LOUVAIN
School of Management

**UNIVERSITE CATHOLIQUE DE LOUVAIN**

**LOUVAIN SCHOOL OF MANAGEMENT**

# Cross-platform user interface design patterns: the case of ERP systems.

Promoteur :

*Saerens Marco*

Copromoteur :

*Nguyen Thanh-Diane*

Mémoire-recherche présenté par

Verheggen Maxime

en vue de l'obtention du titre de
Master 120 crédits en ingénieur de gestion

ANNEE ACADEMIQUE 2015-2016

*I would like to thank Mrs. Diane Thanh Nguyen, Mr. Jean Vanderdonckt and Mr. Marco Saerens who ensured the guidance and direction of my thesis. I thank them for their help, expertise and assistance.*
*I am also thankful to all other people who provided support in the research and writing of this thesis, with a specific mention to Mr. Tristan Dorange for his advice and help.*
*Finally I thank my parents for their permanent support and encouragement in all the time of my university studies.*

**Table of content**

## List of figures

## List of tables

# 1. Introduction

## 1.1. Problem context

Nowadays, people around the world are connected permanently on the internet accessing to varied contents. All of these contents are reached through several kinds of devices having each one of them their particularities. In the last decade, these kinds of devices, referred to the platforms, has evolved widely from desktop to the smartphones passing by tablets. This evolution has gone along with the screen resolution and size, the context in which they are used as the range of users. The challenge for the developer is to offer the same user experience to each kind of these devices. To answer to that challenge three factors has to be taken into account: The Platform, the Environment and the User.

The **platforms** have changed a lot since the first computer. Now, it is possible to encounter a large variety of devices (desktop, laptop, netbook, smartphone, tablet, phablet, smart tv, smartwatch…). One of the biggest challenges with this variety is to handle the screen size. Indeed, laptops have larger screen, bigger resolution, keyboard and a pointer control. The interaction occurs with windows, icons, menus and pointers (WIMP). On the opposite, smartphones run with a much smaller display based on a touch screen. The control has to be large enough to suit the fingers. Tablets are quite the same but larger and provide a richer user interface. This platforms diversity gives a lot of trouble for the developers which have to code the same application in different programming language.

The **Users** has been also diversified from the first home computer until now. We can observe two trends in this last decade, the number of user has increased considerably and the young people are more and more used to the new technologies and manipulate them frequently. As the figure 1 shows, the percentage of American adults who use computers has doubled during 1990 and 2014. The figure 2 explicits the percentage of teens cell phone owners by age in the US between 2004 and 2008.

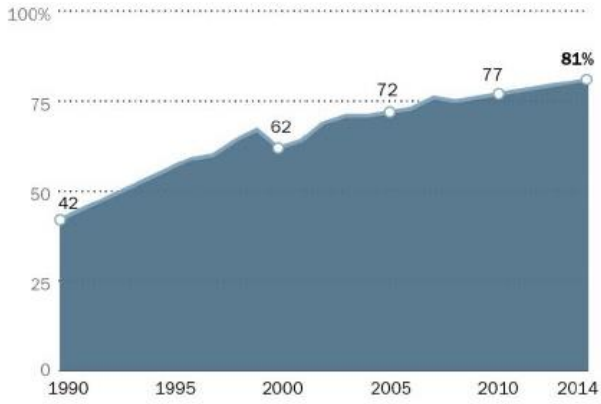



**Figure 1: The use of computer by American adults between 1990 and 2014 (Fox & Rainie, 2014)**

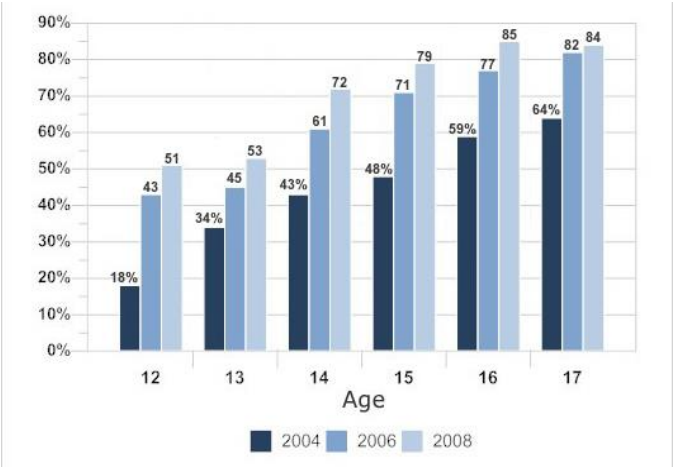



**Figure 2: Percentage of teen cell phone owners by age, 2004-2008 (Lenhart, 2009)**

The **Environment** determines how and what devices are used and what are the user expectations. In the early age of computers, as the designation desktop induces people were static, sitting on a chair and working on their desk. That was the most common context to use a computer. But with the new technologies this environment has evolved and is much more varied. Besides, users can move from an environment to another very quickly. Currently, people are using devices in motion, in the street, in the transport... They are taking away devices and they switch from one to another while keeping doing the same activity.

These three factors (platforms, users and environment) constitute the contexts of the problem. In this problematic, **patterns** come into play because they can offer solutions for developers. Indeed, they aim to provide a generic solution to recurrent problems.

Christopher Alexander defines patterns as the following way:
Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. (Alexander, 1977, p x)

### 1.2. The objective

The problem context leads to raise the general question of this master thesis which is:

“Is it feasible to apply cross-platform user interface design patterns to interactive systems?”

In order to ensure an accurate understanding of the question the terms used will be explained. The design patterns are defined by a problem, a context and a solution. It provides a standard solution to a recurring problem that can be reusable in different contexts. The user interface (UI) design patterns are specific types of design patterns usually related to the interfaces of interactive systems. Several collections of UI design patterns exist but they are mainly focused on a specific platform which raises the cross-platform issue. Indeed in the current literature, patterns have a limited level of abstraction. This low abstraction level reveals a difficulty to implement them in different contexts of use. As in the pattern collections of van Welie and Tidwell, patterns concern directly a Final User Interface in particular. This is too specific to be a generic solution for diverse contexts and therefore prevents to address a cross-platform use. Thus, the cross-platform challenge is to make applicable UI design patterns to various contexts while retaining their specificities. Finally, an interactive system corresponds to any system in general where an interaction occurs between the user and the system.

This research question is a vast topic and requires a complex multifactorial approach overtaking the scope of a master thesis. With a view to provide a potential answer, the following restrictions will be applied and justified.
The studied interactive systems are the ERP because these information and management systems are the more frequently used. Three well-known ERP softwares will be analysed (Oracle, Microsoft Dynamics and Odoo). In this last decade, ERP softwares have known a dazzling evolution and they are widely spread among the enterprises. An ERP is a software

that unifies the information system by integrating different functionality around a unique database. From a managerial point of view, it has become an indispensable tool to get a competitive advantage. So in regards to this successfully information system, it seems relevant to limit this research to this interactive system. (Interactive systems restriction)

As it is hard work to consider all the design patterns methods, this thesis is limited to the object-oriented method. The OO-method is a widely dealt topic in the literature and it is subject to a commercial software environment with for example Olivano. Contrarily to the others, it is one of the only methods which provides a minimal coverage of the interface and that can be used for the information systems. (Design pattern method restriction)

The OO-method includes a lot of patterns with possible combinations between each other's. However the concern is about ERP, an interaction system that executes services. One specific pattern complies with this kind of interaction: The Service Interaction Unit. More precisely, the SIU is in charge of the interaction between the user and the system in order to execute a service. It is one of the most employed patterns in the information systems and surprisingly it is undocumented in the literature. (Pattern restriction)

Although the proposed method has the will to be cross-platform thanks to a platform-independent decision tree, it will be illustrated only for some cases. (Illustration restriction)

These explanations and restrictions allow refining the general research question in new one more accurate. "Applied on the case of ERP systems in a cross-platform context, how to suggest a model to implement the Service Interaction Unit?"

The objective of this master thesis is to suggest a model which could facilitate the application of the SIU in a cross-platform context. This model will be applied in the ERP as study case. But to construct this model, some additional interrogations ensuing from the previous question have to be solved. What is a cross-platform pattern? And how to apply this cross platform pattern? Across this thesis, we will try to reply at best to these questions and we will convey a proposition of a model that could help the user in the graphical implementation of the SIU as a cross platform pattern.

## 1.3. Methodology

### 1.3.1. Reminder of the problem context

The factors identified in the problem context are the platforms, the environments and the users. A solution to these contexts is the patterns. In this master thesis we will see how the Service Interaction Unit can be associated with the ergonomic rules to answer at best to the user expectations. This answer will focus on the case of ERP systems.

### 1.3.2. Assumptions

To go forward in this research, some assumptions have been made.

First, we assume there is a lack of common structure for the implementation of human computer interface. So far, a lot of studies have been done on the patterns but they are mostly oriented on the patterns' description. Yet their goal to provide a reuse solution to a recurrent problem is not achieved because they don't take into account the ergonomic rules. Indeed the ergonomic rules are a really important factor in the human computer interfaces because they give some guidance in their fulfilment. Besides, ergonomic rules and patterns are addressed separately instead of being approach in a common global way. Some authors as Martijn van Welie have already tried to fulfil this lack of cohesion. It leads to the next assumption.

The existing user interface design patterns are too oriented on the final user interface and by consequences are not sufficiently general to be applied to a wild problem space. They don't take into account the early phases of the conception. This is due to a low level of abstraction. That is why Pastor and Molina introduced the conceptual patterns. Conceptual patterns are higher level of abstraction allowing to coverage of the whole life-cycle from the conception to the implementation.

The last assumption is the fact that service interaction unit pattern is not enough documented to be fully efficient.

### 1.3.3. Research methodology

The methodology followed is shown in figure 3.

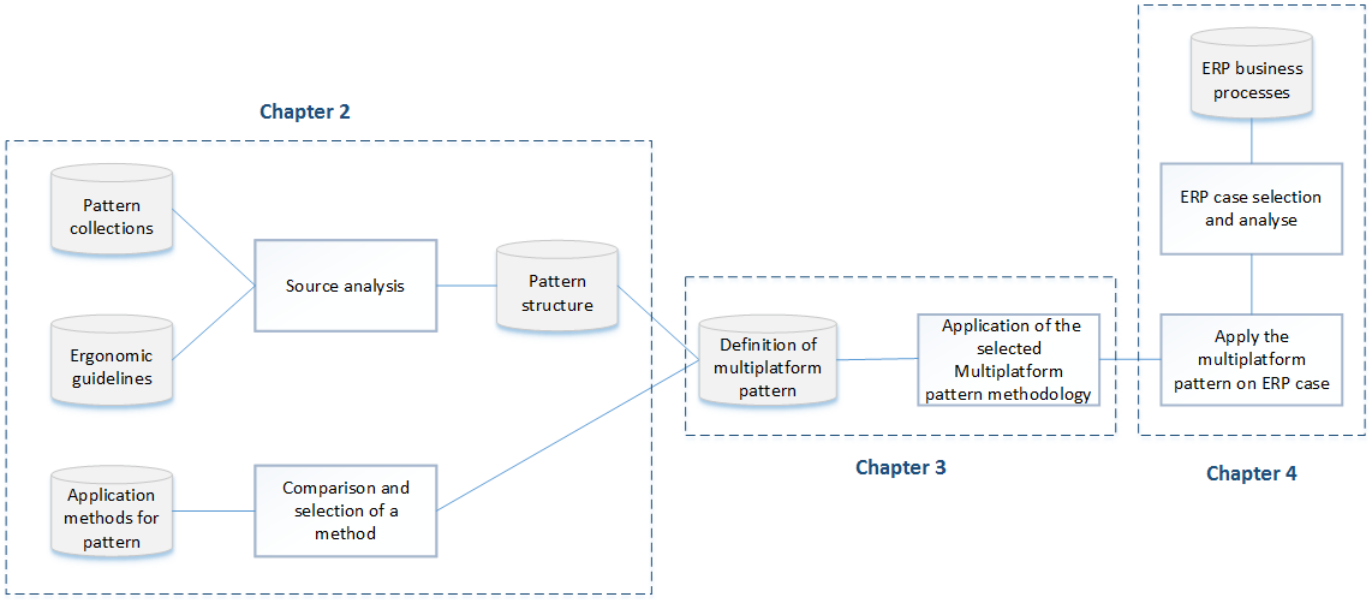



**Figure 3: The methodology diagram**

The first step is to gather adequately information in the aim to have a wide portfolio. In this purpose, we will review the existing collections of pattern, ergonomic rules and the different application methods for patterns. To refine this exercise, we will stay focus on the user interface designs. After analysing the pattern collections and ergonomic guidelines, we will be able to provide a pattern structure. Coupling with an application method, the result will answer at the question of the definition of a cross-platform pattern. This will be covered in the chapter 2, state of the art.

The second step is to answer the question how to apply this cross-platform pattern? Based on the definition of chapter two, we will illustrate the Service Interaction Unit as a multiplatform pattern. For this purpose, the SIU will be represented through the selected methodology coming from chapter 2. This second step is discussed in chapter 3.

Chapter 4 will consist of analysing several business processes common to the enterprises. Then, the most interesting case regarding to the SIU pattern will be deeper studied. Afterwards, we will compare how different ERP vendors apply the SIU for this specific case. Finally for a particular stage of the former business process, we will show how we could suggest an application of the SIU. This confrontation could lead to improve our definition or

to furnish proposition for the ERP developers. This chapter will answer to the main question by exemplifying the SIU implementation with the suggested model in the ERP.

### 1.4. Roadmap

This dissertation is divided in several sections which will guide us all along the reply process to the objectives noticed in section 1.2. Section 2 presents the state of the art of the patterns including the notion of pattern, the types of patterns, their link with ergonomic and the method to apply them. This state of the art enables the achievement of the section 3. Section 3 will detail the execution of a specific methodology on the Service Interaction Unit in order to create a model to help the user in the design interface. The SIU will be defined and examined meticulously to have a fully understanding of this pattern. Then section 4 proposes to confront this model to a study case in the scope of the ERP's. An analysis based on business process modelling will be conducted between different ERP vendors. As the outcome of this analysis, we will use the established model to suggest improvement for the interaction taken on the business process. Section 5 will sum up the thesis and will expose the possible further works.

# 2. State of the art

## 2.1. What is a pattern?

In this section, the notion of pattern will be browsed starting from its origin to its description. The goal is addressing the comprehension of what is a Pattern and what it is standing for. We will observe that patterns are not restricted to the computer science but are applied to a very broad field.

### 2.1.1. Pattern origin

The first one who illustrated the concept of pattern is Christopher Alexander. He formulated this concept in the context of the architecture of buildings. In his book, *The Timeless Way of Building,* the patterns describe a recurring problem and offer proper solutions. As result, a working method is given to everyone in order to improve the conception of building. His underlying theme is well-conceived buildings exhibit inherent and shared qualities that can be generic. “In essence, patterns are structural and behavioural features that improve the habitability of something” (Tidwell, 2010, p xviii). So in the light of this definition, the notion of pattern can be applied to a very large domain.

In psychology, a pattern is a thinking mechanism that is basic to the brain's operation, helping one to perceive things quickly. In dress making, a pattern is a pleasing shape that is applied repeatedly. In linguistics, a pattern is the manner in which smaller units of language are grouped into larger units. In archaeology, a pattern is a group of phases having several distinguishing and fundamental features in common. (Coad, 1992, p 152).

Alexander also developed a context-problem-solution structure to describe patterns (figure 4). In other words, a pattern is a solution to a recurring problem in a specific context.

- The context is a situation that creates problems.
- The problem is a problem that comes repeatedly in that situation
- The solution is the way to solve this recurring problem.

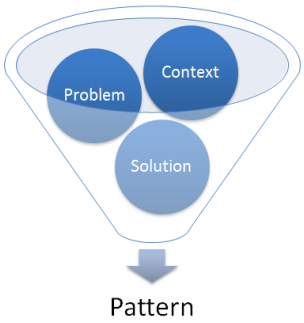



**Figure 4 : Pattern structure**

Then 4 authors (Gamma, Helm, Johnson & Vlissides, 1994) also known as the gang of four (GOF) introduced patterns in object-oriented programming. In their book *Design Patterns: Elements of Reusable Object-Oriented Software*, they expand the notion of pattern in a formalized description traduced in a precise template. This template contains several attributes that specifies the pattern characteristics (see table 1). Through a better documentation, patterns were more intelligible and popularised in order to facilitate their use. With the gang of four, patterns were used for software design pattern. But patterns are put into practice in other orientation such as human-computer interaction, software architecture, security …

### 2.1.2. Pattern definition

Patterns are, in essence, a common structure which applies to several kinds of situations and are reusable. You can reuse solutions that worked in the past to current issues. They improve the usability of things by making them more beautiful and useful. Patterns can be interpreted as a description of best practices within a domain. (Tidwell, 2010)

We can say that patterns are concrete and not general. Concrete enough because they make the link between fundamental design principles (for example “prevent errors”, don’t make the users think”...) and low-level “grammar” (such as text fields, widgets…). Apply the patterns permits to design a real functional interface while being under those principles and integrating those tools. However they need to be valid across different platforms and systems. So they are not too specific in the aim to define a sufficient abstraction which can stay true even if the context change. (Tidwell, 2010)

Patterns can be considered as different suggestions and you are free to pick or not some of them depending on your context.

Alexander defines "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander et al., 1977, p x).

In opposition with the pattern functioning, an Anti-Pattern works in the reverse way. They first illustrate what is wrong and then gives a solution. They explain how to not reproduce this mistake. But seeing mistakes can be inspiring but it does not directly help to solve problems (van Welie, van der Veer & Eliens, 2001). Jeff Johnson exposed large range of bloopers detected on websites (Johnson, 2003). These bloopers can be considered as anti-pattern.

### 2.1.3. Pattern description

The gang of four have drawn up a template to describe the pattern using attributes. This template has been a starting point to the pattern's description. Since then, a lot of authors documented patterns thanks to the GOF template. Other authors kept the idea but developed new templates in order to improve it or to suit it better the studied domain. The consequence is an abundance of diverse formats, varying sometimes attributes only by their name. A lack of standardization and coherence is striking. In table 1, we confront different templates related to well-known authors. This table is the result of analysis from (Gamma et al., 1994 ; Kruschitz & Hitz, 2010 ; Engel, Herdin & Maertin, 2012 ; Nguyen, Vanderdonckt & Seffah, 2016 b).

As we can notice in table 1, there is some convention between some authors. For example the attribute Name is common to the all of the presented formats. On the opposite, they are attributes that are specific to one format such as Literature for van Welie. Another observation is sometimes attributes refer to the same thing but have a different designation.

We can see that for Applicability (GOF) and Context (Tidwell). These remarks highlight a lack of consistency between the formats.

| Gang of four | Coplien | Portland | Tidwell | van welie | van Duyne |
|---|---|---|---|---|---|
| / | / | / | / | / | Pattern ID |
| Pattern Name | Name | Name | Name | Name | Name |
| Intent | Problem + Resulting context | User decision | What + Why | Problem + Why | Problem |
| Also Known As | / | / | / | / | / |
| Motivation | / | Task and Task window | Examples | More examples | / |
| Applicability | Context | User decision | Use when | Use when | Background |
| Participants | / | / | / | / | / |
| Consequences | Forces | Task | / | / | / |
| Collaborations | / | / | / | / | / |
| Implementation | Rationale | Task | How | How | / |
| Sample code | / | / | / | / | / |
| Known uses | / | / | Figure | Solution | Figure |
| Related patterns | / | Task and Task window | In other libraries | / | Other pattern to consider |
| / | / | / | / | Literature | / |
| / | / | / | / | Comments | / |
| / | Solution | Task | Solution | Solution | Solution |
| / | Author | / | / | / | / |

**Table 1 : Comparison between different pattern description formats**

The attribute which links the pattern to other patterns is an interesting feature. Because it gives us the relation between the different patterns and by these binds it is possible to retrieve a global view of all the patterns. This global view is what we called the pattern language.

### 2.1.4. Weaknesses of the description formats

Weaknesses are noticed among all the kinds of description formats. We list them below. (Nguyen et al., 2016 b)

W1. **Lack of consistency**: Collections describe patterns according to different range of attributes, taxonomy criteria, conditions of application. The level of details varies from a format to another. Some attributes appear to be homonyms. All of these points make them inconsistent. For example, van Welie does not link any pattern and the gang of four have only a related pattern.

W2. **Lack of structure**: Structure is not uniform. Application conditions are strewn across attributes.

W3. **Lack of implementation information**: When conditions of application are not omitted, the links with implementation are almost non-existent. Only the gang of four give a concrete implementation with class diagram.

W4. **Lack of pattern evaluation**: There is for most of the patterns no evaluation about their ability to solve a problem. There is no evidence to prove a solution is efficient. van Welie enables users to comment his patterns but it is not an sufficient assessment.

W5. **Limitation of contextual coverage**: The context of use (user, platform and environment) is not fully supported. Patterns related to user interface are merely restricted to the device or platform. Besides, it is usually focused on one device at a time. So it does not help to address multiple-user interface issue.

W6. **Lack of abstraction**: Nearly all patterns are expressed for a specific operating system or render in a specific environment which prevent to switch to another one.

W7. **Lack of ergonomic approach**: There is none or not enough details about ergonomics

W8. **Limited usability consideration**: Applying a pattern in a particular case means choose between design options. This choice is led by a usable solution with varying degree of quality. But patterns give almost no information about their usability or user experience.

### 2.1.5. Pattern Language

As formerly said, the link between the patterns can be seen as a network of patterns. This network is called the pattern language. “A pattern language is a complete set of patterns for a given family of design problems in a given domain. A pattern language describes problems by means of high-level design patterns, which may be solved by lower-level design patterns.” (Kruschitz & Hitz, 2010, pp 227-228). The language is clearly a deconstruction of a problem to make issues easier to solve. In order words, problems are broken down into several manageable problems for which a solution is conceivable. (Alexander et al, 1977)

van Welie and van der Veer (2003) use the metaphor of the puzzle to highlight this concept. Each piece of the puzzle is a pattern and they are a solution to a specific problem. But putting together they form a puzzle and we can see an entire picture which is far more valuable. Besides they organized a pattern language based on the Alexander's hierarchical approach. This top-down approach begins with high level problem such as Business Goal and can be refined to more detailed problem like Task level and Action level. Figure 5 shows this hierarchical approach for the web shopping example.

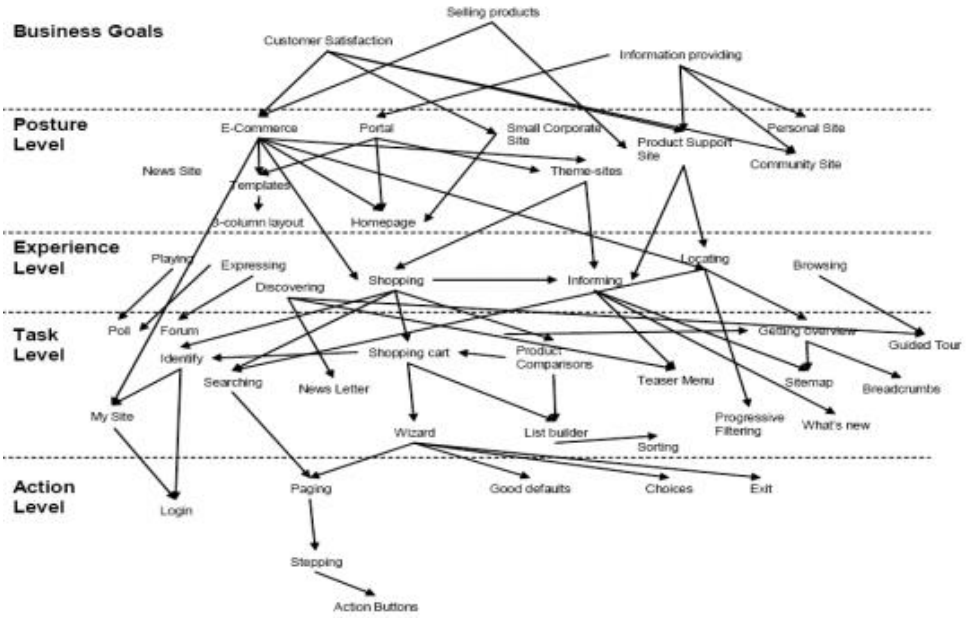



**Figure 5 : Pattern language for the web shopping (van Welie & Van der Veer, 2003)**

In their book *Pattern oriented software architecture: a system of pattern,* Buschmann, Meunier, Rohnert, Sommerlad and Stal (2001) have a similar view but span three levels of abstraction: Architectural patterns are high-level, Design patterns are mid-level and Idioms are low-level.

- Architectural patterns provide the skeleton of overall system architecture.
- Design patterns address problems encountered after the overall structure.
- Idioms deal with language-dependent patterns.

The patterns presented by the Gang of four are situated at the design pattern level. To conclude, a pattern language sub-sums rules and guidelines which point out when and how to handle its patterns to solve a problem, which an individual pattern could not succeed. (Tešanovic, 2005)

A last definition of pattern language is given by Seffah (2015). He says that a pattern language must satisfy three essential elements. First, a language must have one standard pattern definition format. Secondly, the language must group patterns. Finally, the language must describe pattern interrelationships.

### 2.1.6. Pattern Collection

Besides to be organised as a language, patterns can be grouped in collections. Collections are a pattern repository subdivided into small number of broad categories. A catalog or collection ensures fast navigation within this repository. Unlike the pattern language, patterns in a collection show almost no relationships among each other and thus there is nor interconnected system. (Tešanovic, 2005 ; Kruschitz & Hitz, 2010)

Several pattern collections exist and their organisation varies according to the author's vision and the subject concerned. Among others, we can list collection of the gang of four, Tidwell, van Welie, van Duyne, Yahoo library, Coad and Yourdon, Pastor and Molina...

The gang of four elaborated a collection spanning a wide range of patterns (Gamma et al., 1994). So in order to facilitate the user's navigation, there are clustered by a common purpose and scope (table 2).

| SCOPE | PURPOSE | | |
|---|---|---|---|
| | Creational | Structural | Behavioral |
| **Class** | Factory Method | Adapter | Interpreter |
| **Object** | Abstract Factory<br>Builder<br>Prototype<br>Singleton | Adapter<br>Bridge<br>Composite<br>Decorator<br>Facade<br>Flyweight<br>Flyweight<br>Proxy | Chain of Responsibility<br>Command<br>Iterator<br>Mediator<br>Momento<br>Observer<br>Observer<br>State<br>Strategy<br>Visitor |

**Table 2 : The GOF patterns collection (Gamma et al.,1996)**

**By purpose**, there are either:

- Creational relates to the creation of an object.
- Structural is about the composition of the classes and objects.

- Behavioural deals with the interaction between classes or objects and how they share the responsibility.

**By scope**, they are distinguished between classes or objects. Class patterns focus on the inheritance relationships between classes and their subclasses which are static. On the contrary, object patterns concentrate on object links which are dynamic. Indeed objects vary at run-time. Even if all the patterns are concerned by the inheritance, only the class patterns will deal with this relationship. (Tešanovic, 2005)

Even though having the same purpose, patterns defer if they are class or object patterns. For example, the structural class patterns use inheritance to compose classes while the structural object patterns describe ways to assemble objects.

But it is possible to use other ways to classify patterns. Some of them work frequently together as composite and Iterator or Visitor. Others can be substitutes from each other such as Abstract Factory and Prototype.

Tidwell (2010), for her part, developed a collection of user interface patterns (table 3). By user interface, she means websites, desktop applications and elements in between. Her collection design patterns is segmented in key concept in interaction design and visual design such as user's behaviour, navigation, content… It contains 11 key concepts including about 125 patterns. There is a limit to this collection because it's mainly focused on desktop. In the second edition of her book however, a chapter about mobile has been added.

Yahoo has a user interface pattern collection separated in 5 categories. The originality resides in a rating for each pattern. This rating is based on the assessment of their designers and user experience researchers. (Yahoo! Developer Network, n.d.)

| Category | number of pattern | Patterns |
|---|---|---|
| What users do | 14 | Safe exploration; Instant gratification; Satisficing; Changes in midstream; Deferred choices; Incremental construction; Habituation; Microbreaks; Spatial memory; Prospective memory; Streamlined repetition; Keyboard only; Other people's advice; Personal recommendations |
| Organizing the content | 10 | Feature, search and browse; News stream; Picture manager; Dashboard; Canvas plus palette; Wizard; Settings editor; Alternative views; Many workspaces; Multi-level help |
| Getting around | 13 | Clear entry points; Menu page; Pyramid; Modal panel; Deep-linked state; Escape hatch; Fat menus; Sitemap footer; Sign-in tools; Sequence map; Breadcrumbs; Annotated scrollbar; Animated transition |
| Organizing the page | 13 | Visual framework; Center stage; Grid of equals; Titled sections; Module tabs; Accordion; Collapsible panels; Movable panels; Right/left alignment; Diagonal balance; Responsive disclosure; Responsive enabling; Liquid layout |
| Lists of things | 12 | Two-panel selector; One-window drilldown; List inlay; Thumbnail grid, Carousel; Row striping; Pagination; Jump to item; Alphabet scroller; Cascading lists; Tree table; New-item row |
| Doing things | 11 | Button groups; Hover tools; Action panel; Prominent "done" button; Smart menu items; Preview; Progress indicator; Cancelability; Multi-level undo; Command history; Macros |
| Showing complex data | 11 | Overview plus detail; Datatips; Data spotlight; Dynamic queries; Data brushing; Local zooming; Sortable table; Radial table; Multi-Y graph; Small multiples; Treemap |
| Getting input from users | 11 | Forgiving format; Structured format; Fillin-the-blanks; Input hints; Input prompt; Password strength meter; Autocompletion; Dropdown chooser; List builder; Good defaults; Same-page error messages |
| Using social media | 12 | Editiorial mix; Personal voices; Repost and comment; Conversation starters; Inverted nano-pyramid; Timing strategy; Specialized streams; Social links; Sharing widget; News box; Content leaderboard; Recent chatter |
| Going mobile | 11 | Vertical stack; Filmstrip; Touch tools; Bottom navigation; Thumbnail-and-text list; Infinite list; Generous borders; Text clear button; Loading indicators; Richly connected apps; Streamlined branding |
| Make it look good | 7 | Deep background; Few hues, many values; Corner treatments; Borders that echo fonts; Hairlines; Contrasting font weights; Skins and themes |

**Table 3 : Tidwell patterns collection (Tidwell, 2010)**

## 2.2. Type of Pattern

So far we have reviewed the concept of pattern and the literature around this notion which give us the basis to take up the following section. In this section, we will refine our knowledge to determine the type of pattern that answer at the purpose of this thesis. We will expose into which branch this type of pattern fits and will define its features.

### 2.2.1. Descriptive vs Generative Pattern

Until now, we have mainly presented patterns that are referred as being descriptive. They are classified in this way because it consists to outline a description of the pattern, its problem, the context in which the problem occurs and the potential solutions that could solve the issue. Patterns are a mean to have a mapping between the problem space and the

design space. Descriptive patterns usually are fully descriptive (sufficiently described to be self-contained) and generative (sufficiently relevant for the widest problem space). (Vanderdonckt & Simarro, 2010).

In opposition to descriptive, pattern is said to be generative when they contain an object-oriented representation that is able to generate the final code. Generative patterns are on their side fully characterized by expressivity (the ability to be expressive enough to provide a working system) and by generativity (constructed as to facilitate the automated generation of code). (Vanderdonckt & Simarro, 2010)

Thus descriptive patterns describe recurring phenomena but do not describe how to reproduce these phenomena. Whereas generative patterns do not only document features of good systems but also teach how to build such systems. (Tešanovic, 2005)
Vanderdonckt and Simarro (2010) suggest combining both qualities of generative and descriptive into a genuine pattern. The four aspects (expressivity, generativity, descriptivity and genericity) will be contained in a new description template. This combination opposes to consider one dimension at a time. As it is reported in figure 6, with genuine patterns the four properties are expected to be maximized.

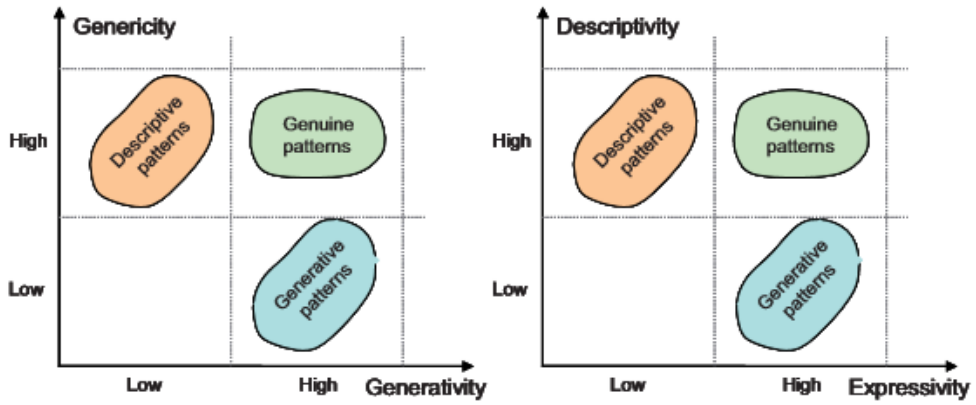



**Figure 6: Type of patterns according to four criteria (Vanderdonckt & Simarro, 2010)**

### 2.2.2. User Interface design pattern vs Software design pattern

In application design, patterns are divided in two main categories (figure 7). The first one, software design pattern, is focused on software design. The kernel pattern relates the functional part of software design. Patterns of the gang of four are subsumed in this category. The second group is about the human computer interaction (HCI) pattern. Human computer interaction concerns the interactive systems for human use. With the new technologies and devices, interaction with a system has changed a lot and nowadays, people are able to interact vocally, through touch screen, by movement recognition… In fact interaction with the system is the first action a user does when he accesses to this system. Human Computer Interaction (HCI) studies the means for human to communicate and control a machine. For instance, how users interact with a machine in order to execute a service. HCI struggles to provide some guidance for interactions with the help of patterns in the aim to develop applications.

User interface (UI) patterns are a specific branch of the HCI and are concentrated on interface design. An interface is the space where interactions between human and machine occur. Collection of UI design patterns are notably represented by Tidwell and van Welie. Conceptual pattern is concept introduced by Pastor and Molina to bring patterns at the early stage of the conception of user interface. In this master thesis, we will focus on the user interface and especially on the service interaction unit from the conceptual patterns.

