Softening the Learning Curve of Software Development Tools

João Filipe Nunes Fernandes

Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores

Júri

Presidente: Profª. Drª. Ana Paiva
Orientador: Prof. Dr. Mário Rui Gomes
Co-Orientador: Prof. Dr. Rui Prada
Co-Orientador: Eng. Lúcio Ferrão
Vogal: Prof. Dr. Andreas Miroslaus Wichert

12 de Outubro de 2011
Acknowledgments

In the first place I would like to thank my family for the constant support. Even though my academic life has left me without much time to be with them, I enjoy the brief moments of their company.

Also, Bernardo, Pedro, Roberto Fernandes, Hugo and Roberto Silva deserve to be mentioned for their support and good moments provided. I would also like to thank the handful of friends that I have made during this period of my career. Rui and Sara thanks for the fun while developing games. Lobato, thanks for believing in me, the guidance and good subjects to discuss. Sérgio, Ana, Cachal, Luís, Cecilia and Nuno, thanks for the help and great moments. David, thanks for sharing the interest in other subjects aside from computer science. Carolina, thanks for the always interesting conversations and moments.

I would also like to thanks Prof. Mário Rui Gomes for the opportunity and efforts made so that this dissertation could be realized. Also Prof. Rui Prada deserves to be acknowledged for the effort invested in my work.

This work would not be possible without the OutSystems R&D, to which I owe my gratitude. For this fact I would like to emphasize my gratitude towards Lúcio Ferrão, which provided his support throughout this work. I would also like to thanks Tiago Simões for the support, and also Jaime Vasconcelos, Mário Araújo and José Caldeira for the great discussions that contributed to this work.

12 de Outubro de 2011
João Filipe Nunes Fernandes
Abstract

In the software industry, albeit widespread techniques to make the development cycle more agile, it is not easy for a company to implement, maintain and deliver tools that support the learning stages of their users. As such, the learning processes and tools tend to receive fewer budget on the product backlog. With this in consideration, a framework for computer-assisted learning was devised. This framework is intended to increase the subject matter retention rate, minimize the learners’ frustration levels, while keeping the development and maintenance costs low.

The learning curve presented by an Integrated Development Environment is discussed. The embedded tutorial system built in OutSystems IDE presents low retention rates, which makes users perform poorly when a transfer test is presented. This dissertation presents a framework which tries to improve the performance of the users after being exposed to a tutorial, without increasing frustration levels. The several components of the framework are detailed and it is argued that the framework can be implemented on other inherently complex tools aside from IDE’s. It is also discussed how the embedded tutorial system was extended with the developed framework and the yielded results.

The challenge lies in combining the simplicity of interactive tutorial systems, with the tailoring offered by Intelligent Tutoring Systems. This approach presents some degree of adaptation to the learner, while keeping the product simple. It provides a minimal viable product that is able to soften the learning curve of inherently complex applications.
Resumo

Não obstante as tentativas para tornar o ciclo de desenvolvimento ágil, na indústria de software, não é fácil para uma companhia implementar, manter e apresentar ao utilizador final ferramentas que suportem as várias etapas da aprendizagem. Como tal, os processos e ferramentas relacionadas com a aprendizagem, tendem a receber menos recursos no ciclo de desenvolvimento.

Tendo este problema em consideração, foi desenvolvida uma framework para aprendizagem assistida por computador. Esta framework pretende aumentar a retenção dos materiais de aprendizagem, bem como minimizar a frustração do utilizador. Pretende-se igualmente manter os custos de desenvolvimento baixos. A curva de aprendizagem de um ambiente de desenvolvimento integrado (IDE) é discutida. O sistema de tutoriais embbedidos integrado no IDE da OutSystems apresenta uma baixa retenção, o que por sua vez faz com que os utilizadores demorem algum tempo a solucionar uma tarefa. Esta dissertação apresenta uma framework que pretende melhorar os níveis de retenção de utilizadores que sejam expostos a este sistema de tutoriais, sem aumentar os seus níveis de frustração.

Os vários componentes da framework são detalhados, e é argumentado que esta poderá ser integrada noutros produtos e sistemas inerentemente complexos. É igualmente discutido como é que esta solução foi integrada no sistema de tutoriais embbedidos e são apresentados os resultados obtidos. O desafio está em combinar a simplicidade dos tutoriais interactivos, com a assistência oferecida pelos sistemas de tutoriais inteligentes.
Keywords

Learning
Interactive Tutorials
Embedded Tutorials
Improving Knowledge Retention
Software Training

Palavras Chave

Aprendizagem
Tutoriais Interactivos
Tutoriais Embebidos
Aumento da Retenção de Conhecimento
Formação em Software
# Contents

1 Introduction .................................................. 3
   1.1 Motivation ................................................. 3
   1.2 Problem .................................................. 3
   1.3 Goals .................................................... 4
   1.4 Context .................................................. 4
   1.5 Challenges .............................................. 5
   1.6 Contribution ............................................ 5
   1.7 Outline .................................................. 6

2 Detailing the Problem ........................................... 9
   2.1 Learning A New Development Platform ......................... 9
   2.2 The Product Owner Point Of View .......................... 10
   2.3 The User Point Of View .................................. 14

3 Psychology Background ......................................... 17
   3.1 “Signifiers” ............................................. 18
   3.2 Schemata ............................................... 18
   3.3 Scripts .................................................. 19
   3.4 Dreyfus Model of Skill Acquisition ........................ 19

4 Related Work: Learning Tools ................................... 21
   4.1 Non-Interactive Media ................................... 21
   4.2 Video Tutorials .......................................... 22
   4.3 Discussion Forums ....................................... 23
   4.4 “Gamification” .......................................... 23
   4.5 Abstraction Layer ....................................... 26
   4.6 Interactive Tutorials .................................... 27
   4.7 Intelligent Tutoring Systems ............................. 29
   4.8 e-Learning ............................................... 32
   4.9 Discussion ............................................... 33
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>OutSystems Learning Path</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Service Studio</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Embedded Tutorial in Service Studio</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>Task Types Comprised on a Tutorial</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Tutorial Window</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>Tutorial Authoring Tool</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Dreyfus Model Of Skill Acquisition</td>
<td>20</td>
</tr>
<tr>
<td>4.1</td>
<td>Flow State</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Ribbon Hero</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Google Advanced Search</td>
<td>26</td>
</tr>
<tr>
<td>4.4</td>
<td>Ruby Interactive Tutorial</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Intelligent Tutoring System</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>The Lumière Project</td>
<td>31</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison Between Learning Tools</td>
<td>34</td>
</tr>
<tr>
<td>4.8</td>
<td>Comparison Between Learning Tools and Maintenance Costs</td>
<td>35</td>
</tr>
<tr>
<td>6.1</td>
<td>User Modelling and Adaptation Modules</td>
<td>42</td>
</tr>
<tr>
<td>6.2</td>
<td>Generation of Adapted Baseline</td>
<td>43</td>
</tr>
<tr>
<td>6.3</td>
<td>Saving the User Performance</td>
<td>43</td>
</tr>
<tr>
<td>6.4</td>
<td>User Evaluation States</td>
<td>45</td>
</tr>
<tr>
<td>6.5</td>
<td>Increasing Challenge</td>
<td>47</td>
</tr>
<tr>
<td>7.1</td>
<td>Implementation of the User Evaluation States</td>
<td>52</td>
</tr>
<tr>
<td>7.2</td>
<td>Increasing Help System</td>
<td>53</td>
</tr>
<tr>
<td>7.3</td>
<td>Help Event Fired for the First Time</td>
<td>54</td>
</tr>
<tr>
<td>7.4</td>
<td>Help Event After Second Call</td>
<td>54</td>
</tr>
<tr>
<td>7.5</td>
<td>Implemented Lost Event</td>
<td>55</td>
</tr>
<tr>
<td>7.6</td>
<td>Amount of Text to Consider in Reading State</td>
<td>56</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Two Ways to Draw a Line in AutoCAD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
Acronyms

AI  Artificial Intelligence
IT  Information Technology
IDE Integrated Development Environment
ITS Intelligent Tutoring System
UI  User Interface
WYSIWYG What You See Is what You Get
1 Introduction

1.1 Motivation

In the Information Technology (IT) industry, developing a new technology might not be enough for a company to succeed. From the clients’ point of view the investment on new development tools must always consider the training costs. Software learning is usually a tiresome task, involving training courses, documentation, and lots of valuable time, both from the learner, but also for the company developing the product. From this arises the necessity for empowering the users with the skills and tools to learn.

