Applying Transformation Templates to Diversify User Interfaces Generated by Model-Driven Engineering [⋆]

Nathalie Aquino[1], Luca Cernuzzi[1], and Oscar Pastor[2]

[1] Departamento de Electrónica e Informática,
Universidad Católica "Nuestra Señora de la Asunción",
Tte. Cantaluppi y G. Molinas, Asunción, Paraguay
nathalie.aquino@uc.edu.py, lcernuzz@uc.edu.py

[2] Centro de Investigación en Métodos de Producción de Software,
Universitat Politècnica de València,
Camino de Vera s/n, 46022 Valencia, Spain
opastor@pros.upv.es

Abstract. Model-driven engineering of user interfaces aims to manage the inherent complexity of the development of user interfaces and to decrease the effort needed to develop them. However, from the end-user perspective, user interfaces generated by model-driven engineering usually present usability problems and it is not always easy to customise them. Transformation Templates arise as a means to improve model-driven engineering of user interfaces, providing several design options to customise user interfaces, at a concrete level. In this work we show how to apply Transformation Templates to OO-Method/Integranova, a model-driven engineering approach for the development of information systems. We present different user interface designs that could be generated, targeting the web platform and devices with small screen. Some of these new designs could have better usability properties than the original design.

Keywords: Model-driven engineering, MDE, user interface, transformation templates.

1 Introduction and Motivation

Historically, the development of graphical user interfaces has been challenging in terms of skills, times and resources [1]. Recently, the wide variety of hardware and software platforms from which interactive systems are accessed to, as well as the diversity of languages and tools for their development, have added more difficulties to their implementation [2].

[⋆] This work has been funded by CONACYT through the PROCIENCIA program, with resources from "Fondo para la Excelencia de la Educación e Investigación - FEEI", FONACIDE. This work has been developed under the project "Mejorando el proceso de desarrollo de software: propuesta basada en MDD" (14-INV-056).

Model-driven engineering (MDE) of user interfaces helps to manage these difficulties and diversities. In this context, several works can be mentioned, such as UsiXML [3], Maria [4], Just-UI [5], among others. Meixner et al. [6] present an interesting introduction to MDE of user interfaces. Meixner et al. [2] also analyse its past, present and future.

However, the literature also reports issues related to MDE of user interfaces [7, 8]. In general, MDE does not provide the possibility of generating different, customised user interfaces. Instead, generated interfaces are usually always similar to each other, which could be good when standard patterns are important, but which could also be an issue when end-users have special interactive requirements that cannot be satisfied with the MDE process. Furthermore, the literature has also reported usability problems in user interfaces generated by MDE [9–11].

OO-Method [5] is an object-oriented and MDE method that allows the automatic generation of software applications from conceptual models. It is supported by a commercial tool named Integranova M.E.S. In a previous work, we have analysed the usability of the user interfaces generated by OO-Method/Integranova in different platforms (desktop, web) and devices (with small, standard and large screens) [11]. We found that usability could be improved for the web platform and devices with small screen (PDAs, tablets, smartphones, etc.).

Furthermore, we have also previously presented Transformation Templates [12] as a means to improve MDE of user interfaces, providing several design options to customise user interfaces at a concrete level.

Therefore, in this work we show how to apply the Transformation Templates approach in OO-Method/Integranova, in order to show that different types of user interfaces can be generated, considering the web platform and devices with small screen. Some of these new designs could have better usability properties than the original designs.

The rest of this paper is organised as follows: Section 2 briefly describes the OO-Method/Integranova technology and presents the results of a usability evaluation of user interfaces generated with these tools. Section 3 presents the Transformation Templates approach. Section 4 applies the Transformation Templates approach to OO-Method/Integranova and presents alternative user interface designs that could be generated. Finally, Section 5 discusses about benefits of the Transformation Templates approach and the expected improvements in usability.

2 User Interfaces Generated by OO-Method/Integranova

This section presents an MDE software development method named OO-Method and its corresponding tool, Integranova. Furthermore, the Presentation Model of OO-Method is described. The section ends presenting a usability evaluation of the user interfaces generated using this tool.

2.1 OO-Method/Integranova and its Presentation Model

OO-Method [5] is a software development method that is MDE-compliant. It involves models of the future interactive system at different levels of abstraction and provides a transformation mechanism among them. It is supported by a commercial software suite named Integranova M.E.S., which was developed by Integranova Software Solutions [3].

Four system views are specified in the OO-Method development process: 1) the Object Model; 2) the Dynamic Model; 3) the Functional Model; and 4) the Presentation Model. Once these models are achieved, they are submitted to model compilation. For different possible target computing platforms (C# or ASP running on .NET or .NET 2.0; EJB, JSP, or JavaServer Faces running on Java), the source code of a fully functional application is automatically generated and structured according to a three-tiered architecture: interface, application, and persistence.