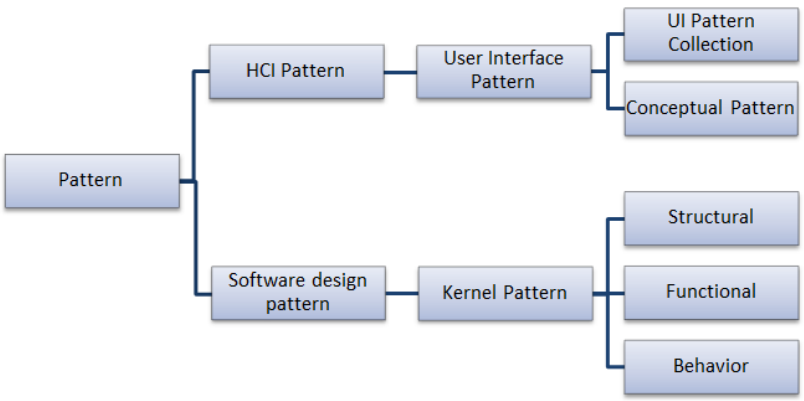



**Figure 7: Pattern classification**

### 2.2.3. Conceptual Pattern

Conceptual Patterns form a pattern collection situated in the section of User Interface Patterns. This concept was introduced by Pastor and Molina in order to describe the interfaces with a higher level of abstraction. In the User Interface field, most of the works cope with design patterns. As a consequence, the early phases (requirements and analysis) are neglected from a pattern point of view. Yet, Conceptual patterns suggest correcting that by introducing patterns in those early phases and propagate them in the later phase, design and implementation. So, the whole life-cycle from the conception to the implementation will be covered by a pattern-oriented development. (Molina, Meliá & Pastor, 2004)

The conceptual pattern collection determines the specifications of the user interface of information systems. Starting at the analysis, the goal is to have patterns related to the requirements of the system. But design and implementation issues are discarded and will be discussed in their respective phase. Therefore, conceptual pattern targets the problem space and not the solution space. By delaying the solution space considerations to another step, it prevents to take decision about the implementation and design at the analysis phase. (Molina et al., 2004)

The main advantage is to help analyst to gather and organize in a common language the information obtained from the user requirements. Moreover, it allows avoiding errors in the early phase which is much faster and easier to correct. This conceptual pattern language provides the advantage of having documented analysis concepts which admits an easy reuse of these concepts. Figure 8 shows how this language is organised. (Pastor& Molina, 2007)

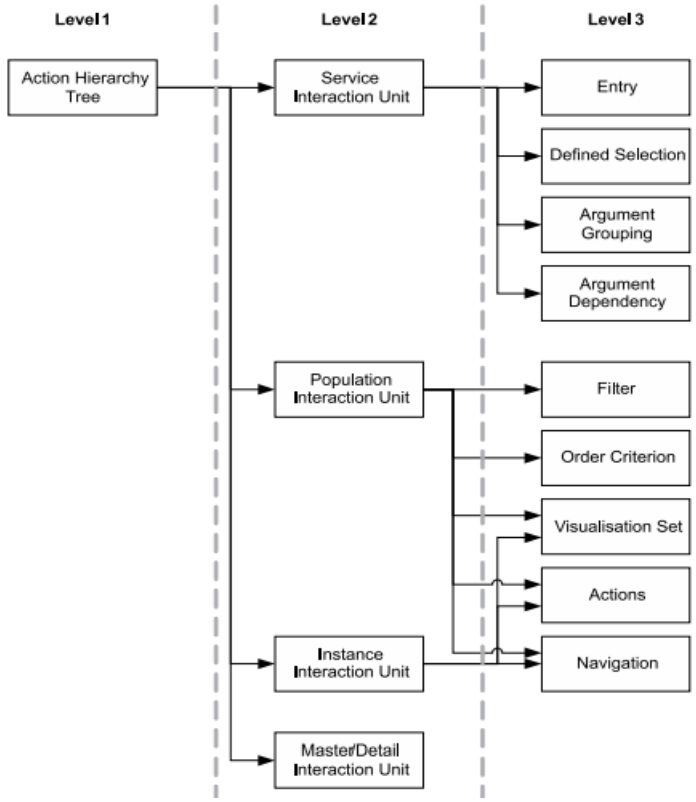



**Figure 8: Structure of the presentation model (Pastor & Molina, 2007)**

Through figure 8, we note the language is segmented in three levels. (Pastor & Molina, 2007)

- Level 1 contains the hierarchical action tree (HAT) which regulates the access points of the user to the application. It organizes the access to the system functionality through a tree-shaped abstraction. The nodes are the interaction units in level 2 and the leaves are the basic elementary patterns.
- Level 2 explains the interaction units and the steer between them. These interaction units determine the interactive operations that can be performed (execute a service, visualise details…).
- Level 3 shows the basic elements (elementary patterns) that composed the interaction units of level 2.

Interaction Unit "describes a particular scenario of the user interface through which users are able to carry out specific tasks (such as executing services or searching for objects)." (Pastor & Molina, 2007, p 171). So, an interaction unit is a particular scenario where users interact with the system through the UI. Those Interaction Units, representing the basic specification element for the interaction, are complex patterns compiling simple patterns or other interaction units. This approach offers advantages. (Pastor & Molina, 2007):

- Extensibility: new interaction units can be added in the model
- Homogeneity: every interaction context can be represented by an interaction unit.
- Design Independence: The model doesn't gather design issues or details.

There are four different types of interaction units expressing four different basic kinds of interaction scenarios (Pastor & Molina, 2007):

- Execution of a service (Service Interaction Unit).
- Manipulation of an object (Instance Interaction Unit).
- Manipulation of a collection of objects (Population Interaction Unit).
- Manipulation of multiple related collections of objects (Master/Detail Interaction Unit).

The last part is the basic elements also known as Elementary Patterns or Auxiliary Patterns. Basic elements are "the building blocks from which abstract user interfaces can be constructed. They represent specific aspects of the interaction between a human and a system, and can be composed into interaction units" (Pastor & Molina, 2007, p151). Pastor reckons eleven basic elements. Some of them are meaningful only in a specific context i.e. argument grouping can only be used in the context of a service execution. That is why they are linked to the appropriate usage context(s).

#### 2.2.3.1. Service Interaction Unit

In this master thesis, we will detail and observe an interaction unit in particular which is the Service Interaction Unit (SIU). The Service Interaction Unit also known as the Service Presentation Patterns is in charge of the communication between the system and the user in order to execute a service. In other words, it models the context of the execution of a service. The user has to enter arguments to launch a service. It results from 4 individual patterns (basic elements) situated at level 3: Introduction/Entry, Defined Selection, Argument Grouping, Argument Dependency. (Pastor & Molina, 2007)

- **Introduction/Entry** pattern determines how argument data can be entered by the users. It is a constraint on the introduction of values. This pattern has the following properties:

a. *Name*: has to be unique in the system
b. *Data Type*: ensures the compatibility in the system (real, string, float…)
c. *Edit Mask*: specifies the format of data to be entered ( phone number could be ### / ## ## ##)
d. *Valid Value Range*: provides a value range and controls the value entered (days in a week is 1 to 7)
e. *Default Value*: provides a predefined value to save time to the user. But it can be modified.
*f.* *Help Message*: is text message that informs the user about the data that has to be entered
*g.* *Validation Message*: is shown to the user when an error occurs in the validation process specifying the cause of this error.
*h.* *Comments*: an additional information that can be useful for Documentation.

- **Defined Selection** provides a pre-existing collection of values where the user has to select one of them. It helps us to define a set of valid values beforehand. Its properties are:
    a. *Name*: is unique and enables us to identify right element.
    b. *Data Type*: controls the data type compatibility.
    c. *Value Set*: is a list of values (code) and their corresponding translation (label). Sometimes, values are meaningless to the user so it is preferable to show comprehensible information. e.g.: values=d, label= available
    d. *Help Message*: is a text that helps the user about the data type and the model where the defined selection is applied.
    e. *Comments*: is additional information about the defined selection that can be useful for Documentation.

- **Argument grouping** organizes the way to present the input arguments for a service to the user. It is an ordered list establishing priority. Besides, arguments can be arranged in groups and sub groups. The properties of argument grouping are:
    a. *Argument Orders* put in order the navigation steps of the service argument through the selected argument group.

b. *Arguments Groups* are defined by
    i. *Alias* is useful to identify the argument group on the user interface
    ii. *Help Message* gives information about the argument group and the argument contained in it
    iii. *Comments* documents the argument group for the engineer.
c. *Containment relationships of groups and arguments*

- **Argument dependency** makes us able to specify the dependency relationship of an input argument (value/state) of a service and another input argument (value/state) of the same service. Argument dependency is based on the event/condition/action-type rules for each input argument. That is to say, when an event occurs for a precise argument, such as entering its values or activate/deactivate it, a specific action is performed if a condition is met. Argument dependency is explained through the following data:

    a. *Interface Event* describes what happens to the argument to which the argument dependency is carried out. There are two types of events.
        i. *SetValue(v)* indicates the value that receives the argument.
        ii. *SetActive(v)* indicates if the argument has been activated or not.

        An agent is associated to each event which determines the application of the argument dependency in case of the event is activated by this agent. Agent can be:
        i. *User,* when the event is triggered by a user.
        ii. *Internal,* when the event is triggered by the action of another argument dependency.
        iii. *, when the event is caused by User and Internal.

    *b.* *Condition* is a boolean formula that must be satisfied in order to execute the action.
    *c.* *Actions* is made up of a list of elements including
        *i.* *Argument* refers to the name of an argument of the service.

ii. *Event* is the action that has to be performed on *Argument* (SetValue,SetActive)

iii. *Formula* expresses the value that will receive *Argument* if *Event* is *SetValue*, either *Argument* is activated/ deactivated if *Event* is *SetActive*.

Recently, four other basic elements were added to the SIU, the Population Preload and the Conditional Navigation (Aquino, Vanderdonckt, Panach, & Pastor, 2011), Status Recovery (which is a dependency) and Complementary Information (Pastor, Insfran & Pelechano, 2000). But due to a lack of information on them, we will stick on the former model.

#### 2.2.3.2. SIU Instance

We will take the rental car case to illustrate the Service Action Unit and its building blocks (basic elements). Suppose you want to rent a car and you rent it online via a rental car company website. To execute this service you have to communicate your wish to the rental car company. Once transmitted, they will be able to satisfy your demand. The popular and well used way to obtain information in order to execute a service is across forms. A form will indicate which information the rental car company requires and how to transmit it.
So, we will first illustrate the four basic elements and we will summarize them through the SIU. This example is inspired of the rental car case from (Pastor & Molina, 2007).

For example, a customer could specify the number of kilometres he is planning to drive. This entry element is defined by the following properties:

- Name: arg_Kilometres
- Data type: Real
- Edit mask: ##,###.##
- Valid value range: ( is determined within the lower and upper bound)
- Lower bound: 0.00
- Upper bound: 6,000.00
- Default value: 500.00 (default can be based on an average)

- Help message: Please enter a real number between 0.00 and 6,000.00 (provides information to the user about the data to be entered)
- Validation message: Only real values between 0.00 and 6,000.00 can be entered (text shown if there is a validation error)
- Comments: This element helps the user in the entry of kilometre values ( information posted for documentation purpose)

The user could face a situation where he has to select an object among a list of objects. For example, he has to choice a car brand. The properties of such defined selection are:

- Name: arg_carbrand
- Data type: Integer (prevents Real or String -type to be assigned)
- Value Set: represents the list of codes (1, 2, 3, and 4), corresponding to the defined selection element, and their labels (Citroën, Ford…).

| Code | Label |
|---|---|
| 1 | Citroën |
| 2 | Ford |
| 3 | Renault |
| 4 | Volkswagen |

**Table 4 : Codes and their labels**

- Help Message: Please choose a brand among the list.

The car company orders the way in which the required information will appear. Besides, arguments can be grouped by common purpose, theme... Argument Grouping clarifies the purpose for the user and avoids getting him lost. Here we can organize arguments in groups and subgroups such as the following table.

| Group | Argument |
|---|---|
| Classification | arg_carbrand |
| | arg_model |
| | arg_fueltype |
| Engine | arg_enginetype |
| | arg_power |
| | arg_co2 |
| Extra | arg_gps |
| | arg_bluetooth |
| | arg_screen |

**Table 5 : Group of arguments**

- Argument order: Arguments and groups show a certain order from general to specific. Classification->Engine->Extra and brand->model->fuel (for classification)
- Argument Groups: The Arguments of service “rent a car” are divided into three groups.
    - Alias: “classification”, “engine” and “extra”
    - Help message: for instance, “ Please complete if you wish any extra”
- Containment relationships of groups and arguments: Classification contains the brand, the model of the brand and the fuel to use. Engine concerns more specifications about the motor such as its type (electric, hybrid or combustion), its power in cc and the $co^2$ emitted. Finally, extra groups are supplementary features (gps, bluetooth or screen).

Now, let us imagine that the user wants to rent for brand x only the diesel fuel type. It means that the value Diesel will be assigned to the argument arg_fueltype of service “rent a car” whenever the user enters brand x for argument arg_carbrand for the same service. This argument dependency is represented in table 6.

| **Event** | | |
|---|---|---|
| *Agent* | *Argument* | *Event* |
| User | arg_carbrand | SetValue(v) |
| **Conditon** | | |
| V= « carbrand X » | | |
| **Action** | | |
| *Argument* | *Event* | *Formula* |
| arg_fueltype | Setvalue(v) | “Diesel” |

**Table 6 : Dependency between argument carbrand and fueltype**

Event section indicates that the rule is applied when the user triggers the SetValue(v) event for the argument arg_carbrand. So when the user enters a car brand. When this rule is activated, condition is evaluated. This step determines if the associated action will be achieved. Here the condition indicates that the value “v” (assigned by the user for arg_carbrand) must equal the “carbrand X”. If condition is met the action is done. The action in our example is to assign the value “Diesel” to the argument arg_fueltype.

## 2.3. Ergonomic rules and UI patterns

At this step, we have looked over entirely the notion of a pattern and their different categories. We have seen two major branches, Software design pattern and User Interface design pattern. We introduced the Service Interaction Unit and the conceptual patterns. As said before, in the framework of this master thesis the focus will stay on the User Interface patterns. So here, we will make the point about the ergonomic in the Human Computer Interface. How they are linked and how they are used. In addition, the concept of usability will be approached with respect to the UI patterns. We have already distinguished two shortcomings in the pattern description format (weakness 7 and 8). Which relate the lack of ergonomic approach and the limited usability consideration.

### 2.3.1. Ergonomic in the HCI

According to the norm DIN/EN/ISO 6385 (as cited in Wegge & Zimmermann, 2007, p295) ergonomic is "scientific discipline concerned with the understanding of interactions among human and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance". In our case, ergonomic makes sure that the interaction between the user and the system (through the user interface) is optimized for human well-being and a system performance.

Experts as Nielsen (Nielsen, 1995), Scapin and Bastien (Scapin & Bastien, 1997) and Schneiderman (Schneiderman, 1997) have worked with a user point of view (user-centered) in order to improve applications. In addition, institutions as the US government defined a set of rules to be applicated on the US governmental applications (Misfud, 2011 a). But also big companies such as Microsoft (Microsoft, n.d.) and Sun Microsystems (Sun Microsystems Inc., 2001) established guidelines. With these rules, they tried to standardize applications to avoid users get lost.

Table 7 regroups by topics guidelines from (Nielsen, 1995 ; Scapin & Bastien, 1997 ; Schneiderman, 1997). Guidelines can be gathered in 8 topics (consistency, errors, guidance and feedback, familiar language, user control, minimized effort, system adaptability and

FAQ). It results from these that guidelines are really large indication and not enough concrete to handle it during the implementation. Moreover, a rule can be a combination of other rules. For instance, Guidance (Scapin) combines Offer informative feedback, Design dialog to yield closure and Support internal locus of control (Schneiderman). We choosed to resume three well known authors but as said before, it exists various guidelines created by institutions, searchers, companies, designers… Unfortunately, there is no current standard for ergonomic guidelines.

| Guidelines | Nielsen | Scapin & Bastien | Schneiderman |
|---|---|---|---|
| **Consistency:** Interface/ code choices are maintained in a same context and are different in a different context. | Consistency and standards | Consistency + Significance of codes | Strive for consistency |
| **Errors:** Allow to avoid/reduce errors and to correct them + good indications | Error Prevention + Help users recognize, diagnose, and recover from errors | Error management | Offer simple error handling |
| **Guidance and feedback:** Orient, advice and inform the user adequately while he interacts with the machine. | Visibility of system status | Guidance | Offer informative feedback + Design dialog to yield closure + Support internal locus of control |
| **Familiar language :** Don't speak geek | Match between system and the real world | Compatibility | / |
| **User Control :** Possibility of undo, redo and cancel | User control and freedom | Explicit Control | Permit easy reversal of actions |
| **Minimized effort:** Require at least user's short-term memory | Recognition rather than recall + Aesthetic and minimalist design | Workload | Reduce short-term memory load |
| **System adaptability:** Tailored system to user's experience | Flexibility and efficiency of use | Adaptability | Enable frequent users to use shortcuts |
| **Provide a good FAQ** | Help and documentation | / | / |

**Table 7 : Summary of ergonomic guidelines into 7 features**

### 2.3.2. Patterns vs ergonomic guidelines

As soon as you start to design an interface, it is possible to work either with ergonomic guidelines or with patterns. But a choice has to be made. In this master thesis, the goal is to suggest a way that mixes patterns and ergonomic guidelines. Though before to mix, it could be interesting to see the differences between the two alternatives.

A first contradiction is the purpose they are looking for. Ergonomic rules capture design knowledge into small rules which can be used by designer when they build interfaces. Whereas patterns capture a proven design knowledge and decline it in terms of problem, context and solution. Eventhough they seem to have a similar objective, they also differ by their format. On one hand, design knowledge may be expressed in a pattern template. On

the other hand, using template to express guidelines does not make patterns. It is important for a pattern to provide a proven solution approved by designers. Inversely, guidelines are not matched with rational. (van Welie et al., 2001)

In the previous section we have already raised some weaknesses of the ergonomic guidelines. From all these observations, a list of shortcomings can be dressed (van Welie et al., 2001):

**S1.** Guidelines are often too simple or too abstract

**S2.** Sometimes difficult to select one

**S3.** It is not easy to interpret them

**S4.** They can conflict with each other

**S5.** Their validity is subject to authority issue

These shortcomings come from the same origin: guidelines are too general and they are used for specific context. So it is crucial to know the context in order to select the right guideline and to understand its rational. However they are not enough precisely described to grasp this context. As a result design problem cannot be solved adequately. (van Welie et al., 2001)

Another issue is to understand the nature of the problem and why the guideline is like that. For example, consider the guideline “Left align labels in dialog window”. We wonder what is the real problem addressed by this rule? It is clearly not how to layout label. (van Welie et al., 2001)

After reviewed all the problems linked to guidelines, we can notice that patterns offer advantages and answer to the highlighted shortcomings. As patterns are context and problem- oriented, they allow users to fully understand when and why a pattern should occur. Hence, after identifying the when and why, pattern provides an adequate and concrete solution and its rational. Guideline says do this or do not that, pattern focuses on do this. (van Welie et al., 2001 ; Seffah, 2015)

### 2.3.3. Usability

We have seen the differences between the ergonomic guidelines and the patterns just as the advantages of patterns compared to guidelines. Now we will introduce the notion of usability. Usability is part of the ergonomics but restricted to a precise domain. In this thesis, we also wish to integer usability just like ergonomic to the patterns.

The norm ISO 9241-11 (as cited in Wegge & Zimmermann, 2007, p297) defines Usability as "Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." In other words, "usability deals with methods to assess the effective, efficient and satisfactory use of a product in a given context of use" (Wegge & Zimmermann, 2007, p297).

Figure 9 (van Welie et al., 2001) exhibits the usability concept through a layered model. On the first layer, we encounter the ISO definition of usability split up in in efficiency, effectiveness and satisfaction. This level is more an abstract way to capture usabilty and is not really practicable. Nonetheless, these three pillars give us a solid basis to start developing the usability.

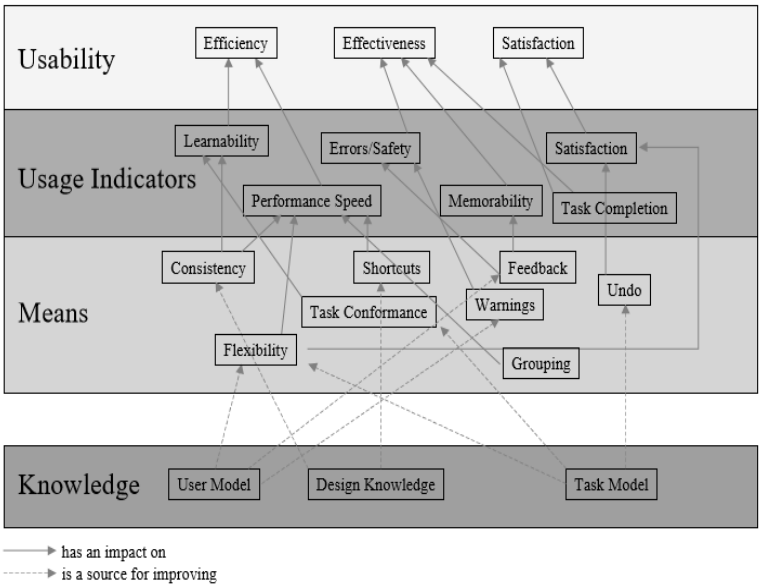



**Figure 9: The model of usability (van Welie et al., 2001)**

The second level contains the usage indicators which represent feasible and observable indicators of the usability level. They contribute to the abstraction of usability. A fast performance speed contributes to a better efficiency. The level required for the indicators

depends on the context. An e-commerce want to accentuate the satisfaction without to ignore the 2 other characteristics.

The third level concerns the means. Means are not goals to reach but a technical way to reach goals. Thanks to the means upper levels characteristics are attainable. Means impact the usability and must be handled optimally. The means presented here, can be enlarged by others. Moreover, they can be grouped according to ergonomic principles. For example, the ones we established in table 7.

The last level is the knowledge domain. It is separated in human, design and task. Thanks to these domains, designers determine how to find the best trade-off to optimize the usability. When using guideline to express design knowledge, it is clear that guideline should embody the impact of means on usage indicators.

We observe that notion of ergonomic guidelines and usability is very broad and could enrich patterns. Unfortunately, researches on that matter are poorly. So to improve user interface design patterns, it could be useful to incorporate ergonomic guidelines. Since usability depends on the usage indicators, the rationale section of pattern description will explain how ergonomic used in the solution leads to an improvement of the usage indicators.

In this thesis, we will stress out the performance brought by this incorporation but we do not prove or verify them. We are not going to create data about performance of those patterns. Hence, there is too little researches concerning pattern performance.

### 2.4. Pattern application methods

The methodology to develop information system generally rests on three pillars: Model, Method and Tool.

- Model expresses the modelling of the required aspects that have to be dealt all along the development.

- Method determines the succession of the operations necessary to the development of (in our case) the UI based on pattern. The method relies on the preliminary chosen model.
- Tool is a software tool which applies the method in an efficient way.

This section will approach methodologies that are suitable for user interface patterns. Then a comparison will be done to select the one comforting our context. As the tools are inherent to the methods, we can consider that model and method are the key decisional factors. To recall, this master thesis is in the light of the cross-platform context and the ergonomic rules.

### 2.4.1. Complementary notions

Before we get to the heart of the pattern application methods, we need to explain complementary notions used in some of the presented methodologies. So in the detail to not burden the methodologies explanations, they will be developed in this section.

#### 2.4.1.1. Cameleon Reference Framework

The Cameleon Reference Framework (Calvary et al., 2003) is a model framework which defines UI development life cycle into four levels of abstraction in a context-sensitive approach. Context-sensitive means that the user interface exhibits some awareness and adaptability in regards to the context in which it evolves. The four levels of abstraction are: Task and Domain, Abstract user interface (AUI), Concrete user interface (CUI) and Final user interface (FUI). These levels are linked by 3 kinds of transformation (see figure 10):

- Reification going from higher abstraction level to the lower, here from AUI to CUI.
- Abstraction is the reverse action which means from the lower abstraction level to the higher (CUI to AUI).
- Translation is a transition from one context to another but at the same abstraction level.

Besides, the CRF is a multi-path UI development. In other words, designer is allowed to start the development from any entry point in the framework. Therefore, designers are free to

follow the path they want and they are not forced to achieve steps as dedicated by the sequential levels.

As we can see in figure 10 (Aquino et al., 2011), CRF is composed of four abstraction levels which are described below.

- Task and Domain provides the end-user's tasks and the domain objects manipulated by these tasks. The context of use (user, environment and platform) can be added if necessary. Task Model is a break down structure of a principal task with constrains on and between subtasks. Usually it is represented by the Concur Task Tree (CTT) notation (Paterno, 1999). The Domain Model completes the task model by its object oriented (class, attribute, method) and its relationships expressed usually by UML class diagram.
- The Abstract User Interface "describes potential UIs independently of any interaction modality and implementation" (Vanderdonckt & Simarro, 2010, p 15). It expresses the UI in terms of the interaction space rendering of the task and domain level. The conceptual patterns of Molina are situated in this level.
- The Concret User Interface "specifies a UI independently of any technological space, but for a given interaction modality" (Nguyen et al., 2016 b, p 276). It is a mockup of the FIU look and feel.
- The Final User Interface is the final rendering and is the source code of any markup or programming language which can be interpreted or compiled.

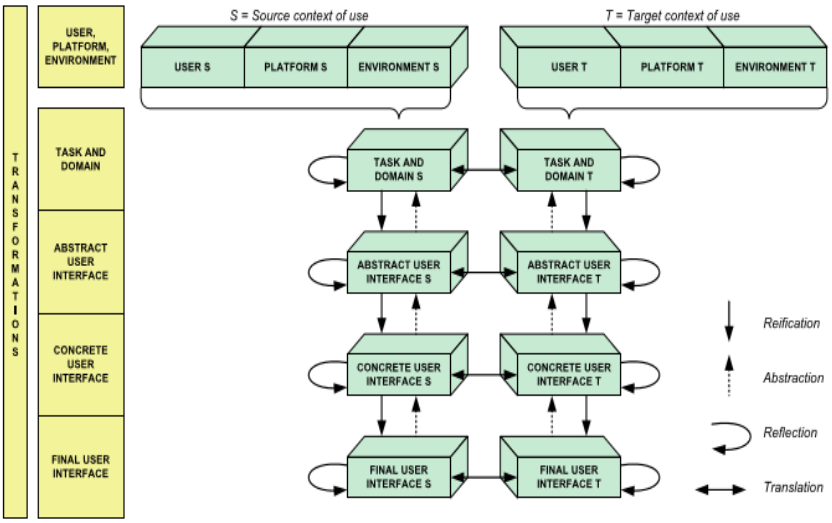



**Figure 10: Cameleon Reference framework (Aquino et al., 2011)**

The Cameleon Reference Framework is compliant to the Model-Driven Architecture (MDA) (Vanderdonckt & Simarro, 2010). MDA is a part of the model driven engineering (MDE). MDE suggests a way to design software which structures the software development with models. MDA has been developed by Object Management Group in order to standardize the model-driven approach. “The MDA separates business and application logic from underlying platform technology.” (Object Management Group [OMG], 2016, para. 1) MDA provides also several models and makes the bridge between them. Following the MDA models and their related CRF abstraction level (Vanderdonckt, 2005):

- Computing Independent Model (CIM) is independent of any implementation of any interactive systems. It corresponds to the Task and Domain models
- Platform Independent Model (PIM) is independent of any interaction modality (graphical, modal, virtual…). It matches the AUI model.
- Platform Specific Model (PSM) is independent of any mark up and programming language. It is equal to the CUI model.

#### 2.4.1.2. User Interface Description Language

A UIDL “aimed at describing user interfaces with various levels of details and abstractions, depending on the context of use”. (Limbourg et al., 2004, p 55)

Since the beginning of Human Computing Interaction, many UIDLs addressing diverse aspects of UIs have been created on basis of the XML language. The main advantage of XML results in the fact that it can be easily transformed in various programming language. Yet the important number of UIDLs, there were still some shortcomings. Limboug et al. (2004, p 55) sum them up in the following list.

- “Making one UI design independently of any modality of interaction (e.g., graphical UI, vocal UI, virtual UI, multimodal …) so that a design at this level may initiate more concrete designs once a particular modality has been selected.
- Supporting the integration of any model used in the UI development process, such as, but not limited to: the context of use, the user, the platform, the environment, the devices used…

- Expressing explicitly mappings between models and elements when appropriate to address the mapping problem.
- Supporting the continuous and seamless manipulation of models from the abstract level to the concrete levels, such as in the Model-Driven Architecture (MDA) of the OMG Group.
- Expressing UIs at a given instant of usage so as to capture relevant information to ensure runtime migration."

The USIXML, which stands for USer Interface eXtensible Markup Language, was developed in order to cover these shortcomings. "USIXML is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory and Vocal User Interfaces, Virtual Reality, and Multimodal User Interfaces" (usixml.org, N.D.). USIXML allows describing any type of user interface without to worry about the programming language and the computing platform. The description is independent of the context of use and due to this characteristic USIXML can be applied in different context of use.

Its advantage provides the possibility for designers to elaborate their user interface even if they don't have any programming skills. Naturally, programmers can use it but it is not the primary goal.

The UsiXML describes, at a high level of abstraction, the UI components of an application such as widgets, controls, containers, interaction modalities… Thanks to its high level of abstraction, it is possible to link the different levels of abstraction during the interface development. Indeed, in the Cameleon Reference Framework, the three types of transformation, abstraction, reification and translation (Vanderdonckt, 2005) can be done thanks to the UsiXML. So, the passing between AUI and CUI is insured by the UsiXML. And for the passing from the CUI to the FUI, the UsiXML is implemented in a programming language e.g. HTML. All these advantages of the UsiXML bridge the gap encountered by the XML.

### 2.4.2. OO-Method

OO-Method is an object oriented method notably used in the tool OlivaNova (currently IntegraNova). OO-Method is articulated with Oasis and Just-UI in order to produce application from specifications and requirements of the system. Oasis is a formal object-oriented specification language. With Oasis specifications can be directly implemented in a programming language.(Pastor, Gómez, Insfrán & Pelechano, 2001) Just-UI is an extension of the OO-Method presentation model. This extension captures specification of the user interface based on the conceptual pattern language. (Molina et al., 2004) Figure 11 details the phases of the OO-Method (Molina, Pastor, Marti, Fons & Insfram, 2001).

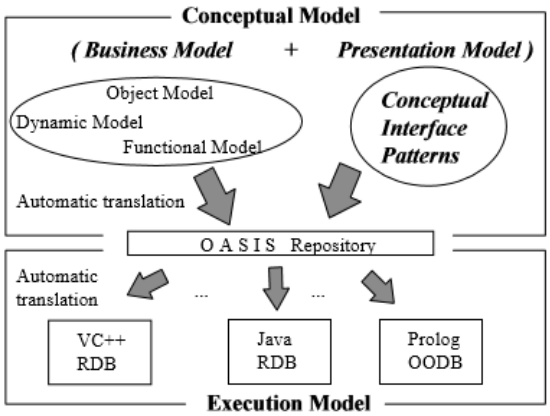



**Figure 11: The OO-Method phases (Molina et al., 2001)**

We can distinguish two models: the conceptual model and the execution model.

The conceptual model determines the components of the system without to worry about the implementation. At this level, the goal is to obtain precise specifications of the system we want to implement. To serve this purpose four sub models have to be accomplished. (Pastor et al., 2000)

- Object Model depicts the structural and static relationships of the different classes, objects… using the UML diagrams.
- Dynamic Model specifies the valid object lives and their interactions. The valid object lives description is done with State Transition Diagrams. An Object Interaction Diagram is used in order to represent object interactions.
- Functional Model captures the semantic attached to each state changes.
- Presentation Model defines how a user interacts with the system. This modelling is helped by Just-UI and gives the requirement of the Human Computer Interface according to the conceptual patterns.

From the Conceptual Model build with the four previous sub models, a formal and object-oriented specification in Oasis is obtained. As figure 11 shows us, this obtaining is automatically done. Then from the Oasis specifications another automatic transformation will be done to get the Execution Model. By this way, an application prototype is provided due to a chosen programming code allowing cross-platform.

But to enjoy this freedom, we need to abstract first the Execution Model to valid the transformation of the conceptual model into the execution model. Then once abstracted, a last step is to transform this execution model into code for the application prototype. (Pastor et al., 2000)

### 2.4.3. Engel & Forbrig method

Engel and Forbrig have achieved a method that supports the construction and adaptation of user interface models. This method is embedded in a tool called Pattern-Based Modelling and generation of Interactive Systems (PaMGIS). The basic concept is to mix methods and techniques from model-driven and pattern based development. In addition, HCI patterns description is composed of model fragments that can be applied as building blocks for the models or can influence model transformations. (Engel, Märtin & Forbrig, 2015)

The Model driven part is inspired by the Cameleon Reference Framework. The figure 12 shows how the framework is conceived with PaMGIS. There are six abstraction levels. We can notice that the Concept Model lists all the artifacts required by the Task Model and will be expressed in a XML compliant. Then it follows the classical transformation scheme of reification from Domain Model to FUI. But each transformation step can be filtered by the Context of Use Model. The Context of Use is split into User, Device, UI implementation and Environment model.

- User Model identifies specific characteristics of the users.
- Device Model comprises all the relevant features of the end device
- UI Implementation Model holds information about the UI element of the targeted platform.

- Environment Model is about the environmental influence factors (lighting conditions, noise..)

