamodels, and domain models (fig-16-a). The specification of the filter is done with checkboxes for supporting multiple choices. (fig-17) is the result of the filters specified in fig-16 i.e. the selection of domain models and meta-models that are linked through an instantiation relationships.

4.3.7. Search

Search is done by entering a text in an input field (fig-18-a). Search can be performed on the model name (fig-18-d) or on category (fig-18-c). For preventing users from errors, MoVi supports autocompletion (fig-18-b).

4.4. Implementation

MoVi (fig-19) processes models in three steps:

– Model collection (left part of the figure). It is the result of an initial activity that collects the models,

– Pre-processing formats raw models to make them compatible with the next step. It consists of registering model properties (e.g. name, description, image, type, etc),

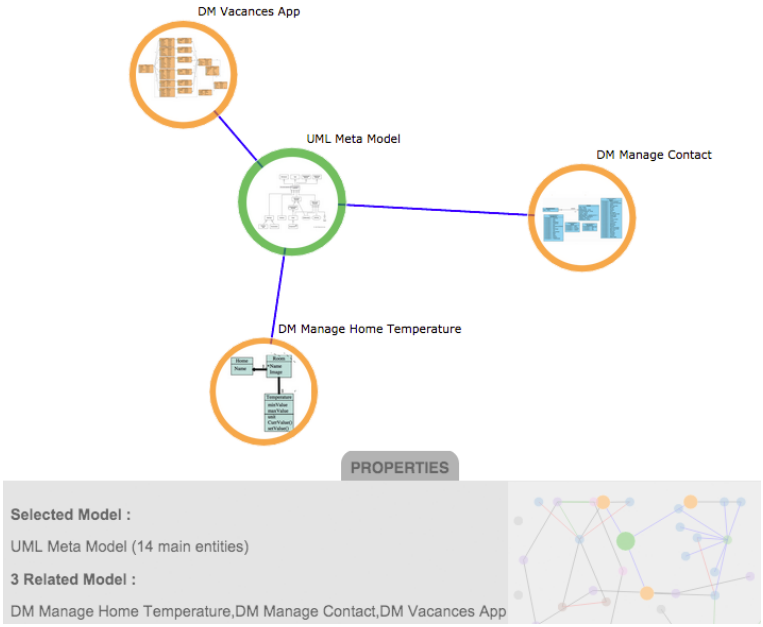

Figure 17. Filter result

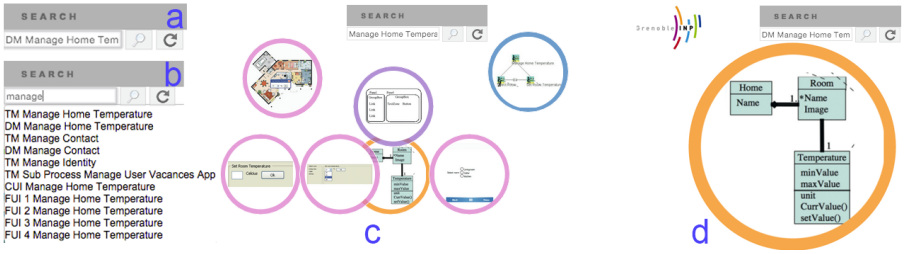

Figure 18. Search

grouping models, and generating JSON (JavaScript Object Notation) file. When JSON file changes, the visualisation is updated accordingly.

– Visualization covers the generation of the appropriate visualization and the features described before (overview, zoom and filter, search, extract).

To make easily accessible, MoVi is implemented as a web based application. It uses the data driven document (D3) library (Bostock *et al.*, 2011) which provides features for generating visualization for data. D3 combines DOM (document object model) as document structuration, HTML5 for markup language, CSS for aesthetic, JQuery for controling menu effect, JavaScript for performing interactions, JSON for format data input, and SVG for having data as a graphic representation. The visual appearance depends on the json files which contains a link to the model images. There

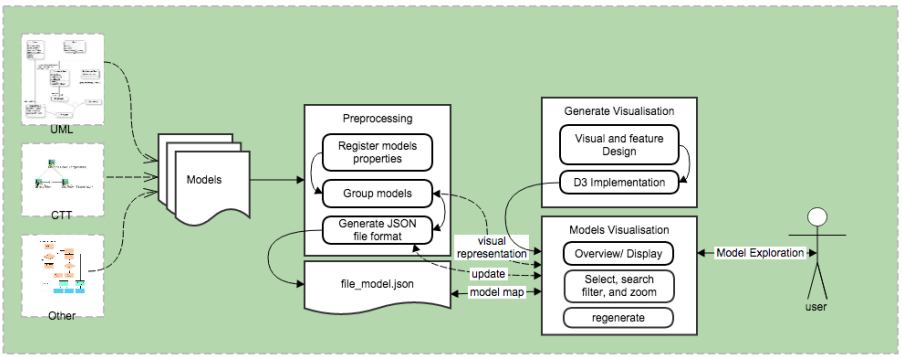

Figure 19. MoVi processing steps

are several possible layouts in D3 like divergent forces, multiple foci, graph constructor, force-directed tree, force-directed symbols, force-directed, images and labels.