The OO-Method Presentation Model is composed of interaction units, which, in turn, are composed of elementary patterns.

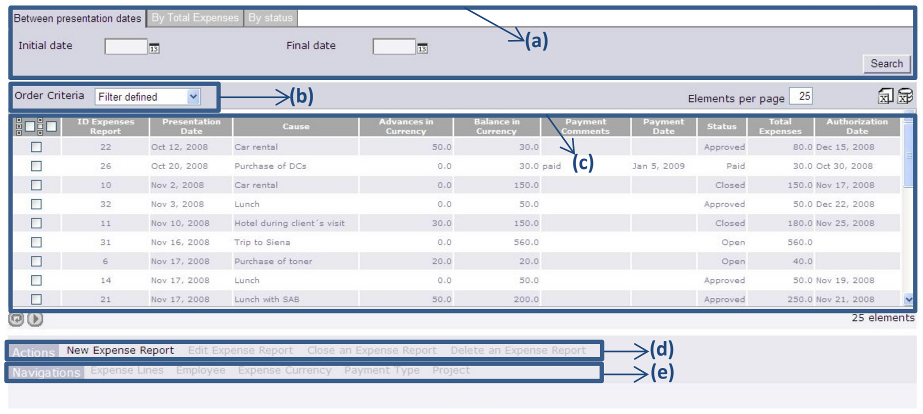
Interaction units represent the main interactive operations that can be performed on domain objects. There are four interaction units: 1) the Service Interaction Unit (SIU), which allows the modification of objects, their attributes, and relationships; 2) the Population Interaction Unit (PIU), which shows a group of similar objects; 3) the Instance Interaction Unit (IIU), which shows a single object at a time; and 4) the Master/Detail Interaction Unit (MDIU), which shows a hierarchical view of relationships among objects.

Elementary patterns are used to build interaction units and to restrict and specify their behaviour. For example, if a PIU is being specified, then five elementary patterns could be attached to it: 1) the Filter pattern, which filters a set of objects to display only the needed ones; 2) the Order Criterion pattern, that specifies the order in which objects will be shown; 3) the Display Set pattern, that restricts which attributes of objects are going to be presented; 4) the Navigation pattern, which specifies navigation among objects; and 5) the Action pattern, which specifies functions that can be triggered from a selected object.

As an example, Figure 1 presents a PIU generated by OO-Method/Integranova, indicating its corresponding elementary patterns. The example corresponds to an Expenses Report application that allows the expenses of the employees of an organisation to be managed. In particular, this user interface corresponds to a list of all the expense registries.

From the explanation of the OO-Method Presentation Model decomposition, the SIU and PIU can be considered to be the most relevant interaction units of the approach, since the MDIU is a composition of other interaction units, and the IIU is a special case of a PIU in which just one object at a time is shown. Therefore, in this work we focus on SIUs and PIUs.

[3] http://www.integranova.com



	ID Expenses Report	Presentation Date	Cause	Advances in Currency	Balance in Currency	Payment Comments	Payment Date	Status	Total Expenses	Authorization Date
☐	22	Oct 12, 2008	Car rental	50.0	30.0			Approved	80.0	Dec 15, 2008
☐	26	Oct 20, 2008	Purchase of DCs	0.0	30.0	paid	Jan 5, 2009	Paid	30.0	Oct 30, 2008
☐	10	Nov 2, 2008	Car rental	0.0	150.0			Closed	150.0	Nov 17, 2008
☐	32	Nov 3, 2008	Lunch	0.0	50.0			Approved	50.0	Dec 22, 2008
☐	11	Nov 10, 2008	Hotel during client´s visit	30.0	150.0			Closed	180.0	Nov 25, 2008
☐	31	Nov 16, 2008	Trip to Siena	0.0	560.0			Open	560.0	
☐	6	Nov 17, 2008	Purchase of toner	20.0	20.0			Open	40.0	
☐	14	Nov 17, 2008	Lunch	0.0	50.0			Approved	50.0	Nov 19, 2008
☐	21	Nov 17, 2008	Lunch with SAB	50.0	200.0			Approved	250.0	Nov 21, 2008

Fig. 1. User interface generated from a PIU with filter (a), order criterion (b), display set (c), actions (d), and navigations (e). Corresponds to the list of expense registries of the Expenses Report application.