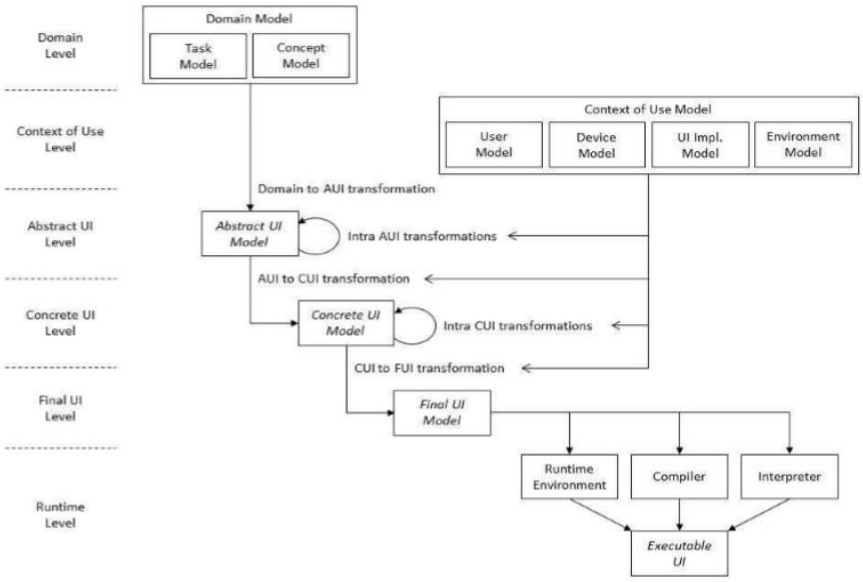



**Figure 12: Models and abstraction levels according to PAMGIS (Engel et al., 2015)**

These four models specify the context of use. When this context of use is established, it comes into play to refine and to keep only adequate elements determined in the Domain Model. It controls the subsequent transformations of the UI models. For example some task determined in the Domain model might be impractical within a certain context of use. (Engel et al., 2015)

A runtime engine provided with the development framework can create executable UI prototypes from AUI, CUI and FUI. The advantage is to identify and prevent design problems in early phases of the development. (Engel et al., 2015)

The pattern-based part alleviates the complexity of the model-driven processes. As previously mentioned, patterns furnish building blocks that might be useful to elaborate the domain and UI model. They can also serve as valuable inputs for the transformation steps. To achieve this goal, it is important to have a uniform pattern description equipped with the required information and showing the relation between patterns. So the PaMGIS Pattern Description Language (PPSL) has been created. PPSL extends the PLML with new attributes

such as Deployment which contains the model fragments and Relationships which identifies the links between patterns forming the pattern language. (Engel et al., 2015)

### 2.4.4. Decision Tree

Decision trees are one of the most employed classification method. Their success is explained by their formalism to express knowledge which are easy to interpret by expert and non-expert user. The structure is displayed as a top-down strategy where the purpose is to partition the tree in many subsets mutually exclusives. The goal of the decision tree is to obtain an answer after a series of questions. Tree is composed of nodes, branches and leafs. A node (subset) is an intermediate step where a question is asked. A branch is a possible answer at a node question, branch binds nodes. Then a leaf is the end node where there is no more question but only an answer. An answer is reached when you arrive to a leaf.
So going down a tree comes under decisions and they are considered as a tool helping to make choice. (Elouedi, Mellouli & Smets, 2000)

UsiMAD is a tool which guides the designers to elaborate a user interface through several choices. This tool is composed of the MAD framework which works with a decision tree and provides representation of widgets at various abstraction levels and in different development environments (Balsamiq mockup, HTML, Android, Windows). UsiMAD applies the Master/Detail pattern across four abstraction levels (Task and Domain, AUI, CUI and FUI) defined by the Cameleon Reference Framework (see appendix 1). (Nguyen, Vanderdonckt, Seffah, 2016 a) As observed in the appendix 1, the MAD framework inserts the conceptual pattern language in the CRF abstraction levels. So the HAT and Interaction Unit are the first subparts of the AUI.

UsiMAD is a generative design pattern-based approach for cross-device services. The choice of a generative pattern allows transforming the patterns characteristics into code generation thanks to the USIXML. This code can be generated for different programming languages and the consistency is guaranteed due to the high level of abstraction provided by the USIXML. This approach provides a way to address the problem of cross-platform. (Nguyen et al., 2016 a)

As said in the beginning, a FUI is reached after having answered to a set of questions profiling as a decision tree. For example, you first have to decide if you want the data to be displayed combined or separated. Each time, the choice made will lead you to a mutually exclusive proposition and at the end to the FUI (here implemented in HTML). But this FUI can be achieved in another way. Indeed UsiMAD combined the pattern and usability and ergonomic criteria. So it is possible to be led to the FUI by choosing the guidelines you want to see effective on your UI. The tool will generate the adequate propositions in function of your approved criteria. (Nguyen & Vanderdonckt, 2015)

In a pattern context, decision tree can be useful to better target designer's needs. It allows designers to be led towards the use of a certain pattern or another but answering perfectly the context.

### 2.4.5. Methodologies comparison and selection

In order to select a methodology to pursue this thesis, we will compare the three aforementioned methodologies. This comparison (see table 8) will be done with criteria that are indispensable for our context. As we have already mentioned, the approved methodology will have to develop a user interface which can be responsive to a multi-platform context and which integrates ergonomics guidelines. The criteria used for this comparison are:

- Ergonomic criteria: if the methodology incorporates the ergonomic criteria to build the UI.
- High level of abstraction: if the methodology provides patterns in a high level of abstraction.
- XML-compliant: if the methodology uses XML-compliant to generate the source code.
- CRF abstraction levels: if the methodology incorporates the Cameleon Reference Framework (which is a standardization of abstraction level)
- Design Guidance: if the methodology gives good design guidance to the user
- MUI: if the methodology is multiple user interface and if there is a consistent user experience

"Multiple User Interfaces (MUIs) refer to a user interface that provides access to the same information or service using multiple computing devices or platforms while providing the end user with a similar user experience" Calvary et al. (Cited in Nguyen et al., 2016 a, p 151).

MUI's is opposed to cross-device user interfaces which mean different UIs for different devices or platforms. Thus consistency might not be guarantee. (Nguyen et al., 2016 a)

| Criteria | OO-Method | Engel & Forbrig Method | Decision Tree |
|---|---|---|---|
| **Ergonomic criteria** | × | × | ✓ |
| **High level of abstraction** | ✓ | ✓ | ✓ |
| **XML-compliant** | × | ✓ | ✓ |
| **CRF-abstraction levels** | × | ✓ | ✓ |
| **Design guidance** | × | × | ✓ |
| **MUI** | ✓ | × | ✓ |
| *Legend: × criterion is not respected, ✓ criterion is respected* | | | |

**Table 8 : Comparison between OO-Method, Engel & Forbrig Method and Decision Tree**

As table 8 shows, ergonomic criteria are fully incorporated in the decision tree methodology with the USIMAD tool. Engel & Forbrig method and OO-method do not take ergonomic enough into account. Even if Pamgis offers a usability feedback comment for the patterns, it is more a lesser information. OO-Method does not use a XML-compliant to express its different models and to generate the source code. Besides, OO-method does not apply the CRF standard. The Decision tree with USIMAD and OO-Method with Just-UI is Multiple User interfaces. Pamgis is cross devices user interfaces. Only a decision tree gives a truly guidance all along the development process and leads the user to the specific FUI required.
So in the light of these criteria, the decision tree methodology coupling with the USIMAD tool will be the one applied to pursue this thesis.

We can add that the advantages offered by UsiMAD are:

- The used decision tree gives a large range of solutions at each node avoiding a simple answer like yes or no.
- The multiple context of use that are addressed thanks to the different programming languages. Big screen are possible with Windows, mobile with Android, web with HTML.

- The representation in all the abstraction level and not only focused on the AUI. The CUI and FUI are well described.
- The combination of the FUI and the ergonomic rules which was poorly done before. Each FUI has its related ergonomic rule.
- Some guidance is provided to the user. The guidance can be of two ways. On one hand, user chooses the parameters he wishes to appear on the widget. On the other hand, user can select ergonomic rules he wish to apply. Then widgets corresponding to these rules are suggested.

Now that we have scanned the literature about patterns and their particularities, we can close this chapter. We have sufficient elements to take up the next chapters of this thesis. In the following chapters, we will apply on the Service Interaction Unit what we have noticed in this state of the art. Then we will observe how the SIU is working in the Enterprise Resource Planning and we will compare these observations with our model.

# 3. SIU through the methodology

This chapter has the goal to elaborate a multi-platform pattern with all the information gathered across the State of the Art chapter. To reach this purpose, we will apply the former selected methodology on the Service Interaction Unit. It involves revisiting the MAD framework (decision tree) with the SIU through the different CRF abstraction levels. Ergonomic guidelines will be combined with the pattern in order to attempt to provide a MUI pattern. But first, the SIU will be described in a format which responds to the cross-platform context. To remind, this master thesis will propose a way to match pattern and ergonomic rules in a cross-platform context. The result will not be verified and this task will be let for further work.

## 3.1. Cross-platform pattern description

Patterns have a lot of description format according to the authors. In section 2.1.4., we have raised a series of weaknesses links to this diversity. But to overcome this scarcity of norms, HCI community brought in 2003 some standardization for HCI design patterns. They created a new language and format called the Pattern Language Markup Language (PLML). Which by its turn have known some extension such as PLMLX and XPLML. These extensions add some attributes not covered in the PLML format. The PLML format includes among other things attributes such as: Forces which reveals the forces in the environment that are resolved by the patterns, Confidence which rates the solution given by the pattern, and Implementation which gives code fragment or technical details. (Engel et al., 2012)

The PLML gives an answer to the weaknesses W1, W3, W4 and partially to W2 and W7. The PLML Forces attribute is related to the 8 ergonomic criteria of Bastien & Scapin but it's not enough. Structure is still not complete because matching and searching patterns remains a difficulty despite the fact that PLML provides standardization and responds to several weaknesses. There are still shortcomings with no reply (W2, W5, W6, W7 and W8) and new ones appear. Indeed, PLML is a natural language-based way for writing patterns and suffers from ambiguity and inconsistency. Besides PLML is more used to describing the UI than really supporting the UI design process. (Vanderdonckt & Simarro, 2010)

These remarks allow us to update our list of weaknesses. Regarding the changes bring by the PLML, W2 and W5 are adapted and W9 is added, the others remain unchanged. Below the updated list of weaknesses.

W1. **Lack of consistency**: Collections describe patterns according to different range of attributes, taxonomy criteria, conditions of application. The level of details varies from a format to another. Some attributes appear to be homonyms. All of these points make them inconsistent. For example, van Welie does not link any pattern and the gang of four have only a related pattern.

W2. **Lack of structure**: Patterns are described in Document Type Definition (DTD) with a flat structure. Thus it does not help to match and search structured patterns. Besides tags are defined in a broad way without decomposition and prevent software from using structured tags.

W3. **Lack of implementation information**: When conditions of application are not omitted, the links with implementation are almost non-existent. Only the gang of four give a concrete implementation with class diagram.

W4. **Lack of pattern evaluation**: There is for most of the patterns no evaluation and evidence about their ability to solve a problem. van Welie enables users to comment his patterns but it is not a sufficient assessment.

W5. **Limitation of contextual coverage**: The context of use (user, platform and environment) is a general tag that prevents refinement. So the separation of concerns is not fully supported. Patterns related to user interface are merely restricted to the device or platform. Besides, it is usually focused on one device at a time. So it does not help to address multiple-user interface issue.

W6. **Lack of abstraction**: Nearly all patterns are expressed for a specific operating system or render in a specific environment which prevent to switch to another one.

W7. **Lack of ergonomic approach**: There is none or not enough details about ergonomics

W8. **Limited usability consideration**: Applying a pattern in a particular case means choose between design options. This choice is led by a usable solution with varying degree of quality. But patterns give almost no information about their usability or user experience.

W9. **Lack of expressivity**: Some tags are too narrow on one aspect and do not cover the counterpart i.e. Forces tag but no counter forces tag.

In order to fill these weaknesses, Nguyen et al. (2016 b) present the User Interface Pattern Language Markup Language (UIPLML). UIPLML is pattern language, a description format and a software application which supplies all the information needed to develop a User Interface. Appendix 2 shows the meta-model of the UIPLML in a UML 2.0 class diagram. UIPLML compiles all the attributes from the existing collections and provides new ones to support the MUI development. For instance, the UI pattern is improved with a SWOT analysis. The context is deeply depicted through the ContextProblem and the ResultingContext. A level of evidence ranges the pattern from 1 to 7. There are 3 different approaches of implementation (by code fragment, by description in a high level of notation, by giving a reference in the four CRF abstraction levels).
Besides, UIPLML application allows a clear visualization of the pattern across a expandable/ collapsible tree. The application is built on four pattern databases (MUI Patterns, Context-aware patterns, Culture-aware patterns, Accessibility patterns) and gives a lot of possibility for searching and matching patterns. The application has an editor viewpoint for those having the access and a end user viewpoint. So in that way patterns can be up to date. Usability and Ergonomic guide are brought by the editors contributions and can inform the users in their patterns queries. (Nguyen et al., 2016 b)

As UIPLML fully solves the stated weaknesses, we will use the UIPLML description format to express the SIU. Besides UIPLML description is a completing tool to USIMAD to develop a cross-platform pattern.

### 3.2. SIU description format

In table 9, the Service Interaction Unit is illustrated within the UIPLML description format formerly presented.

| Attributes | Explanation |
|---|---|
| <Pattern Name> | Service Interaction Unit |
| <Pattern Alias> | Service Presentation |
| <Problem> | The user wants to execute a service. For this purpose he needs to provide inputs for this service to be executed. Inputs can be required into several ways through arguments. In this scenario, the Service Interaction Unit handles arguments via auxiliary patterns (entry, defined selection, argument grouping and argument dependency). |
| <Pattern Synopsis> | Service Interaction Unit manages argument data-type as well as argument display and relationships in order to execute a service. |
| <Classification/Template> | Structural /object centric |
| <Solution> | SIU organizes the argument presentation, suggests default value for arguments and puts in place controls for the input to be entered. In addition, it allows providing information about the services and their arguments, but also when error occurs (feedbacks). |
| <Restriction> | SIU has to be used with its auxiliary patterns. Inputs must satisfy the SIU requirements to execute the service. |
| <Forces> | This pattern facilitates a frequent interaction occurring in many contexts. It makes tasks easier for the user and saves him time thanks to arguments organizations and error prevention. |
| <Weaknesses> | Argument labels can be ambiguous. SIU focuses on form-based interaction thus screen size impacts the form organization. |
| <Applicability> | SIU is used when a user wants to execute a service that requires inputs from him. |
| <Structure> | SIU has an aggregation link with its auxiliary patterns. |
| <Collaborations> | Objects operate through their dependencies and attributes. |
| <Participants> | The four auxiliary patterns |
| <Known Uses> | Every time there is a form, this pattern should be used. ERP are software with high use of forms. |
| <Consequences> | Need to adapt this pattern on different platforms. Platform information is necessary. |
| <Pattern Link> | Population Interaction Unit, Instance Interaction Unit, Filters pattern, Display Set pattern. |
| <Rationale> | This pattern simplifies and optimizes the interaction between the user and the system to execute a service. |
| <Context of Use> | As forms are encountered for all type of users and all kind of environment, this pattern has no restriction. |

| | |
|---|---|
| <Implementation> | Take into account the aggregation link between SIU and its auxiliary patterns. |
| <Threats> | In the literature, this pattern is generally covered in multiple fragments rather than one entity. So the risk is a lack of structural guidance letting the user with several solutions without knowing which one to apply. |
| <Opportunities> | Forms which are the well-known expression of this pattern are frequently used and can be handled without lots of computing skills by using easy to use software. |
| <Level of Evidence> | 4 (experts description) |

**Table 9 : SIU in the UIPLML description format**

### 3.3. Ergonomics rules and SIU pattern

In this section, we have tried to gather all the fragments (patterns, ergonomic rules) related to the SIU in order to enrich it. To proceed, we set up a list of indispensable points mixing patterns and ergonomic criteria and completing the auxiliary patterns. We took elements from the ergonomic criteria of Bastien and Scapin, the " giving input" section of van Welie, the "getting input from Users" of Tidwell, "form bloopers" of Jeff Johnson and several ergonomic guidelines from big companies and institutions, related to the fact of entering input in order to execute a service.

The most common method of interaction to execute a service is the data-entry form. Forms are used no matter the context, devices and user type. So, there is a real need to improve the SIU pattern with ergonomic guidelines. To reach this purpose we have analyzed the form regarding the gathered elements. This analysis consists in splitting up the form in two sections: the tools used in a form and the presentation of these tools in a form.

#### 3.3.1. Tools used in a Form

The first tool is the **field**. The field can capture a large variety of data. The auxiliary pattern that can be associated to, is the introduction/entry basic element of the SIU. In fact, the properties (see 2.2.3.1) of these basic elements describe what a field must have and must be. Consequently, we will first interpret the different kind of field with the properties and then we will complete these properties with ergonomic rules. They are different kinds of

fields. Table 10 presents them in function of the properties of the entry/introduction pattern.

| Name | Free -text Fields | Text Fields | Number Fields | Mix Fields |
|---|---|---|---|---|
| Data type | String (ex: leave a comment) | String (ex: author's name) | Integer, Real (ex: phone number) | String (ex: national registration number) |
| Edit Mask | / | depending on the required input (Johnson, 2003) | depending on the data required (Johnson, 2003) | depending on the required input (Johnson, 2003) |
| Valid Value Range | arbitrary number of character (Johnson, 2003) | According to the text required and the text length (Ergolab, n.d.) | according to the data required and the data length (Ergolab, n.d.) | according to the data required and the data length (Ergolab, n.d.) |
| Default Value | / | choose by common sense, experience and site data, can be also based on user's data or arbitrarily (good default pattern) (Tidwell, 2010 ; Johnson, 2003) | choose by common sense, experience and site data, based on user's data or arbitrarily (good default pattern) (Tidwell, 2010 ; Johnson, 2003) | choose by common sense, experience and site data, can be also based on user's data or arbitrarily (good default pattern) (Tidwell, 2010 ; Johnson, 2003) |
| Help Message(if necessary) | / | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) |
| Validation Message | / | Depending on the error but indicate on the same page than the form (Same-page error messages pattern): what and where is the error and what is expected. ( Tidwell, 2010 ; van Welie, 2008) | Depending on the error but indicate on the same page than the form (Same-page error messages pattern): what and where is the error and what is expected. ( Tidwell, 2010 ; van Welie, 2008) | Depending on the error but indicate on the same page than the form (Same-page error messages pattern): what and where is the error and what is expected. ( Tidwell, 2010 ; van Welie, 2008) |

**Table 10 : Combination of ergonomic rules and the Entry auxiliary pattern in regards to fields.**

Hereunder are ergonomic rules completing these four fields. We distinguish ergonomic guidelines in two parts, those common to the four fields and those specific to a field-type.

*Common ergonomic rules*

- Labels:
    a. must be close to the field with the same alignment. Left alignment optimizes form reading. But if the longest label is six characters longer than the smallest label, alignment should be to the right (see figure 13 and figure 14).(Mayhew,

1992) Label on the top is used to faster the reading or to suit a small screen.(Misfud, 2011 b)

b. must have a consistent style. If question, keep question but not mix it with noun phrase or command… (Nielsen, 1995 ; Bastien & Scapin, 1997)

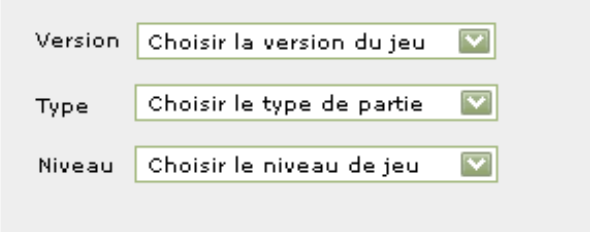



**Figure 13: Right alignment of the labels (Ergolab, n.d.)**

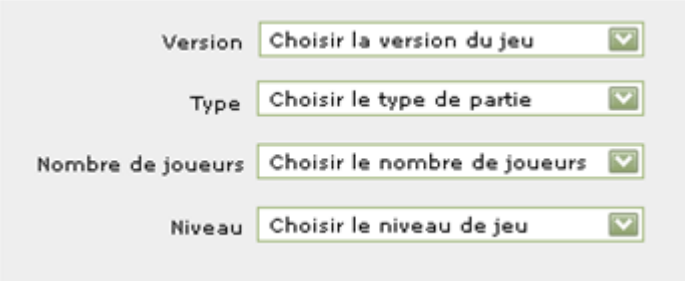



**Figure 14: Left alignment of the labels (Ergolab, n.d.)**

- Principle of compatibility is necessary. Texts need to be written with words the target understands. Don't speak geek to non-enlightened. (Johnson, 2003 ; Tidwell, 2010) (familiar language from table 7)
- Compulsory input focus on the first field to avoid user wasting times by clicking on it. (johnson, 2003 ; U.S. Department of Health & Human Services, n.d.)
- Indicate clearly which fields are required. (Johnson, 2003 ; Misfund, 2011 b)
- Respect the information density principle by balancing the information load (help Message and Validation message) (Minimized effort from table 7) (U.S. Department of Health & Human Services, n.d.)

*Specific ergonomic rules for text, data, mix field*

- Control the data entered when there is a specific type with edit mask with i.e. structure format pattern (figure 15) (Tidwell, 2010) and/or data-type recognition.


**Figure 15: Edit mask example (Toxboe, n.d.)**

- Accept common format. For example, time can be expressed differently around the world, Zip codes structure varies in function of the country. Tidwell (2010) provides the forgiving format pattern.
- If uppercase or lowercase letters are irrelevant don't make the field case-sensitive.
- Arrange the fields in the form of a sentence that enlightens the user with the fill-in-the-blanks pattern (figure 16) (Tidwell, 2010). E.g. "text field" is married to "text field"


**Figure 16: Fill-in-the-blank pattern example (Tidwell, 2010, p362)**

- For date provide a picking date calendar. (Toxboe, n.d.)
- Let's prefill the field with an example or a question that prompts the user about what to do (input prompt pattern). (Tidwell, 2010 ; Toxboe, n.d.)


**Figure 17: Input prompt pattern example (Google Belgium webpage)**

- Help the user when is typing by showing a set of selectable possible answers, autocompletion pattern (Android Developers, n.d. ; Tidwell, 2010).

The second tools used in a form are controls. Controls can be **radio button, menu, checkbox and slider.** They offer the possibility of a choice to the user. It is completely determined by the defined selection basic element.

| Name | Menu | Radio button | Checkbox | Slider |
|---|---|---|---|---|
| Data Type | can be any type | can be any type | can be any type | Can be any type |

| | | | | |
|---|---|---|---|---|
| Value Set | more than 5 values available (UX Movement, 2013) | less than or equal 5 values available (UX Movement, 2013) | binary values available (Microsoft, n.d.) | Depending on the evaluation type: Likert scale varies with 4, 5, 6, and 7 values available (Likert, 1932). Range slider strictly numerical values but no restriction set. |
| Help Message | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) | Can be in-line help box differentiated from normal content, help box triggered by clicked icons, help box triggered by pointer resting on field. (Toxboe, n.d.) |

**Table 11 : Combination of ergonomic and the defined selection pattern in regards to control tools**

Hereunder are ergonomic rules completing these four control tools. We distinguish ergonomic guidelines in two parts, those common to the four controls and those specific to a control-type.

*Common ergonomic rules*

- Control the data-type by this set of choices.
- Labels alignment and style use the same rules as fields.
- Principle of compatibility is the same as fields. Besides keep clear labels (see figure 18) e.g. don't switch the meaning of on/off. (Johnson, 2003)


**Figure 18: Bloopers in the worded labels (Johnson, 2003, p151)**

- Respect the information density principle by balancing the information load (help Message). (Minimized effort from table 7) (U.S. Department of Health & Human Services, n.d.)
- Provide as much as possible default values by common sense, experience and site data, can be also based on user's data or arbitrarily (good default pattern) (Tidwell, 2010). E.g. select the country based on the IP address. Checkbox has by definition a default (on or off). (Johnson, 2003)
- Controls must always be editable. (Johnson, 2003)

*Specific ergonomic rules*

- No default for menu and radio button is preferable in the case to avoid any presumptions or in the case of a social, political or legal requirement. For example, male or female? It is more natural for menu than radio button to not have a default. (Johnson, 2003)
- Set a prompt serving as menu label or an instruction. Use this option only if prompt are carefully worded. Avoid the prompt "please choose" and replace it by please choose + object nature. See figure 19. (Toxboe, n.d.)

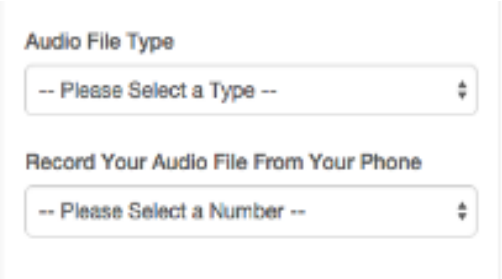



**Figure 19: Prompts examples (Talkroute, n.d.)**

- When facing a binary choice (on/ off), adopt a checkbox if change are not immediate. Otherwise use a toggle. (Microsoft, n.d.)
- Concerning menu, radio button and checkbox, if the non-yet option is an important choice make it explicit and/or by default (Johnson, 2003)

### 3.3.2. Presentation of the Form

The way to display the form is guided by some ergonomic rules and patterns. This display responds also to the argument grouping basic element of the SIU. Actually the argument grouping orders the form arguments and can display them by groups and subgroups. The form may be displayed in a single page or in multiple pages. Each page may have one group or multiple groups. The rule that matters about the number of groups by page relates from an arbitrary choice. Indeed, this choice depends on the organization themes of the form.

*ergonomic rules*

- Analyze the target task to discover the steps the users expect in the aim to establish the order and the organization (compatibility). (Bastien & Scapin, 1997)
- Organize form as a conversational flow.
    a. Orders the labels logically to reflect this natural flow. For example, ask name field before bank account field. (Misfud, 2011 b)
    b. Label should address one topic at a time preventing user confusion and skipped input. (Misfud, 2011 b)
    c. Groups and form pages should reflect the natural pause of a conversation. (Misfud, 2011 b)
    d. Remove clutter such as banner and unnecessary navigation to not distract users. (Misfud, 2011 b ; Sun Microsystems Inc., 2001)
- Groups should be designed in function of the task characteristics according to the logical sequence of use, or the use frequency or the relative significance. Related information should be grouped together. (U.S. Department of Health & Human Services, n.d. ; Ergolab, n.d.)

- Groups have clear separation. The separation can be done by a space separator, a line separator or a group box. (Google, n.d. ; Microsoft, n.d.) Figure 20 shows a form separation by group box.

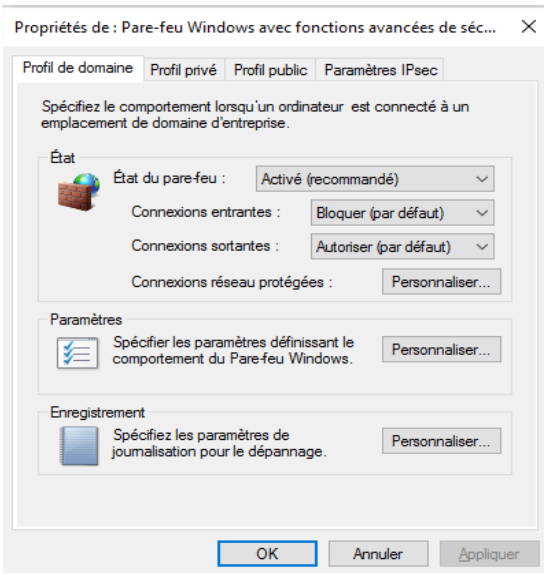



**Figure 20: Group box example (Windows 10)**

- One page form advantage is a faster task execution but one page deals with higher information load and dependent field complexity (actualisation by reloading). (Ergolab, n.d.)
- Multiple page form advantage are low information load, error management simplification and easier field dependency but multiple page have slower task execution and hard to estimate the task time. (Ergolab, n.d.)
- In case of multiple page form, steps in the task achievement have to be shown to the user (see figure 21). Wizard, tabbed dialog boxes and accordion are good to show the process flow. (van Welie, 2008 ; Google, n.d.) Wizard is a succession of forms, tabbed dialog boxes works like navigation between forms and accordion displays collapsible forms (Yahoo! Developer Network, n.d.). Figure 21 shows a wizard.

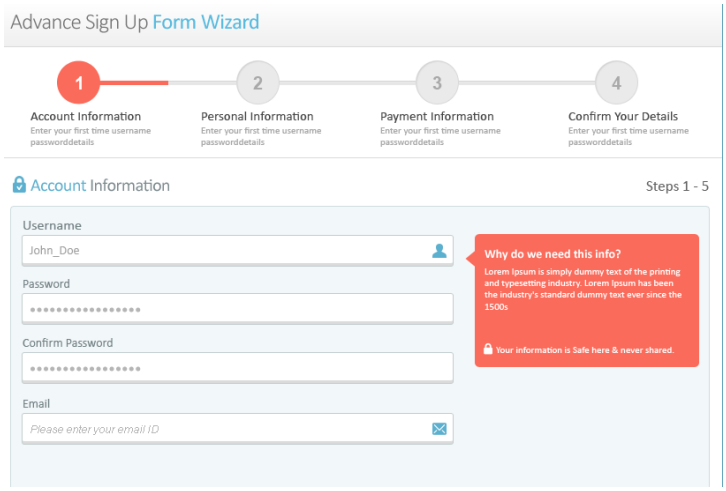



**Figure 21: Wizard example (Envatomarket, 2014)**

- Give feedback when a task or a step is achieved. (Guidance and Feedback from table 7)
- Interface choices are maintained in a same context and are different in a different context. (Consistency from table 7) (Sun Microsystems Inc., 2001)
- Follow the explicit controls with the explicit user actions and the user control. That means the system executes only the action asked by the user and the fact that user can cancel this action. (Bastien & Scapin, 1997)

The relation between arguments is determined by ergonomic rules and patterns. Specifically, argument dependency is something that must run really well and must be well designed. On one hand, the auxiliary pattern Argument Dependency of the SIU explains how dependency works and how it is possible to attribute new value/state to an argument from a trigger event. On the other hand, relationship may concern an argument and a collection of object (population). In the case of object as input argument, the object selection among a population may be helped by a filter (Molina et al., 2001). So we are able to distinguish three kinds of relationships: argument dependency to fill other arguments or to hide/ disclose other arguments and argument dependency to filter a population.

*Ergonomic rules*

- Check all the interactions to avoid infinite loop. E.g. within a group of arguments, an event triggers a new action on argument1, argument1 creates new value for argument2 which triggers a new action on argument 3, which makes a loop by triggering a new action on argument1 etc…

- Avoid redundant request by using given input to determine related arguments. Redundancy is available only for security matters but it has to be explained. (Johnson, 2003)
- Use dependency to adapt dynamically the form user will face. For example, use dependency to prefill other arguments. Status recovery pattern (Pastor et al., 2000) But also to hide/ disclose and filter arguments.

### 3.4. Extended USIMAD with the SIU

The natural continuation of the previous sections is to develop the SIU through the Cameleon Reference Framework. To perform this exercise, we will first elaborate the task and domain of the Service Interaction Unit. Secondly, we will decompose the SIU in the abstract user interface level and with these features, we will manage the concrete and final user interface. Finally based on this breakdown, we will construct a decision tree to lead the user through the different abstraction levels (AUI, CUI and FUI). All this analysis is focused on how to elaborate an interaction across forms.

#### 3.4.1. Domain Model

The domain model is a conceptual model that incorporates the behaviour and the data.

“A domain model captures the most important types of objects in the context of the business. The domain model represents the ‘things’ that exist or events that transpire in the business environment.” (Jacobson, Booch & Rumbaugh, 1999, p 87)

The domain is represented by the UML class diagram. In the figure 22, the service interaction unit is expressed in the UML class diagram. On one side, we can observe the SIU as a class in its own right representing the service that will be executed. On the other side, we have the four auxiliary patterns (entry, defined selection, argument grouping and argument dependency) representing specificities that support the service execution. The relation that links the SIU is an aggregation. The SIU has a collection of exactly one of each auxiliary pattern. However each auxiliary pattern can be part of several Service Interaction Units. Finally, erasing the SIU does not mean the auxiliary pattern removal.

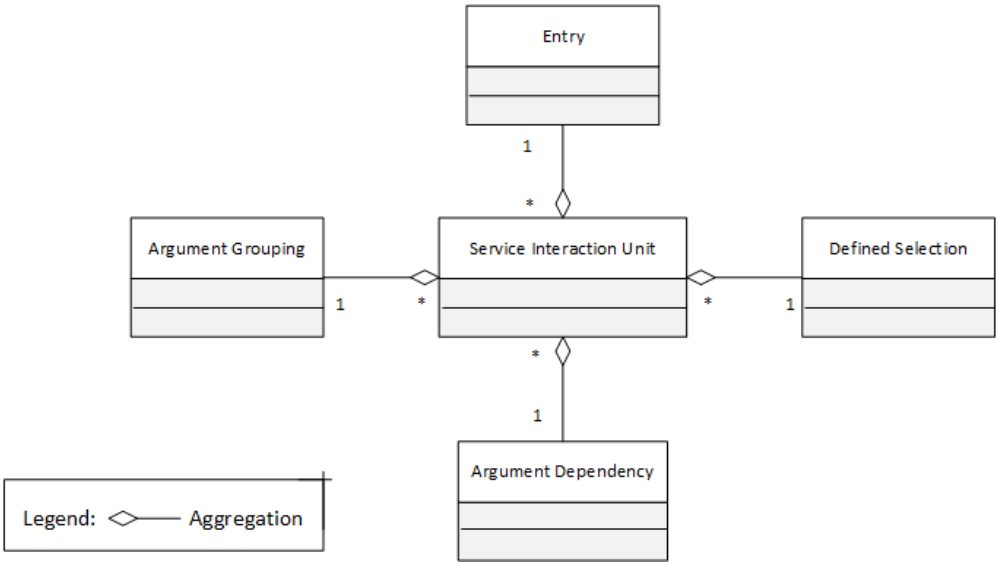



**Figure 22 : Domain model of the SIU**

### 3.4.2. Task Model

Based on this domain model, we are able to elaborate the task model. The task model allows assembling all the actions made by a user in order to achieve his goal (here, a service achievement). Each task is ordered with constraints and subtasks. This breakdown structure gives a more important level of precision. Indeed a global task, e.g. hire new employees, is common to all enterprise but the process (subtasks) can be completely different for each entity. The task model presented below is drawn in respect to the UsiXML Task model and is expressed with the Concur Task Trees notation. (Paterno, 1999) In our case, the Task Model is an illustration of the tasks a user might encounter when he faces a form.