Most of times the overhead of a tool to support and help the user in the learning process does not come from the development cycle, but from the costs of maintaining the content of the learning materials up to date. This is critical in the industry, on which requirements keep changing, technology evolves every day and new features are constantly in development. It is the case of the IT industry, where as a product grows in size and complexity, the creation of content to help the users imposes serious budget restrictions.

With it, one observes two concurrent points of view: the user and the software company, where the first is seeking a way to evolve in expertise and master a given technology or product, and the latter is focused on delivering training methods and tools that accompany a given product, which are intended to aid the user. The development of training materials is costly and as such, the de facto procedure is to develop a minimal viable product which is able to satisfy the user needs while minimizing the impact on the project backlog.

Throughout this document these two points of view are analysed, to contextualize a framework developed to address this problem. It provides a balance between softening the learning curve, while keeping the development and maintenance costs as low as possible, trying to achieve a good return over the investment.

1.2 Problem

Expert and novice users have opposite needs. While the first needs a rich environment, where several features exist and are specialized to solve particular problems (micro approach to problem solving), the novice needs not only a simpler interface, but also a simple process in order to solve a task or a problem (macro approach to problem solving). This implies that is difficult to build software that can find a good balance, and correctly support both ends of the expertise spectrum.

A clear example of products that face this problem are image editing software, 3D modelling tools and Integrated Development Environments (IDE’s). These tools most often deploy cutting edge technology, and are used by experts in their fields. They often present a highly steep learning curve to the novice user.

For products and technology that are inherently complex, it becomes a challenge to support the novice user in the learning process, while keeping all the functionalities of the product. This means that advanced users can be very productive and effective using these tools, but it can take months or years of practice and frustration for a novice user to master it. This implies that each product or tool will present
a learning curve. The learning curve depicts both the initial difficulty of learning how to use a tool, and how much there is to learn after being familiar with that tool.

Even though there are a set of tools and techniques developed to improve the ease of use a product, software training is still a problem to the novice user. The learning path tends to be dispersed; learning materials are not tightly coupled with the software system, which introduces additional complexity to the learning process. Because of these factors, the learning systems and materials present a low retention rate, not being effective at softening the learning curve. Most often, after being exposed to the learning materials, users will perform poorly given a transfer test.

This poses a serious problem, since the learning materials are costly to develop and are not effective at their goals: making the users able to solve a given problem without further need of assistance.

1.3 Goals

This dissertation seeks to develop a framework that delivers high retention rate at a low maintenance cost. This is intended to guide the user, while learning a new technology and empower him to solve the task or problem that he needs in order to advance in the learning curve. Summarizing this dissertation intends to:

- Increase the learner’s retention rate. This means that while keeping the time constant, the learner should be able to absorb more information. This will allow delivering an efficient learning experience without increasing the frustration levels;
- The developed solution should not jeopardize the experience for the higher stages of the learning curve. The focus is in smoothing the learning curve for the novice user, but the users that are already at a higher level in the learning curve cannot be harmed as for example slowing their performance;
- Not to significantly increasing the cost of maintenance after the developed solution is implemented;

1.4 Context

The best way to understand a company point of view is to be integrated in the development cycle of a real product, so that one can closely watch the decisions taken and the motivation behind them. This provides a better insight on the tradeoffs performed when developing new tools and content that help the users to evolve in mastery, or even the simple overhead introduced by keeping the documentation up to date.

This dissertation was developed on an industry aware environment, in collaboration with the research and development team of the OutSystems company. This allowed a better comprehension on the topics discussed throughout this document, since one had the opportunity to watch, discuss and participate in the development cycle of Service Studio, an IDE for a domain specific language.

OutSystems\(^1\) provides a platform that delivers solutions to the IT industry, thus their core product, the Agile Platform, supports full life-cycle development of enterprise Web applications. In order to soften the learning curve of their platform, the Agile Platform deploys an embedded tutorial system. This is relevant because while some other companies choose only to deliver services and keep their platforms in-house, OutSystems has its platform open for use to the general public. With this, there is an increasing responsibility not only to deliver fast solutions when providing services but also to empower the users of their platform to do so.

\(^1\)http://www.outsystems.com/
This opportunity proved critical, since one has became aware of the decision processes in the development of this product. One came to discern what is relevant for the user, but also what concerns the company has when designing new features, in order to have a minimal impact on the project budget.

Also, as part of the development team, one had the opportunity to participate and conduct usability test sessions. This not only proved essential to understand and suggest changes that came from real usability issues, but also to understand how the users learnt and evolved. These sessions were continuously performed during the development of this work, and as such, provided the opportunity for:

- Attend several hours of usability sessions;
- Understand the most frequent problems when interacting with the platform and tutorial system;
- Understand the user expectations and mental models.

1.5 Challenges

To tackle and solve the problems this dissertation addresses, several topics were analyzed, so that an appropriate solution could be developed. Some of the critical challenges that were faced while developing and implementing the solution, can be enumerated:

- Decreasing the mental overload when solving a problem. Given the same problem, the learner should be able to solve it more quickly and present a better solution to that problem;
- The frustration levels of the users cannot increase. One must monitor and validate if the frustration levels presented by the framework are not higher than the deployed solution;
- Since the focus of the product is not on teaching, the solution cannot compromise the efficiency of the product itself. The solution must be small and computationally inexpensive;
- Probing is already deployed in order to gather metrics. This system provides feedback about the interaction of the user with the tutorial system. The implemented solution must be able to maintain the consistency of this feedback loop;
- The system where the solution was implemented is mature and has more than 1 million lines of code. Understanding the architecture and introducing the changes in a system of this dimension is by itself a challenge.

1.6 Contribution

This dissertation presents a framework that can be implemented on any software tool, but targets those who are intended to be used in an industry environment, where the focus is not on the learning process, but the accomplishment of a task with minimal effort and resources, while having a good return on investment over the learning process. So this framework can be implemented into a minimum viable product which is focused on not increasing the project maintenance costs, correctly balancing the user assistance and keeping the user frustration levels low. In order to increase the subject matter retention rates, this framework balances procedural and declarative knowledge. The contributions that this document provides can be summarized as:

- A framework for computer assisted learning to soften the learning curve of a product, increasing learning retention rates;
- This framework combines procedural and declarative knowledge;
• Guidelines to arrange the content of a tutorial to maximize learner attention and avoid frustration. Four task types are presented;

• Guidelines to present Inquisitive Tasks inside the tutoring system. If correctly presented they allow the learner to build and foster mental models that will make him more efficient when problem solving.