2.2 Usability Evaluation of User Interfaces Generated by OO-Method/Integranova

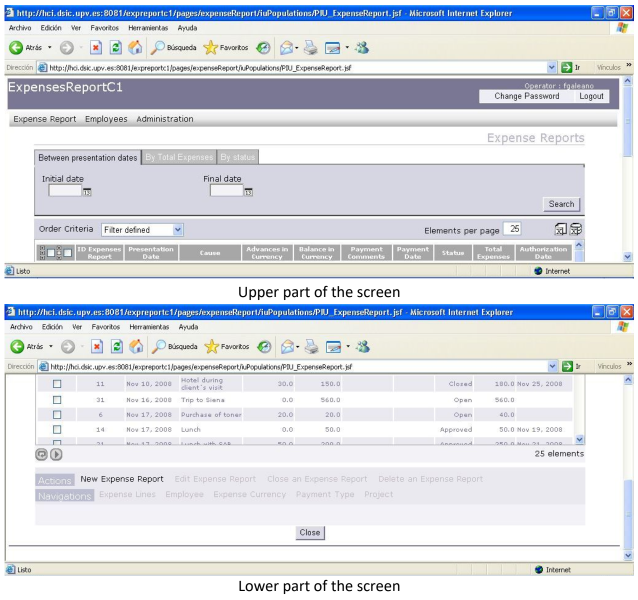
As earlier commented, we have previously evaluated the usability of user interfaces generated by OO-Method/Integranova. An exploratory usability evaluation was carried out in an experimental controlled context. All details concerning this empirical study (motivation, planning, operation, analysis of data, interpretation of results) were reported in [11]. This section presents just a brief description, including a summary of results.

Usability was examined in terms of satisfaction, effectiveness and efficiency. The evaluation focused on interfaces generated from PIUs and SIUs of the previously mentioned Expenses Report application. Furthermore, the evaluation was carried out using different platform versions of the interfaces (web and desktop) and using devices with different screen sizes (small, standard and large).

Figure 2 and Figure 3 present two of the twelve different combinations of interaction unit (PIU, SIU), platform (web, desktop), and screen size (small, standard, large) that were evaluated. Figure 2 corresponds to a PIU that presents the list of expense registries of the Expenses Report application. Figure 3 corresponds to a SIU that allows new expense registries to be added. Both figures correspond to the web platform and the device with small screen (an ultra-mobile PC with 7” screen).

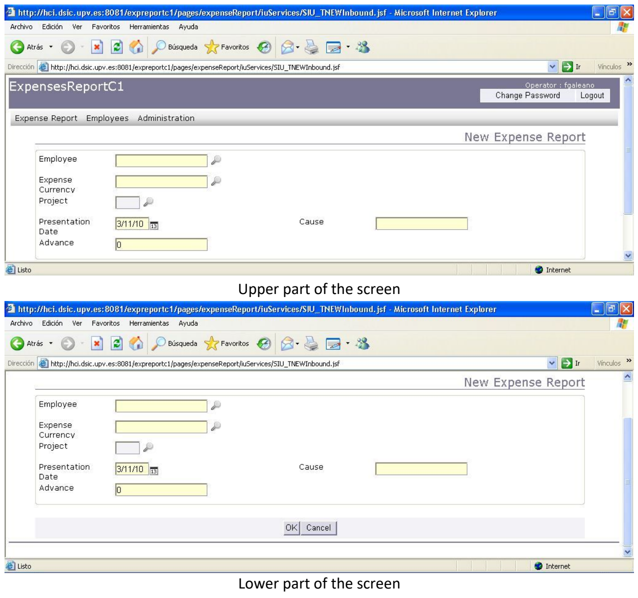
Table 1 summarises the results of this experiment. Perceptions of satisfaction and effectiveness were good, although there is still room for improvements. Efficiency results were most affected by the use of different platforms and devices.

In general, the web platform obtained worse results than the desktop platform. And the device with the small screen obtained worse results than devices with standard or large size screens.


http://hci.dsic.upv.es:8081/expreportc1/pages/expenseReport/iuPopulations/PIU_ExpenseReport.jsf - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://hci.dsic.upv.es:8081/expreportc1/pages/expenseReport/iuPopulations/PIU_ExpenseReport.jsf Ir Vínculos
ExpensesReportC1
Operator : fgaleano
Change Password Logout
Expense Report Employees Administration
Expense Reports
Between presentation dates By Total Expenses By status
Initial date
Final date
Search
Order Criteria Filter defined
Elements per page 25
Listo Internet
Upper part of the screen
25 elements
Actions New Expense Report Edit Expense Report Close an Expense Report Delete an Expense Report
Navigations Expense Lines Employee Expense Currency Payment Type Project
Close
Lower part of the screen