4.5. Users' Feedbacks

In order to help us in the design of MoVi, we realized a formative experiment. It means that it is not a final evaluation of the prototype. This evaluation aims at getting users' feedback about two asumptions concerning the prototype features and usability. The results will be used to improve MoVi according to users' requirements.

4.5.1. System setup and Participants

We started with providing many and various models with heterogeneous types of relationships shown in (fig-9-a). The models consist of 44 HCI models and 48 relationships. The 44 models are categorised as model or meta-model and as domain model, task model, abstract user interface, concrete user interface, final user interface, platform and other type of models. The 48 relationships are categorised as abstraction, generalization, instantiation, composition and association.

Twelve persons with various profiles (age, occupation, and working experience) gave their feedbacks about MoVi. The participants are between 21 and 32 years old. They are students and researchers with various domain of expertise: MDE and HCI, HCI, formal methods, Architecture, system, compilation, and visualization.

Table 2. Subject profile

Number	Subject profile
3	Subject who is not familiar either with MDE neither HCI.
1	Subject who is familiar with MDE but not familiar with HCI.
3	Subject who is familiar with HCI but not familiar with MDE.
5	Subject who is familiar with both MDE and HCI.

4.5.2. Asumption 1 (A1)

The first asumption that we want to explore is that MoVi helps designers to find a subset of models from a large set of models. MoVi masters model complexity (multi models, relationship, multi levels of abstraction) by providing visualization features to explore models. So each subject was asked to perform three model exploration tasks using MoVi, and to give comment about MoVi. Tasks had different levels of difficulties. But they did not aim at figuring out the subject model understanding. Participants had to answer questions about their opinions. We also measured how long they need to accomplish task (timer), calculated the number of errors that are made to find a particular model (wrong response), calculated the number of terminated and non terminated tasks.

4.5.3. Asumption 2 (A2)

The second asumption concerns MoVi support for models exploration and model understanding. Designers could explore the existing models using MoVi to support them learn about model and design new model. Participants used MoVi features to accomplish model exploration tasks and to have an idea of how to reuse existing models. The tasks aimed at figuring out participants' understanding of models. So participants were asked to create a new model using existing related models. They had to explore the model ecosystem to get knowledge about HCI models. Then they had to note their experience using MoVi and to give strength and weakness points.

4.5.4. Results

Even if the number of participants was limited, we got several feedbacks that are useful to improve the MoVi prototype. First of all, all tasks were terminated with only few errors (only 5% of errors). The duration lasted from 5 to 20 minutes for an average of 13 minutes. This difference cannot be fully explained. It may be due to the variable task difficulties, to the sequencing of tasks or to their formulation.

From the users' point of view, participants perceived that MoVi helps in models exploration by using features like filter, search and extract. It is consistent with the first asumption which states MoVi can support model exploration tasks by implementing features to navigate between multimodels and their relationships. However, participants using MoVi to explore models, did not use all the proposed features (table 3). The most useful features were search, extract, zoom, and filter. Overview and legend were only cited once. Details on demand did not seem to be important. But this might be due to the fact that required tasks do not focus enough on details.

For the second asumption, results also relate to the information that users need to accomplish tasks (table 4)). Here we can note the importance of legends and examples.

We also categorised users' comments based on the application aspects in tables 5 and 6. From these preliminary results, we can notice that the prototype is perceived as usable (model navigation and model properties features) and useful (model representation and model examples). However some improvements were suggested about

Table 3. Helpful features to accomplish the tasks

Number of citations	Helpful features
7	Search
5	Extract
4	Zoom
4	Filter
1	Overview
1	Legend

Table 4. Necesary information

Number of citations	Important information
9	Model and relationship names and their category (Legend)
5	Examples of models of the same type
3	Information that provides similar model

usability (e.g. for menu and model label setting) and usefulness (addition of models version, search of a group ...).

4.6. Discussion

Even if the first users' feedbacks are encouraging, MoVi still suffers from limits that we have to consider.