Before we present the Task Model, concepts of the CTT notation are defined in tables 12 and 13. Table 12 shows the possible temporal operators ordering the tasks and table 13 is about the different types of task. (Paternò, Santoro & Spano, 2012)

| Temporal operators | Explanation |
|---|---|
| T1 >> T2 | Second task cannot begin until first task performed. |
| T1 []>> T2 | When performed first task information is used by second task. |
| T1 \|> T2 | Interruption of task 1 by task 2 and resumption of task 1 when task 2 performed. |
| T1 [] T2 | Mutual exclusive choice. |
| T1 [> T2 | Complete interruption of the first task by the second |
| T1 \|=\| T2 | Tasks can be done in any order but has to be performed before to switch to another. |
| T1 \|\|\| T2 | Tasks can be performed in any order or at the same time and do not need to be completed to switch to another. |
| T1 \|[]\| T2 | Concurrent tasks can exchange information. |

**Table 12 : Temporal operators in the CTT notation**

| Illustration | Type | Explanation |
|---|---|---|
|  | Interaction | All the tasks performed by the user and requiring an interaction with the system. |
|  | Abstraction | Task composed by different types of subtask. |
|  | User | Task only performed by the user. |
|  | Application | Task only performed by the system. |

**Table 13: Types of task in the CTT notation**

With these concepts, we are able to analyse figure 23 which depicts the Task Model of the SIU. We observe that the user, who wants to execute a service, has first the possibility to provide information or to exit the service. If he chooses to provide information, he is allowed to complete form 1 or another form, depending on the number of form. Once he is completing a form, he faces two kinds of tasks that have to be performed at any order: enter value and select value. There are as many enter/ select value tasks as there are number of arguments calling for them. When all forms are completed the user can submit it. This action stops the provide info task. Afterwards, the system thanks to the information gathered executes the service and in the end it shows the results.

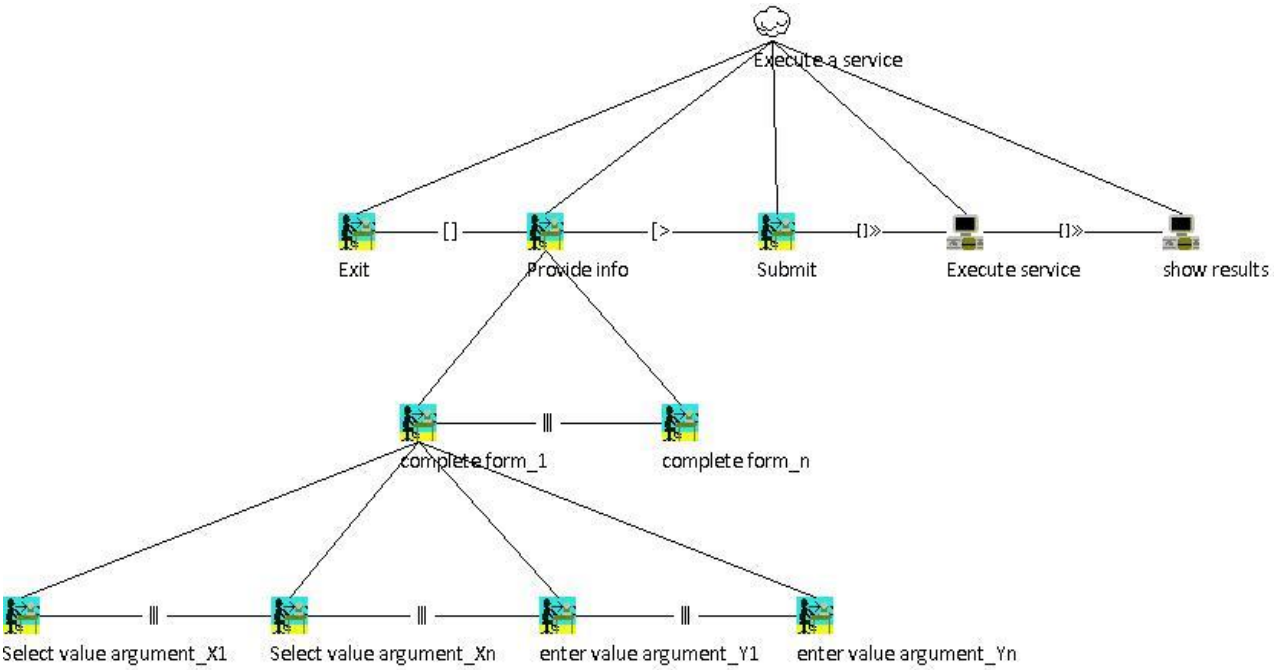



**Figure 23: Task Model of the SIU expressed in the CTT notation**

### 3.4.3. SIU across the AUI, CUI and FUI

The original MAD framework (appendix 1) is created with a hierarchical tree representing the distinct abstraction levels. In this framework, the AUI is separated in sublevels which refine the Master/Detail in specific parameters. Then these parameters are mocked up in the CUI before to be designed in a specific programming language in the FUI. The first two AUI sublevels relate from the conceptual model described in section 2.2.3. Sublevel 1 just tells us that the rest of the sublevels are organized in a hierarchical action tree. Sublevel 2 shows the four Interaction Units. But USIMAD has a third sublevel enriching the conceptual model. In fact, the conceptual model (figure 8) has the weakness to not illustrate tangible use of the conceptual patterns. So the following sublevel determines parameters proper to the master/detail pattern.

In the aim to revisit USIMAD with the SIU, we will present here an adapted MAD framework for the SIU. Given that the most used interaction to receive input takes place in forms, we will configure this framework with the help of the ergonomic rules and patterns established in section 3.4.

Appendix 3 represents the SIU across the all the abstraction levels. As we can observe the first two levels of the AUI are identical to the MAD framework. That is to say, the hierarchical action tree in level 1 and the Service Interaction Unit in level 2. Then the level 3 consists of four parts which refine the SIU in parameters proper to build a form.

In figure 24 gives us the full conception of these four parts of level 3. It starts with the concern types of the form. Three concerns are possible: input, organization and relationship. Starting from these three options the following levels (3.1, 3.2 and 3.3) offer specific refinement to determine the parameters. For example, if the concern is about the input, an input refinement (level 3.1) is suggested to the user between entry, selection and evaluation. For example, level 3.2 determines for entry two possibilities. User can need input with a unique data type or mixing data type. If he needs unique data type, he has the choice between text, number and masked number type (level 3.3). There is no case where mask is used only with text. Indeed, it is always encountered for a mix of data. E.g. mail address. Sometimes the refinement does not need to go as deep as level 3.3. In the case of form organization, if the user chooses to display the form in one single group, the refinement stops at this level. It will be directly mocked up in the CUI.

Now that we have seen how is configured the Service Interaction Unit in the Abstract User Interface, we are able to take the CUI and the FUI up. Figure 25 makes the link between the AUI and these two last abstraction levels. The Concrete User Interface gives the corresponding widgets of the SIU parameters and their mockups. These mockups have been designed with Balsamiq software. Starting from the level 3.3, the immediate change (binary choices) has the toggle widget in the CUI. Then we have the mockup of the toggle. The last step consists in interpreting the CUI mockups into a programming language. Final User Interface level exhibits them in HTML and Android. For the sake of visibility, figure 25 only shows the representation in HTML. But the whole abstraction levels are available in appendix 3.

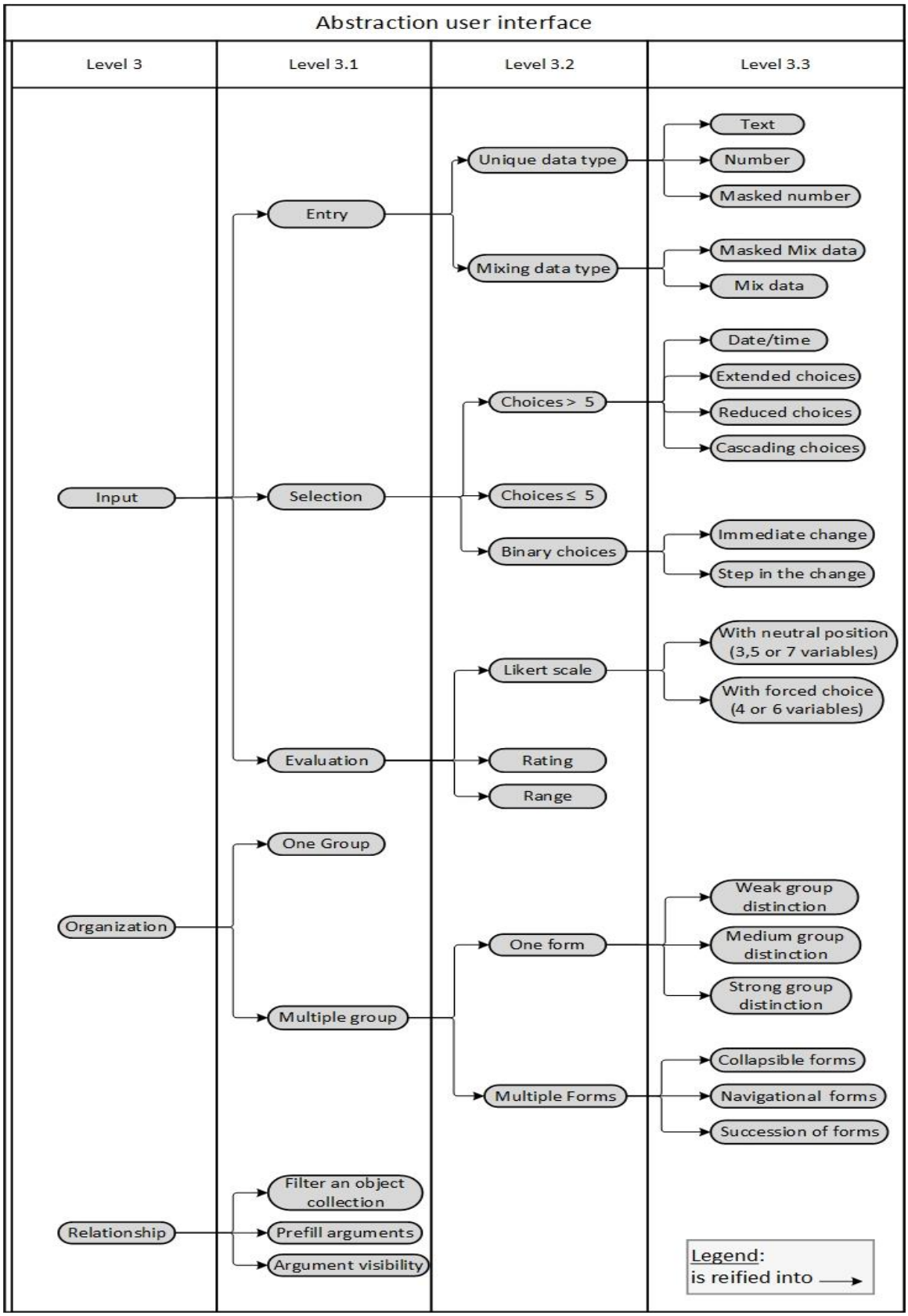



**Figure 24 : Abstract User Interface of the SIU**

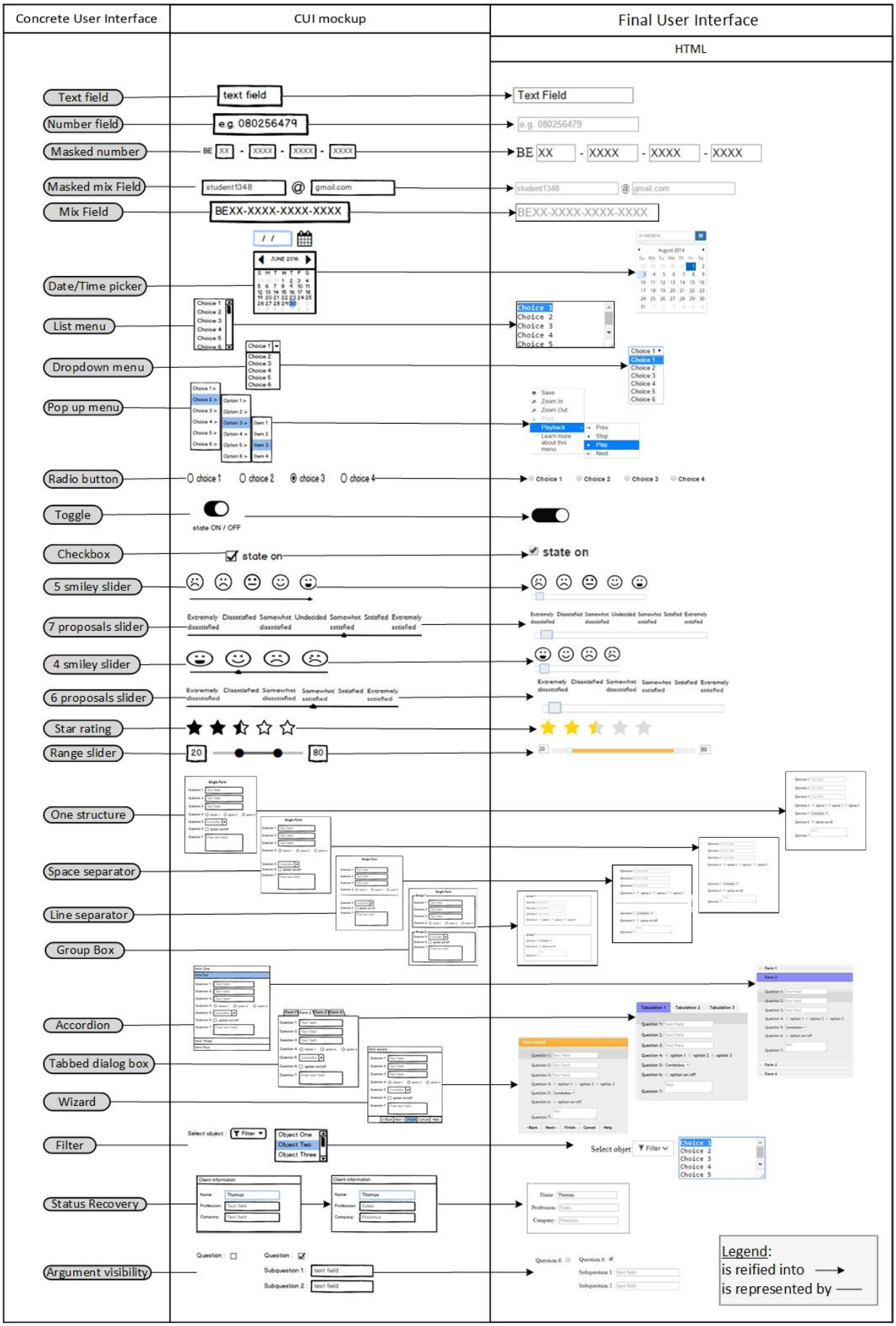



**Figure 25 : SIU across the CUI and the HTML FUI**

### 3.4.4. Decision Tree

Now that we have all the parameters to establish the SIU, we can improve the guidance with a decision tree. Thanks to this decision tree the user choice will be facilitated to reach a widget. This decision tree is not restricted to a binary answer (yes or no) which enables a wide range of possibilities. The combination of a decision tree and the abstract parameters is very helpful to develop a form. A fully representation of this decision tree is available in appendix 4. But for a better comprehension, we will analyse it step by step.

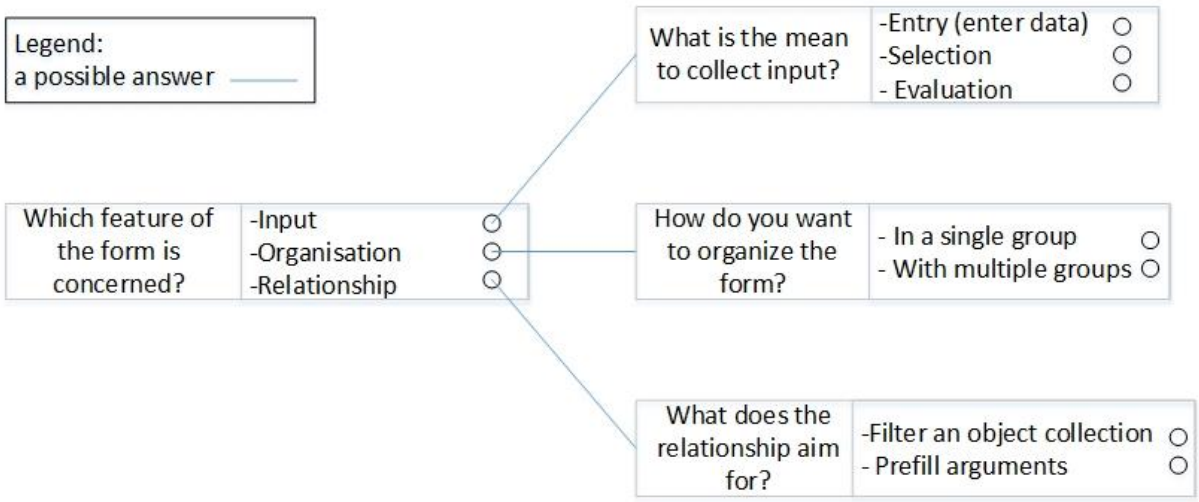



**Figure 26 : Form feature choice in the SIU decision tree**

The first question asked to the user is which form feature is concerned (figure 26). The user has the possibility to handle the input, the organization or the relationship. This question will be the starting point to go further in our analysis. Indeed, we will first present the decision tree inherent to the input. Then we will approach the organisation and finally the relationship.

*Input*

The input is oriented by three means to collect input: an Entry (enter data), a selection and an evaluation.

The entry is further refined by asking how the user needs the data. He may want a unique type of data or a mix of data type. In the case of a unique data type, the propositions are Text (letters and characters) and Number with or without mask. The respected widgets are Text Field, Masked number field and Number Field. Fields are predefined to only accept

input corresponding to a Text or a Number. Otherwise, it takes us back to the previous proposition which was a mix of data. The mix of data is refined by having mask or not. The corresponding widgets are mixed data field and masked mixed data field. These fields allow number and text. (Figure 27)

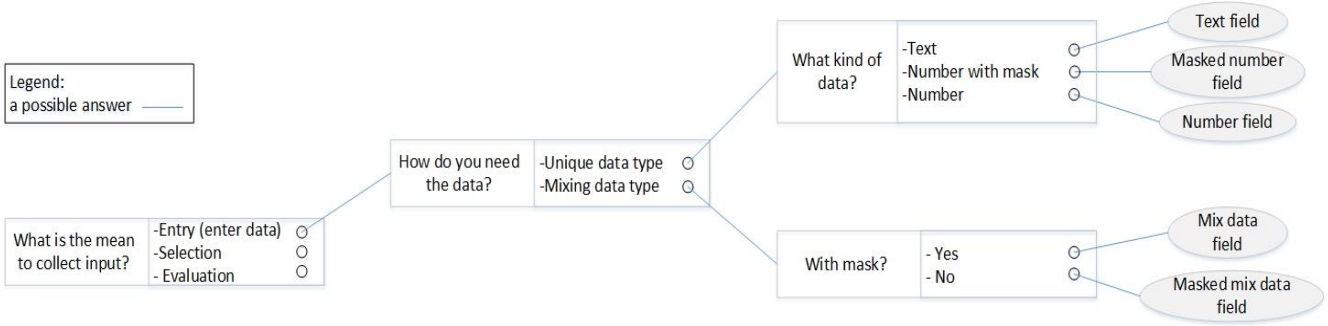



**Figure 27 : Steps of the entry mean in the SIU decision tree**

The second means to collect data is a selection. We first need to determine the number of value composing a set. If the set is larger than 5, there are 4 possibilities to display it. If the set is equal or less than 5 choices, the only representation allowed is a radio button. If the selection concerns a binary choice, another parameter is required to know when the change of state occurs (on or off). Toggle corresponds to an immediate change while the checkbox happens once the form is submitted. (Figure 28)

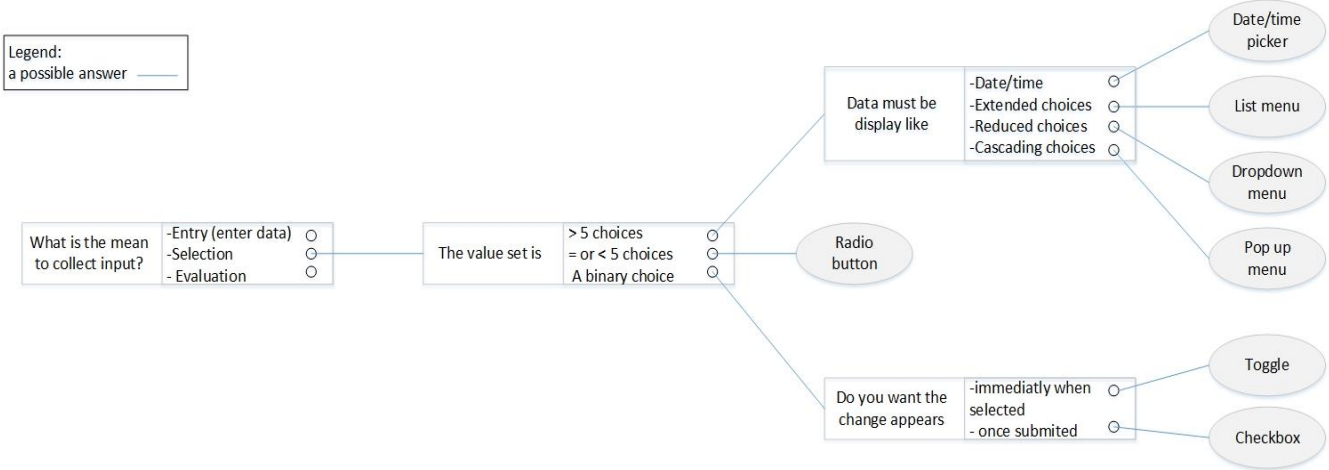



**Figure 28 : Steps of the selection mean in the SIU decision tree**

The last means used for input is the evaluation. To determine the selection, the user has to opt for a tool between a Likert scale, a rating and a range (figure 29). The Likert scale can be configured with a neutral position or with a forced choice. This configuration depends on the number of variables and even numbers compel to make a choice. The number of variables for a Likert scale cannot exceed seven (Likert, 1932). We arbitrarily choose two kinds of

illustrations that are often used: the smileys and the proposals. Four widgets are available and all work with a slider. Another tool enabled is the rating through the very widespread star rating widget. The last evaluation tool is the range which gives the range slider widget.

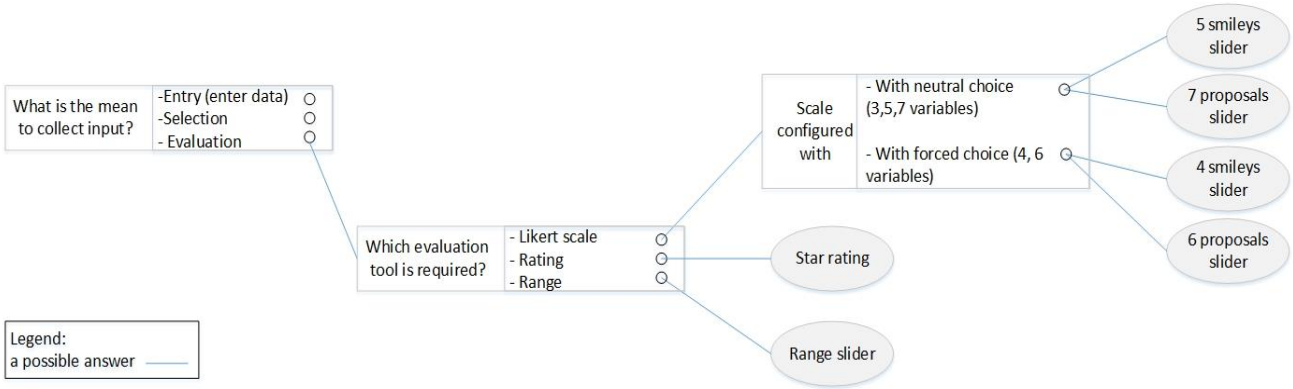



**Figure 29 : Steps of the evaluation mean in the SIU decision tree**

*Organisation*

The form can be elaborate in a single group or using multiple groups (figure 30). When the form is composed of a unique group it will be display in one structure (one page). On the contrary having multiple groups enables to display them in one form (one page) or in several forms. When multiple groups are contained in a form, they can have weak, medium or strong distinction between them. Three widgets have this role respectively the Space separator, the Line separator and the Group box. When groups are displayed through several forms, the user must select how to arrange forms with each other's. They can be ordered as being collapsible (Accordion), as a navigation (tabbed dialog box), as a succession (Wizard).

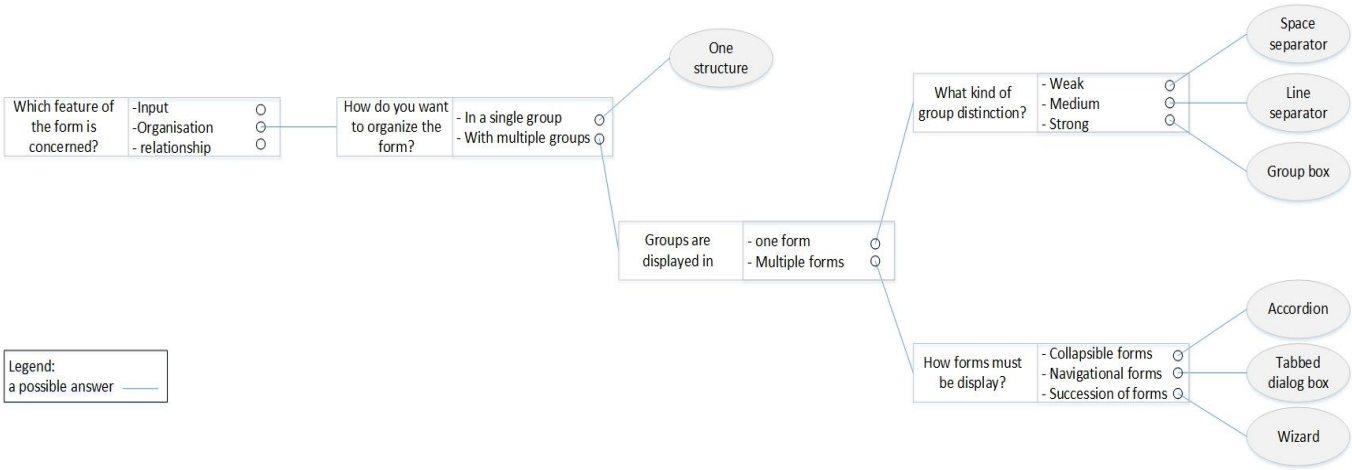



**Figure 30 : Steps of the organisation features in the SIU decision tree**

*Relationship*

Form arguments are in some cases linked to facilitate the form use by diminishing the user task. That is why there is a dedicated section to these relations in our decision tree (figure 31). Most of the time, the relationship between arguments is used to filter objects among a collection. In this case, the relationship highlights the filter use but the argument filter is determined by the input selection. But it can also pre-fill arguments earlier registered with the status recovery and it can hide/disclose arguments with the argument visibility. The pre-filling and the visibility operate on the basis of ECA (Event, Condition, Action) mentioned in section 2.2.3.1.

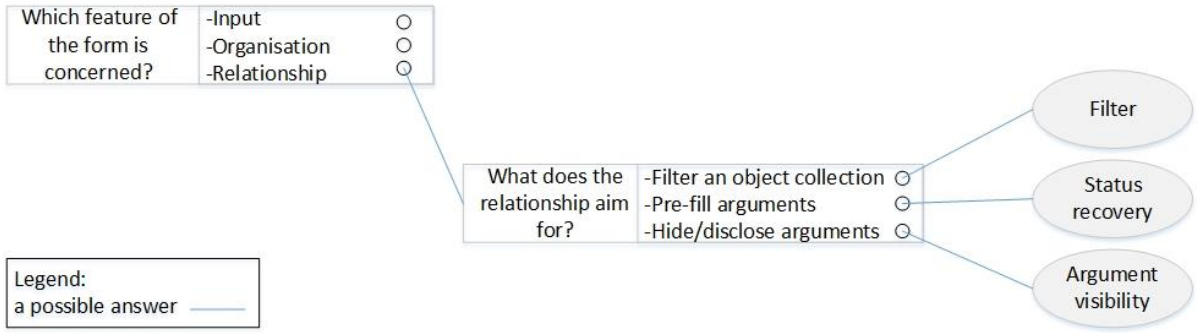



**Figure 31 : Steps of the relationship feature in the SIU decision tree**

We have now looked over the decision tree and we can notice that it could be a very helpful tool for the user. Moreover this tree does not provide yes or no answers but instead it suggests a wide range of choices.

Through this chapter, we have expressed the SIU as a cross-platform pattern. Firstly, the Service Interaction Unit has been described in a MUI compliant format. Then an analysis of the ergonomic guidelines present in forms and a repository of them have been carried out. And finally, the SIU has been interpreted in the light of the decision tree methodology with the help of the ergonomic guidelines repository.

## 4. Study case: SIU in the light of a specific ERP process

The goal of this chapter is to confront our cross-platform SIU pattern to a study case. We will examine how forms are used and built in this study case and we will compare them to our decision tree. As already mentioned in chapter 1, this study case corresponds to a real process met in the Enterprise Resource Planning (ERP). So as to do this comparison, this chapter will be advanced in five steps. First of all, we will detail what is an ERP. Afterwards, we will list very famous ERP processes and we will select one of them with a high use of form. Then, we will analyse deeply this process with the help of the Business Process Modelling Notation (BPMN). Last but not least, we will compare our way to construct a form with the forms used in Odoo, an open ERP software. By means of this confrontation, we will be in position to adjust our decision tree or to suggest Odoo possible improvement for their forms.

### 4.1. What is an Enterprise Resource Planning?

"Enterprise resource planning (ERP) system is a business management system that comprises integrated sets of comprehensive software, which can be used [...] to manage and integrate all the business functions within an organisation". (Shehab, Sharp, Supramaniam & Spedding, 2004, p 359)

This definition gives us a clear comprehension of the ERP purpose. Besides to enable the business functions integration, the ERP centralizes the information in a unique database shared by all the business departments. So the Enterprise Resource Planning is part of the information systems. Usually, ERP are composed by a set of mature business applications and tools for finance and accounting, sales and purchases, manufacturing, inventory and supply chain, human resource and marketing. (Investopedia, 2005) Figure 32 (Shehab et al., 2004) provides a good illustration of what is an ERP.

Traditionally, application systems used by organizations treat the transactions separately like silo. These systems are designed with strong boundaries between the specific functions for which only specific applications cater. Conversely, ERP counteracts these boundaries by treating these transactions not as stand-alone activities but rather as parts of interconnected

processes constituting a global business. (Shehab et al., 2004) This unified system prevents from errors of data transaction from a system to another by suppressing incompatible systems. (Picardo, 2016)

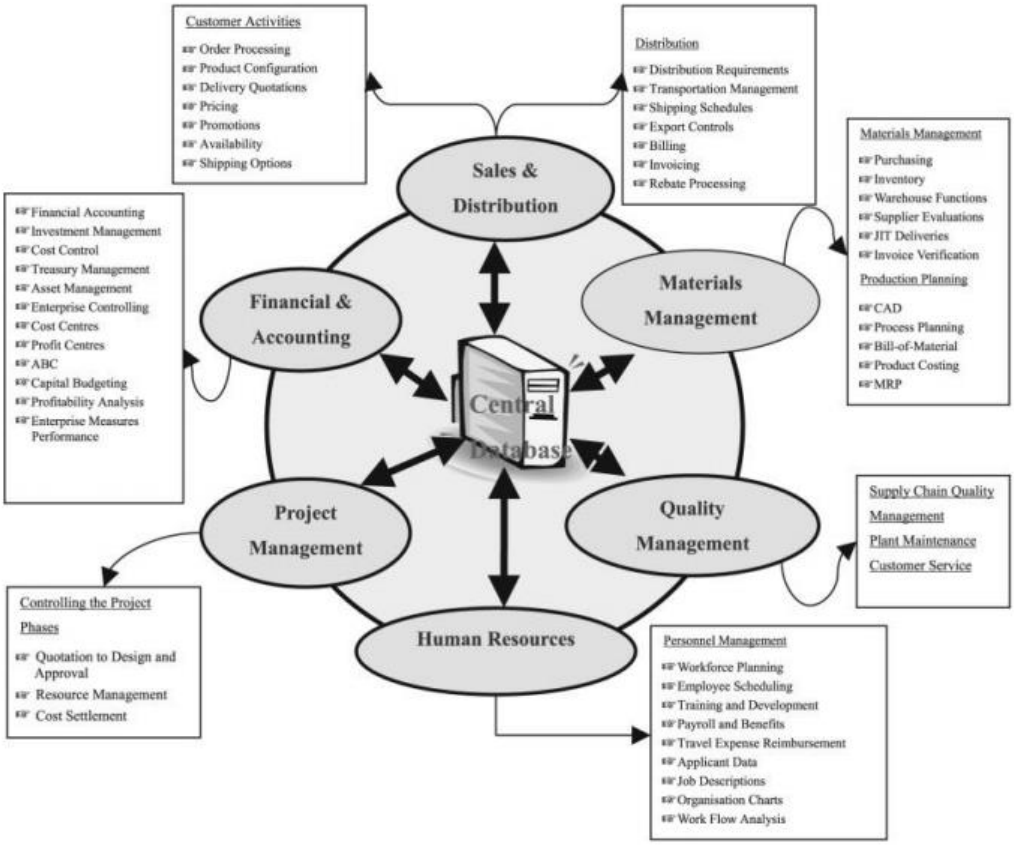



**Figure 32 : ERP system modules (Shehab et al., 2004)**

The ERP origin started with the materials requirement planning (MRP), a system developed in the seventies specialized in the planning and scheduling of materials required for the product manufacturing. In the eighties, MRP II was introduced as a new software system handling shop floor management, project management, human resources and finance (Rashid, Hossain & Patrick, 2001). MRP II was the precursor of the ERP. The real boom of the ERP occurred in the nineties. While MRP II was dealing with internal resources, ERP also strives to take into account supplier resources driven by dynamic client demands. From that point, the entire value chain of an enterprise was encompassed in the ERP. Another change is related to the evolution of the information technology. At the beginning, Information systems worked with mainframe-based computing then, with the internet era, they moved

to client/server architecture. Now, to comply with e-business, ERP vendors are working on browser/web server architecture. (Shehab et al., 2004)

Overtaking an ERP implementation comports some risks that enterprises have to bear in mind. ERP are expensive to implement so enterprises have to balance this with the incremental return they could benefit from it. In every change, there is a risk of failure and can take more time than expected. But these risks are justified by a greater degree of control in real time, a simplification and streamlining of the business process and a higher productivity thanks to a better allocation of the resources and information. (Picardo, 2016)

The principal vendors of ERP systems are SAP, Oracle, Sage, Infor and Microsoft Dynamics. These software enterprises are mainly focused on bigger enterprise mainly because of the drawbacks listed above. Such enterprise can afford large investment of money combined with a risk of failure. But company as Odoo, a Belgian ERP vendor, has targeted this neglected segment (SME's) offering a very competitive price. This was possible with its open source vision. Indeed particular developers are able to enhance the solutions and suggest adaptation to the open source ERP. Moreover the principal costs are R&D related, avoiding marketing and commercial excessive expenses. Odoo relies on the word of mouth through the open source community to make its reputation. This strategy has met the success with 2 million of users around the world and a fifty percent market share among the open source ERP. (Smile open source solutions, n.d.)