	ID Expenses Report	Presentation Date	Cause	Advances in Currency	Balance in Currency	Payment Comments	Payment Date	Status	Total Expenses	Authorization Date
☐	11	Nov 10, 2008	Hotel during client´s visit	30.0	150.0			Closed	180.0	Nov 25, 2008
☐	31	Nov 16, 2008	Trip to Siena	0.0	560.0			Open	560.0	
☐	6	Nov 17, 2008	Purchase of toner	20.0	20.0			Open	40.0	
☐	14	Nov 17, 2008	Lunch	0.0	50.0			Approved	50.0	Nov 19, 2008
☐	21	Nov 17, 2008	Lunch with SAP	50.0	200.0			Approved	250.0	Nov 21, 2008

Fig. 2. User interface generated from a PIU that lists the expense registries of the Expenses Report application, as it was seen in the web platform and device with small screen


http://hci.dsic.upv.es:8081/expreportc1/pages/expenseReport/iuServices/SIU_TNEWInbound.jsf - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://hci.dsic.upv.es:8081/expreportc1/pages/expenseReport/iuServices/SIU_TNEWInbound.jsf Ir Vínculos
ExpensesReportC1
Operator : fgaleano
Change Password Logout
Expense Report Employees Administration
New Expense Report
Listo Internet
Upper part of the screen
OK Cancel
Lower part of the screen

Employee			
Expense Currency			
Project			
Presentation Date	3/11/10	Cause	
Advance	0		

Fig. 3. User interface generated from a SIU that allows new registries to be added in the Expenses Report application, as it was seen in the web platform and device with small screen

Table 1. Summary of results of a usability evaluation of user interfaces generated by OO-Method/Integranova

Usability Aspect	Results
Satisfaction	- The perception of satisfaction was good for user interfaces generated from PIUs and SIUs.
	- Devices with different screen sizes have not affected the satisfaction perceived using interfaces generated from PIUs or SIUs.
	- Different platforms have not affected the satisfaction related to user interfaces generated from PIUs. In the case of SIUs, satisfaction was slightly better using the desktop platform than the web platform.
Effectiveness	- Subjects reached high percentages of task completion (means above 80%) when using interfaces generated from PIUs and SIUs. The achieved effectiveness is good.
	- The effectiveness related to interfaces generated from PIUs has not been affected by the use of devices with different screen sizes or diverse platforms.
	- The effectiveness related to interfaces generated from SIUs has been affected by different combinations of device and platform. Effectiveness was better using the combination of small screen with desktop platform than using the combination of standard size screen with web platform.
Efficiency	- The efficiency related to interfaces generated from PIUs and SIUs has been affected by the use of devices with different screen sizes. In the case of PIUs, efficiency was better using the large screen than using the small screen. In the case of SIUs, efficiency was better using standard or large screens than using the small screen.
	- The efficiency related to interfaces generated from PIUs and SIUs has been affected by different platforms. In both cases, efficiency was better using the desktop platform than using the web platform.

Currently, the OO-Method Presentation Model allows a user interface to be specified in a way that is independent from platforms and devices, and the transformation logic is internally defined in the Integranova tool, which generates the user interface code. It can be considered that the tendency to have better results for standard or large screens and for the desktop platform is related to the fact that the OO-Method/Integranova approach is mainly used to develop organisational information systems [5]. It can also be considered that the device with the small screen obtained worst results because the kinds of user interfaces that people are used to using in small devices are different from the types of user interfaces generated with Integranova.

Therefore, the OO-Method/Integranova approach should incorporate enhancements in order to generate multi-device/platform user interfaces with improved usability.

3 Transformation Templates

A Transformation Template [12] aims to explicitly specify the structure, layout, and style of a user interface, according to preferences and requirements of end-users, as well as in line with different hardware and software computing platforms and environments in which the user interface will be used.

The main concepts that characterise the Transformation Templates approach are related to context, to user interface models, and to the Transformation Templates themselves.

Context refers to the context of use of an interactive system. We have defined context according to the Cameleon Reference Framework [13], which is widely accepted in the human-computer interaction community. According to this framework, a context of use is composed of the stereotype of a user, who carries out an interactive task, with a specific computing platform in a given surrounding environment. Conceptualising context, we can define Transformation Templates for different contexts of use.

A *user interface meta-element* represents, in a generic way, any meta-element of a user interface meta-model. A *user interface element* represents an element of a user interface model. These generic representations allow the Transformation Templates approach to be used with different MDE approaches.

A *parameter type* represents a design or presentation option related to the structure, layout or style of the user interface. Defining a parameter type subsumes specifying the list of user interface meta-elements that are affected by it, as well as its value type. The *value type* refers to a specific data type (e.g., integer, URI, colour, etc.) or to an enumeration of the possible values that a parameter type can assume. A parameter type, with all or a set of its possible values, can be implemented in different contexts of use. In order to facilitate decision-making regarding these implementations, we propose that each possible value receive an estimation of its importance level and its development cost for different relevant contexts of use. In this way, possible values with a high level of importance and a low development cost can be implemented first in a given context, followed by the other options if appropriate. Furthermore, for each relevant context of use, usability guidelines can be assigned to each possible value of a parameter type. These guidelines will help user interface designers in choosing one of the possible values by explaining the conditions under which the values should be used.