Firstly, MoVi inputs are model images and properties that are collected manually from many sources. Different sources and model formats require preprocessing to unify models and properties to make them compatible with the format required for data manipulation, JSON. Unfortanetely JSON is not compatible with model editors. So there are some drawbacks regarding the input format. We recognise that this kind of input lacks of compatibility and of dynamicity for direct model modification. But it is possible to connect the MoVi environment to various model editors.

Secondly, MoVi presents model visualisation in graph representation. This kind of representation has some drawbacks. We need layouts that can manage more models while preserving the ability to show overviews of model details without zooming the nodes. Graph representation also has a simple concept of relationship between nodes. This simple representation decreases designers' effort to remember a lot of visual variables in MoVi. But a limitation is that we need to consider map organisation which calculates node position while preserving edges between nodes.

Finally, MoVi can also facilitate model creation by providing model examples and counterexamples. Nevertheless, it is only a first step in studying model reuse and some improvements must be made to improve MoVi in this way.

Table 5. User experience to use MoVi - Strengthes

Strong points	**Analysis**	**Eval. group**
Search is good	Usable to explore models	A1
Extraction quite helpful	Usable to explore models	A1
Filtering is good	Usable to explore models	A1
Overview feature is helpful	Usable to explore models	A1
Filter and search feature	Usable to explore models	A1
We can see inside the nodes	Usable to explore models	A1
Very easy to group and visualize the relationships	Usable to explore models	A1
Different colour impact was very good to distinguish models	Visual encoding design supports model exploration	A2
The interface is pretty (colours to distinguish the type of model and relationships between them)	Visual encoding design supports model exploration	A2
The way models are displayed is really nice	Visual encoding design supports model exploration	A2
Good representation of model and their link	Visual encoding design supports model exploration	A2
Having finished example which guide new designs	Functionality support to learn models by example	A2
Learn models by example	Functionality support to learn models by example	A2
Keeping meta-model	Support multi model exploration	A2

5. Conclusion and Future work

This paper presents MoVi, a proof-of-concept tool for supporting models exploration by applying interactive visualization features. MoVi presents a novel way of working with MDE models, employing a network structure. It presents various kinds of models, metamodels and relationships used in the HCI field. MoVi supports the mantra principles: overview, zoom and filter, details-on-demand, view relationship, extract and search for extending visualization feature design in order to support multi model exploration. The first users' feedbacks show that MoVi is useful even if it needs to be improved in terms of usability and exploration features.

Moreover, exploring models is complementary to other actions such as modifying or adding models directly in the MoVi environment. So we would like to integrate some model editors into MoVi. Models could be processed for MoVi (semi) automatically. Currently, we mostly use pre-defined model collection and preprocess them manually. Therefore, automated model collection could help in providing ready-state model instants. Moreover such automation could permit to realize some quality

Table 6. User experience to use MoVi - Weaknesses

Weak points	**Analysis**	**Eval. group**
Filtering is a bit messy	Usable to explore models	A1
Abbreviation in search (Task Model = TM)	Usable to explore models	A1
Do not use same colour for node and link	Usable to explore models	A1
Add select all button in filter	Usable to explore models	A1
The label of the models are not well positioned	Usable to explore models	A1
Menu could be fixed	Usable to explore models	A1
Difficulty to search model Based on the model group	Usable to explore models	A1
Change icon refresh	Usable to explore models	A1
Overview is not clear to distinguish selected models	Usable to explore models	A1
There is colour but no shape as a visual variable	Visual encoding design supports model exploration	A2
Tree exploration with "contextual" node with low opacity	Functionality support to learn models	A2
Add matrix displays for bigger number of models	Functionality support to learn models	A2
Add version relationship to know different model versions	Functionality support to learn models	A2

measurements and controls. A seminal work related to this approach is using Model Driven Engineering metrics for real time system (Monperrus *et al.*, 2008).

In the future, MoVi can also be improved in several other directions. We could incorporate algorithms from data mining and machine learning for model clustering. It would provide more possibilities to group models depending on different categories (functionality, model pattern, etc). We expect that in this way, users will get more knowledge from existing models. We would also like investigating 3D rendering in order to provide better model understanding (Pilgrim, Duske, 2008). An history feature could be interesting to log user activity when exploring models.

Even if it remains a lot of work to finalize MoVi, it seems to be a good solution to explore model ecosystems thanks to interactive visualization features, particularly thanks to the seeking mantra. We hope that this idea can be a first step in a new way of managing models.