More concretely, this dissertation presents a framework for computer assisted learning, which was implemented on an IDE and tested in a real world scenario. It is aimed at helping novice users to smoothly walk their way into a competency [2]. It was developed having in consideration not to jeopardize the overall user experience, and the performance of the platform itself. Although this framework was implemented on top of the Agile Platform, a general framework is described, so that it can be implemented on other software systems that present the same challenges to the novice users. Some of the implemented features that are of special interest to these systems are:

• Development of a tutoring system that evolves according to the user, finding a good balance in the level of assistance provided to the user;

• Development of a system where the user modelling decays with time and events. This allows to provide different adaptation levels. In this particular case, at the beginning no help is given to the learner, and after increasing the time taken for the learner to provide a correct solution, the level of assistance increases;

• Several levels of assistance are incorporated in the developed system;

• The user classification system depends on the learner past experience with the tutorial system. As the user improves in expertise, the assistance provided will decrease;

• Development of a system that automatically takes into account the user reading speed. This allows for the learner to start addressing the problem posed, before receiving any kind of help. This is performed without intervention from the tutorial authors;

• Introduction of Inquisitive Tasks in the tutorial system. This mechanism supports the delivery of declarative knowledge without being intrusive or increasing the frustration;

• The Inquisitive Tasks follow a framework that aims to build high level mind models, instead of low level approach to problems;

• Development of a modelling system which uses a baseline to assess user level, meaning that it can be fine-tuned by the author of the content and be iteratively refined.

This dissertation will not be addressing games, simulations and pure learning systems, because the former are a different study subject per se and the latter only focus in delivering knowledge (e.g. teaching a programming language). While the research for this document reviewed the literature describing the advances in these two areas, these systems pose different problems, thus deserving a separate analysis.

1.7 Outline

This document is organized as follows:

Chapter 2 introduces the problem in more depth. Both the user and the product owner points of view are discussed. Chapter 3 introduces a psychology background. This is relevant to define some terms that will be used through the rest of the dissertation, but will also be the foundation of the developed solution. Chapter 4 provides an overview on the state of the art. Technologies and tools that intend to soften the learning curve for novice users are analysed. Chapter 5 discusses why interactive tutorials
should be used as a starting point to develop the solution. Chapter 6 discusses the developed framework, and chapter 7 explains how the framework was implemented on an IDE. Finally, chapter 9 discusses the results and presents some future work.

Please note that from now on the terms learner and user will be used interchangeably. Since this dissertation addresses the problem posed by software training, every user of a system can be seen as a learner. Every user is trying to find ways to perform new tasks or at least performing them more efficiently.
2.1 Learning A New Development Platform

Perhaps the best way to understand the problem that this dissertation tackles is through a user story. Imagine Dave Lauper as being a typical senior developer. As such he is used to deploy applications using C# or Java. He has several years of experience in the IT industry. Furthermore, he feels that he must become more productive, abstracting himself from the details of implementation. He needs to approach a solution from a business process perspective, not focusing on implementation details. For this reason he will start searching for a tool or technology that will allow him to do so.

After using a search engine to find a product, Dave Lauper will enter the product’s web site. From there he will spend some time exploring the site, in order to understand the features of the product and if they are useful to his development cycle. Generally he will be seeking information about:

- How the product works;
- Who already uses this product;
- How much does it cost;
- How is the development cycle.

When feeling comfortable enough to try the product, Dave Lauper will develop a small project in order to understand what is possible to achieve using that technology. In an inherently complex platform, such as an IDE, Dave Lauper will not be able to proceed very far with his test project, without some additional help. So at some point this user will search for instructions that are able to teach and guide him in the learning process. This can happen on the product web site, the IDE itself, or often using a search engine. Although Dave Lauper is a C# or java proficient user, he is a novice user in this new development platform. At some point, Sean follows the instructions given in a tutorial, video or help file and regardless of his efforts, he does not seem to advance. He understands that he must have done something wrong, and tries again. This second has not produced the expected result. After some time struggling with this system, Sean will feel tired and frustrated, thus deciding to leave and try another tool.

This scenario exemplifies a problem faced by every company that invests in creating tools and technology to assist in the learning process. It intends to allow users to evolve in expertise and master a given product. The problem is finding the correct balance of guidance, but also not introduce high frustration levels. Only this way one can deliver engaging learning materials, that achieve high retention rates. This fine equilibrium can be the difference between getting satisfied customers and new prospects, or users giving up in the middle of the learning process. Most of times new technology could be developed or a greater investment could be made, but most frequently, companies focus primarily on the core product development, improving or developing new features. As such, the smoothness of the learning curve receives less priority on the product backlog.

One can dissect this user story in order to understand both the user and the product owner points of view. This will allow structuring the problem.
2.2 The Product Owner Point Of View

Instead of using an abstract example, and given the opportunity to observe the development of a real product, this experience is presented as a case study. Will be discussed the problems faced by the OutSystems Platform, in order to better understand the problem posed by software training and how one can address it. Although the issues faced by this particular company and its experience are described, these concepts can be easily generalized, because they impact companies that provide complex products, and as such need to develop means to deliver instructional content to their users.

For a company like OutSystems, the focus is on the acquisition and maintenance of customers, and so, they will try to “buy” Dave Lauper’s time a step at a time, forming a customer pipeline. First they will try to buy five minutes with an appealing Web page, then one hour, and then one day. Only if Dave Lauper grants a week of his time, OutSystems will be able to provide a meaningful learning experience. But the fact is that this company cannot take Dave Lauper’s time for granted, and so this learning path must be consistent and appealing, in order to keep this user engaged.

Although OutSystems has a great investment trying to provide the smoothest transition from another technology like C# or Java, to the specific domain language that they provide, every effort invested in the learning process must consider:

- **Immediate development cost** - How complex is the solution and how many man/hours will be needed the implement it;
- **Coupling with product development** - how is the feature or tool coupled with the product. Does it stand in the critical path of the product development?
- **Maintenance costs** - after the project is completed what effort must be made to update it and how hard is to keep track of the changes over time.

Generally, a thorough analysis is performed before the development of a feature or a change is introduced in the learning experience. This analysis is done considering the above vectors in order to understand if it is possible to have a good return over the investment that is software training. In order to soften the learning curve introduced by their domain specific language and IDE, OutSystems has developed a set of tools that assist their users in the learning process.

In figure 2.1 one can observe the tools provided by OutSystems and the user expertise for which they are adequate. In section 3.4 it is discussed the characteristics of the different learning stages.

**Interactive Tutorials** - it is an interactive tutorial system which is embedded in the IDE itself. It is able to teach novice users but falls short when providing assistance to more advanced users;

**Academy** - it is an e-Learning system that comprises videos, quizzes and discussion topics. It is able to assist a novice user until he becomes competent, but is not effective assisting more experienced users;

**Help Reference Files** - are used both by proficient and expert users alike. Since they are highly technical, they only discuss a single feature at a time, and do not provide context, they are not suited to assist less experienced users;

**Community** - is a discussion forum and as such, the topics are dispersed and most of the active users is already experienced, it will present a barrier to the less experienced ones. Novice users will feel lost with the amount of information comprised on it.
2.2. THE PRODUCT OWNER POINT OF VIEW

OutSystems Embedded Tutorial System

In order to address the steepness of the learning curve within their platform, OutSystems developed an embedded tutorial system. This tutorial system intends to replace text files that served as tutorials in the past. The interactive tutorials were a way devised to accommodate a “learn by doing” approach, and were developed in order to let the learning process being done within the platform itself. This framework was designed having the novice learner in consideration, since it comprises a set of tutorials that users can use and practice within the IDE, that is the first tool of the platform that they interacted with.

In order to provide a better illustration of this system, one can consult images 2.2 and 2.3. While in image 2.3 the user is doing a tutorial, the IDE environment remains the same. In fact it is possible to exit the tutorial and extend the application built in it. This not only allows the user to learn inside the IDE, but it also presents low maintenance costs, since the tutorial uses a meta-language that is able to refer interface elements of the IDE, even if changes are made to them. This means that even if the whole user interface of the IDE is changed, the tutorial would still work correctly.