Table 2 shows an example of the definition of a parameter type named *Filter Widget*. This parameter type is useful for deciding how to present filters in PIUs. Three different possible values have been defined: tabs, combo and accordion. The parameter type has been associated to two contexts of use: desktop and web platforms. Furthermore, Table 3 and Table 4 show that for each context of use and each possible value, importance level and development cost have been estimated, and usability guidelines have been proposed. These usability guidelines were extracted from [14].

A *transformation template* gathers a set of *parameters* for a specific context of use. Each parameter of a Transformation Template corresponds to a parameter

Table 2. Parameter Type: Filter Widget

Name	Description	Affects To	Possible Values	Contexts of Use
Filter Widget	Allows to select the widget to be used to present filters of a PIU	Filter	Tabs Combo Accordion	Desktop Web

Table 3. Parameter Type: Filter Widget - importance level, development cost and usability guidelines for the desktop platform

Value	Importance Level	Development Cost	Usability Guidelines
Tabs	high	low	The number of filters is not too big, less than 8
Combo	low	low	The number of filters is high
Accordion	low	low	The number of panels should be small, less than 8

type and has both, a value and a selector. A *value* is an instance of a value type. The value of a parameter corresponds to a possible value of the corresponding parameter type. A *selector* delimits the set of user interface elements that are affected by the value of a parameter. We have defined different types of selectors that allow the designer to choose a specific user interface element, all the user interface elements of a certain type, the first or last element contained in a specific type of user interface element, or other options.

As a simple example of a transformation template, Table 5 represents a fragment of a transformation template defined for a context of use that addresses a web platform and devices with small screens. This transformation template uses the Filter Widget parameter type to specify that all filters of PIUs will be displayed using accordions (first row), except for the filter with id=12, that will be displayed using a combo (second row).

Transformation Templates add flexibility in MDE approaches because they externalise the design knowledge and presentation guidelines, and make them customisable according to the characteristics of the project being carried out. At the same time, they diversify the kinds of user interfaces that an MDE approach

Table 4. Parameter Type: Filter Widget - importance level, development cost and usability guidelines for the web platform

Value	Importance Level	Development Cost	Usability Guidelines
Tabs	high	low	The number of filters is not too big, less than 8
Combo	low	low	The number of filters is high
Accordion	high	low	The number of panels should be small, less than 8

Table 5. Transformation Template for web platform and devices with small screen

Parameter	Value	Selector
Filter Widget	Accordion	Filter
Filter Widget	Combo	Filter with id=12
...	...	...

Table 6. Parameter Type: PIU Layout

Name	Description	Affects To	Possible Values
PIU Layout	Provides options for positioning elementary patterns of a PIU	PIU	PIU layout 1 PIU layout 2 PIU layout 3

can generate. Furthermore, Transformation Templates defined for one project can later be reused in other similar projects, targeting similar contexts of use.

It is worth noting that Transformation Templates do not replace any implicit transformation logic or explicit transformation languages; instead, they provide a higher-level tier for user interface designers to specify user interface transformations.

4 Applying the Transformation Templates Approach to OO-Method/Integranova

The definition of parameter types is among the first steps of the process to incorporate Transformation Templates in an MDE approach for user interface development. Therefore, this section presents some parameter types defined for OO-Method/Integranova. Then, we use these parameter types to suggest alternative designs for generating PIUs and SIUs, considering the web platform and devices with small screen.

4.1 Parameter Types for OO-Method/Integranova

Besides Filter Widget, which was previously presented, this section introduces four more parameter types for OO-Method/Integranova. For space reasons, estimations of importance level and developments cost, as well as usability guidelines, are omitted.

Table 6 presents the *PIU Layout* parameter type, which allows different configurations for the position of the elementary patterns that conform a PIU (filter, display set, actions, navigations) to be chosen. Figure 4 presents a graphical representation of the possible values.

Table 7 presents the *Display Set Layout* parameter type, which allows to specify if display sets are going to be displayed with tables or using a set of independent fields.