### 4.2. Selection of an ERP process with high use of forms

In appendix 5 to 9, we have analysed four business processes interacting with different business functions of an enterprise. These processes are converting lead into sales, invoicing timesheet, new product manufacturing, recruitment process and rental car. These processes span various activities from the customer relationship management to the sales and passing by the manufacturing and human resources management. However for this thesis, we will focus our analysis on one specific case which brings out the Service Interaction Unit. To do so, we need to establish a set of criteria that have to be met. Besides, in regards to the previous sections, the chosen case will be studied with a form-driven look.

The criteria to select a specific case are based on:

- a high use of forms
- a wide range of form style
- a feasible generic process into various ERP softwares.

The last criterion enables us to eliminate the rental car which is not really practicable in standard ERP software. For example Odoo does not support the rental. Then the first criterion excludes the timesheet invoicing and new product manufacturing processes. There are less forms used than converting lead into sales and recruitment process. Finally, the second condition eliminates the lead conversion into sales. Although it uses a lot of forms, it is not using wide variation of forms as in the recruitment process. So the recruitment process meets all the conditions and it will be our study case. We will analyse it more deeply in the next section.

### 4.3. Business Process Modeling Notation

Now that we have our study case, we will decompose it in order to obtain an inside view of the steps required to achieve the recruitment process. This breakdown is made possible with the Business Process Modeling Notation. This notation is a well-known standard in the business analyse. Through the BPMN, we will highlight where forms are used.

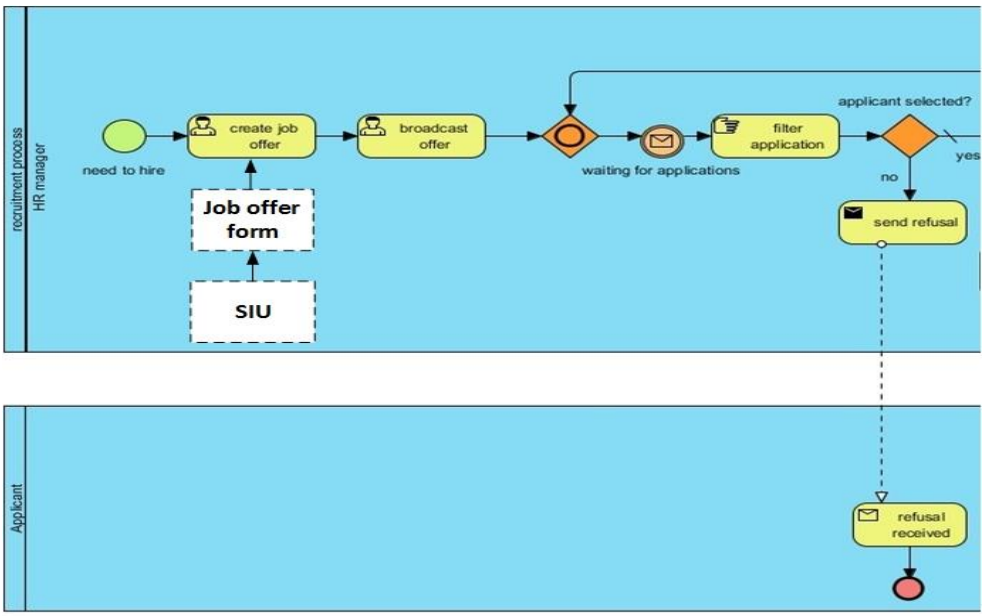



**Figure 33 : First part of the recruitment process BPMN**

Figure 33 shows the first part of this BPMN. The HR manager which has a need to hire new employees starts by creating a job offer and broadcasts it to let know people that the enterprise is looking for new colleagues. Once it is done, he has to wait for applications. Then, he can filter them with its own conditions. For example, this profile has not the diplomas requested. So when there is a non-selection, a gateway indicates that a refusal email is sent to the person. In this first part, a form is used to create the job offer.

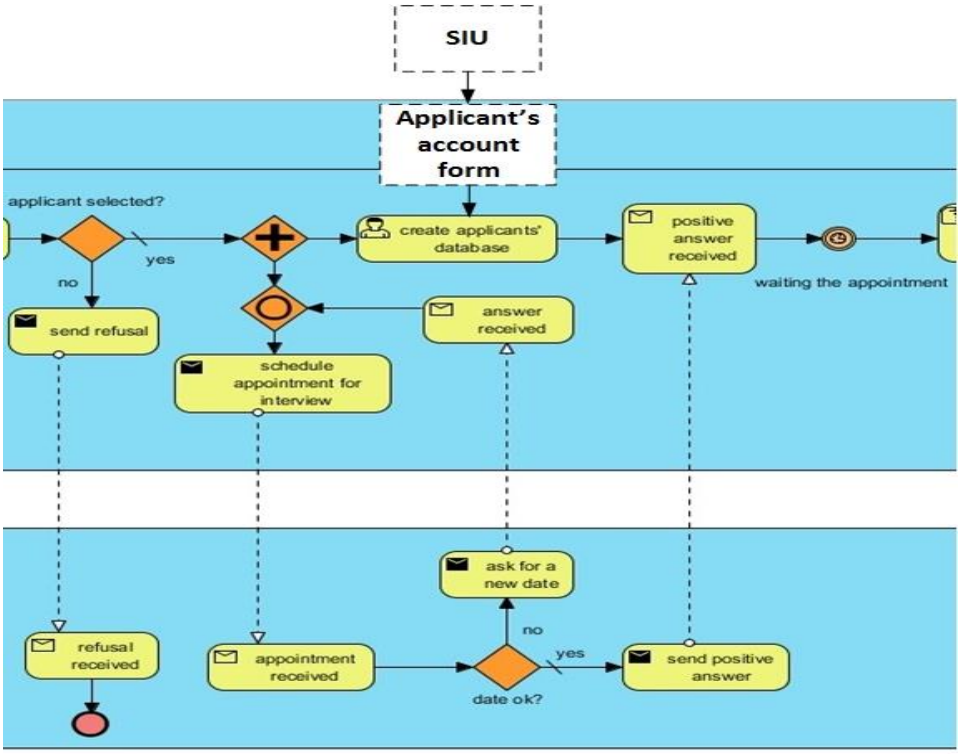



**Figure 34 : Second part of the recruitment process BPMN**

If the candidate is hold, illustrated by figure 34, two actions are done in parallel. The manager sends an email to schedule an interview and he also creates the applicant in a dedicated database. While a schedule is fixed, he has to wait until the date. A new form appears in these steps during the creation of the applicant in a database.

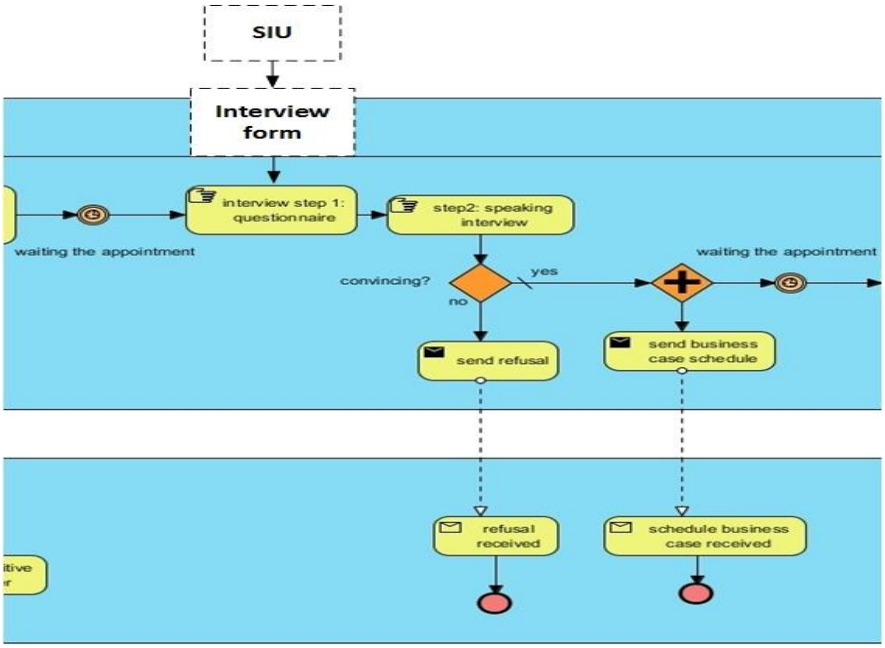



**Figure 35 : Third part of the recruitment process BPMN**

The interview happens in two steps. Beforehand, the candidate must answer to a questionnaire and after the speaking interview occurs. The following stage is the gateway determining if the applicant was convincing or not. In case of a positive feeling, another appointment is fixed to practice a business case. (Figure 35) Here the questionnaire could be created using the forms features formerly described.

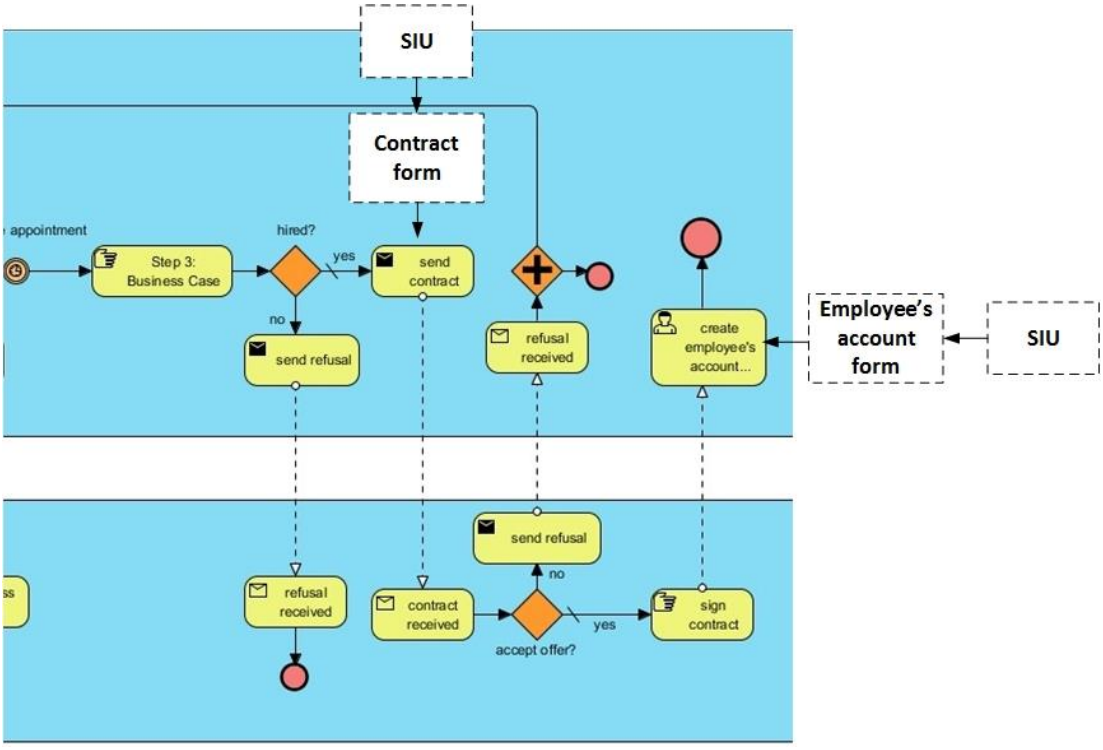



**Figure 36 : Final part of the recruitment process BPMN**

Across figure 34, we will interpret the final part of the recruitment process. This last part begins with the performance of the business case. As soon as manager has taken his decision, he can send a refusal answer or a contract. In case of the candidate refuses the job, the process stops and restarts at the waiting application stage. Otherwise the contract is signed and the manager creates the new employee account in the database. Two additional forms are exploited in the aim to create the contract and the employee's account.

We have perused the entire recruitment process and how forms are implicated in the different stages. To sum up, we have risen five kinds of forms used in five different purposes (create a job offer, create applicant in the database, create a form interview, create a contract and create an employee account). In the following section we will compare how these forms are displayed by different ERP vendors in their recruitment process.

### 4.4. Forms used comparison between different ERP

In this section, the comparison will be organized by each stages identified in the BPMN. At each steps, we will observe how forms are conceived in Odoo, Oracle Taleo and Microsoft Dynamics AX with regards to the dedicated ergonomic guidelines established in section 3.3. The choice of these three ERP has the will to be representative of the ERP industry. On one hand, Odoo is justified by the fact that it is the best open source ERP. On the other hand, Oracle and Microsoft are two of the main licensing ERP.

#### 4.4.1. Comparison

Table 14 illustrates this comparison and assigns a score representing the rules respected out of the number of relevant rules (✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account). For the sake of saving space, we will show only figures for the first form. The rest of the tables and figures are available in appendix 10 to 24. This analysis will enable to observe how ergonomic is respected by the ERP vendors. This will be an opportunity to confront our ergonomic repository and could lead to a modification of our tree. At the end, we will design a form for one of the BPMN stages with the required information using the decision tree as a proposal to correct the ergonomic rules missing.

*Create a job offer*

As we can notice in table 14 Odoo and Oracle obtains a total score of 19/25 and Microsoft Dynamics AX has 19/23. Odoo and Microsoft do not respect the 6 letters rule which implies a right alignment for labels, Oracle complies the label rules by placing labels on the top. Their required inputs are not clearly indicated in Odoo and Dynamics (respectively a thin black line at the bottom field and a red line in the field), instead Oracle uses the well-known asterisk. However they have a compulsory input focus but not Oracle. Prompts are applied in Oracle and Odoo forms but not in the Dynamics form. Help message is triggered by resting pointer on the labels but the timer used by odoo is too long. Dynamics does not provide a process flow, so we don't know the step order even if it seems logical when using it. Oracle does only accept the Anglo-saxon decimal separator. In all cases, ERP does not give any feedback guidance about the action just done, wrong data type entered are not directly blocked. Figure 37, 38 and 39 show the forms used to create a job offer.

| ***Tools Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |
|---|---|---|---|
| *Common criteria* | | | |
| Label | ✕ does not respect the 6 letters rule | ✓ | ✕ does not respect the 6 letters rule |
| Principle of compatibility | ✓ | ✓ | ✓ |
| Required inputs | ✕ not clear, it appears in the validation message | ✓ | ✕ not clear, red line in required fields |
| Minimized effort | ✓ | ✓ | ✓ |
| Data type | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered |
| Default value | ✓ | ✓ | / |
| Prompts (for fields and menus) | ✓ | ✓ | ✕ |
| Help message | ✕ triggered by pointer resting on field but the timer is too long. | ✓timer is good | ✓ |
| Validation message | ✓ | ✓ | ✓ |
| Date/time picker | / | / | ✓ |

| | | | |
|---|---|---|---|
| Value set/range | ✓ | ✓ | ✓ |
| *Specific for fields* | | | |
| Edit mask | ✓ | / | / |
| Compulsory input focus | ✓ | × | ✓ |
| Accept common format | / | × only respect the Anglo-saxon decimal separator. | / |
| Case sensitive | ✓ | ✓ | ✓ |
| Fill-in-the-blanks | / | / | / |
| Autocompletion | ✓ | ✓ | ✓ |
| *Specific for controls* | | | |
| Always editable | ✓ | ✓ | ✓ |
| No Default | ✓ | ✓ | / |
| Checkbox or toggle | / | / | ✓ |
| Non-yet option | / | / | / |
| Score | 12/16 | 12/16 | 11/14 |
| ***Presentation Criteria*** | ***Odoo*** | ***Oracle*** | ***Microsoft Dynamics AX*** |
| Display criteria | | | |
| Conversational flow | ✓ | ✓ | ✓ |
| Group with related information | ✓ | ✓ | ✓ |
| Process flow indicated | ✓ | ✓ | × does not appear clearly on the page |
| Guidance feedback | × no guidance action done | × no guidance action done | × no guidance action done |
| Consistency | ✓ | ✓ | ✓ |
| Explicit controls | ✓ | ✓ | ✓ |
| Relationship criteria | | | |
| Avoiding loop | ✓ | ✓ | ✓ |
| No redundant request | ✓ | ✓ | ✓ |
| Prefilling | / | / | / |

| | | | |
|---|---|---|---|
| Score | 7/9 | 7/9 | 8/9 |
| **Total Score** (respected rules out of the relevant rules ) | **19/25** | **19/25** | **19/23** |
| Legend: ✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account | | | |

**Table 14 : Comparison for the job offer form between Odoo, Oracle and Microsoft**

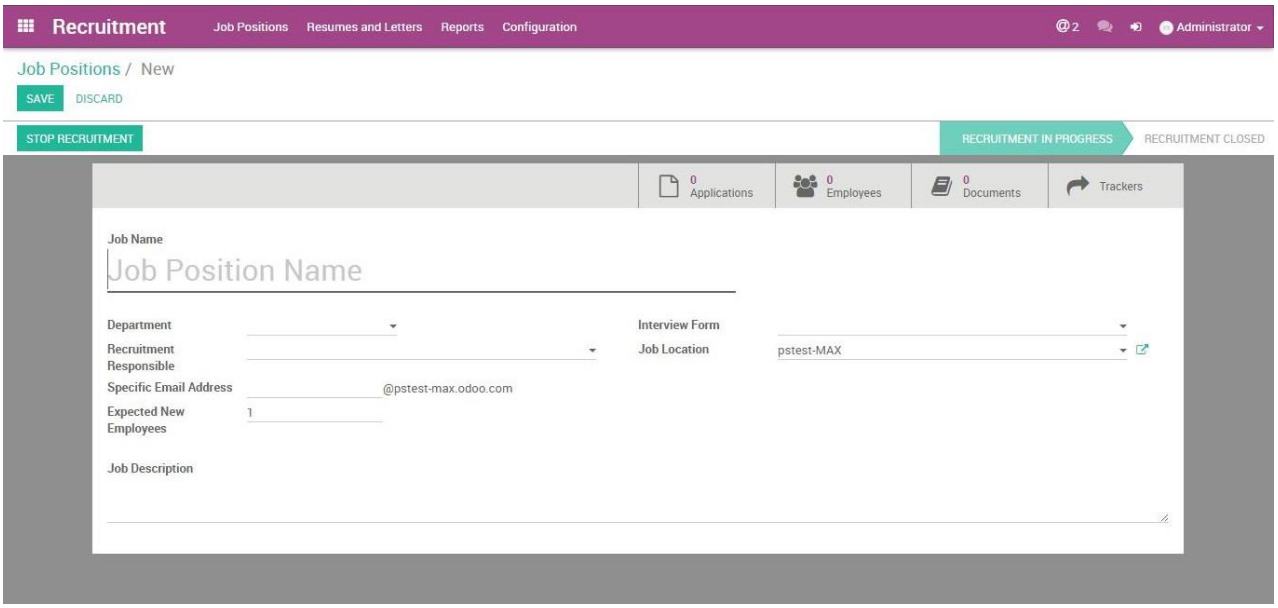



**Figure 37 : Job offer form in Odoo**

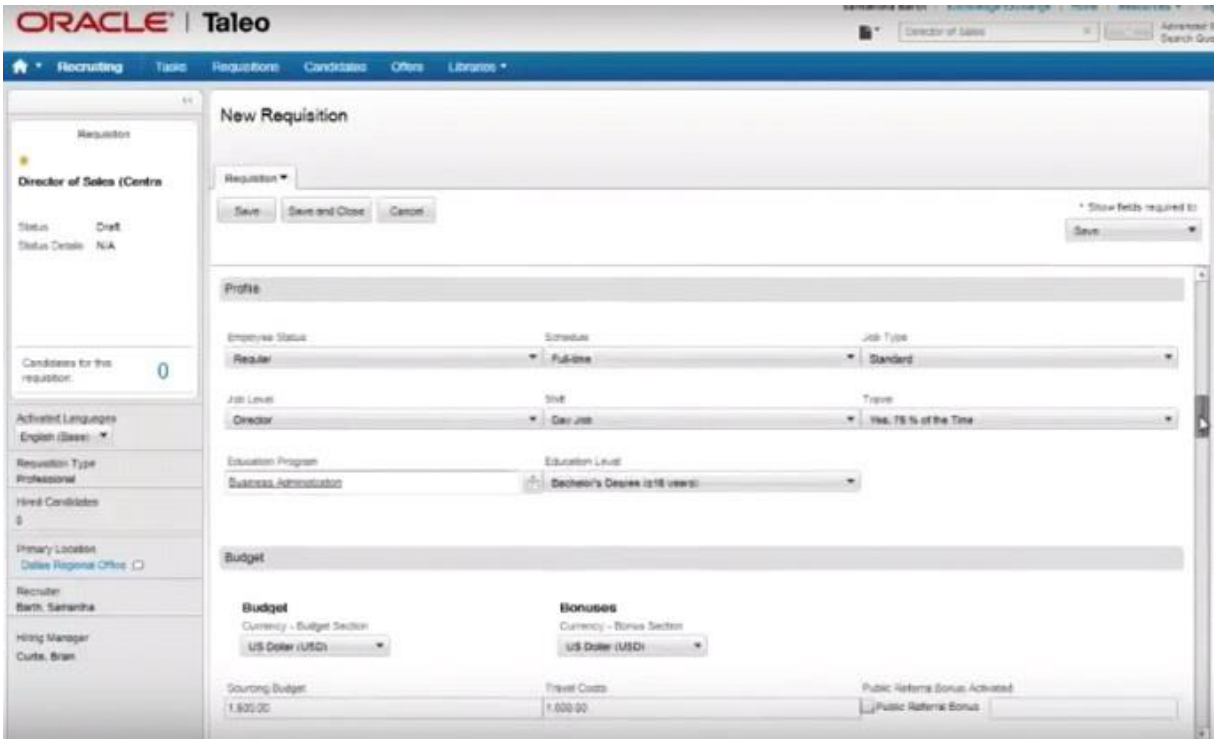



**Figure 38 : Job offer form in Oracle Taleo**

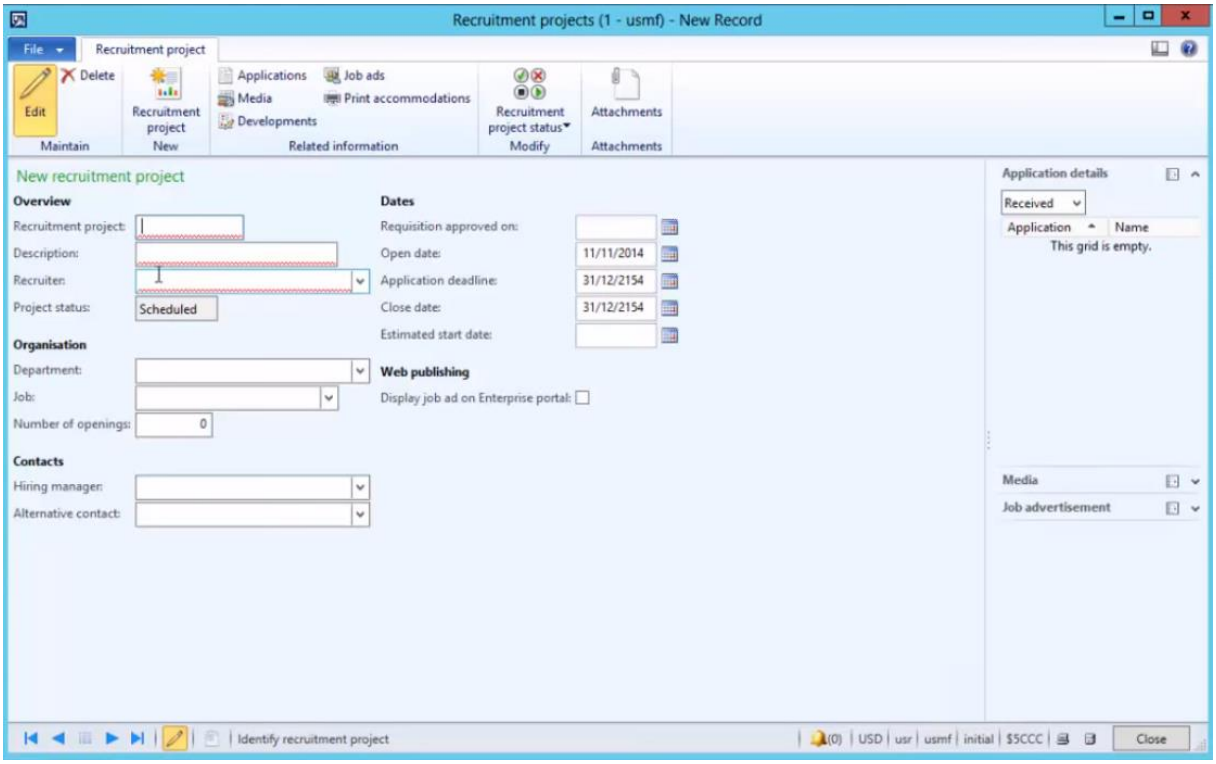



**Figure 39 : Job offer form in Microsoft Dynamics AX**

*Create applicant in the database (Appendix 10 to 13)*

As we notice in appendix 10, Odoo has a score of 20/26, Oracle 21/26 and Dynamics 21/27. Contrarily with the previous form, Odoo respects the guidelines related to the label but Odoo does only accept Anglo-Saxon decimal separator. The other errors registered in the previous form remain unchanged. On its part Oracle uses menus instead of date/time picker, places radio button for a set of value bigger than five. It has a redundant request about the user's name for security reasons but there is no explanation about that. So we consider it as a blooper. Nonetheless, at the end of the process, the user receives a feedback message to confirm when process is finished which is not the case with Odoo and Dynamics. Oracle and Dynamics prefill some fields logically or with the information they receive (e-mail for Oracle and number of applications for Dynamics). A main difference in Oracle is that the user registers himself on the enterprise website and applies on the job offers.

*Create a form interview (Appendix 14 to 16)*

With both Oracle and Odoo, you are able to create your questionnaire or to use one they give by default. Both ERP's has a score of 16/21. In the Odoo default questionnaire, they do not use slider for a likert scale but radio buttons instead. Another divergence is they do not respect the value set rule. They use radio buttons for more than 5 possible choices (9

actually) and they use menu for 2 choices. In this case Odoo indicates clearly what is mandatory with the help of asterisks. Oracle and Odoo do not provide prompt, compulsory input focus and data type control. There are no help messages available but questions seem really clear. Microsoft Dynamics does not provide a default interview form but a template that the user can fulfil such as with Infopath. So this last ERP is not taken into account for this comparison.

*Create a contract (appendix 17 to 20)*

As resumed in appendix 17, Odoo and Dynamics do not have clear required inputs. The help message in Odoo is long to appear when resting the pointer on the label. Nevertheless, there is a good prefilling. Contrarily to Dynamics, Oracle also provides a prefilling but this choice is let to the user by clicking on a button. Odoo and Oracle comply the label guidelines but as before Dynamics not. Besides, Dynamics does not place any prompts and the process flow is still missing. Oracle for its part uses menu instead of date/time picker and compulsory input focus is absent because form start with a menu. Moreover the default value is not constant. For instance, there is the US Dollars for the currency but the Salary field is empty in place of 0.0 like in Odoo and Dynamics which respect the default value. The three ERP's allow only Anglo-Saxon decimal separator (accept common format), wrong data-types entered are not directly blocked and there is no feedback guidance once action is done. The scores are 20/25 for Odoo, 18/24 for Oracle and 18/26 for Dynamics

*Create employee's account (appendix 21 to 24)*

In this form, Odoo and Dynamics do not respect the right alignment for labels (6 letters rule). in Odoo, the required inputs are still not clearly indicated and required fields are already prefilled in Dynamics. The Odoo data type control is not operational for all fields e.g. the work mobile. In Oracle and Dynamics it is still not blocking when wrong data is entered. They also do not use prompts but Odoo does. Like the previous form, Odoo timing to trigger help message is too long. For date/time, Oracle applies menu. There is no edit mask (e.g. working email) in Oracle and Odoo. Besides Oracle do not execute the compulsory input focus. The three ERP's allows only the Anglo-Saxon decimal separator and in the addition for Odoo of no possibility for decimal number in the home-work distance. All of them do not show any

feedback guidance. Finally, Dynamics does not point the process flow out. Odoo and Oracle reach a score of 19/26 and Dynamics 19/25.

### 4.4.2. Analysis

With the above comparison, we are able to analyse the ergonomic and the Service Interaction Unit occurring through the ERP forms. We have noticed at each step using forms what are the non-respected rules. Here, we will provide some suggestion to correct these errors. Nonetheless, several guidelines have always been approved and it means they are common knowledge in the form establishment. To precise, some of them are always followed when there come into play. These guidelines are:

- Principle of compatibility
- Minimized effort
- Validation message
- Case sensitive
- Auto completion (when it is appropriate)
- Always editable
- No default (when it is appropriate)
- Checkbox or toggle (when it is appropriate)
- Non-yet option (when it is appropriate)
- Conversational flow
- Group with related information
- Consistency
- Explicit control
- Avoiding loop
- Redundant request

The recommendations for observed rules that were not respected are:

- When the longest label is six characters longer than the smallest label, alignment should be to the right. Oracle places labels on the top to faster the reading and

sometimes on the right. In both cases it respects the label guideline but for consistency we suggest to choose just one option.

- Required inputs in Odoo and Dynamics are not really clear as a result user can easily skip it and will be warned in the validation message. Obviously by dint of use, he will recognise what is required but it is not user friendly. So we recommend the well-proved asterisk.
- Data-type could be more used when arguments ask only one type of input it is easy to configure the argument to reject directly the wrong input entered (date/ time, phone number). Prompts and edit mask can prevent to wait the validation message.
- Default value helps the user to faster and to diminish its effort. But it has to be well considered. In the case of Oracle form, Salary field needs to have the 0.00 value because it serves also as a prompt.
- Prompt is helpful to indicate the action or the input that is asked and works as a first control by giving an example.
- Help messages are applied but Odoo should reduce the timer because user can easily miss it.
- Date/time picker are always encountered in Odoo and Dynamics but Oracle puts menus. Menus take more space because it needs a menu for the days, months and years so it slows the reading. Moreover user spends more time to fill than using a date/time picker.
- Value set range is not in application in the creation of the applicant account of Oracle and in the interview form of Odoo. They use radio button for more than 5 choices. We suggest replacing them by menus.
- Edit mask is really important to avoid mistakes. So for the email field in Oracle and Odoo employee account form it could be interesting to add one. E.g. <field>@<field>
- Compulsory input focus should always be executed except when the first input is an evaluation or a selection (cf. section 3.4.3)
- The sole case where common format where not respected was for the numbers. All of the ERP's work with the Anglo-Saxon decimal separator. This problem seems unsolvable due a technical setting. However we suggest adapting the used format in function of the user's origin. For example based on its nationality.

- Dynamics never displays the process flow yet it is quite helpful for the user to know what has been done and which steps need to be still performed. A suggestion is to add the flow on the each page of forms like breadcrumbs (Odoo) or carousel (Oracle).
- Guidance feedback is absent for the three ERP's except for some forms in Oracle in (applicant account and interview form) and in Odoo (interview form). Even so user needs to know when a specific action is done. Obviously having dialog windows with click request appearing at each action end is troublesome. But it can be without click request and coming into sight for a few seconds.
- Prefilling should be executed as soon as it is possible. Dynamics does not allow this for the contract creation form and could simply use the data entered in the applicant account to prefill the municipality of residence field...

We observe Fill-in-the-blanks guideline is never applied. It appears too difficult and not efficient to put into practice. Consequently we can remove it from our guidelines grid. The Odoo interview form advocates the usage of radio buttons for a likert scale instead of a slider. We could consider it in our model precisely in this case because it is conformed to the 5 values set rule of the radio button. However some studies (Dobronte, 2012 ; Cape, 2009) argue for the use of sliders in place of radio button. It is said that scales create a more pleasant experience for survey responders and data quality is better with slider. Besides they involve more engagement from the responders. Considering these facts, we will keep using the sliders and we will not modify our model.

### 4.5. Specific form suggestion using the decision tree

Section 4.4 enables to notice the ergonomic and SIU parameters that are not respected in fifteen forms among three ERP softwares. Besides suggestions related to these lacks are proposed in order to improve forms. But to better highlight the possible enhancements, it could be interesting to apply our decision tree providing an example of what could be a reference form. Scores are given to each form, representing the respected rules out of the relevant rules that apply on the form. We will choose a specific stage of the ERP as example to show how this form can be designed with the decision tree. We base this selection on two criteria. First each ERP must have an analysed form for this stage and secondly the stage getting the lowest weighted average of the three scores will be considered. The first

criterion eliminates the interview form as there is no analysed form for Dynamics. The weighted averages with a weight of a third for each form are the following:

- Create a job offer: [(19/25)+(19/25)+(19/23)]/3= 78,2 %
- Create an applicant in the database:[((20/26)+(21/26)+(21/27))/3]/3= 78,49%
- Create a contract: [(20/25)+(18/24)+(18/26)]/3= 74,74%
- Create an employee account:[(19/26)+(19/26)+(19/25)]/3= 74,05%

According to the criteria, the selected case is the form for the employee's account creation.

### 4.5.1. Required information

Before interpreting the decision tree we have to collect all the necessary information that has to appear on the employee's account form. To do so, we list below the principal information collected through the forms used in the three ERP. This information is organized in four groups arranged in subgroups and they are ordered to comply with the conversational flow and related information grouped guidelines. The information is given as example and the reader is free to complete them.

*Personal Information*

Subgroup 1 is the Identification which contains personal title, first name, middle name, last name, birth date, birth place, nationality, native language, national registration number and personnel identification. Status is the subgroup 2 is composed of gender, marital status and number of children. Then subgroup 3 (private address) corresponds to street1, street2, zip code, city and country. Finally subgroup 4 is private contact and gathers phone, mobile and email

*Public Information*

The first subgroup is Working Address with street1, street2, zip code, city and country. Then we have the second subgroup called professional contact assembling working mobile, working phone and working email. Lastly Position subgroup contains department, job title, manager and the working time.

*Contract*

There are two subgroups, contract information and payroll. Contract information groups the contract type, start date, end date, advantages, remarks, insurance and company vehicle. If

there is an insurance we have two more pieces of information: the insurance policy number and start date/ end date. Also when employee enjoys a vehicle, model car and number plate are added. Payroll subgroup gathers the payroll category, monthly wage, currency, start date/ end date, income tax code, bonuses as option (if yes it includes the amount and the condition) and the employee bank account.