In figure 2.2 one can observe the IDE of the Agile Platform, Service Studio. It is comprised of the following elements:

**Tools Pane (1)** - this pane allows the user to compile, publish and perform other useful operations upon an application;

**Tools Tree (2)** - contains the elements that can be added to a flow;

**Flow Pane (3)** - canvas where the user can design the interaction between screens and actions;

**eSpace Tree (4)** - displays all the elements that comprise the application;

**Bottom Pane (5)** - shows warnings and errors in the application. It also presents debugging information, if the user is in a debug session;

**Properties Pane (6)** - allows changing the properties of the selected element.

After understanding the several elements that comprise the IDE, a typical tutorial step is presented. As can be seen in figure 2.3, a tutorial is comprised of:
CHAPTER 2. DETAILING THE PROBLEM

Figure 2.2: Service Studio

Figure 2.3: Embedded Tutorial in Service Studio

**Tutorial Window (1)** - window that provides the instructions for the user to advance in the tutorial;

**Tutorial Arrow (2)** - points to the element that the user needs to interact with, in order to advance.

After a user starts a tutorial, he will need to perform several tasks, interacting with several interface elements, so that he can proceed in the tutorial. A tutorial will typically teach how to build or extend an application using Service Studio. In order combine procedural and declarative knowledge, it is possible to integrate two different kinds of tasks in a tutorial: *action tasks* and *informative tasks*.

While action tasks (figure 2.4a) comprise several actions, which must be completed by the user, informative tasks (figure 2.4b) only requires reading. The latter are used as a summary or to explain in depth a certain topic, providing declarative knowledge. Most of times an action task will be comprised of several tasks in order to allow the user to understand the relation among these several tasks. Other times, the tutorial will just require the user to read some explanatory text. After he is done, the user will click a button in order to advance.

As stated, an action task can contain tasks. So it is relevant to describe the elements that compose the tutorial window itself.
2.2. THE PRODUCT OWNER POINT OF VIEW

(a) An Action Task

![Image of task types]

(b) An Informative Task

Figure 2.4: Task Types Comprised on a Tutorial

In figure 2.5 are detailed the several elements inside a tutorial window. They will provide instructions to be followed by the user, and also allow the navigation between the several steps that comprise the tutorial. The user can skip steps, which is useful if for some reason he is not able to perform the necessary actions.

**Task Title (1)** - presents the title of the current task;

**Introductory Text (2)** - explains what the user will be doing in the current set of tasks displayed;

**Tasks (3)** - it is an action or set of actions that must be accomplished by the user in order to advance in the task;

**Current Task (4)** - the task that the user is currently at;

**Steps Button (5)** - this button allows to navigate between all the steps within the tutorial;

**Help Button (6)** - this button provides help according to the task the user is currently at.

As previously stated, the embedded tutorial is specified in a meta-language. This not only allows the tutorials being up to date even when changes are made the IDE, but it also means that the tutorials can be authored inside the IDE itself.

In figure 2.6 can be observed that while the user is still on the IDE, new options are available, which allows building and authoring a tutorial in a What You See Is What You Get way. This environment will show the screen as will be rendered for the user, but it also provides a way to test the tutorial system. Only with this second option is it possible to preview the arrow that points to the elements in the interface. This is because, certain elements which are pointed by the arrow, only exist at run-time.

The tutorial authoring environment has:
CHAPTER 2. DETAILING THE PROBLEM

Figure 2.5: Tutorial Window

Tools Pane (1) - this pane allows undo, redo, open files and other operations;

Tools Tree (2) - contains the elements that can be added to a tutorial flow;

Flow Pane (3) - canvas where the user can design the interaction between steps in a tutorial;

eSpace Tree (4) - shows all the tutorials, as well as the steps and tasks inside tutorials;

Bottom Pane (5) - shows warnings and errors while developing a tutorial;

Properties Pane (6) - allows to change the properties of the selected element, such as a tutorial, step or task.

It is important mentioning that this tutorial system is deployed in the platform for more than two years from the realization of this dissertation. This means that one had the opportunity to see and conduct tens of hours of usability tests, in which participants had to perform a transfer test after being exposed to a given tutorial.

2.3 The User Point Of View

Dave Lauper is a persona created to illustrate certain attitudes, behaviour and concerns of a certain set of users. In this particular case the concerns of a senior developer. This particular kind of user has several years of experience in the IT industry and, as such, has already dealt with a vast array of technologies. This is important because unlike an inexperienced user, this one has seen the advantages and disadvantages of the industry’s technologies and has both expectations and previous knowledge to work upon.

As will be discussed, having or not previous knowledge about a certain domain or topic has an impact on the overall learning experience. As will be shown, having previous knowledge decreases mental overload when performing a given task [3; 4]. Let us suppose that Dave Lauper is interested in changing the technology that he or his team works with, searching for a product that is able to make them more productive. Dave Lauper knows that most of times, there is a trade-off that must be made in order to be more productive: approach the problems with an high abstraction level, in order to develop solutions at a business process level, instead of approaching them from a technology constrained or algorithm specific perspective.
2.3. THE USER POINT OF VIEW

This trade-off allows the implementation of patterns instead of algorithms, implying that sometimes it will be impossible to access the depths of the platform and manipulate its logic. As a senior developer, Dave Lauper is comfortable with this compromise, since being more productive is his main concern. So Dave Lauper will evaluate the industry offers according to his needs. If there are several products that provide similar sets of features, Dave Lauper will consider several factors in order to choose between them. This issue is discussed on Laudon [5, p. 224], which can be summarized as:

1. **Visibility of the product** - how easy it is to find it;
2. **Market dominance** - how is the product established in the market, what companies are also using it, and product maturity level;
3. **Costs** - direct costs as maintenance and indirect costs as productivity;
   (a) **Learning Associated Costs** - how long it takes to become proficient and start being productive;

In his search for the correct platform or technology, Dave Lauper will perform two distinct use cases: evaluation and learning. The first involves making an analysis of the product and its features, and the latter a more in depth immersion in the learning process. In this dissertation, the focus will be on the latter. In particular the low retention rates presented by the embedded tutorial system, which is deployed on the Agile Platform. This is a problem for Dave Lauper, because if the training materials do not have an high retention rate, this means that his team will have to spend more time learning and practicing before they start being productive. More concretely, Dave Lauper noticed that while the embedded tutorial system is able to assist in the exploration of the Agile Platform, it also does not provide any challenges. As such, a novice user is able to complete all the tutorials by blindly following the arrow, and interacting with the elements which it points to. This means that while Dave Lauper is able to easily complete a tutorial, giving him a sense of accomplishment, when he tries to perform a similar task outside the tutorial environment, he finds out that he is not able to do so. Since he was so tightly guided during the tutorial, he acquired some knowledge during the realization of the tutorial, but was not able to retain it. As will be discussed in chapter 3, there are three steps in the learning process: acquisition, retention and retrieval. If any of those steps is not correctly performed, the learning process becomes jeopardized, and the learner will perform badly in a transfer test.
CHAPTER 2. DETAILING THE PROBLEM

The problem is that while the embedded tutorials delivered to the user a good acquisition step, the same was not true to the retention. This was observed during usability tests, in which participants were asked to perform a transfer test, after experiencing a tutorial. Further confirmation of this fact, came from the Developers’ Bootcamp (a face to face class). The attendees of this class needed to perform fifteen tutorials and solve a set of problems in order to attend the class. When asked about what they felt about the tutorial system, a significant number (four participants out of six) replied that in order to solve the problems they had stopped using the tutorial system. They explained that since the tutorial provided excessive guidance, it became easy to finish a tutorial, but one had to do the same tutorial several times in order to learn from it. These users even stated that in order to solve the problems, they had been using deprecated tutorial files, which were non-interactive media (a set of text files that provided instructions). These files are presented in annex A.4.