Acknowledgements

Les auteur remercient le cluster Connexion (Programme d'Investissements d'avenir / Fonds national pour la société numérique / Usages, services et contenus innovants) pour son soutien financier.

References

Auber D. (2004). Tulips a huge graph visualization framework. In *Graph drawing software*, pp. 105–126. Springer.

Ball T., Eick S. G. (1996). Software visualization in the large. *Computer*, Vol. 29, No. 4, pp. 33–43.

Bastian M., Heymann S., Jacomy M. *et al*. (2009). Gephi: an open source software for exploring and manipulating networks. *ICWSM*, Vol. 8, pp. 361–362.

Benelallam A., Gomez A., Sunye G., Tisi M., Launay D. (2014). Neo4EMF, a scalable persistence layer for EMF models. In *Modelling foundations and applications*, pp. 230–241. Springer.

Bihanic D., Polacsek T. (2012). Models for visualisation of complex information systems. In *16th international conference on information visualisation (IV' 2012)*, pp. 130–135. Montpellier, France.

Blouin A., Combemale B., Baudry B., Beaudoux O. (2015). Kompren: modeling and generating model slicers. *Software & Systems Modeling*, Vol. 14, No. 1, pp. 321-337. Retrieved from http://dx.doi.org/10.1007/s10270-012-0300-x

Blouin A., Moha N., Baudry B., Sahraoui H., Jézéquel J.-M. (2015). Assessing the use of slicing-based visualizing techniques on the understanding of large metamodels. *Information and Software Technology*, Vol. 62, No. 0, pp. 124 - 142. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950584915000373

Bostock M., Ogievetsky V., Heer J. (2011). D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 2301–2309.

Bull R. I., Favre J.-M. (2005). Visualization in the context of model driven engineering. *MDDAUI*, Vol. 159.

Calvary G., Coutaz J., Thevenin D., Limbourg Q., Bouillon L., Vanderdonckt J. (2003). A unifying reference framework for multi-target user-interfaces. *Interacting with Computers*, Vol. 15, No. 3, pp. 289-308.

Card S. K., Mackinlay J. D., Shneiderman B. (1999). *Readings in information visualization: using vision to think*. Morgan Kaufmann.

Chi E. H.-h., Riedl J. T. (1998). An operator interaction framework for visualization systems. In *Proceedings of the ieee symposium on information visualization*, pp. 63–70.

Clasen C., Del Fabro M. D., Tisi M. (2012). Transforming very large models in the cloud: a research roadmap. In *First international workshop on model-driven engineering on and for the cloud.*

Craft B., Cairns P. (2005). Beyond guidelines: what can we learn from the visual information seeking mantra? In *Proceedings. ninth international conference on information visualisation*, pp. 110–118.

Da Silva P. P. (2001). User interface declarative models and development environments: A survey. In *Interactive systems design, specification, and verification*, pp. 207–226. Springer.

Ducasse S., Lanza M., Tichelaar S. (2000). Moose: an extensible language-independent environment for reengineering object-oriented systems. In *Proceedings of the second international symposium on constructing software engineering tools (CoSET 2000)*, Vol. 4.

Engel K., Sommer O., Ertl T. (2000). *A framework for interactive hardware accelerated remote 3d-visualization*. Springer.

Fayyad U. M., Wierse A., Grinstein G. G. (2002). *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann.

Garcia J., Theron R., Garcia F. (2010). Innovations and advances in computer sciences and engineering. In T. Sobh (Ed.),, pp. 325–330. Dordrecht, Springer Netherlands. Retrieved from http://dx.doi.org/10.1007/978-90-481-3658-2_56

Garcia Frey A., Sottet J.-S., Vagner A. (2014). Ame: an adaptive modelling environment as a collaborative modelling tool. In *Proceedings of the 2014 ACM SIGCHI symposium on engineering interactive computing systems*, pp. 189–192.

Genaro Motti V., Raggett D., Van Cauwelaert S., Vanderdonckt J. (2013). Simplifying the development of cross-platform web user interfaces by collaborative model-based design. In *Proc. of the 31st ACM int. conf. on design of communication*, pp. 55–64. New York, NY, USA, ACM. Retrieved from http://doi.acm.org/10.1145/2507065.2507067

Gomez A., Tisi M., Sunye G., Cabot J. (2015). Map-based transparent persistence for very large models. In *Fundamental approaches to software engineering*, pp. 19–34. Springer.