*Competencies*

Education subgroup needs the education level, program and institution. The next subgroup is the certification which asks for certificate name, subject, issuing institution and remarks. Finally the last subgroup is the work experience that incorporates the sector, the company, department, job and some remarks.

### 4.5.2. Form design

The next step to sketch our form is to apply the decision tree (see section 3.4.4) on each piece of information to determine the desired widget. We will first begin with the organisation of our form. Then we will assign the adapted widget in function of the input. And finally we will determine the relationships.

*Form organisation*

We want to organise the form with multiple groups that are displayed in multiple forms. At this point we face three possibilities (succession of forms, navigational forms and collapsible forms). As we have several subgroups composed of numerous pieces of information, it seems more appropriate to navigate easily between the forms and to provide a clear view of each one of them. The succession of forms prevents to have a simple navigation and collapsible forms will be too fussy to support all the details. So the employee account will be designed with the tabbed dialog box. Then we have to decide how subgroups will be placed in each form. Following the decision tree, the subgroups related to the form Personal Information (identification, status, private address and contact) can have a strong, medium or weak distinction. This choice depends on the user's interpretation but a way to conceive the separation levels is like a Russian doll (groups nested in groups). In our case, we do not have any other groups embedded in the subgroups so a weak distinction (space separator) will be sufficient. This scheme is the same for the three other forms (public information, contract and competencies).

*Form inputs*

As it is the same process, we will only explain the path to reach widgets for one example of each arguments cluster occurring in the form. In the subgroup Identification we want to represent the personal title. This information can be an entry or a selection. Using entry for this information can lead to write the same title in different way. For example, Mr or MR or Mr. or MR. or Mister... Preventing this lack of consistency, It is better to choose a selection. We will also apply the no default parameter as we do not know the title in advance. The number of titles can exceed five and we want to optimize the space in the form by reducing the choices. The dropdown menu appears to be best solution. The first name corresponds to an entry with a unique data type and text input resulting in a text field. The national registration number follows the same path but ends in a masked number field. The field can have a format like <XX>.<XX>.<XX> - <XXX>.<XX>. Then we have the birth date which is a selection with more than five choices and related to date. So the widget is a date/time picker. Finally the last cluster of arguments presents in the employee account form is a binary selection. The insurance will be the example. We need to decide between an immediate change (toggle) or a step in the change (checkbox). Here, we don't want to directly operate the change because this information will be completed by insurance policy number and its start/end date.

*Form relationship*

The relationship here concerns the fact that an event sets off the visibility of arguments. Here the insurance triggered by the On state from checkbox will make visible the arguments insurance policy number and its start/end date. Another relationship is the filter applied on the manager argument. Managers will be filtered by department and the filter argument is a dropdown menu (arbitrarily chosen).

Appendix 25, 26, 27 and 28 show the suggested form for the employee account creation. This form is a mockup enabling the implementation in different programming language. In this form, we have corrected the lack of guidelines noticed in appendix 21 with the help of section 4.4.2 and the decision tree. This suggestion could be enriched by possible additional arguments but the main purpose was to enhance the application of the decision tree in a concrete case. Through this chapter we have analysed the recruitment business process and

its related forms in three ERP. Then we have provided solutions to fix these errors. At the end, we have proposed a form mockup for the creation of an employee account thanks to the decision tree.

# 5. Conclusion and further works

## 5.1. Conclusion

It is important to remind that the goal of this master thesis was to suggest a model that could help the user to implement the Service Interaction Unit in the user interface for a cross-platform context. To achieve this purpose, this work has been broken down into several milestones attempting an answer to the question: Applied on the case of ERP systems in a cross-platform context, how to suggest a model to implement the Service Interaction Unit? Initially, we will briefly recall these different milestones. After we will see what are the contributions of this model in regards to the question and at the end we will analyse this contribution with respect to the weaknesses observed in the literature at section 3.1.

The first step was a review of the existing literature in order to take stock of the current notion of the user interface design patterns. Through this review, we have noticed a series of 9 weaknesses and the lack of documentation about the Service Interaction Unit. As these weaknesses are quite related to the concept of cross-platform pattern, they enabled to determine a proper response of what is a cross-platform pattern.

Then based on this assessment, we have tackled how to apply a cross-platform pattern. To do so, we have embraced the decision tree methodology that combines cross-platform pattern features and ergonomic criteria. First of all, the application of this methodology required to describe the SIU with the UIPLML which is a cross platform description template. In a second time, the SIU has been expanded across the abstraction levels of the Cameleon Reference Framework. On one hand we have represented the SIU in the task and domain model. And on the other hand, we have conducted a deep analysis of forms used to interact with the system considering relevant ergonomic rules and the specificities of the SIU. This analysis has served to establish the Abstract User Interface of the SIU through a decision

tree. Each leaves of this tree lead to a mockup in the Concrete User Interface and subsequently to a Final User Interface (here HTML and Android).

Finally, we have applied the SIU model to a study case in the ERP context. We have selected the recruitment process among five common business activities. The recruitment process has been explored back and forth with the help of the business process modelling notation in order to identify interactions via forms. Based on an ergonomic criteria grid, a comparison of these forms used in three ERP software's has been carried out. Afterwards, we have depicted counterparts to the errors noted in the comparison. Lastly, we have suggested a form using the SIU decision tree as a correction for the in use employee account forms.

The proposed model offers several advantages which are explained in the following points.

- To begin, this thesis contributes to document the Service Interaction Unit which is poorly studied in the literature despite the fact that is frequently used. Besides this documentation is not restricted to a simple description of the pattern but to a complete implementation process.
- Then, this model expresses the SIU in a high level of abstraction allowing a use in different contexts and platforms. Indeed from the mockup in the CUI which is platform independent, several final user interfaces can be implemented depending on the platform.
- Another advantage is the fact that SIU is enriched by the ergonomic guidelines which is a lack in the current patterns. This incorporation is really helpful to address correctly an implementation.
- Next, the asset of this decision tree is articulated on three points. First it guides the user all along the conception process of the user interface. Secondly decisions are not limited to binary choices (yes or no) but are enriched by various possibilities. Third, the result is illustrated in a mockup and subsequently implemented in a specific final user interface providing a good idea of the result.
- Afterwards, forms are the most commonly used way for a system to get input from a user. As this model structures and organizes forms it could have a real application and can be put in practice in various domains.

- Last but not least, this SIU model respects the conceptual pattern principle of extensibility. New features can be added in the tree refining its user interface representations.
- Finally the study case reveals that SIU pattern can be directly included in the BPMN enabling a centralisation of the information and a global view of what is at stake. This can facilitate the work of the developers. Existing software's such as Bonitasoft offers to make this link.

All these facts reveal a model that could support the implementation of the SIU. But we will see how this model is critical in regards to the noticed weaknesses of section 3.1. In table 15, we will evaluate if the SIU model solves the weaknesses. Possible answers can be totally, partially or not at all.

| Weaknesses | Answers |
|---|---|
| Lack of consistency (W1) | Totally (UIPLML description format) |
| Lack of structure (W2) | Partially (can be included in the UIPLML application) |
| Lack of implementation information (W3) | Totally |
| Lack of pattern evaluation (W4) | Not at all (SIU has not been tested) |
| Limitation of contextual coverage (W5) | Partially ( only represented in HTML and Android) |
| Lack of abstraction(W6) | Totally |
| Lack of ergonomic approach (W7) | Totally |
| Limited usability consideration (W8) | Not at all (no survey to know if the SIU is user friendly) |
| Lack of expressivity (W9) | Totally (swot analysis) |

**Table 15 : Model weaknesses reviewing**

Our model does not answer to W4 and W8 because it has been tested by users. This a lack and that is why this SIU model is a suggestion. W2 and W5 are partially solved. To fully comply with W2 it should include the SIU in the UIPLML application but it has been released too recently (june 2016). In the case of W5, we have illustrated the CUI in HTML and Android but there are other possibilities to express the CUI. However these two FIU's target two different screen sizes which are representative of the most frequently used devices (smartphones and computer).

## 5.2. Further Works

The last part of the conclusion gives us some tracks for further works. But we can prioritize them in short term and long term. It seems that further works in the short term should be the achievement of the weaknesses that are not totally addressed by the model. A survey is critical in the prioritization. The result would enable to validate the model or to adapt it. Another one is to continue the FUI representation in alternative operating systems such as Windows and IOS. As this thesis is in the scope of the Louvain School of Management, the programming code part has not been undertaken and could complete these FUI illustration. It could be also interesting to incorporate the SIU in the UIPLML application. This would link the SIU to other patterns within a rich repository.

A further works in the long term would be to make the decision tree interactive with a possibility for users to add new features. Then the incorporation in software that implements business process into application would benefit the collaboration between the service interaction unit and the business process modelling.

These propositions of further works aim at improve this model and to facilitate the use of the Service Interaction Unit. Besides they also could implicate more strenuously the user in the SIU development.

## 6. Reference

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language.* New York: Oxford University Press.

Android Developers. (n.d.). *Introduction to Android*. Retrieved July 27, 2016, from https://developer.android.com/guide/index.html

Aquino, N., Vanderdonckt, J., Panach, J. I., & Pastor, O. (2011). Conceptual Modelling of Interaction. In D. W. Embley & B. Thalheim (Eds.), *Handbook of Conceptual Modeling : Theory, Practice, and Research Challenges* (pp. 335-358). Berlin: Springer.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (2001*). Pattern oriented software architecture: a system of pattern.* Chichester: Wiley.

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, 15*(3), 289-308.

Cape, P. J. (2009). *Slider scales in online surveys*. Retrieved July 09, 2016, from http://www.websm.org/uploadi/editor/doc/1436872048Cape_2009_Slider_Scales_in_Online_Surveys.pdf

Coad, P. (1992). Object-oriented patterns. *Communications of the ACM Commun. ACM, 35*(9), pp. 152-159.

Dobronte, A. (2012). *Likert Scales vs. Slider Scales in commercial market research*. Retrieved July 08, 2016, from https://www.checkmarket.com/blog/likert-scales-slider-scales/

Elouedi, Z., Mellouli, K., & Smets, P. (2000). *Decision trees using the belief function theory*. Paper presented at The proceedings of the international conference on Information Processing and Management of Uncertainty IPMU 2000, Madrid. doi=10.1.1.37.756

Engel, J., Herdin, C., & Maertin, C. (2012). Exploiting HCI pattern collections for user interface generation. In A. Zimmermann & P. Lorenz (Eds.), *Patterns 2012*. Paper presented at the 4th International Conferences of Pervasive Patterns and Applications, Nice, France (pp. 36-44). IARIA.

Engel, J., Märtin, C., & Forbrig, P. (2015). A Concerted Model-driven and Pattern-based Framework for Developing User Interfaces of Interactive Ubiquitous Applications. In R. Seiger, B. Altakrouri, A. Schrader, & T. Schlegel (Eds.), *Workshop on Large-scale and model-based Interactive Systems: Approaches and Challenges.*Paper presented at Proceedings of the 1st International Workshop on LMIS 2015, Duisburg (pp. 35-41). CEUR-WS.org .

Envatomarket. (2014). *Responsive Login, Sign Up and Payment Form Wizard*. Retrieved June 13, 2016, from https://codecanyon.net/item/responsive-login-sign-up-and-payment-form-wizard/8020781

Ergolab. (n.d.). *L'ergonomie dans la conception d'un formulaire*. Retrieved May 26, 2016, from http://www.ergolab.net/articles/ergonomie-formulaire.php

Fox, S., & Rainie, L. (2014). *How the internet has woven itself into American life*. Retrieved May 11, 2016, from http://www.pewinternet.org/2014/02/27/part-1-how-the-internet-has-woven-itself-into-american-life/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Pearson Education.

Google. (n.d.). *Accessibility - Usability - Google design guidelines*. Retrieved July 27, 2016, from https://material.google.com/usability/accessibility.html

Investopedia. (2005). *Enterprise Resource Planning (ERP) Definition*. Retrieved July 31, 2016, from http://www.investopedia.com/terms/e/erp.asp

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*. Boston: Addison-Wesley.

Johnson, J. (2003). *Web bloopers: 60 common web design mistakes, and how to avoid them*. Amsterdam: Morgan Kaufmann.

Kruschitz, C., & Hitz, M. (2010). Human-Computer Interaction Design Patterns: Structure, Methods, and Tools. *International Journal on Advances in Software, 3* (1 & 2), 225-236.

Lenhart, A. (2009). *Teens and Mobile Phones Over the Past Five Years: Pew Internet Looks Back*. Retrieved May 11, 2016, from http://www.pewinternet.org/2009/08/19/teens-and-mobile-phones-over-the-past-five-years-pew-internet-looks-back/

Likert, R. (1932). *A technique for the measurement of attitudes*. New York: The Science Press.

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., & Trevisan, D. (2004, May). Usixml: A user interface description language for context-sensitive user interfaces. In K. Luyten, M. Abrams, Q. Limbourg, J. Vanderdonckt (Eds.), *Developing User Interfaces with XML: Advances on User Interface Description Languages.* Paper presented at the Proceedings of the ACM AVI'2004 Workshop, Gallipoli (pp. 55-62). New-York: ACM

Mayhew, D. J. (1992). *Principles and guidelines in software user interface design*. Englewood Cliffs, NJ: Prentice Hall.

Microsoft. (n.d.). *Conception et interface utilisateur*. Retrieved July 17, 2016, from https://developer.microsoft.com/fr-fr/windows/design

Misfud, J. (2011 a). *Official Usability & Web Site Guidelines of Governments From Around the World - Usability Geek*. Retrieved July 17, 2016, from http://usabilitygeek.com/official-usability-web-site-guidelines-of-governments-from-around-the-world/

Misfud, J. (2011 b). *An Extensive Guide To Web Form Usability – Smashing Magazine*. Retrieved June 27, 2016, from https://www.smashingmagazine.com/2011/11/extensive-guide-web-form-usability/

Molina, P. J., Melia, & S., Pastor, O. (2004). The just-ui approach : conceptual modelling of device Independent user interfaces, *Journal of Human-Machine Interaction*, 5(1), 12-39.

Molina, P. J., Pastor, O., Marti, S., Fons, J. J., & Insfram, E. (2001). Specifying conceptual interface patterns in an object-oriented method with automatic code generation. In E. Kapetanios & H. Hinterberger (Eds.), *Proceedings Second International Workshop on User Interfaces in Data Intensive System, UIDIS 2001.* Paper presented at the User Interfaces to Data Intensive Systems, 2001, (UIDIS 2001). Second International Workshop on, Zurich (pp. 72-79). Washington, IEEE Computer Society.

Nguyen, T. D., & Vanderdonckt, J., (2015), Generative patterns for cross-platform user interfaces: The case of the master-detail pattern. In A. Seffah (Ed.), *Patterns of HCI Design and HCI Design of Patterns* (pp. 123-153). Berlin: Springer.

Nguyen, T. D., Vanderdonckt, J., & Seffah, A. (2016 a). Generative Patterns for Designing Multiple User Interfaces. In A. Abadi, L. Flynn, P. Inverardi, & J. Gray (Eds.), *Proceedings of 3rd IEEE/ACM International Conference on Mobile Software Engineering and Systems MobileSoft'2016.* Paper presented at *3rd IEEE/ACM International Conference on Mobile Software Engineering and Systems MobileSoft'2016*, Austin (pp. 151-160). New-York: ACM Press.

Nguyen, T. D., Vanderdonckt, J., & Seffah, A. (2016 b). UIPLML: Pattern-based Engineering of User Interfaces of Multi-Platform Systems. In S. España, J. Ralyté & C. Souveyet (Eds.), *Proceedings of IEEE Tenth International Conference on Research Challenges in Information Science*. Paper presented at the IEEE Tenth International Conference on Research Challenges in Information Science (RCIS'2016), Grenoble, France (pp. 273-284). Piscataway, NJ: IEEE Press.

Nielsen, J. (1995). *10 Heuristics for User Interface Design*. Retrieved May 05, 2016, from https://www.nngroup.com/articles/ten-usability-heuristics/

Object Management Group. (2016). *MDA - The Architecture of Choice for a Changing World*. Retrieved July 19, 2016, from http://www.omg.org/mda/

Pastor, O., & Molina, J. C. (2007). *Model-driven architecture in practice: A software production environment based on conceptual modeling*. Berlin: Springer.

Pastor, O., Gómez, J., Insfrán, E., & Pelechano, V. (2001). The OO-method approach for information systems modeling: From object-oriented conceptual modeling to automated programming. *Information Systems, 26*(7), 507-534.

Pastor, O., Insfran, E., & Pelechano, V. (2000). *Conceptual user interface patterns for object-oriented conceptual models*. Paper presented at the East-European Conference on Advances in Databases and Information Systems (ABDIS'00), Prague, Czech Republic.

Paterno, F. (1999). *Model-based design and evaluation of interactive applications*. Berlin: Springer.

Paternò, F., Santoro, C., & Spano, L. D. (2012). *Concur Task Trees (CTT)*. Retrieved July 29, 2016, from https://www.w3.org/2012/02/ctt/

Picardo, E. (2016). *Enterprise Resource Planning (SAP, ORCL*), Investopedia. Retrieved July 31, 2016, from http://www.investopedia.com/articles/investing/032416/enterprise-resource-planning-sap-orcl.asp?ad=dirN

Rashid, M. A., Hossain, L., & Patrick, J. D. (2001). The Evolution of ERP Systems: A Historical Perspective. In Rashid, M. A., Hossain, L., & Patrick, J. D. (Eds.), *Enterprise Resource Planning: Global Opportunities and Challenges: Global Opportunities and Challenges* (pp. 1-16). Hershey, PA: Idea Group Publishing.

Scapin, D. L., & Bastien, J. M. (1997). Ergonomic criteria for evaluating the ergonomic quality of interactive systems. *Behaviour & Information Technology, 16*(4-5), 220-231.

Seffah, A. (2015). From HCI Patterns Languages to Pattern- Oriented Design. *Human–Computer Interaction Series Patterns of HCI Design and HCI Design of Patterns,* 15-33.

Shehab, E.M., Sharp, M.W., Supramaniam, L., & Spedding, T.A. (2004). Enterprise resource planning, *Business Process Management Journal*, 10 (4), 359 – 386.

Shneiderman, B. (1997). *Designing the user interface: strategies for effective human-computer interaction (3rd)*. Boston: Addison-Wesley Longman Publishing Co., Inc.

Smile open source solutions. (n.d.). *Présentation de Odoo*. Retrieved July 31, 2016, from http://information-systems.smile.eu/Tout-savoir-sur/Erp-pour-l-ecommerce/Presentation-de-odoo

Sun Microsystems Inc. (2001). *Java Look and Feel Design Guidelines, 2nd Edition*. Retrieved July 17, 2016, from http://www.oracle.com/technetwork/java/higtitle-140701.html

Talkroute. (n.d.). *How to Change the Call Menu Audio Prompt (Press #1 for... Press #2 for...)*. Retrieved August 13, 2016, from https://support.talkroute.com/hc/en-us/articles/213859048-How-to-Change-the-Call-Menu-Audio-Prompt-Press-1-for-Press-2-for-

Tešanovic, A. (2005). *What is a pattern?* Retrieved june 25, 2016, from http://st.inf.tu-dresden.de/Lehre/dpf/IntroductoryPapers/tesanovic-WhatIsAPattern.pdf

Tidwell, J. (2010). *Designing interfaces (2nd ed.).* Sebastopol, Canada: O'Reilly Media.

Toxboe, A. (n.d.). *Design patterns*. Retrieved May 26, 2016, from http://ui-patterns.com/patterns

U.S. Department of Health & Human Services. (n.d.). *Usability Guidelines: These guidelines are research based and are intended to provide best practices over a broad range of web design and digital communications issues*. Retrieved July 27, 2016, from http://webstandards.hhs.gov/guidelines

UX Movement. (2013). *Stop misusing select menus*. Retrieved July 27, 2016, from http://uxmovement.com/forms/stop-misusing-select-menus

van Welie, M. (2008). *A Pattern Library for Interaction Design*. Retrieved june 16, 2016, from http://www.welie.com/

van Welie, M., & Van der Veer, G. C. (2003). Pattern languages in interaction design: Structure and organization. In M. Rauterberg, M. Menozzi & J. Wesson (Eds.), *Human Computer Interaction-interact'03*. Paper presented at the IFIP TC13 International Conference on Human-Computer Interaction, Zurich, Switzerland (pp. 527-534). Amsterdam: IOS Press.

van Welie, M., Van Der Veer, G. C., & Eliëns, A. (2001). Patterns as tools for user interface design. In J. Vanderdonckt & C. Farenc (Eds.), *Tools for Working with Guidelines* (pp. 313-324). London: Springer-Verlag.

Vanderdonckt, J. (2005). A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In O. Pastor & J. F. Cunha (Eds.), *Advanced Information Systems Engineering.* Paper presented at 17th International Conference, CAiSE 2005*,* Porto (pp. 16-31). Berlin: Springer.

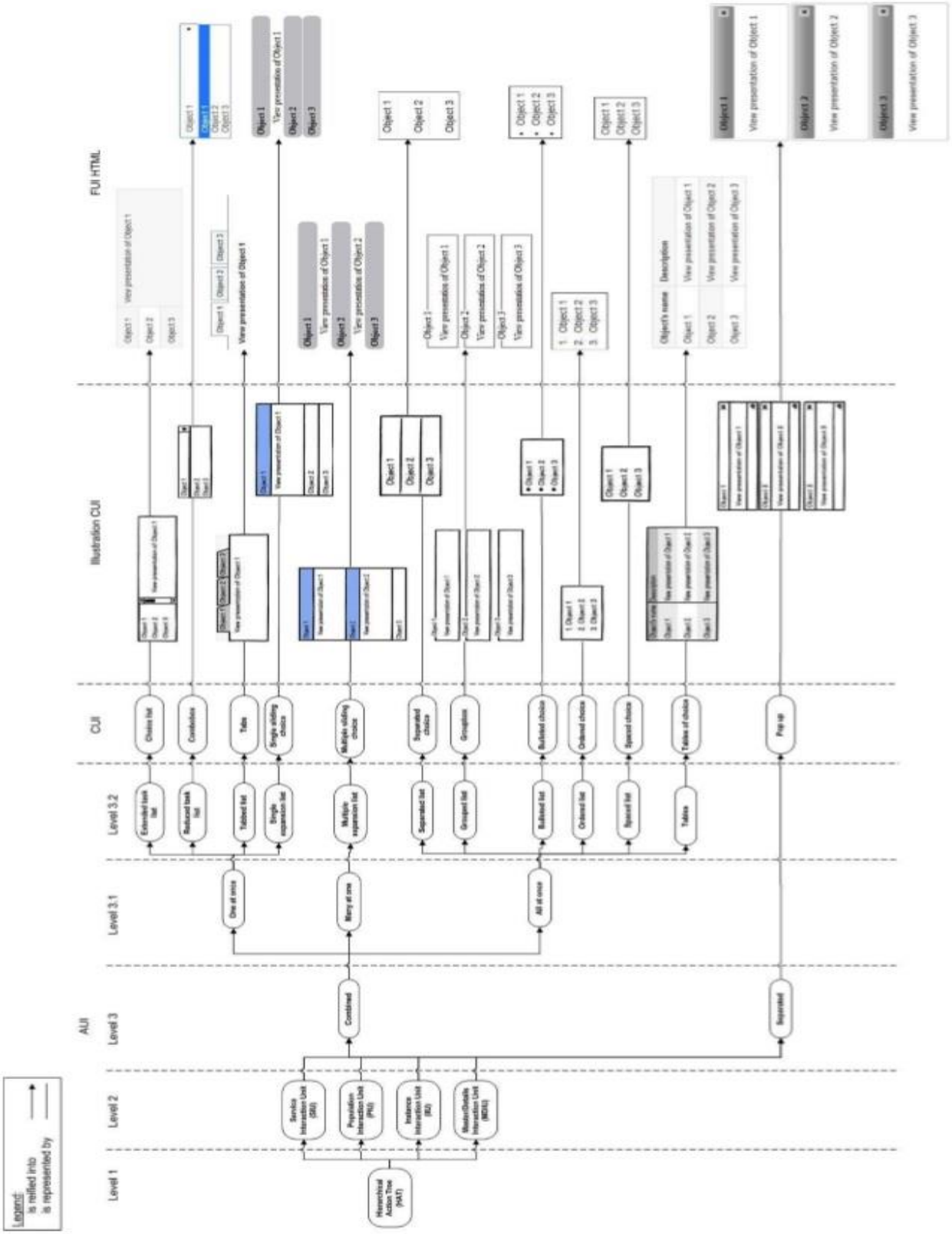
Vanderdonckt, J., & Simarro, F. M. (2010). *Generative pattern-based design of user interfaces*. Paper presented at Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems - PEICS '10, Berlin. New-York: ACM.

Wegge, K. P., & Zimmermann, D. (2007). Accessibility, usability, safety, ergonomics: concepts, models, and differences. In C. Stephanidis (Ed.), *Universal Acess in Human Computer Interaction. Coping with Diversity*. Paper presented at the 4th International Conference on Universal Access in Human-Computer Interaction, UAHCI 200, Beijing (pp. 294-301). Berlin: Springer-Verlag.

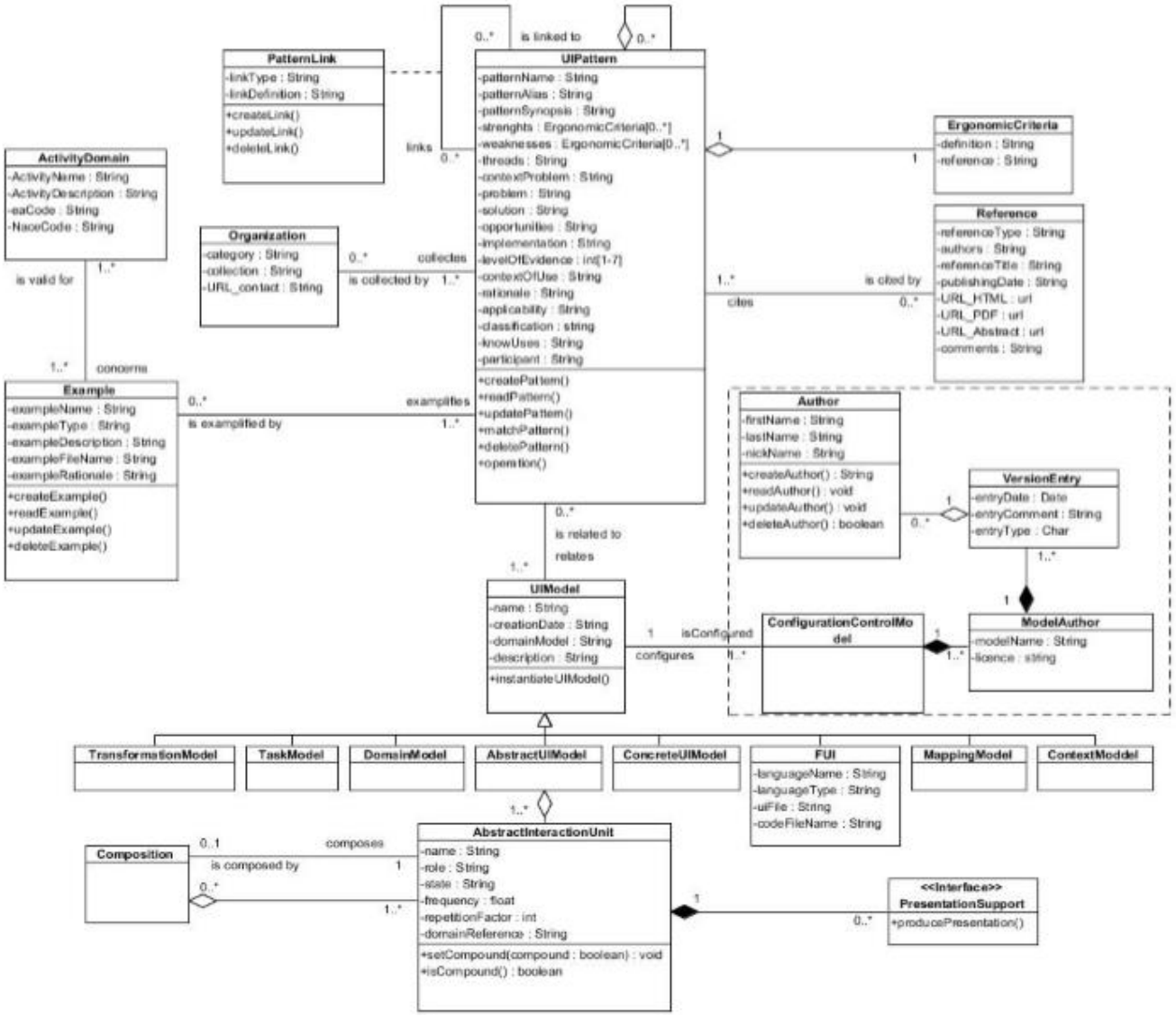
Yahoo! Developer Network. (n.d.). *Yahoo Design Pattern Library*. Retrieved July 17, 2016, from https://developer.yahoo.com/ypatterns/

## 7. Appendix

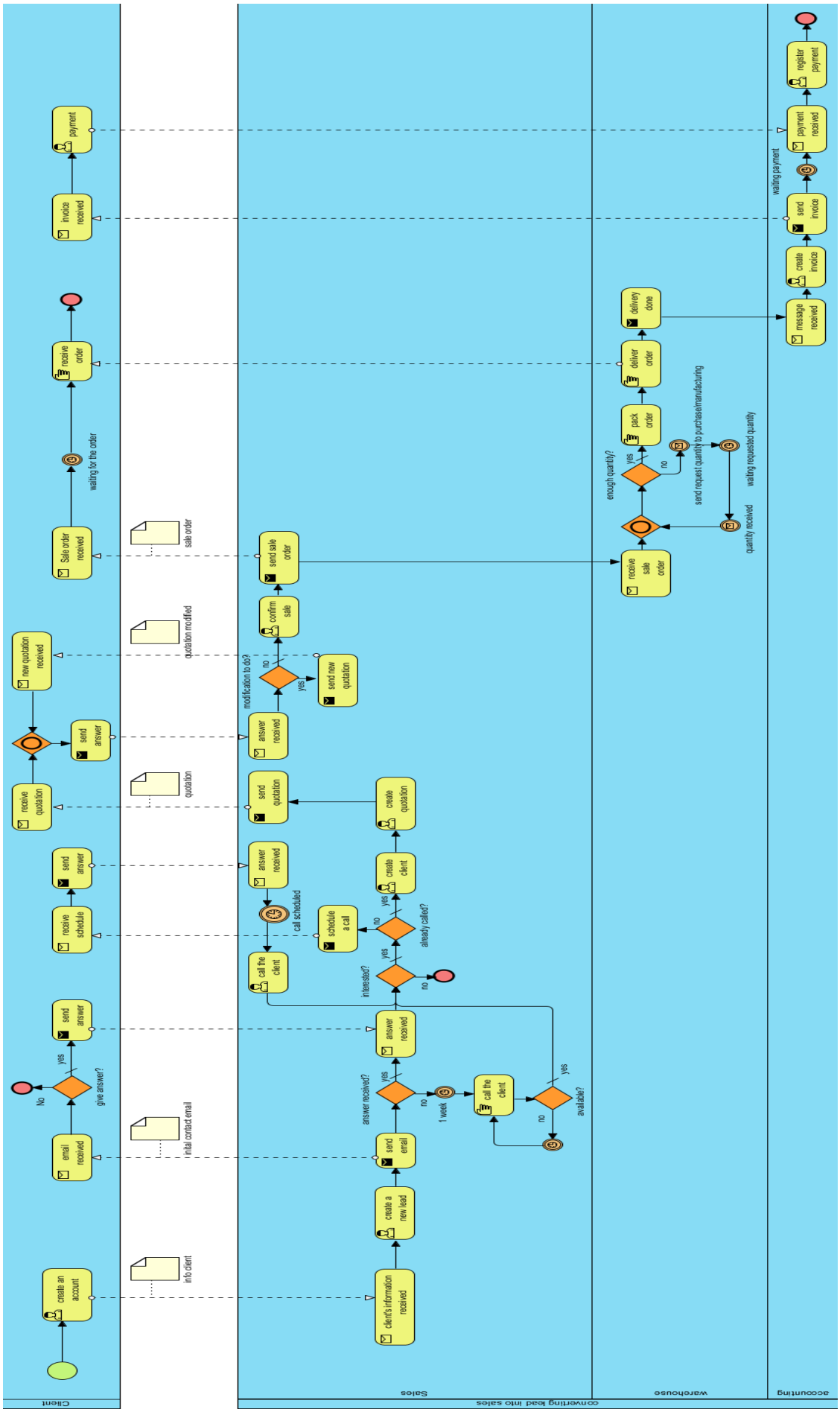
### Appendix 1: The Master/Details pattern through the abstraction levels