This is interesting, given that the interactive tutorials were developed to accommodate the same learning subjects as the outdated instructional files. As these users wanted to evolve in the learning curve, they were willingly to burden the difficulty of mapping the lesson presented in the files and mimicking the actions on the IDE. This pointed to the fact that some users do not consider the interactive tutorials as being a good learning alternative, since the retention rates were low.

One of the observations done, was that some users started to solve the tutorial without even reading it. After observation, was concluded that users would scan the text in search of meaningful actions to perform. The user would scan for such keywords as:

- Click;
- Double Click;
- Drag.

While this can be a good user experience, because a tutorial can be accomplished almost without reading, the fact is that it negatively impacts the learning experience. This is mainly because the learner was able to follow instructions but did not understood how those actions impacted the overall process.

Summarizing, this dissertation addresses the problem posed by software training that delivers low retention rates. Although the particular problem of the Agile Platform is discussed, this is only an instance of a problem faced by other tools which present a steep learning curve. The developed framework is intended to address:

- The low retention rates presented by software training;
- This framework should not increase the user frustration levels;
- This framework must consider the more experienced users. They should not be less productive with the introduction of this framework;

An analysis on the training process will be made, having in consideration a psychology background, so that later one can survey the state of the art having those topics in consideration.
Before an analysis on the tools and concepts that address the same problems that this dissertation
does, it becomes relevant to understand how learning is processed. By doing so, some concepts will be
defined. These definitions will prove useful throughout the rest of this document.

One can state that learning a concept involves three distinct steps[6][7]:

1. **Acquisition** - the step in which the learner is exposed to the learning materials;
2. **Retention** - what can be denominated as “storing the concepts”;
3. **Retrieval** - the step in which the learner finds the correct knowledge within the memory. It makes
   the learner able to start addressing a problem.

Failure to remember means failure at performing at least one of the steps described above. Successful
learning is not only accomplished by the retrieval of the correct information. It is also relevant to know
how to apply that knowledge to a given situation in order to perform a given task. Hence, one can state
that learning is successful once all three steps are performed successfully, and the learner has successfully
applied the new knowledge to solve a task.

It is important to notice that retrieval is not necessarily remembrance. Most of times one has not
been exposed to a solution of a particular problem, anyhow is able to solve it; this is discussed by Sweller
et al on [8]. So, most of our day to day problem solving is not accomplished through accessing the memory
and retrieving some exact experience. Instead, patterns can be identified that allow to infer upon past
experiences in order to successfully solve a new instance of a problem that is already known to us. This is
what is known as **inference**. Since this dissertation addresses the problems posed by software training,
the developed solution focus on efficiently delivery these three steps to the learner. But for inherently
complex systems, it is not sufficient for the learner to be exposed to the learning materials. The learner
must also be active in the process, using new knowledge and solving new instances of the same problems.
This allows him to practice problem solving, which stimulates inference.

Let us now distinguish two different types of knowledge[7]:

- **Declarative** (propositional or descriptive) knowledge, is to know how two or more concepts are
  related. E.g. “Primary keys provide an unique identifier to a row in a table”;

- **Procedural** (imperative) knowledge, is to know how to achieve a given task or solve some specific
  problem. Most of times, this kind of knowledge is difficult to be expressed by an expert. E.g. “How
to deploy a Web application”.

Distinguishing these two knowledge types is relevant to address the problem, because both must be
mastered in order for the user of the software to be able to successfully accomplish his goal (e.g. building
a Web application using an IDE). It is not sufficient to only deliver one kind of knowledge to the learner.
If the user was only exposed to declarative knowledge, he would not be able to efficiently solve a problem
such as deploying a Web application. This would be because although he would possess knowledge about
facts, he would not know how to efficiently combine that knowledge into a practical solution. On the
other hand, if only procedural knowledge was delivered, he would know where the relevant elements were on the interface, but would not have the knowledge to use those features to build an application, nor would be able to solve a different instance of the same problem.

If these two types of knowledge are not experienced in an integrated way, the acquisition, retention, retrieval cycle can be compromised. So the user could face difficulties while problem solving. To correctly address this issue, the learning subject must be adequately broken into appropriate frameworks that deliver both procedural and declarative knowledge in an integrated way. As previously stated, achieving a balance of procedural and declarative knowledge is one of the main goals of this dissertation.

Having discussed how the learner acquires new knowledge and the different knowledge types, one can define concepts and terminology that will assist in the development of a framework to address the problems discussed.

3.1 “Signifiers”

One can state that “signifiers” (or perceived affordances) are information spread over the surrounding environment. As such, they will be relevant in the acquisition step. According to this concept when one observes a button on a user interface, more knowledge than just its color, shapes and textures is gathered; exemplifying one also comprehend that it affords pushing. This concept was first presented by J.J. Gibson in [9]. Later Donald Norman extended this idea to software and design. It was first presented in [10], and a few concepts (like the word signifier itself) were later introduced in [1].

The problem is that sometimes “signifiers” are misleading, because there is a gap between the perceived affordance and the real one. This is common in graphical interfaces, when one perceives that a button affords pushing when in fact it does not, or will not respond in the way the user was expecting. The above example leads to faulty mental models. Since the learner will acquire wrong mental models, the retention step becomes compromised. This implies that when designing learning systems, one should be careful to minimize these gaps. It is essential to provide a correct and concise mental model to the learner, so that he is able to create schemata thus being able to apply that knowledge to other domains.

3.2 Schemata

Schemata are cognitive structures that organize elements of information according to the way they will be dealt. This is the key part of retention and retrieval. This topic is discussed by Donald Norman in [6] and by Paas and Sweller in [4]. Schemata are what allow humans to generalize a concept. A simple example is that after observing several different “A” letters, one can identify almost any character that resembles an “A” (even hand-written). This is possible because one has built schemata that stores the properties of what an “A” should have and generalize it. One will never need to see all the instances of a class to make a generalization. This is interesting, because when new information is obtained and it fits previous built schemata, it can be easily remembered, but in turn, if the new knowledge does not fit any schema, it can be ignored, change an existing schema or force the construction of new schemata.

This is relevant since schemata acquisition leads to automation (the cognitive processing starts being done at a subconscious level) hence reducing working memory load [4; 8; 11]. Having previous schemata allows an expert user to perform problem solving, not focusing on details. Those details are processed subconsciously, reducing the memory load and consequently, the attention needed to solve the problem. Also, in software training, schemata acquisition is done through the stimulation of both declarative and procedural knowledge. It is important to notice that schemata are easily acquired if context switching is avoided. This is because context switching introduces extraneous processing.

Cognitive processing is finite, and as such, only a limited amount of processing will be useful to the learning process. Let us distinguish the three different types of processing made, when considering the learning process presented by Mayer et al in [7]:
3.3. SCRIPTS

- **Essential Processing** (intrinsic), in which the learner is processing the study material and the processing requirements are related with intrinsic properties of the learning elements;

- **Generative processing**, is cognitive processing aimed at deeper understanding the concepts that were exposed, and basically this involves building or changing schemata.

- **Extraneous processing**, which is cognitive processing that does not support the instructional objective. This is mainly done because the learner is processing something else (e.g. flipping pages in a book trying to map concepts). Extraneous processing does not support learning in the sense that it does not lead to schemata acquisition;

In order to develop a framework that delivers high retention rates in software training, one needs to minimize extraneous processing. A few factors that increase extraneous processing are materials that grasp the learner attention even though are not relevant to the learning process, materials presented in a confusing way, or misleading “signifiers”.

3.3 Scripts

As discussed above, learning is only successful if the learner is able to apply the learnt knowledge. While schemata allow us to store generalized knowledge, scripts are a set of actions that one can use in order to solve a particular class of problems. When a problem is found, the first step performed is trying to find which class of problems does it belongs to. If one are able to identify it (find the correct schema), then set of generalized actions (script) will be retrieved, which will provide guidelines, helping to solve, or at least to tackle the specific problem. This implies that the developed framework should not only deliver new knowledge, but also be presented in such a way that the learner is stimulated to generalize it, building scripts that make him able to solve new instances of the problems. Scripts will allow the learner to become independent of the training materials, being able to solve problems outside the learning environment.