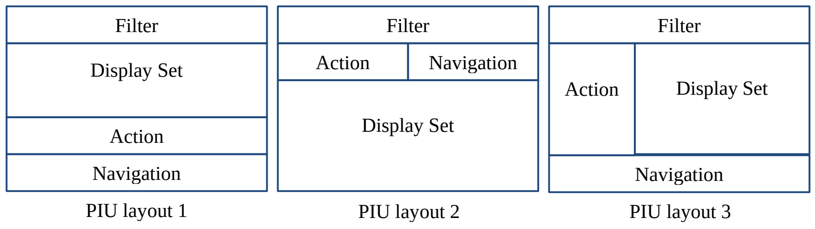


Fig. 4. Graphical representation of the possible values of PIU Layout

Table 7. Parameter Type: Display Set Layout

Name	Description	Affects To	Possible Values
Display Set Layout	Allows to specify if a display set is going to be displayed with a table or with a set of independent fields	Display Set	Table Reduced table Independent fields

Table 8 presents the *SIU Layout* parameter type, which allows to specify if input arguments of a SIU are going to be displayed in one or two columns, or if they are going to be placed according to the current algorithm used by Integranova.

Table 9 presents the *Label Alignment* parameter type, which allows the alignment of a label with regard to an input field to be specified.

4.2 Alternative Designs for PIUs and SIUs

This section presents two alternative designs for PIUs, as well as two for SIUs, targeting a context of use composed by web platform and devices with small screen. These alternative designs could be generated if, in the future, OO-Method/Integranova incorporates the Transformation Templates approach and implements the previously defined parameter types for the considered context of use.

The first alternative design for PIU is presented in parts (a), (b), (c) and (d) of Figure 5. This design is obtained using the following parameters and values:

Table 8. Parameter Type: SIU Layout

Name	Description	Affects To	Possible Values
SIU Layout	Allows to specify if a SIU is going to be displayed in 1 or 2 columns, or following the standard algorithm	SIU	Standard algorithm 1 column 2 columns

Table 9. Parameter Type: Label Alignment

Name	Description	Affects To	Possible Values
Label Alignment	Allows the alignment of a label with regard to an input field to be specified	Filter SIU	Left Up Inside

- *PIU Layout = PIU layout 2*, therefore, filters appear on top (in blue), followed by actions (in orange) and navigations (in green) at the same level, and the display set appears at the bottom (see Figure 5(a)).
- *Filter Widget = Accordion*, therefore, an accordion is used to present the three different filters of this PIU. In Figure 5(b) and (d) the accordion is closed. In (c) all filter options are displayed. In (b) one filter is open to be used.
- *Display Set Layout = Independent fields*, therefore, registries are not displayed using a table, but as a sequence of independent fields (see Figure 5(a)).
- *Label Alignment = Inside*, therefore labels of input arguments in the filter appear inside the input texts (see Figure 5(b)).

The second alternative design for PIU is presented in Figure 5(e). This design is obtained using the following parameters and values:

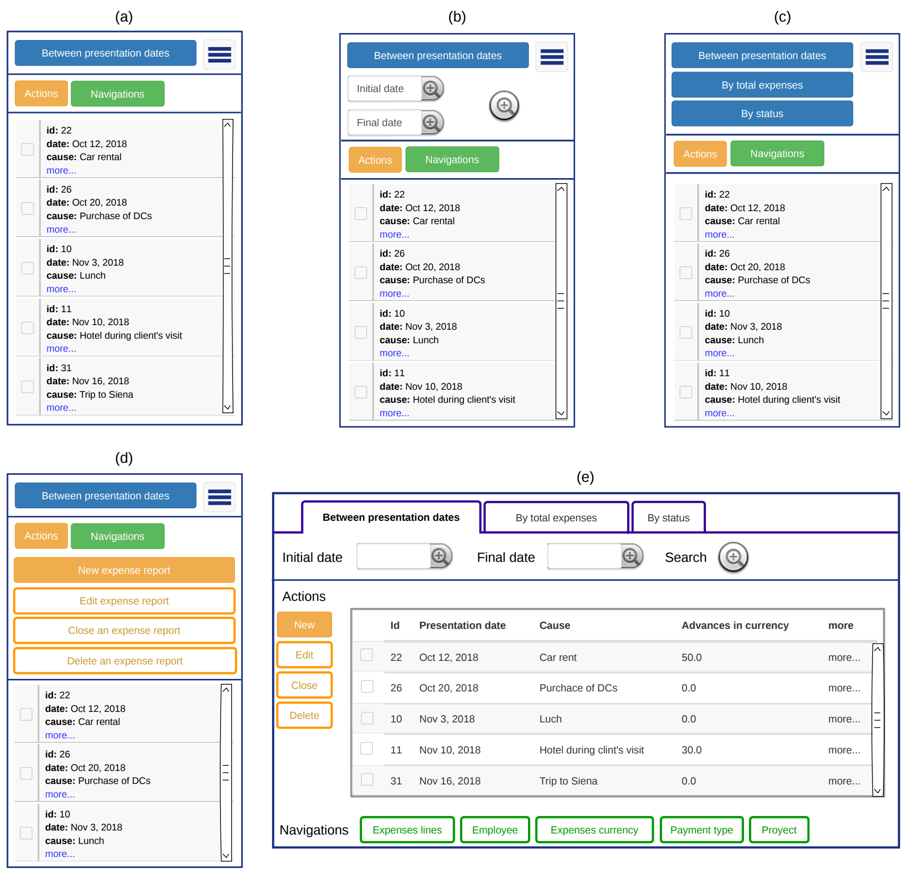
- *PIU Layout = PIU layout 3*, therefore, filters appear on top, actions appear to the left, the display set appears to the right, and navigations appear at the bottom.
- *Filter Widget = Tab*, therefore, tabbed panels are used to display the three different filters of this PIU.
- *Display Set Layout = Reduced table*, therefore, registries are displayed using a table, but only four of a total of ten columns are displayed (see Figure 2), providing also a link to access a different view with the missing columns.
- *Label Alignment = Left*, therefore labels of input arguments in the filter appear to the left of input texts.