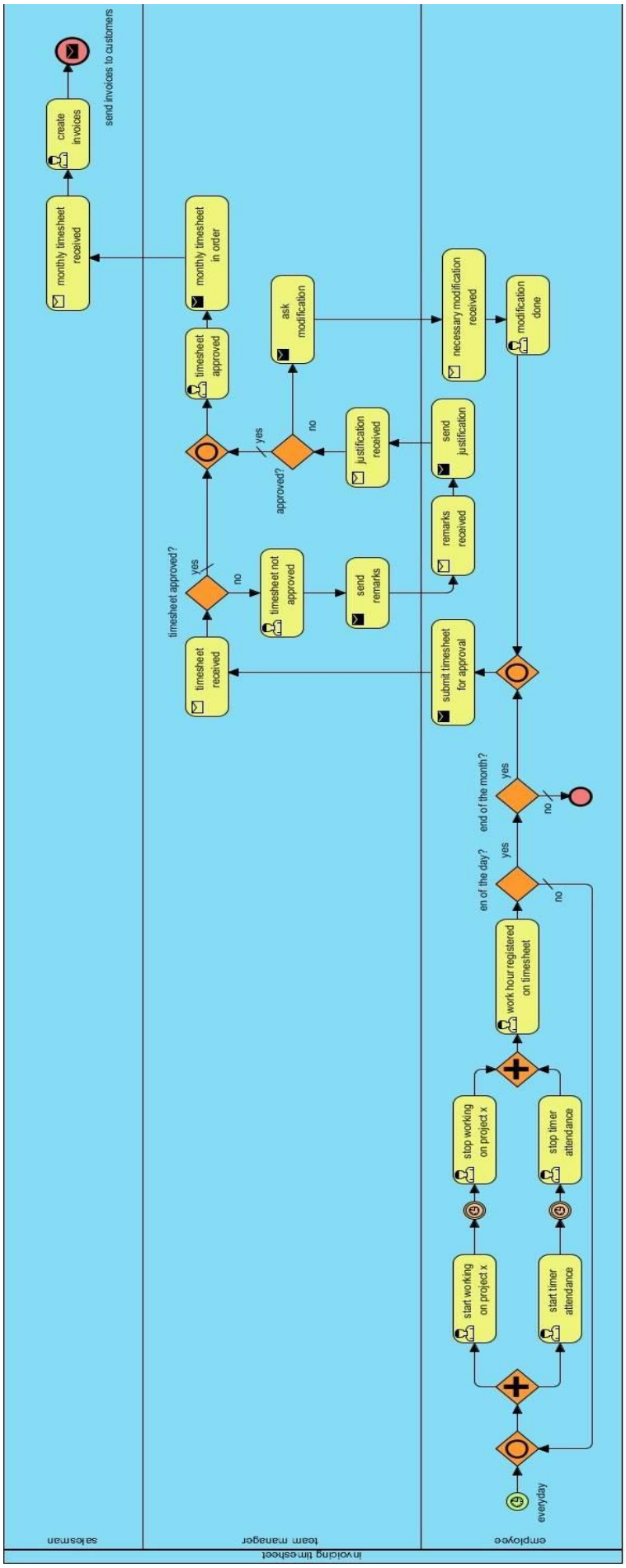
**Appendix 2: The meta-model of UI patterns as described in UIPLML.**



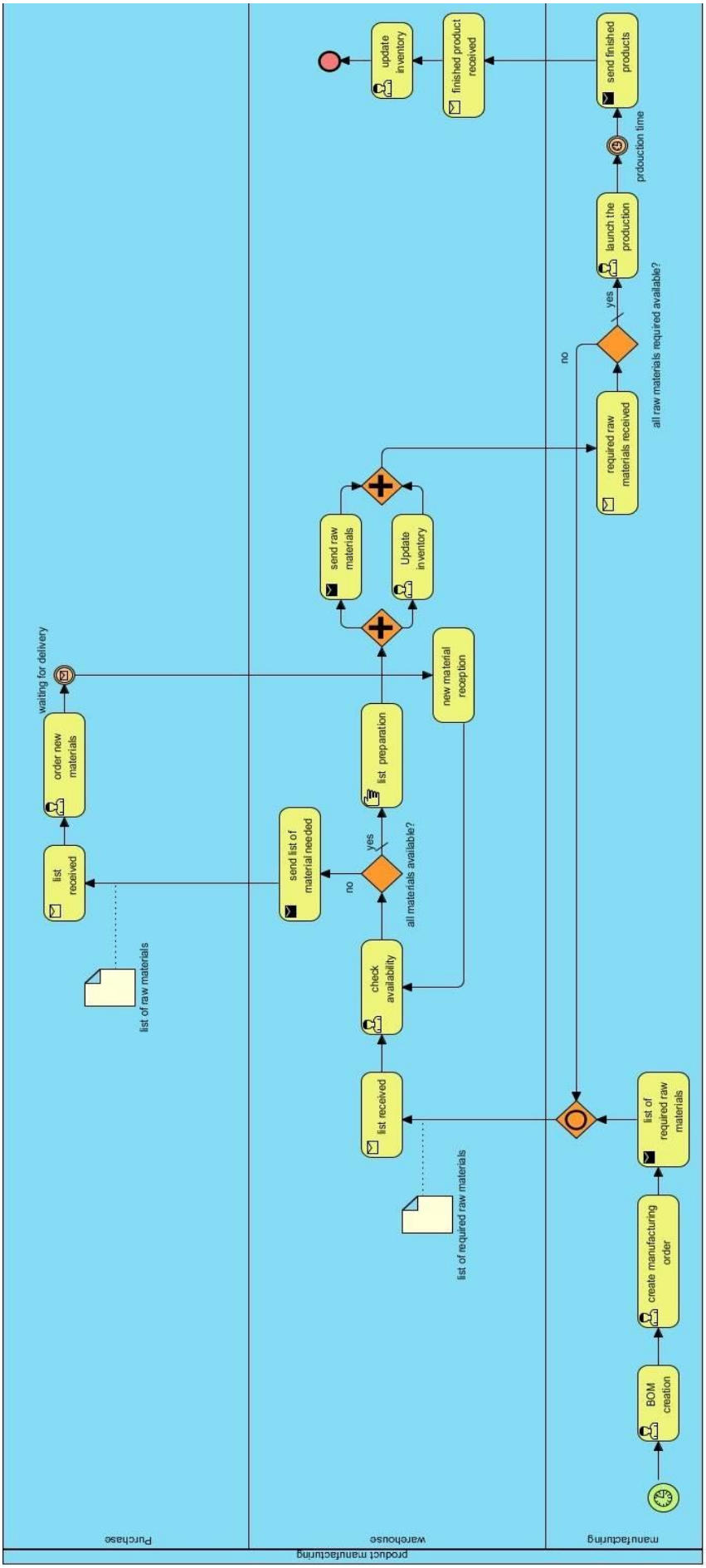
## Appendix 5: Converting lead into sales BPMN



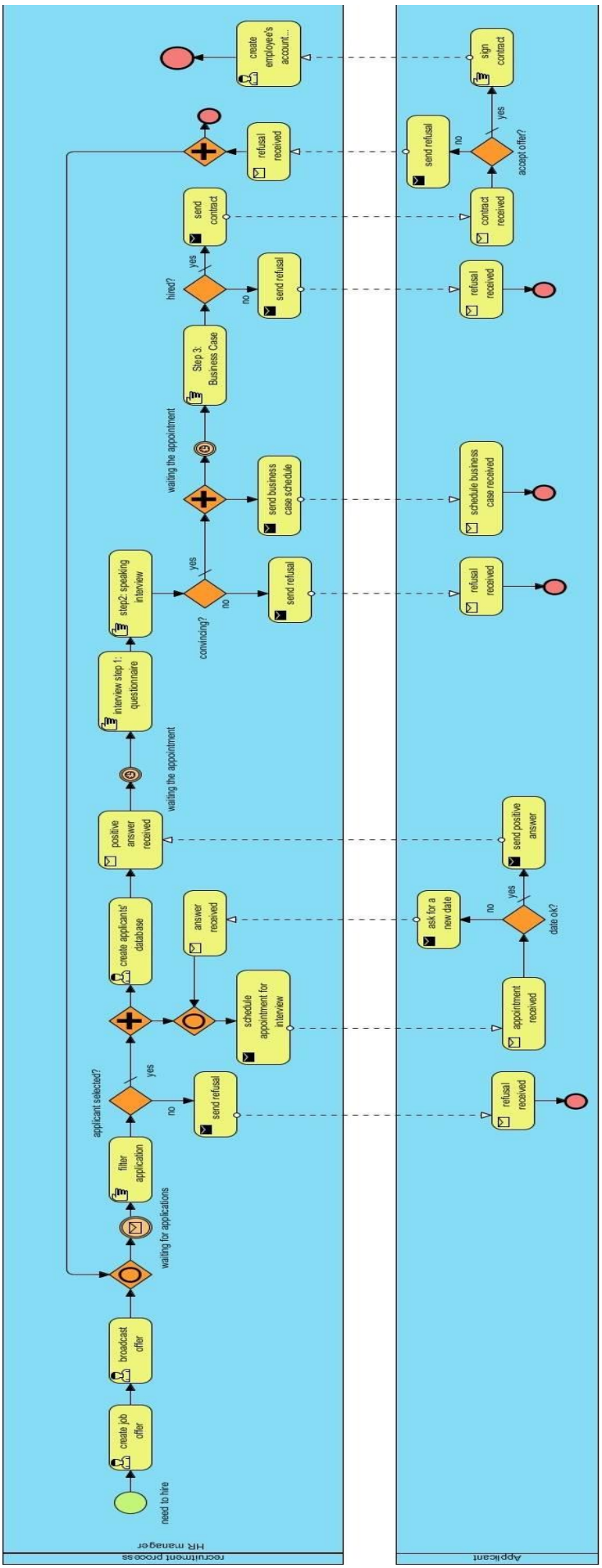
## Appendix 6: Invoicing timesheet BPMN



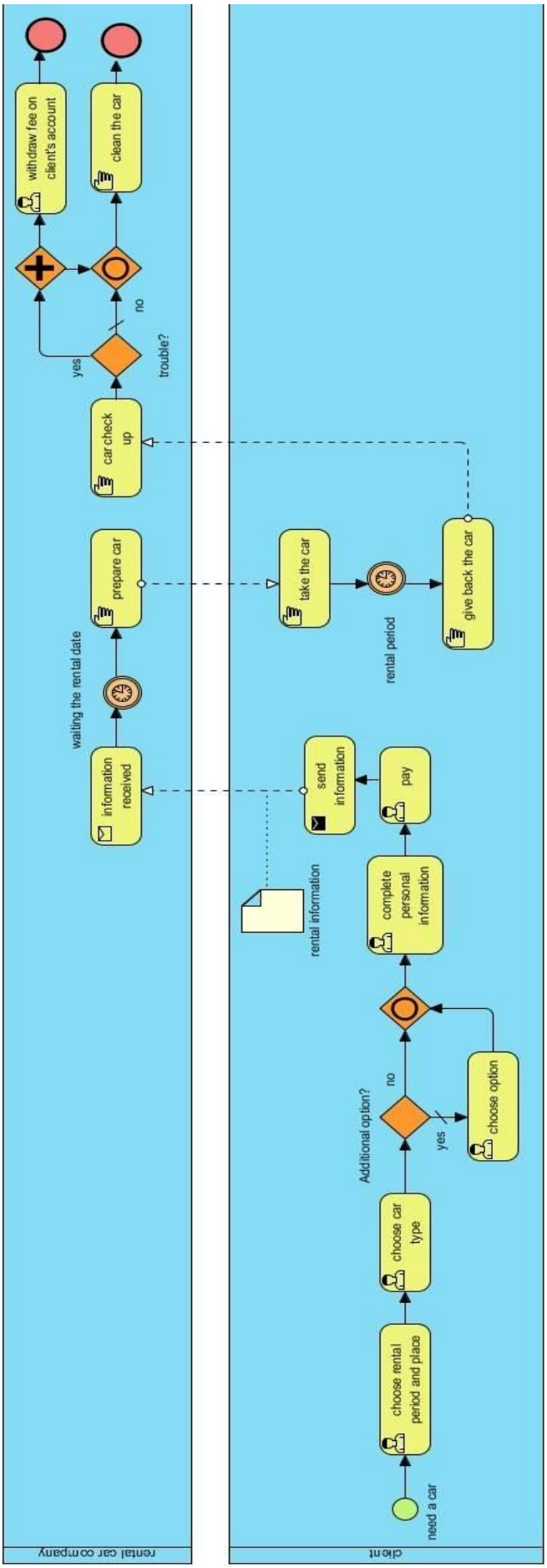
**Appendix 7: New product manufacturing BPMN**



**Appendix 8: Recruitment process BPMN**



**Appendix 9: Rental car BPMN**

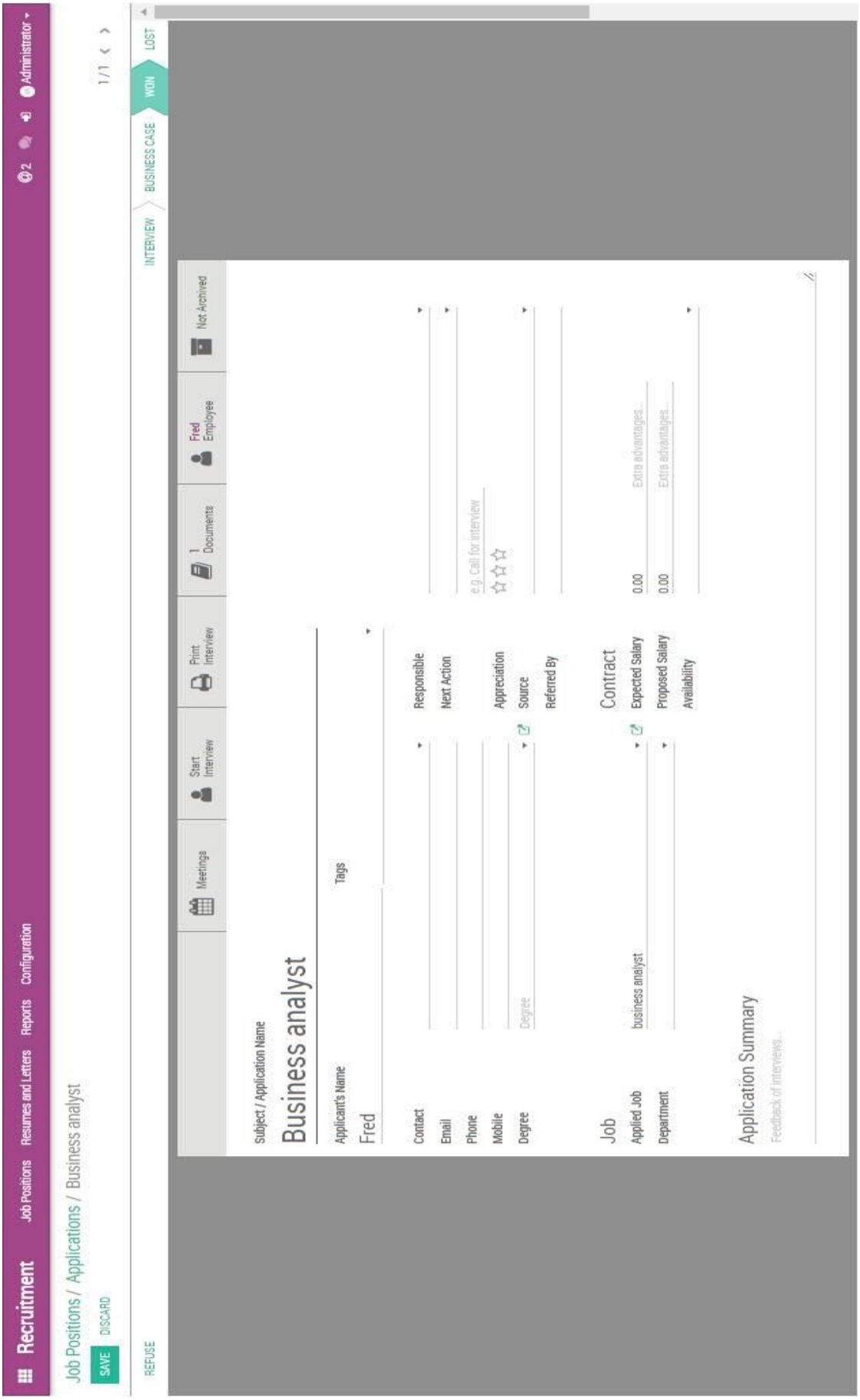


**Appendix 10: Comparison table for new applicant form**

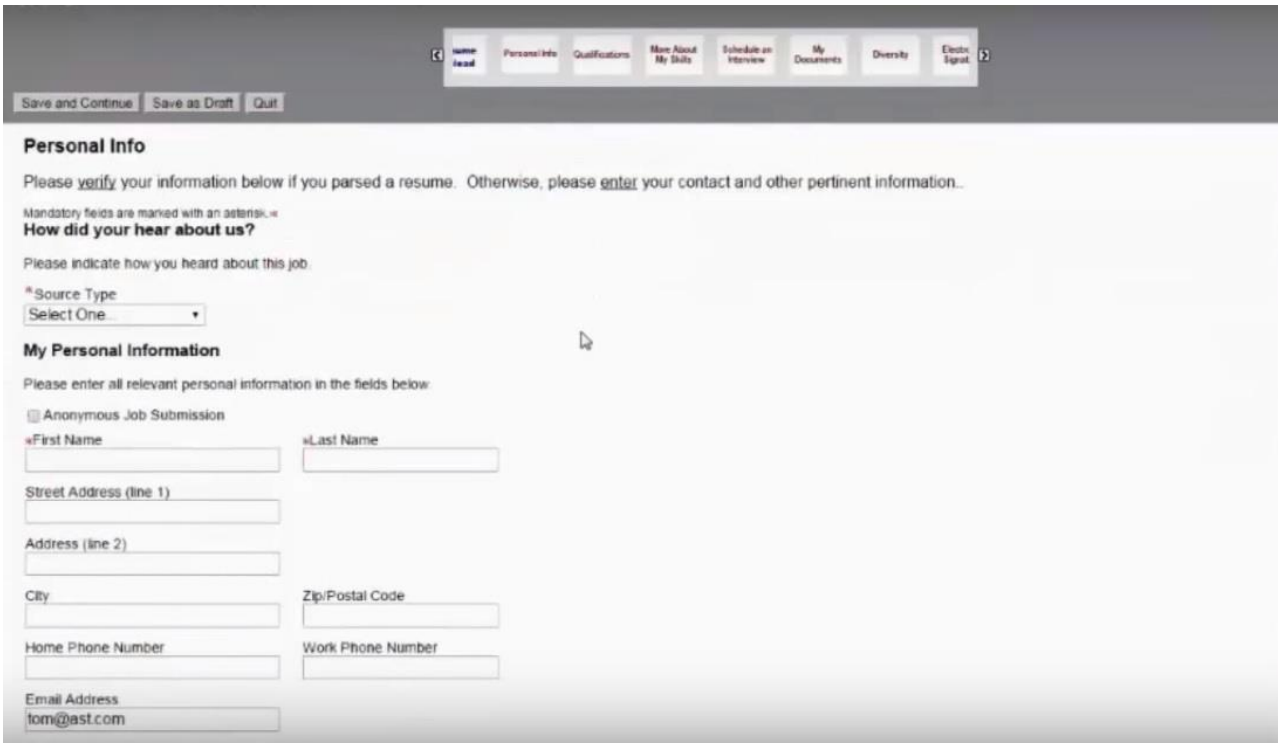
| *Tools Criteria* | *Odoo* | *Oracle Taleo* | *Microsoft Dynamics AX* |
|---|---|---|---|
| *Common criteria* | | | |
| Label | ✓ | ✓ | ✕ does not respect the 6 letters rule |
| Principle of compatibility | ✓ | ✓ | ✓ |
| Required inputs | ✕ not clear, it appears in the validation message | ✓ | ✕ not clear, red line in required fields |
| Minimized effort | ✓ | ✓ | ✓ |
| Data type | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered |
| Default value | ✓ | ✓ | ✓ |
| Prompts (for fields and menus) | ✓ | ✓ | ✕ |
| Help message | ✕ triggered by pointer resting on field but the timer is too long. | ✓ | ✓ |
| Validation message | ✓ | ✓ | ✓ |
| Date/time picker | ✓ | ✕ use of menu | ✓ |
| Value set/range | ✓ | ✕ use radio button for more than 5 choices | ✓ |
| *Specific for fields* | | | |
| Edit mask | ✓ | / | / |
| Compulsory input focus | ✓ | ✕ | ✓ |
| Accept common format | ✕ only respect the Anglo-saxon decimal separator. | ✓ | ✓ |
| Case sensitive | ✓ | ✓ | ✓ |
| Fill-in-the-blanks | / | / | / |
| Autocompletion | ✓ | / | ✓ |
| *Specific for controls* | | | |
| Always editable | ✓ | ✓ | ✓ |
| No Default | ✓ | ✓ | ✓ |

| | | | |
|---|---|---|---|
| Checkbox or toggle | / | / | / |
| Non-yet option | / | ✓ | ✓ |
| Score | 13/17 | 13/17 | 14/18 |
| ***Presentation Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |
| *Display criteria* | | | |
| Conversational flow | ✓ | ✓ | ✓ |
| Group with related information | ✓ | ✓ | ✓ |
| Process flow indicated | ✓ | ✓ | ✕ does not appear clearly on the page |
| Guidance feedback | ✕ no guidance action done | ✓ | ✕ no guidance action done |
| Consistency | ✓ | ✓ | ✓ |
| Explicit controls | ✓ | ✓ | ✓ |
| *Relationship criteria* | | | |
| Avoiding loop | ✓ | ✓ | ✓ |
| No redundant request | ✓ | ✕ ask two times the name | ✓ |
| Prefilling | / | ✓ | ✓ |
| Score | 7/9 | 8/9 | 7/9 |
| **Total Score** (respected rules out of the relevant rules ) | **20/26** | **21/26** | **21/27** |
| Legend: ✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account | | | |

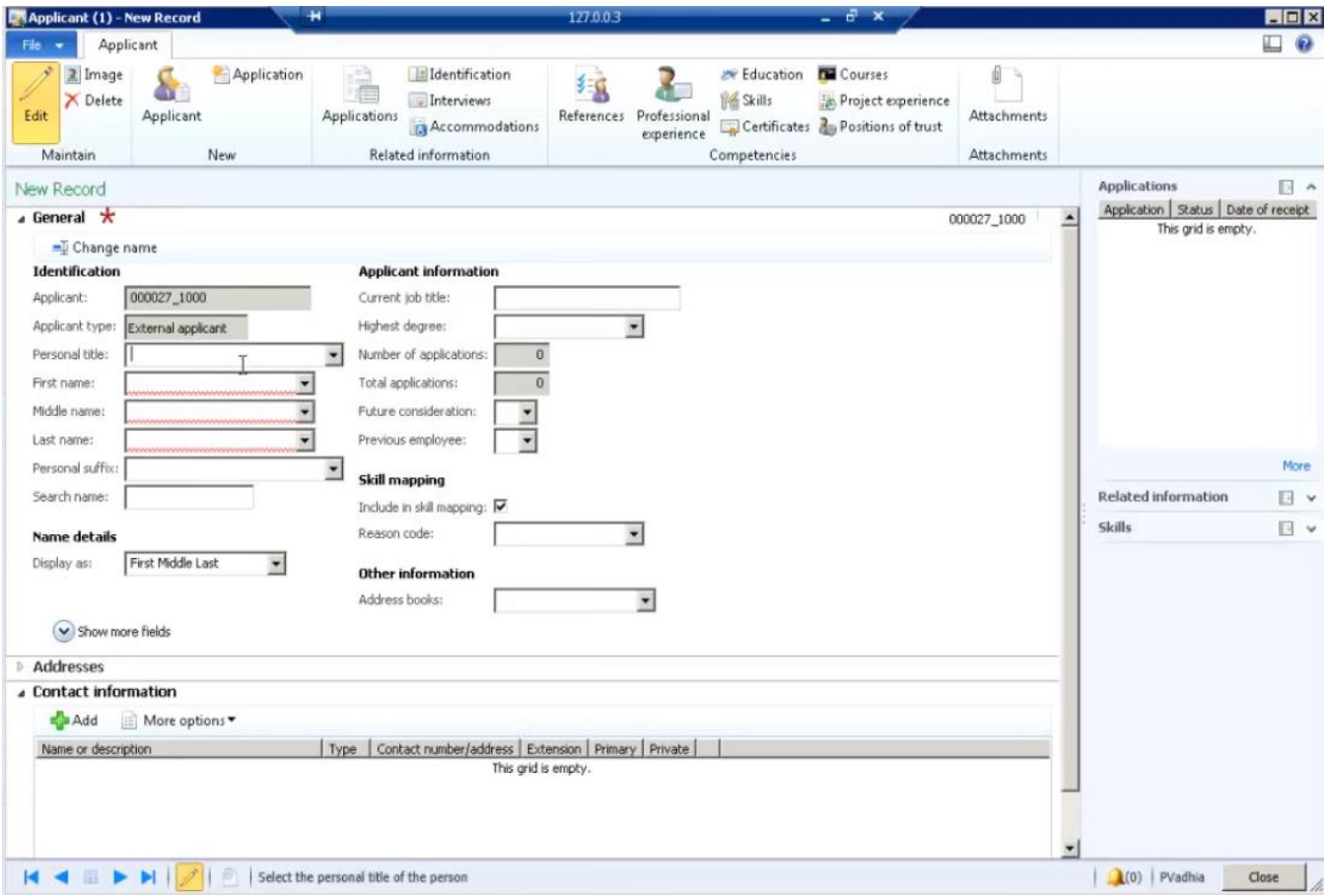
**Appendix 11: New applicant form in Odoo**



## Appendix 12: New applicant form in Oracle Taleo




## Appendix 13: New applicant form in Microsoft Dynamics AX

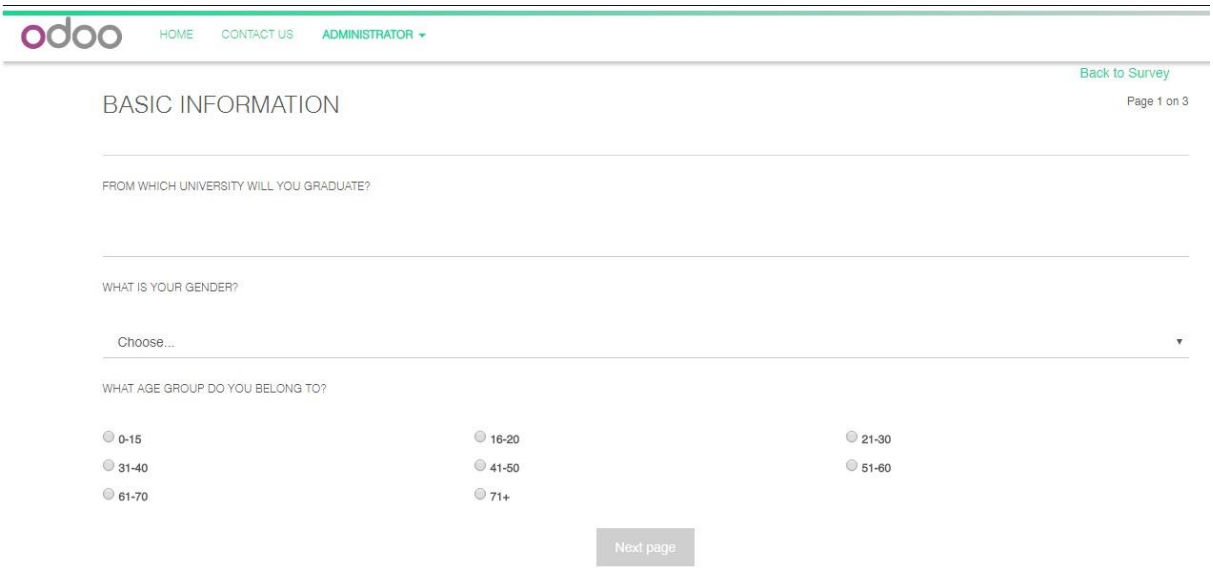


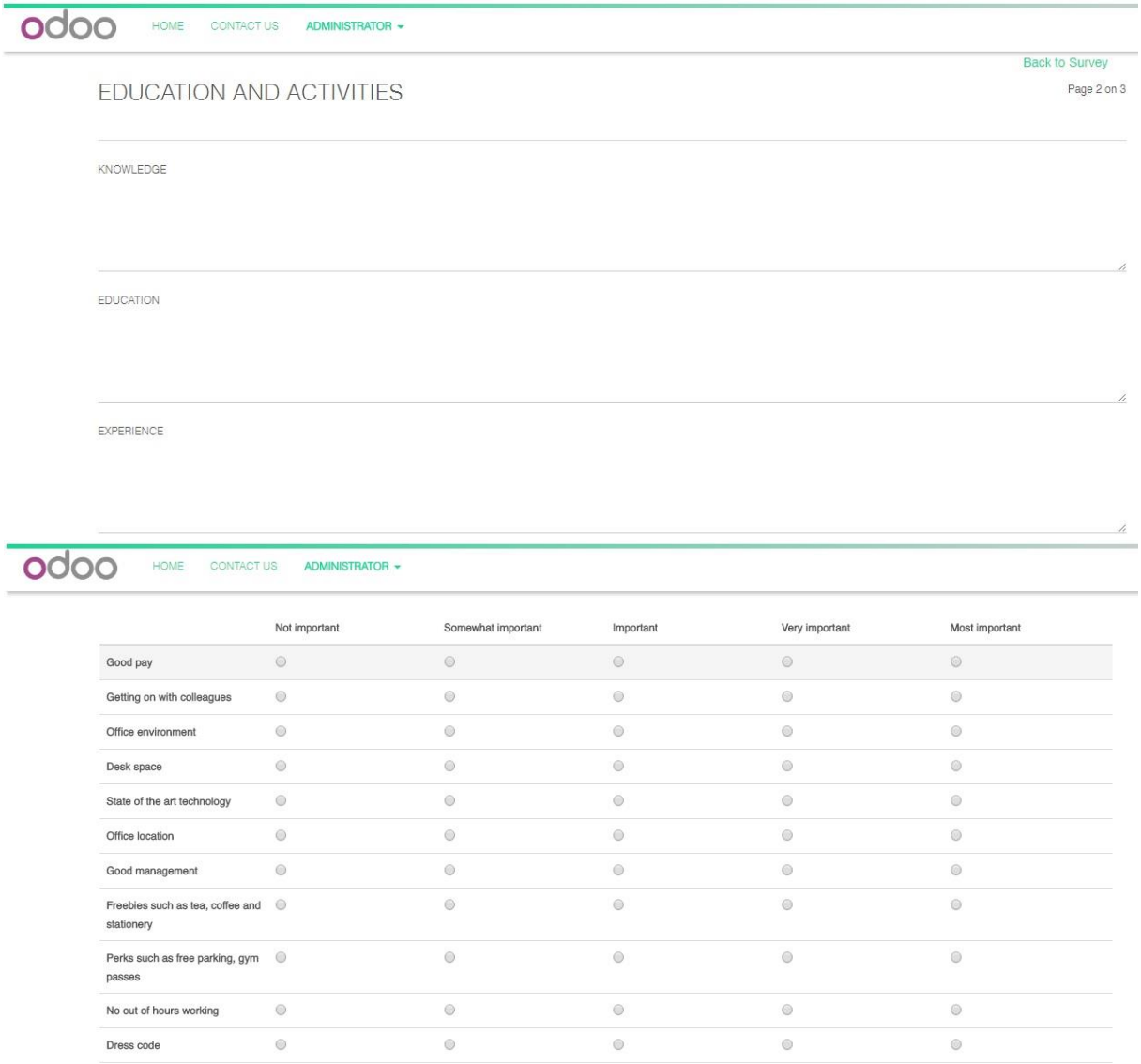
## Appendix 14: Comparison table for interview form

| ***Tools Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |
|---|---|---|---|
| *Common criteria* | | | |
| Label | ✓ | ✓ | |
| Principle of compatibility | ✓ | ✓ | |
| Required inputs | ✓ | ✓ | |
| Minimized effort | ✓ | ✓ | |
| Data type | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered | |
| Default value | / | / | |
| Prompts (for fields and menus) | ✕ | ✕ | |
| Help message | ✕ | ✕ | |
| Validation message | ✓ | ✓ | |
| Date/time picker | ✓ | ✕ use of menu | |
| Value set/range | ✕ | ✓ | |
| *Specific for fields* | | | |
| Edit mask | / | / | |
| Compulsory input focus | ✕ | ✕ | |
| Accept common format | / | / | |
| Case sensitive | ✓ | ✓ | |
| Fill-in-the-blanks | / | / | |
| Autocompletion | / | / | |
| *Specific for controls* | | | |
| Always editable | ✓ | ✓ | |
| No Default | ✓ | ✓ | |
| Checkbox or toggle | / | / | |
| Non-yet option | / | / | |
| Score | 9/14 | 9/14 | |
| ***Presentation Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |

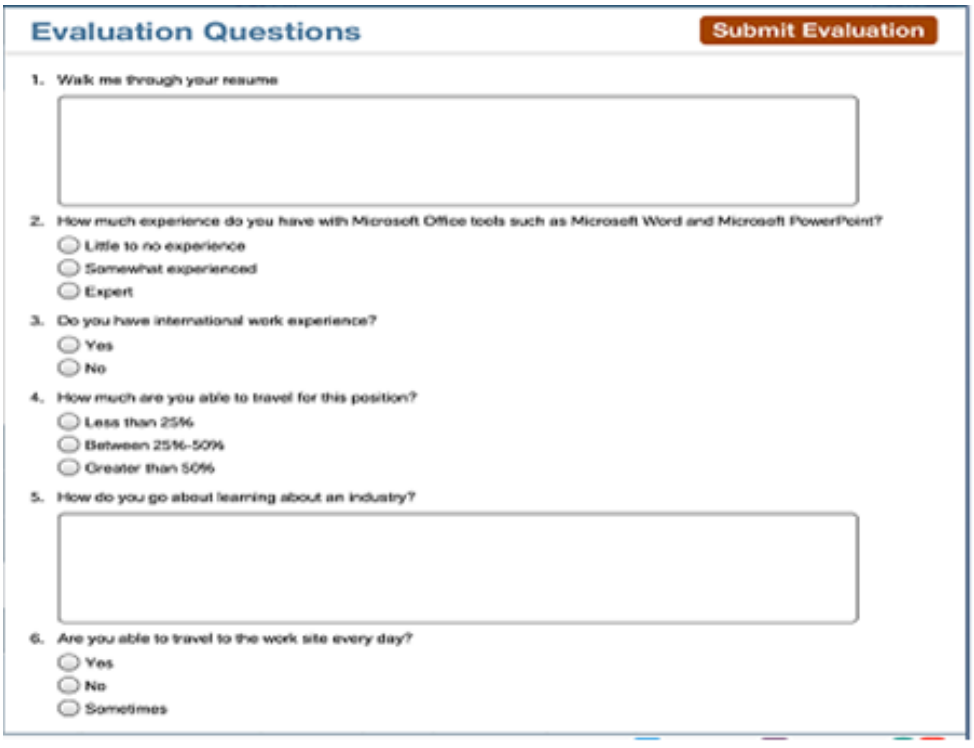
| *Display criteria* | | | |
|---|---|---|---|
| Conversational flow | ✓ | ✓ | |
| Group with related information | ✓ | ✓ | |
| Process flow indicated | ✓ | ✓ | |
| Guidance feedback | ✓ | ✓ | |
| Consistency | ✓ | ✓ | |
| Explicit controls | ✓ | ✓ | |
| *Relationship criteria* | | | |
| Avoiding loop | / | / | |
| No redundant request | ✓ | ✓ | |
| Prefilling | / | / | |
| Score | 7/7 | 7/7 | |
| **Total Score** (respected rules out of the relevant rules ) | **16/21** | **16/21** | |
| Legend: ✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account | | | |