Having the above factors in consideration, the developed framework to must:

- Use appropriate “signifiers”. This allows for a better understanding of the visual elements thus reducing extraneous processing;

- Provide integrated declarative and procedural knowledge. This will allow not only to keep the frustration levels low, the user engaged but also will stimulate schemata acquisition;

- Provide repetition and opportunities for the user to think and solve the problems with the lowest assistance possible. This will allow the stimulation of the acquisition-retention cycle, leading to script acquisition.

Having described the elements that compose the learning process and how one can use them to provide high retention software training, let us now evaluate the difference between the several stages of the learning expertise.

3.4 Dreyfus Model of Skill Acquisition

Now it is useful to model the different stages of learning, according to the constructed schemata and make a distinction of the several properties that separate the several learning stages. Scientific literature presents several models that make this distinction, but this dissertation adopts what has been known as the Dreyfus model of skill acquisition, which is described on [2; 12].
Figure 3.1: Dreyfus Model Of Skill Acquisition

According to this model, what distinguishes the several expertise levels is the increasingly capability of the advanced levels to execute a task at subconscious level. It makes him able to act more smoothly and performing other tasks (read cognitive processes) at the same time. Also the point of view when tackling a problem and solving it, is different (i.e. an expert does not think of performing a step by step approach to solve a problem. Instead he takes a global approach, understanding patterns that can be implemented to solve that problem).

- **Novice** - needs to have a decision tree to choose what to do in order to perform a task. As they do not have expertise, they might not know which rules apply to a particular situation, and have little situational judgement;

- **Advanced beginner** - in this stage the learner is starting to break away from fixed rules and exploring new ways to do the task, but probably they will have difficulties acting on their own. The learner gives the several attributes of the task the same level of importance and attention;

- **Competent** - the learner is starting to build mental models and take advantage of advices given by more experienced users. He is able to understand how the performed actions fit on the long-term plan;

- **Proficient** - at this stage the learner is able to observe deviations from the normal patterns. He understands which attributes of a task deserve more attention, and decision making becomes easy when faced with different options. Also, learners in this stage have already addressed some issues that decreased performance when problem solving;

- **Expert** - this category is continually looking for ways to improve and better ways to accomplish a task. As part of their experience, they have developed gut feelings and can understand the difference between irrelevant details and crucial ones.

Knowing the learners’ stage within this model is relevant to provide a better learning experience and keep the learner engaged in the learning process. The goal of this dissertation is providing high retention software training to users that are at the bottom left of the learning experience (novices) in the Agile Platform, but at the middle-top of the learning experience (competent - experts) in software development with C#, Java or similar technologies. In order to develop a framework that tackles this problem, let us review the state of the art on software training, and understand their adequacy to the several stages of the learning expertise. The next chapter surveys the state of the art regarding software training. Both academic and industry related research will be discussed.
Related Work: Learning Tools

Now it becomes relevant to review what approaches have been made in order to soften the learning curve, and increase software training retention levels. Since this dissertation focuses on computer-assisted learning, only tools and technology that fall in this category will be discussed.

4.1 Non-Interactive Media

Books, instructional materials, and other non-interactive media can provide help when learning a new subject or evolving existing knowledge. An example is almost any book from the IT shelf in a library or a help reference file that ships with any software product.

Some of the main features of this approach to software training are:

- **Searchable** - allows to search for content, meaning that it will be easy to find help about a given feature, among several others. One example is the help reference from TortoiseSVN\(^1\), a subversion client.

- **Linked with the product** - Presents some kind of connection with the product. One implementation can be pressing the “F1” key makes a context-aware page to open with the help file related to the screen element the user was selecting. An example can be Microsoft Office Excel\(^2\). Writing a function and pressing “F1” will provide help on that particular function.

- **Detailed** - These tools are highly accurate and extensive, providing information almost about every feature implemented. This proves valuable, especially for advanced users. One example is Microsoft’s web page for development reference\(^3\). Another example is Unity script reference\(^4\).

One can divide non-interactive media into two distinct categories: **continuous** and **discrete**. The first is developed to incrementally support the learner. The latter is only for reference and is intended to explain in depth a feature in a self-contained way, but does not provide high-level context to the learner.

**Continuous Non-Interactive Media**

These systems support the learner evolution in the learning curve. An example of continuous non-interactive media can be tutorial books or online instructions. They make a soft transition between the topics discussed, allowing the learner to get a sense of where a particular element fits in the overall process. It is relevant to present some real-world implementations, for a better illustration. To learn Unity (a game development tool) the learner can extend a game by following instructions\(^5\). To learn Blender (a

\(^1\)http://tortoisessvn.net/
\(^2\)http://office.microsoft.com/
\(^3\)http://msdn.microsoft.com/en-us/library/
\(^4\)http://unity3d.com/support/documentation/ScriptReference/
\(^5\)http://unity3d.com/support/resources/tutorials/3d-platform-game
CHAPTER 4. RELATED WORK: LEARNING TOOLS

3D modelling tool) the learner can follow instructions presented in files that can be downloaded\(^6\). These documents are developed to help users evolve in expertise, starting from a novice stage. To provide context to the learner, these books contain screenshots, schematics and other illustrations. Another good aspect of these systems is that they are well structured and indexed. This way the learner is able to navigate across sections and understand how they relate to each other. The disadvantage of these tools is that they force the learner to shift the attention between tutorial (simulated environment) and tool (real environment). This increases extraneous processing. Since non-interactive media is approached in a passive way, it becomes difficult to provide opportunities to make the learner active in the learning process. While a book can be enough to transmit declarative knowledge, the same is not true for procedural knowledge.

Discrete Non-Interactive Media

On the other side, discrete non-interactive media, focus on explaining every feature in a self-contained way. It can be hard for the learner to grasp high level concepts from this kind of tools, since they provide few context. The best example of discrete non-interactive media are help reference books and files. These tools are mostly used by advanced users, since they might need to consult details of implementations or other small characteristics in order to explore to the maximum the potentials of the platform. The best example of discrete non-interactive media are the help reference files contained in any software product.

Help Reference

These help systems are devised to assist the user with properties and descriptions of that screen element or feature. While it is a great way to present detailed information about properties and functionalities of the various tools and screen elements, it is not adequate to assist the novice user. This is mainly because a novice user cannot understand how a particular element fits in an overall process.

4.2 Video Tutorials

Video tutorials are a mean devised to provide context to the learner. By watching the procedure instead of just reading about it, the learner receives richer information. This brings him closer to a one-on-one learning experience. There are wide spread examples. If a learner wants to learn the complex interface of 3D Studio Max (another 3D modelling software), he can watch video tutorials provided in the product web site\(^7\). To learn Adobe Photoshop (an image editing software) the company provides a site with several video tutorials\(^8\). And to learn Unity one can also find video tutorials in the product web site\(^9\).

With higher bandwidth this learning method has been spreading and is probably one of the most used within novice users, because it is direct - the video provides context and almost no previous knowledge is required. This learning method is becoming increasingly accessible to the general public. With a network connection is possible to have access to free video tutorials done by other learners or even experts. This contrasts with books that are costly and have too much content, but lack context. This helps learners to accomplish specific tasks, which is rewarding.

There are two approaches while learning using these tools:

- **Step by step** - where the learner stops the video every step and tries to mimic it, on the platform;

\(^6\)http://gryllus.net/Blender/3D.html
\(^7\)http://download.autodesk.com/us/3dsmax/skillmoviesv2011
\(^8\)http://tv.adobe.com/
\(^9\)http://unity3d.com/support/documentation/video/
4.3 DISCUSSION FORUMS

- **Observe and try** - where the learner watches the whole video and at the end tries to accomplish the same task, or apply that knowledge to another instance of the problem.