Guychard C., Guerin S., Koudri A., Beugnard A., Dagnat F. (2013). Conceptual interoperability through models federation. In *Semantic information federation community workshop.*

Heer J., Shneiderman B. (2012). Interactive dynamics for visual analysis. *Queue*, Vol. 10, No. 2, pp. 30.

Kent S. (2002). Model driven engineering. In *Integrated formal methods*, pp. 286–298.

Lanza M., Ducasse S. (2005). Codecrawler-an extensible and language independent 2d and 3d software visualization tool. *Tools for Software Maintenance and Reengineering, RCOST/Software Technology Series*, pp. 74–94.

Le Pallec X., Dupuy-Chessa S. (2012). Intégration de métriques de qualité des diagrammes et des langages dans l'outil ModX. In *Conférence en ingénieriE du logiciel (CIEL)*, pp. 1–6. Rennes, France.

Maletic J. I., Leigh J., Marcus A. (2001). Visualizing software in an immersive virtual reality environment. In *Proceedings of ICSE*, Vol. 1, pp. 12–13.

Monperrus M., Jézéquel J.-M., Champeau J., Hoeltzener B. (2008). Model-driven engineering metrics for real time systems. In *4th european congress ERTS embedded real-time software.*

Montero F., Lopez-Jaquero V. (2007). IdealXML: An interaction design tool. In G. Calvary, C. Pribeanu, G. Santucci, J. Vanderdonckt (Eds.), *Computer-aided design of user interfaces V*, p. 245-252. Springer Netherlands. Retrieved from http://dx.doi.org/10.1007/978-1-4020-5820-2_20

Moody D. (2009, November). The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.*, Vol. 35, No. 6, pp. 756–779.

Nassar M. (2003). Vuml: a viewpoint oriented uml extension. In *Proceedings of the 18th ieee international conference on automated software engineering*, pp. 373–376.

Paterno F., Mancini C., Meniconi S. (1997). Concurtasktrees: A diagrammatic notation for specifying task models. In *Human-computer interaction INTERACT'97*, pp. 362–369.

Pilgrim J. von, Duske K. (2008). Gef3d: a framework for two-, two-and-a-half-, and three-dimensional graphical editors. In *Proceedings of the 4th ACM symposium on software visualization*, pp. 95–104.

Puerta A., Eisenstein J. (2002). Ximl: a common representation for interaction data. In *Proceedings of the 7th international conference on intelligent user interfaces*, pp. 214–215.

Ropinski T., Preim B. (2008). Taxonomy and usage guidelines for glyph-based medical visualization. In *Simvis*, pp. 121–138.

Santos S. dos, Brodlie K. (2004, June). Gaining understanding of multivariate and multidimensional data through visualization. *Computers and Graphics*, Vol. 28, No. 3, pp. 311–325.

Scheidgen M., Zubow A. (2012). Map/reduce on EMF models. In *Proc. of the 1st int. workshop on model-driven engineering for high performance and cloud computing*, p. 7.

Schmidt D. C. (2006, February). Model-driven engineering. *IEEE Computer*, Vol. 39, No. 2. Retrieved from http://www.truststc.org/pubs/30.html

Shneiderman B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE symposium on visual languages*, pp. 336–. Washington, DC, USA, IEEE Computer Society.

Sottet J.-S., Calvary G., Favre J.-M., Coutaz J. (2009). Megamodeling and metamodel-driven engineering for plastic user interfaces: MEGA-UI. In A. Seffah, J. Vanderdonckt, M. Desmarais (Eds.), *Human-centered software engineering*, p. 173-200. Springer London. Retrieved from http://dx.doi.org/10.1007/978-1-84800-907-3_8

Sottet J.-S., Ganneau V., Calvary G., Coutaz J., Demeure A., Favre J.-M. *et al.* (2007). Model-driven adaptation for plastic user interfaces. In *Human-computer interaction–INTERACT 2007*, pp. 397–410. Springer.

Sottet J.-S., Vagner A. (2013). Genius: Generating usable user interfaces. *arXiv preprint arXiv:1310.1758*.

Telea A. C. (2014). *Data visualization: principles and practice*. CRC Press.

Viegas F. B., Wattenberg M., Van Ham F., Kriss J., McKeon M. (2007). Manyeyes: a site for visualization at internet scale. *Transactions on Visualization and Computer Graphics*, Vol. 13, No. 6, pp. 1121–1128.

Yu H., Ma K.-L., Welling J. (2004). A parallel visualization pipeline for terascale earthquake simulations. In *Proceedings of the 2004 ACM/IEEE conference on supercomputing*, p. 49.

Article soumis le 5/06/2015

Accepté le 15/02/2016