The first alternative design for SIU is presented in Figure 6(a). This design is obtained using the following parameters and values:

- *SIU Layout = 1 column*, therefore, input arguments are presented in one column.
- *Label Alignment = Inside*, therefore, labels of input arguments appear inside the input texts.

The second alternative design for SIU is presented in Figure 6(b). This design is obtained using the following parameters and values:

- *SIU Layout = 2 columns*, therefore, input arguments are presented in two columns.
- *Label Alignment = Up*, therefore, labels of input arguments appear above input texts.



(a)

id: 22
date: Oct 12, 2018
cause: Car rental
more...

id: 26
date: Oct 20, 2018
cause: Purchase of DCs
more...

id: 10
date: Nov 3, 2018
cause: Lunch
more...

id: 11
date: Nov 10, 2018
cause: Hotel during client's visit
more...

id: 31
date: Nov 16, 2018
cause: Trip to Siena
more...

(b)

id: 22
date: Oct 12, 2018
cause: Car rental
more...

id: 26
date: Oct 20, 2018
cause: Purchase of DCs
more...

id: 10
date: Nov 3, 2018
cause: Lunch
more...

id: 11
date: Nov 10, 2018
cause: Hotel during client's visit
more...

(c)

id: 22
date: Oct 12, 2018
cause: Car rental
more...

id: 26
date: Oct 20, 2018
cause: Purchase of DCs
more...

id: 10
date: Nov 3, 2018
cause: Lunch
more...

id: 11
date: Nov 10, 2018
cause: Hotel during client's visit
more...

(d)

id: 22
date: Oct 12, 2018
cause: Car rental
more...

id: 26
date: Oct 20, 2018
cause: Purchase of DCs
more...

id: 10
date: Nov 3, 2018
cause: Lunch
more...

(e)

Id	Presentation date	Cause	Advances in currency	more
22	Oct 12, 2018	Car rent	50.0	more...
26	Oct 20, 2018	Purchase of DCs	0.0	more...
10	Nov 3, 2018	Luch	0.0	more...
11	Nov 10, 2018	Hotel during clint's visit	30.0	more...
31	Nov 16, 2018	Trip to Siena	0.0	more...

Fig. 5. Alternative designs for PIU

5 Discussion

Applying the Transformation Templates approach to OO-Method/Integranova we can obtain the following consequences: 1) the transformation logic becomes explicit and customisable, since it can be expressed in transformation templates by means of parameters with their corresponding values and selectors; 2) it becomes possible to target specific contexts of use, including specific platforms, since transformation templates implies that a specific context of use must be selected; 3) it becomes possible to generate different types of user interfaces, using the different possible values of parameter types, as presented in Section 4.2; and 4) among the different user interfaces that can be generated using Transformation Templates, some of them could have better usability properties than the original designs, considering a particular context of use. At this point, it is worth noting that currently, the Integranova tool has not implemented yet the Transformation Templates approach, therefore these are expected benefits.