These two approaches do not necessarily harm the learning experience. Literature has shown that if neither of them overburdens the working memory, the learner is able to correctly apply the new knowledge afterwards [13]. Another argument supporting video tutorials is that one can consider it the same as being exposed to a worked example (watching a problem being solved step by step). There are arguments in the literature pointing to the fact that worked examples can be most adequate for novices if intrinsic processing is high [14].

The main problem with these tools is that they do not imply generative process. The learner can be passive in the learning process, achieving a task through imitation thus not internally processing it in order to generate new knowledge. This is crucial for the learning process.

4.3 Discussion Forums

From the beginning of the Internet and more recently with the boom of Web 2.0, groups of people have united to share knowledge. This is the emergence of communities dedicated to helping each other, fostering new knowledge, and sharing within a community of learners. Some examples are forums where users exchange knowledge about the Java programming language\(^{10}\), and communities built to perform questioning and answering (Q&A) about software development\(^ {11}\). Companies have been increasingly adopting this technology. To pose questions about Blender to users there is a community\(^ {12}\). In order to exchange knowledge with other Unity developers, the company has a forum\(^ {13}\).

These tools are great in addressing *ad-hoc* problems. They are able to answer questions that were not previously identified by the product developers. Also, the content can be updated by the users themselves, which will provide richer content and at the same time, ease the maintenance cost for the product developers. Since the content is constantly updated, instrumentation techniques can be used to identify the common problems both in the training process, as well as on the product itself. This can provide a feedback loop that can positively impact the product quality.

On the downside, since these tools have emergent properties, it becomes hard and costly to maintain the redundant content. Also these tools tend to be avoided by the novice user. They often feel that these communities are used by experts and will be reprimanded in case their question is seen as trivial, hence neither participating actively nor posting their questions. In order to deal with these issues, companies often assign resources dedicated to moderate the discussions and to ensure the stability of the community.

4.4 “Gamification”

One can observe that most part of games are basically a structured framework where the gamer is taught how to smoothly master a set of skills that will allow him to overcome increasingly difficult challenges. This means that games, at their core, have a learning curve and the gamer will evolve in expertise. So game designers have been carefully tailoring the user experience and finding an equilibrium between mastery and frustration since the first games were developed.

“Gamification” is an approach that tries to apply game mechanics to non-game applications in order to teach behaviours to the users. For a clarification of this term please refer to [15]. Some of the mechanics introduced in these systems are: achievements, levels, points, rewards and so on. This

\(^{10}\)http://www.javaranch.com/

\(^{11}\)http://stackoverflow.com/

\(^{12}\)http://www.blender.org/community/user-community/

\(^{13}\)http://forum.unity3d.com/
feedback loop intends to keep the user engaged. This way it increases the probability that the user will stay more time within the system. Usually this approach presents the learning materials in a slower pace. This way, it is able to deliver the knowledge and practice that is needed to master a system, without frustrating the learner. Industry implementations of these tools are still difficult to find. While there are companies that develop experiences or products around this concept, is difficult to find an implementation of a “gamification” system that supports the learning curve of a complex product such as an IDE, 3D modelling tool or image editing software.

These tools are able to immerse the learner, thus making him spend more time in contact with the learning system. Trough game mechanics, repetition of tasks is stimulated. While typically the learner does not like to repeat a task, “gamification” focus on developing approaches that stimulate repetitive tasks. With the introduction of game mechanics, repetitive tasks become challenges and are able to increase the interest of the learner. Another positive factor is that these tools often deploy a scoring system, so the learner will be competing not only with other learners, but with himself. This is able to provide short term goals and increase the learner motivation to evolve in the learning curve.

As the learner will be competing, this can also negatively impact the learning process. One example is learners that will cheat in order to easily achieve a higher score or collect some reward. To deal with this issue, developers need to constantly monitor and adjust the system in order to not incentive this kind of behaviour. Also on the downside is the fact that the learning content will be expensive to develop since it must be closely coupled with the game mechanics.

**Ribbon Hero**

Ribbon Hero is a plug-in for Microsoft Office suite that helps the user evolution through “gamification”. Its main goal is to teach a set of skills that the learner can use in the real world environment (in this particular case within the Microsoft Office suite).

All the learning processes are developed around the concept of earning points, while trying to keep the learner in a flow state. This state is described by Csikszentmihaly on [16] and summarized in figure 4.1.

![Flow State](image)

**Figure 4.1: Flow State**

For the Microsoft Office Word, there are four different categories, each one containing several different challenges:

- Working with text;
- Page design and layout;
4.4. “GAMIFICATION”

- Getting artistic;
- Quick points.

This last category is developed to stimulate constant practice of the set of skills learnt in the other categories. In order to earn points in this category the learner must practice outside the learning environment. When the learner uses a feature while working in a document, he will be notified that some points were earned.

![Figure 4.2: Ribbon Hero](image)

One of the central points in Ribbon Hero is the fact that the learner can monitor the progress at all times through a pointing system. This provides a feedback loop that incentives the learner to be active in the learning process. This provides a sense of accomplishment every time the learner sees the progress bar filling as a task is completed.

Each challenge has a set of clues that are intended to assist the learner, in case he is not able to advance in the task. They are presented at the right pane in figure 4.2. These hints are direct, meaning that if the user needs to use three UI (User Interface) elements in order to complete a task. A hint will direct the user to one of the elements that he needs to interact with in order to advance.

After the completion of a task the user is encouraged to perform the same task but with a higher challenge. Some of the sentences used are:

- “Can you do it without any hints?”;
- “Reuse what you just learnt within the next couple of days to receive an even higher score.”;
- “Can you do it faster?”;
- “Can you do it in fewer steps?”.

This system is event-based and there is no strict sequence that needs to be followed in order to complete a task composed of several steps. This is a design decision that simplifies the challenge for the learner. Doing so does not necessarily introduces incorrect mental models, since most sets of basic tasks in Microsoft Word can be used interchangeably. This means that:
{ changing a text to bold, changing font to arial } ⇔ { changing font to arial, changing a text to bold}
4.5 Abstraction Layer

Another approach to software training is the introduction of an abstraction layer that can be used by novice users. In this approach, there are at least two ways to solve the same problem. One path is developed to be used by novice users and other for more advanced ones. What is important for this research is that these systems provide clear feedback to the novice users on how to perform the same action using the tools or mechanisms that an expert would use. Since it is non-intrusive, at first the user might not notice this feedback loop. But as time passes, the users understand and appreciate this system. Succinctly, successful abstraction layers are those who stop being used as the user evolves in expertise. This provides a smooth way of helping users evolve according to the Dreyfus Model.

These tools provide integrated support for novice and experts. It teaches using a “learn by example” paradigm. In these systems, novices are provided with feedback that shows how they can be more productive and evolve. On the downside, since these tools are highly coupled with the product, thus being in the critical path of the product development. Also almost no content is used, that can be consulted by the learner. If the learner wants to know more about a feature, he will have to use the help reference files.

Below two implementations of this technique are presented.

Google Advanced Search

Even though Google’s web site is fairly simple, it allows the users to do some tasks that are not trivial, if the user is novice. One example of this is the use of reserved keywords to modify the parameters of the search using boolean logic. This can be used in case the user wants to make the conjunction or disjunction of two different keywords on his search. Google found a way to balance the use of this feature by both novice and expert users. The solution was to let experts use the reserved keywords on a typical search, and introduce an abstraction layer for novice ones.

This abstract layer is the Google Advanced Search. The introduction of this layer allows novice users to specify complex search behaviours in a simple way, by using a form so that the user can specify the keywords in the proper input field that has a description of the produced behaviour. With it, a novice user is able to perform the same query as an expert, without having to know or memorize reserved keywords or specific syntax.