## Appendix 15: Default interview form in Odoo






## Appendix 16: Default interview form in Oracle Taleo

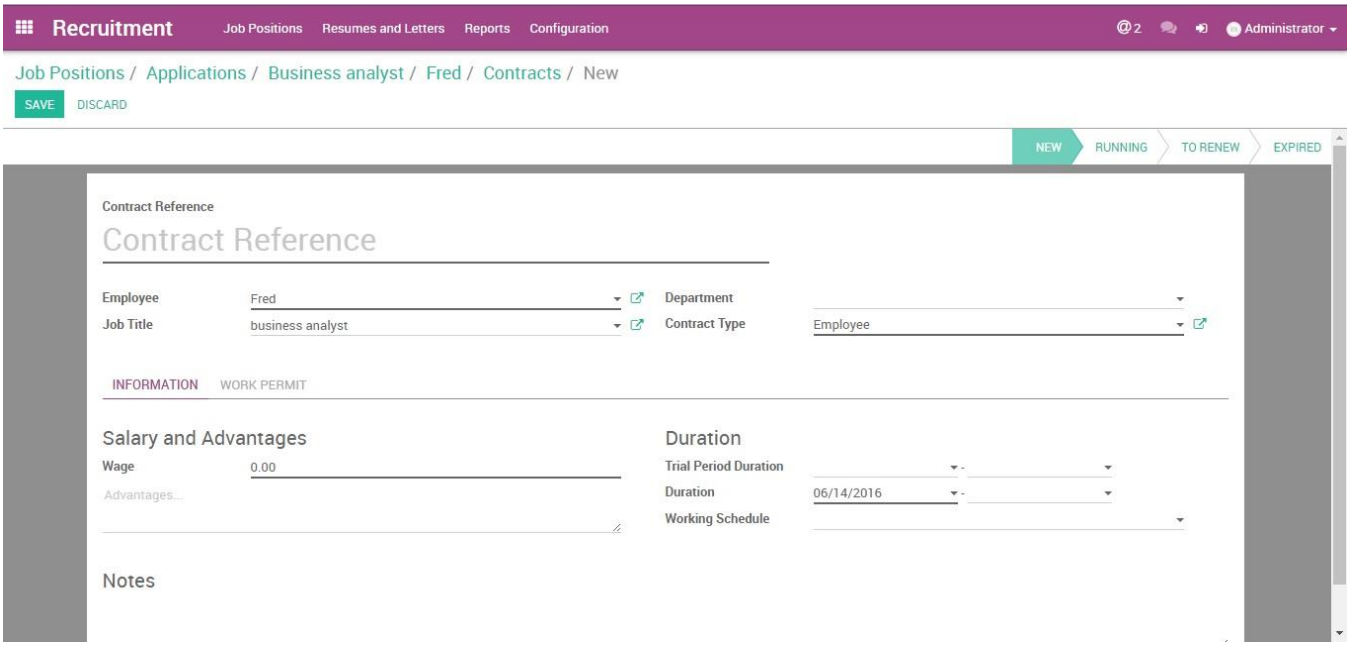


## Appendix 17: Comparison table for contract creation form

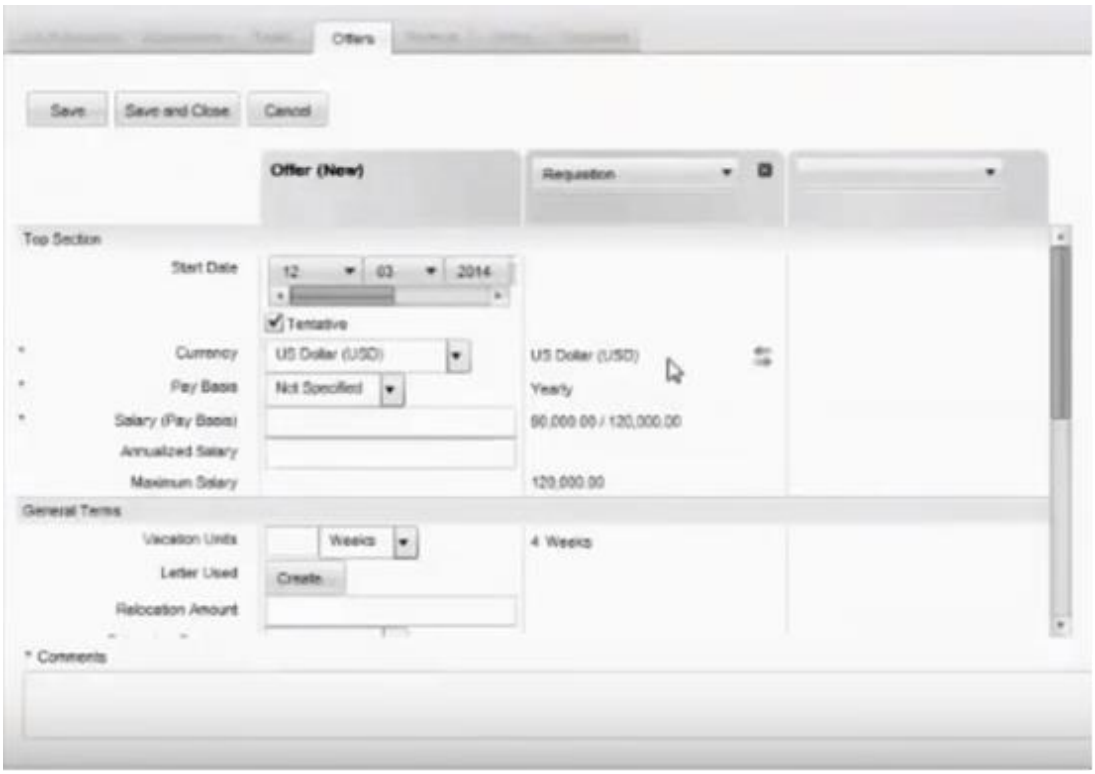
| ***Tools Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |
|---|---|---|---|
| *Common criteria* | | | |
| Label | ✓ | ✓ respect the 6 letters rule | ✕ does not respect the 6 letters rule |
| Principle of compatibility | ✓ | ✓ | ✓ |
| Required inputs | ✕ not clear, it appears in the validation message | ✓ | ✕ not clear, red line in required fields |
| Minimized effort | ✓ | ✓ | ✓ |
| Data type | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered |
| Default value | ✓ | ✕ depends on the case | ✓ |
| Prompts (for fields and menus) | ✓ | ✓ | ✕ |
| Help message | ✕ triggered by pointer resting on field but the timer is too long. | ✓ | ✓ |
| Validation message | ✓ | ✓ | ✓ |
| Date/time picker | ✓ | ✕ use of menu | ✓ |
| Value set/range | ✓ | ✓ | ✓ |
| *Specific for fields* | | | |
| Edit mask | / | / | / |
| Compulsory input focus | ✓ | ✕ | ✓ |
| Accept common format | ✕ only respect the Anglo-saxon decimal separator. | ✕ only respect the Anglo-saxon decimal separator. | ✕ only respect the Anglo-saxon decimal separator. |
| Case sensitive | ✓ | ✓ | ✓ |
| Fill-in-the-blanks | / | / | / |
| Autocompletion | ✓ | / | ✓ |
| *Specific for controls* | | | |
| Always editable | ✓ | ✓ | ✓ |
| No Default | / | / | / |

| | | | |
|---|---|---|---|
| Checkbox or toggle | / | / | ✓ |
| Non-yet option | / | / | / |
| Score | 12/16 | 10/15 | 12/17 |
| ***Presentation Criteria*** | ***Odoo*** | ***Oracle*** | ***Microsoft Dynamics AX*** |
| *Display criteria* | | | |
| Conversational flow | ✓ | ✓ | ✓ |
| Group with related information | ✓ | ✓ | ✓ |
| Process flow indicated | ✓ | ✓ | ✕ does not appear clearly on the page |
| Guidance feedback | ✕ no guidance action done | ✕ no guidance action done | ✕ no guidance action done |
| Consistency | ✓ | ✓ | ✓ |
| Explicit controls | ✓ | ✓ | ✓ |
| *Relationship criteria* | | | |
| Avoiding loop | ✓ | ✓ | ✓ |
| No redundant request | ✓ | ✓ | ✓ |
| Prefilling | ✓ | ✓ in option | ✕ |
| Score | 8/9 | 8/9 | 6/9 |
| **Total Score** (respected rules out of the relevant rules ) | **20/25** | **18/24** | **18/26** |
| Legend: ✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account | | | |

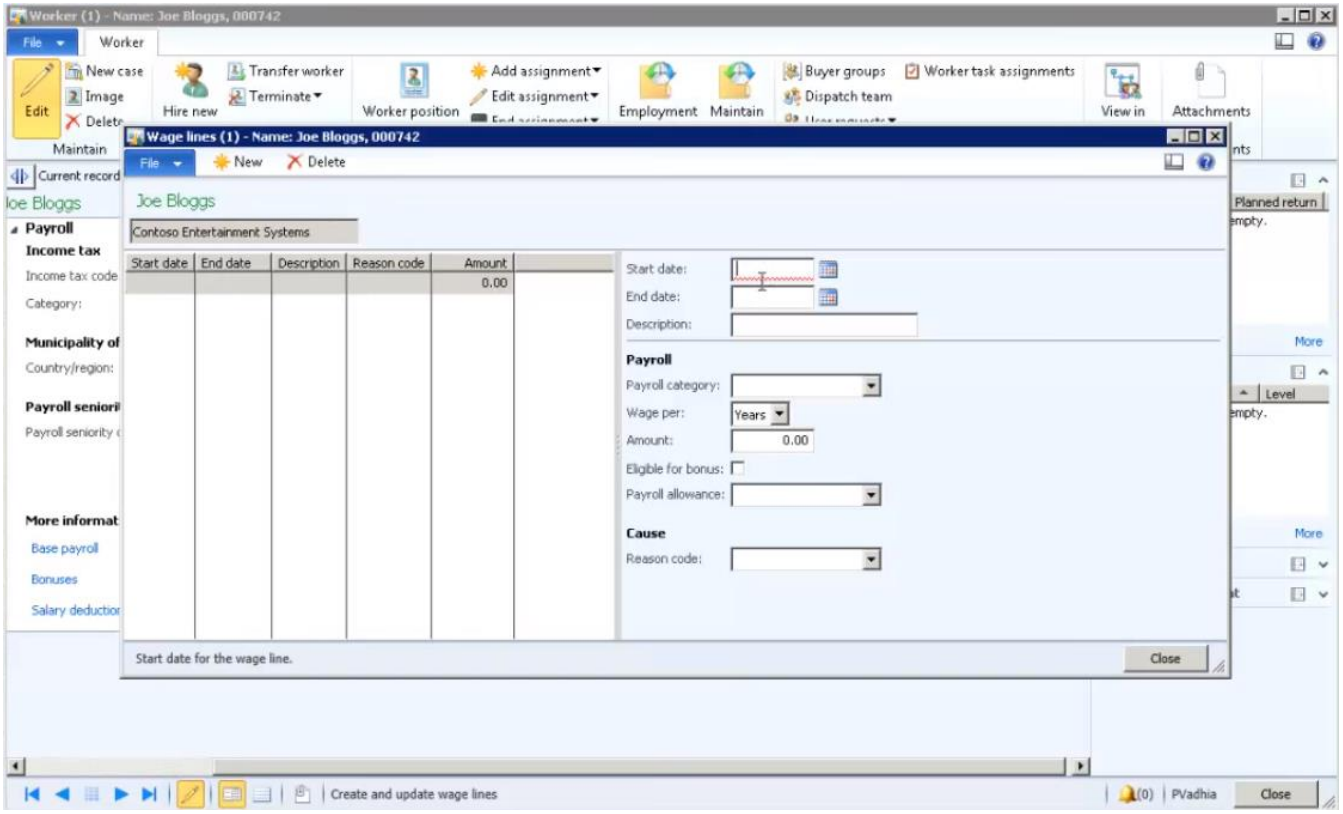
## Appendix 18: Contract creation form in Odoo




## Appendix 19: Contract creation form in Oracle Taleo



## Appendix 20: Contract creation form in Microsoft Dynamics AX

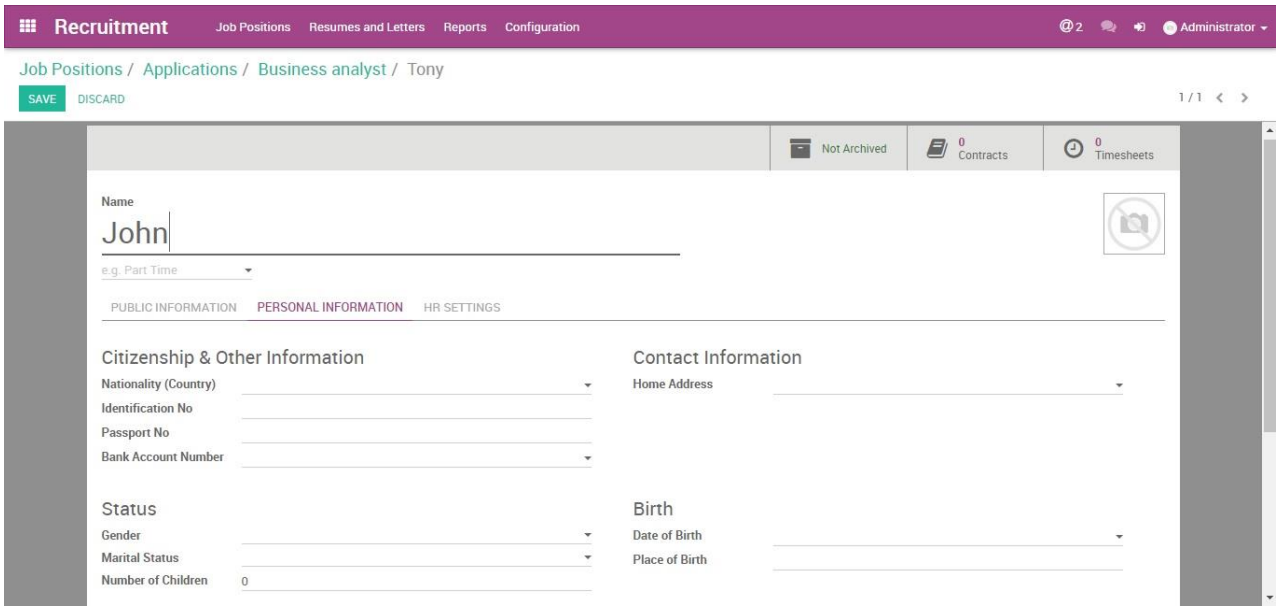


## Appendix 21: Comparison table for new employee account form

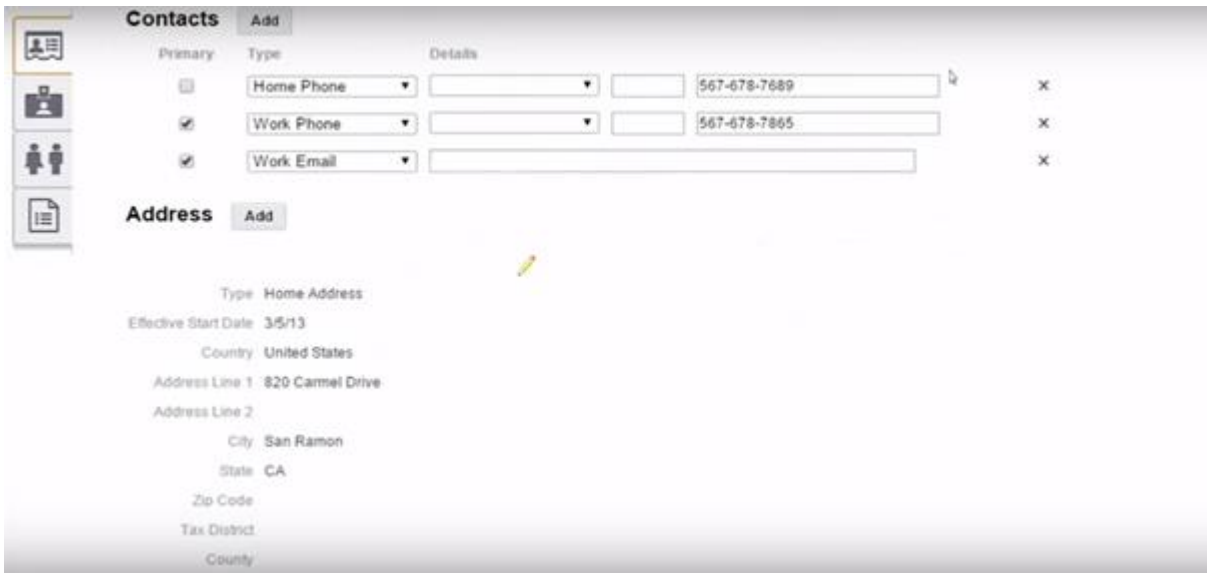
| *Tools Criteria* | *Odoo* | *Oracle Taleo* | *Microsoft Dynamics AX* |
|---|---|---|---|
| *Common criteria* | | | |
| Label | ✕ does not respect the 6 letters rule | ✓ respect the 6 letters rule | ✕ does not respect the 6 letters rule |
| Principle of compatibility | ✓ | ✓ | ✓ |
| Required inputs | ✕ not clear, it appears in the validation message | ✓ | / |
| Minimized effort | ✓ | ✓ | ✓ |
| Data type | ✕depending on the arguments (Mobile number no control) | ✕not blocking when wrong type entered | ✕not blocking when wrong type entered |
| Default value | / | / | ✓ |
| Prompts (for fields and menus) | ✓ | ✕ | ✕ |
| Help message | ✕ triggered by pointer resting on field but the timer is too long. | ✓ | ✓ |
| Validation message | ✓ | ✓ | ✓ |
| Date/time picker | ✓ | ✕ use of menu | ✓ |
| Value set/range | ✓ | ✓ | ✓ |
| *Specific for fields* | | | |
| Edit mask | ✕ | ✕ | ✓ |
| Compulsory input focus | ✓ | ✕ | ✓ |
| Accept common format | ✕ only respect the Anglo-saxon decimal separator. | ✕ only respect the Anglo-saxon decimal separator. | ✕ only respect the Anglo-saxon decimal separator. |
| Case sensitive | ✓ | ✓ | ✓ |
| Fill-in-the-blanks | / | / | / |
| Autocompletion | ✓ | ✓ | ✓ |
| *Specific for controls* | | | |
| Always editable | ✓ | ✓ | ✓ |
| No Default | ✓ | ✓ | / |
| Checkbox or toggle | / | / | / |

| | | | |
|---|---|---|---|
| Non-yet option | / | / | / |
| Score | 11/17 | 11/17 | 12/16 |
| ***Presentation Criteria*** | ***Odoo*** | ***Oracle Taleo*** | ***Microsoft Dynamics AX*** |
| *Display criteria* | | | |
| Conversational flow | ✓ | ✓ | ✓ |
| Group with related information | ✓ | ✓ | ✓ |
| Process flow indicated | ✓ | ✓ | ✕ does not appear clearly on the page |
| Guidance feedback | ✕ no guidance action done | ✕ no guidance action done | ✕ no guidance action done |
| Consistency | ✓ | ✓ | ✓ |
| Explicit controls | ✓ | ✓ | ✓ |
| *Relationship criteria* | | | |
| Avoiding loop | ✓ | ✓ | ✓ |
| No redundant request | ✓ | ✓ | ✓ |
| Prefilling | ✓ | ✓ | ✓ |
| Score | 8/9 | 8/9 | 7/9 |
| **Total Score** (respected rules out of the relevant rules ) | **19/26** | **19/26** | **19/25** |
| Legend: ✓ the rule is respected, ✕ the rule is not respected, / the rule does not come into account | | | |

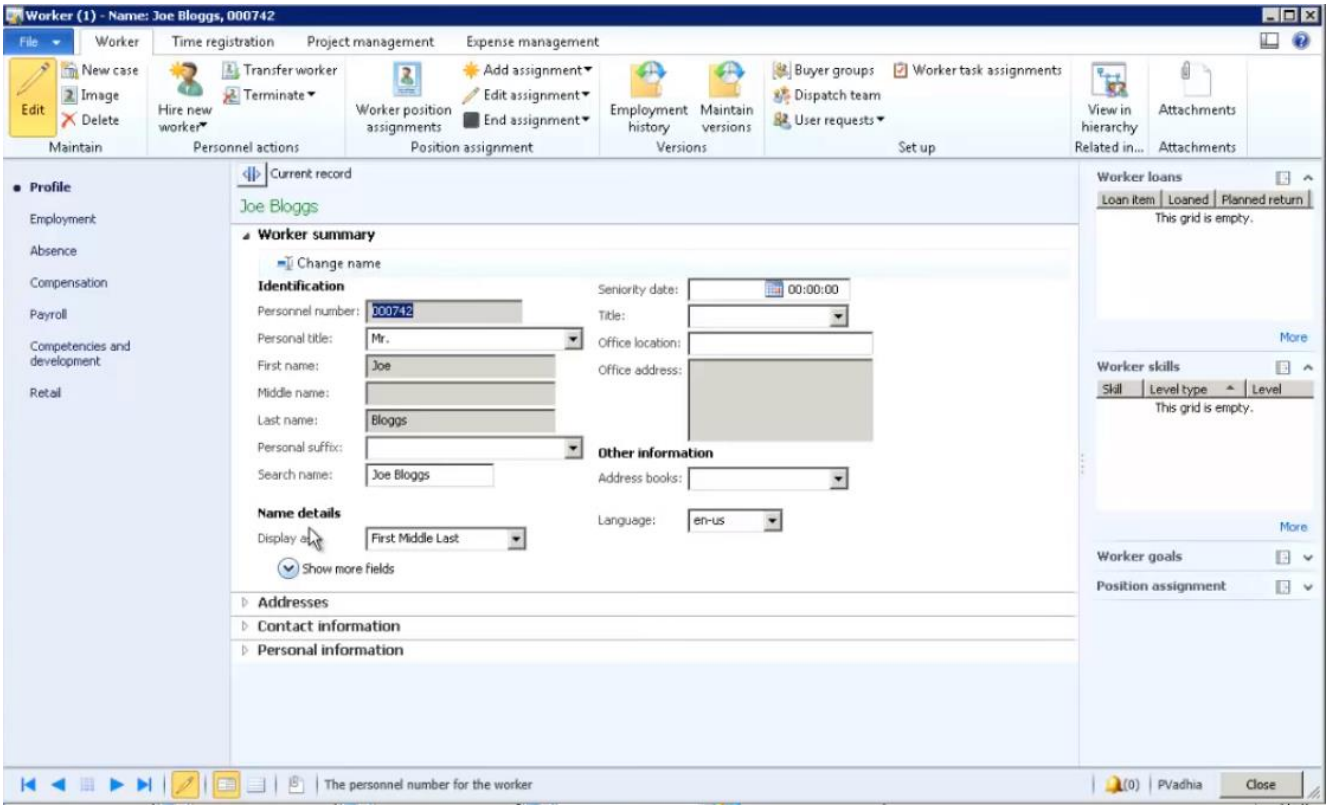
## Appendix 22: New employee account form in Odoo



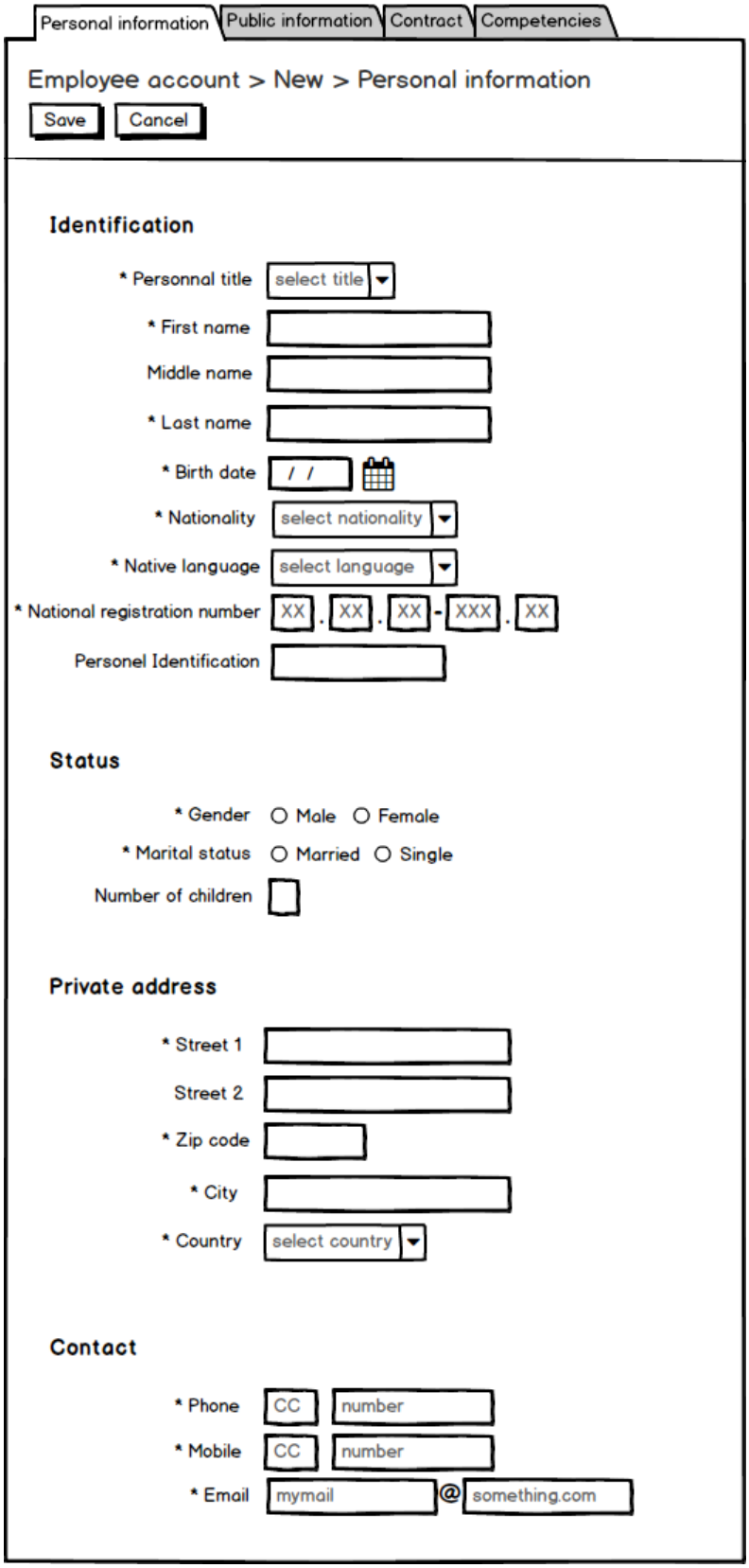

## Appendix 23: New employee account form in Oracle Taleo



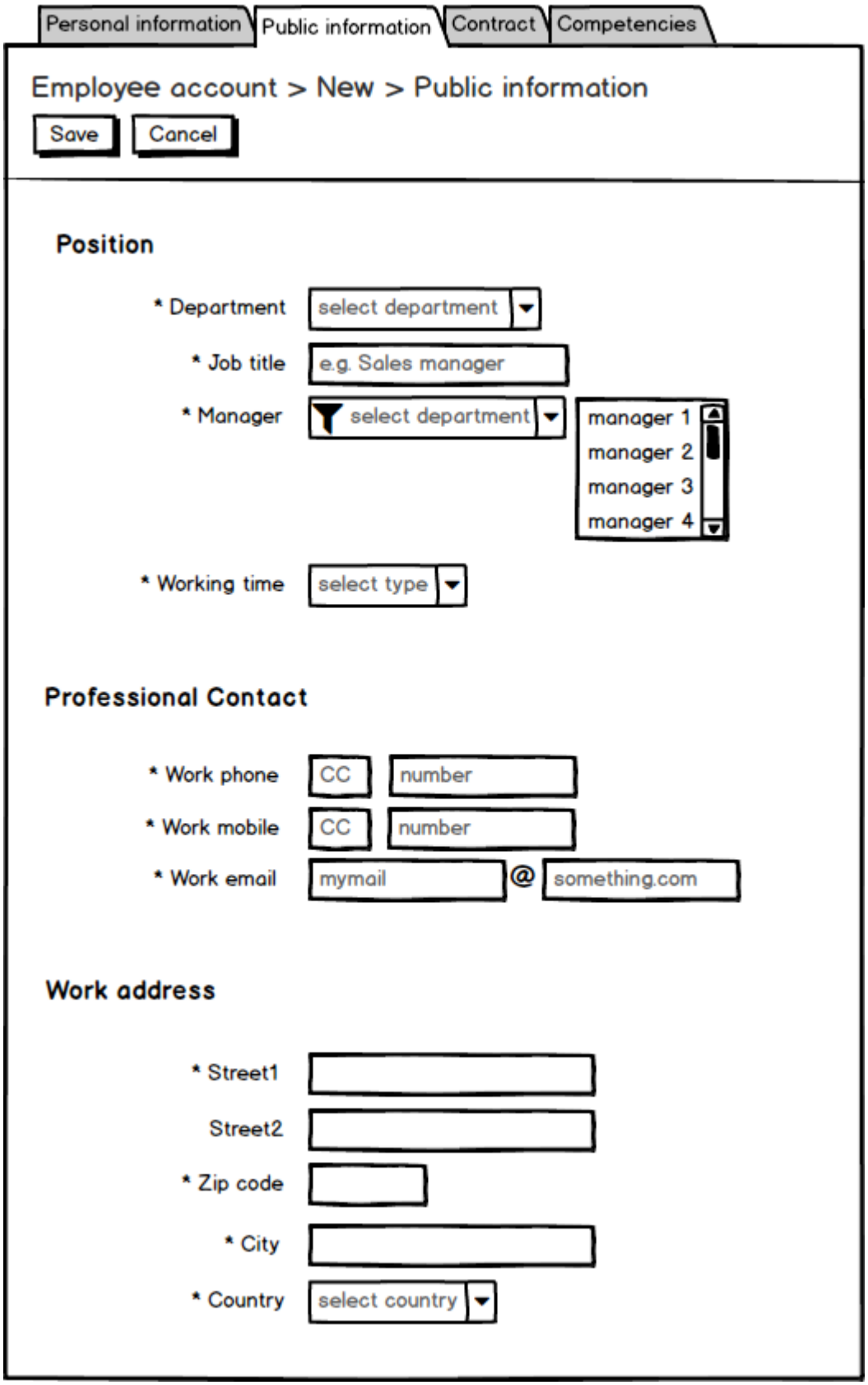
## Appendix 24: New employee account form in Microsoft Dynamics AX



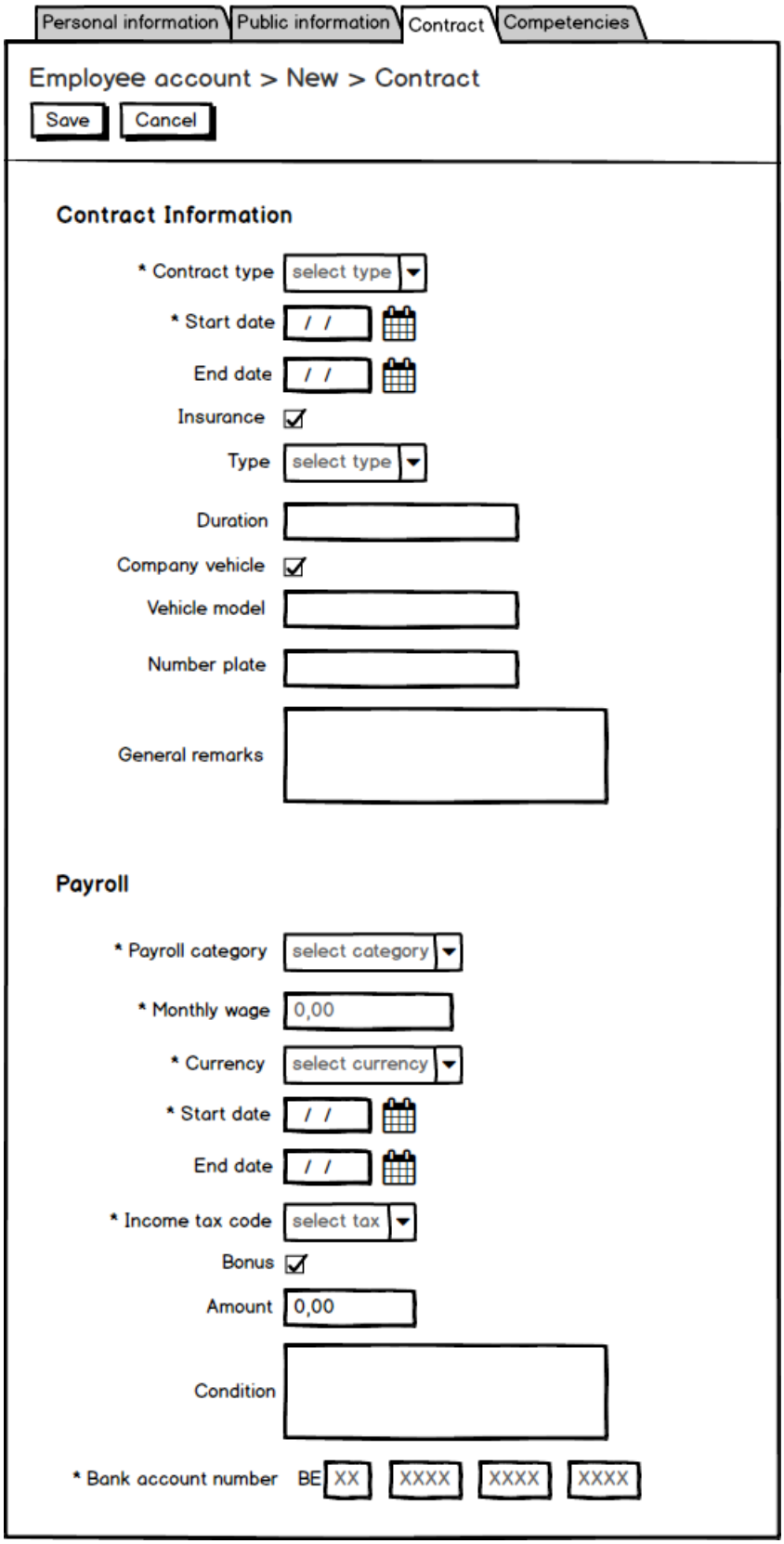
**Appendix 25: Personal information of the suggested form for the employee's account**



**Appendix 26: Public information of the suggested form for the employee's account**



**Appendix 27: Contract of the suggested form for the employee's account**



**Appendix 28: Competencies of the suggested form for the employee's account**