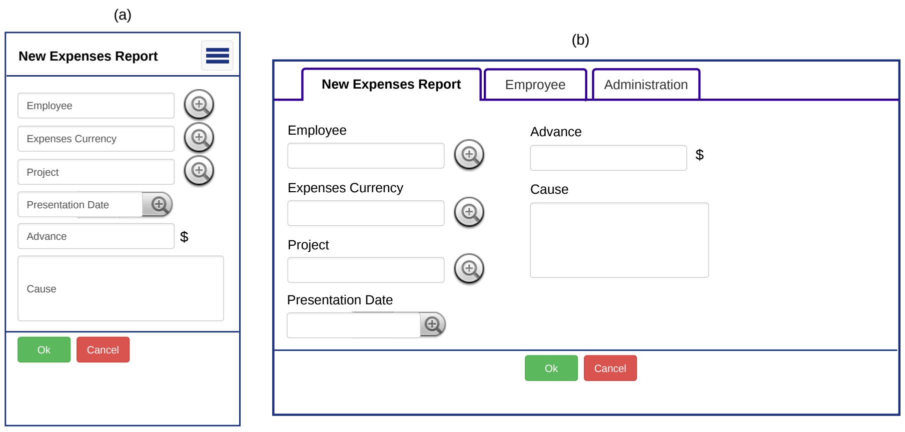


Fig. 6. Alternative designs for SIU

Elaborating a little bit more about usability problems, it is useful to highlight that our approach considers that for each context of use, usability guidelines should be provided for all possible values of a parameter type. Therefore, when defining a transformation template, the designer will be able to see these guidelines and to choose the most appropriate value, considering the corresponding context of use. For instance, it could be possible that the alternative designs for PIU presented in Figure 5 obtain better usability results than the design presented in Figure 2, since usability guidelines for small devices suggest that: 1) the most relevant information (the content of the display set in this case) should be presented first, in upper and central positions; 2) the use of tables should be avoided. As a future work we plan to perform an empirical evaluation in order to formally verify if these new designs improve the usability with regard to the original ones.

Finally, we recognise that currently there is a trend to use responsive web design when targeting to the web. However, it is still valid to follow guidelines for specific hardware or software platforms. Furthermore, it should be taken into account that the Transformation Templates approach is not specific for the web. It can also be applied in desktop or native platforms.

References

1. Myers, B.A., Rosson, M.B.: Survey on User Interface Programming. In Bauersfeld, P., Bennett, J., Lynch, G., eds.: Conference on Human Factors in Computing Systems, CHI 1992, Monterey, CA, USA, May 3-7, 1992, Proceedings, ACM (1992) 195–202
2. Meixner, G., Paternò, F., Vanderdonckt, J.: Past, Present, and Future of Model-Based User Interface Development. i-com **10**(3) (2011) 2–11
3. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In

Bastide, R., Palanque, P.A., Roth, J., eds.: Proc. of 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems, EHCI-DSVIS 2004 (Hamburg, July 11-13, 2004). Volume 3425 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (2005) 200–220

4. Paternò, F., Santoro, C., Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact. **16**(4) (2009)
5. Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
6. Meixner, G., Calvary, G., Coutaz, J.: Introduction to Model-Based User Interfaces. Technical report, World Wide Web Consortium (2014) http://www.w3.org/TR/mbui-intro/.
7. Montero Simarro, F., López-Jaquero, V., Vanderdonckt, J., González, P., Lozano, M.D., Limbourg, Q.: Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. In Gilroy, S.W., Harrison, M.D., eds.: Interactive Systems, Design, Specification, and Verification, 12th International Workshop, DSVIS 2005, Newcastle upon Tyne, UK, July 13-15, 2005, Revised Papers. Volume 3941 of Lecture Notes in Computer Science., Springer (2005) 161–172
8. Coutaz, J.: User interface plasticity: model driven engineering to the limit! In Sukaviriya, N., Vanderdonckt, J., Harrison, M., eds.: Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System, EICS 2010, Berlin, Germany, June 19-23, 2010, ACM (2010) 1–8
9. Abrahão, S., Iborra, E., Vanderdonckt, J.: Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool. In Law, E.L.C., Hvannberg, E.T., Cockton, G., eds.: Maturing Usability - Quality in Software, Interaction and Value. Human-Computer Interaction Series. Springer (2008) 3–32
10. Schramm, A., Preußner, A., Heinrich, M., Vogel, L.: Rapid UI Development for Enterprise Applications: Combining Manual and Model-Driven Techniques. In Petriu, D.C., Rouquette, N., Haugen, Ø., eds.: Model Driven Engineering Languages and Systems - 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Proceedings, Part I. Volume 6394 of Lectures Notes in Computer Science., Springer (2010) 271–285
11. Aquino, N., Vanderdonckt, J., Condori-Fernández, N., Dieste Tubío, Ó., Pastor, O.: Usability Evaluation of Multi-Device/Platform User Interfaces Generated by Model-Driven Engineering. In Succi, G., Morisio, M., Nagappan, N., eds.: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, 16-17 September 2010, Bolzano/Bozen, Italy, ACM (2010)
12. Aquino, N., Vanderdonckt, J., Pastor, O.: Transformation Templates: Adding Flexibility to Model-Driven Engineering of User Interfaces. In Shin, S.Y., Ossowski, S., Schumacher, M., Palakal, M.J., Hung, C.C., eds.: Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010, ACM (2010) 1195–1202
13. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers **15**(3) (2003) 289–308
14. Galitz, W.O.: The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. John Wiley & Sons, Inc., New York, NY, USA (2002)