In figure 4.3 can be observed:

- Final Query (1) - the user can observe the query that will be made in the search engine;
- Arguments (2) - the user can insert words in these fields, to build a query;
- Additional Information (3) - a brief description about the functionality is provided.
4.6. INTERACTIVE TUTORIALS

While the user inputs keywords in the form, at the top of the page is displayed the query that is being built. This way the user can start to learn by example. This is particularly important, because instead of being given a generic example of a query, the user will be able to see their specific query as a learning example. At first sight this tool is very simplistic, but the fact is that it enables novice users to evolve according to the Dreyfus Model at performing queries within Google’s search engine.

AutoCAD

AutoCAD is a software application for both 2D and 3D design. This software provides its users with typically two or three ways to accomplish the same task. The relevant topic for learning is that the novice user interacts more with the UI than the expert. While the novice searches for cues on the environment in order to solve problems, the expert is able to use shortcuts and combinations of commands that allow him to be more efficient. To support both ends of the Dreyfus Model, and assisting the novice user evolution, this software displays all the commands in a command line, even when the user interacts with the UI.

Let us analyse two ways of making a line from point $(0,0)$ to point $(100,100)$ in this software tool.

<table>
<thead>
<tr>
<th>Graphical User Interface</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the line tool</td>
<td>LINE (or L)</td>
</tr>
<tr>
<td>In point $(0,0)$ of the canvas perform a left click using the mouse</td>
<td>0,0</td>
</tr>
<tr>
<td>In point $(100,100)$ of the canvas, perform a left click with the mouse</td>
<td>100,100</td>
</tr>
</tbody>
</table>

Table 4.1: Two Ways to Draw a Line in AutoCAD

Even though the number of interactions with the system is the same, the execution of the command through the command line will take less time to be accomplished because it is more difficult and time consuming to find the point $(x,y)$ with the mouse. It becomes important to support the user’s evolution so that he can become increasingly effective and efficient in the use of these commands. The way that AutoCAD accomplishes this, is by showing in the command line all the equivalent commands, while the user executes them through the graphical interface. So the tools palette is just an abstraction layer that tends to be less used as the user evolves.

4.6 Interactive Tutorials

Another approach to soften the learning curve for newcomers are interactive tutorials. This way a user can be guided while performing a task. These tutorial systems can be embedded on the product itself, providing a guided environment for the learner to evolve.

As they are integrated with the product, it decreases the gap between learning environment and real one. This minimizes extraneous load, since almost no mapping must be done. They also present a good balance of procedural and declarative knowledge, meaning that the learner will have the opportunity to be active in the learning process. On the downside, these systems are not adequate for expert users, since it will not be able to explain in detail the topics covered. They usually focus on the first levels of the learning curve. Since these tools are highly coupled with the product, their development and maintenance impacts the development cycle of the product. And since most of these tutorials are scripted, this means that they can constrain the actions made by the learner. The learner sometimes will not be able to explore at will.
Try Ruby

Try Ruby\textsuperscript{14} is a web site that provides an interactive Ruby\textsuperscript{15} tutorial. In this site the user can find a typical console, where all the features and built-in methods of Ruby (release 1.9.2) are available. In addition, there are a few special commands that allow the user to interact with the tutorial, for example starting or jumping to the next step. This is a step by step tutorial, and as such, the user must only write one expression in order to make the tutorial advance. The first few steps are intended to teach some syntactic elements of the language. To teach this, the tutorial is very flexible and as long as the user inserts the required method, the tutorial will move forward.

One example of this feature is presented on the first step of the tutorial. It instructs the user to try inserting “2+6” in the console. As the important lesson is that the user learns how to use the addition operation, the tutorial will move forward as long as the user inputs an addition expression $<$int$>$ + $<$int$>$.

![Try Ruby Tutorial](image4.4a.png) ![Try Ruby Summary](image4.4b.png)

Figure 4.4: Ruby Interactive Tutorial

For every tutorial step, the user is presented with an explanatory text and a new expression to be inserted. A step of this tutorial is presented in image 4.4a.

One important feature is the way the instructional text is presented. The command or method that is relevant to the step is highlighted using a red background color. This way, the learner can scan the text for these keywords. Close to these highlighted keywords there are explanations about them. Also highlighted, are the commands that the learner must insert to advance in the tutorial. These are presented in a white background color.

This provides that even if the user does not pay the deserved attention to the explanatory text, those keywords will be instantly noticed.

Another feature is that after the completion of several steps, a summary is presented to the user. An example is shown in figure 4.4b, that reviews the learnt syntax so far and gives a concise example of where that command can be used to solve a real problem. One example is that when the tutorial teaches the “length” method, after inserting an expression with this method on the console, the user is told:

\textsuperscript{14}http://tryruby.org/ \textsuperscript{15}http://ruby-lang.org/about
4.7 INTELLIGENT TUTORING SYSTEMS

“Well, I’m sure you’ve been to a website that screamed, Hey, your password is too short! See, some programs use this simple code.” (referring to the use of the “length” method).

This tutorial provides guided steps, while the learner is inside a fully functional Ruby console. This means that even if the user is in a tutorial, all the behaviour and state of the environment is saved and modified as if in a regular Ruby environment. So in fact this tutorial is also an abstraction layer that helps guiding the user exploring the language. Even when inside the tutorial, the user can explore and develop software with no restrictions.

4.7 Intelligent Tutoring Systems

In the past years developments were made in an area that addresses some of the problems that this dissertation does, providing adaptive learning experiences. This was the Intelligent Tutorial Systems (ITS) community. Examples in the industry are hard to find, since most of these systems are used within the academic community. Academic examples are presented in [17] and [18].

![Figure 4.5: Intelligent Tutoring System Architecture](image)

This is an Artificial Intelligence (AI) based approach that aims at adapting the content, and learning process to the learner. This is accomplished by encoding detailed knowledge derived from experts at a particular domain. Intelligent Tutorial Systems are typically composed of three different modules [19]:

1. **Expert module** - this module is where the domain knowledge is coded and can be inferred upon. There are several ways to implement it:

   - **Black-box** - this can be implemented as a simple input-output system. It can be accomplished by anticipating the learner behaviour and when that particular behaviour is detected by the system, it will generate a response, programmed to that behaviour;
   - **Glass-box** - uses a representation of knowledge, using rule based formalism, so that the encoded knowledge can be inferred upon;
• **Cognitive modelling** - encodes the way humans use knowledge. This approach allows for different knowledge types (declarative, procedural, and qualitative).

2. **Student module** - this module will allow identifying the learner and how it relates to the learning subject, its difficulties and knowledge, in order to assess the learner actions within the learning environment, and develop a model of his/her knowledge;

3. **Tutor module** - is the one that considering the learner’s and the knowledge coded within the expert module, will select how to act to deliver knowledge or knowledge building opportunities to the learner. It can present new exercises, provide corrective feedback or other means that it finds fit to help the learner. As learner’s expertise evolve, this module may allow presenting increasingly complex scenarios, explain deeper the subject or just let the learner explore the learning environment or subject.

Usually these systems are deployed as single applications. As such, only a few are integrated as part of a product where the emphasis is not in the learning process, as the problem this dissertation addresses. One example that is interesting to analyse is described below. While this system is not a tutoring system, it tries to assess the user difficulties and provide assistance. What makes this example valuable is that it was developed under the industry context. This system also comprises a student module that performs user classification, and uses Bayesian belief networks to implement a functionality similar to the expert module described above.

**Lumière Project**

Lumière Project [20] was an approach that intended to assess the user goals and difficulties using Bayesian Belief Networks. The implemented solution was patented by Microsoft\(^\text{16}\). The most known contribution of this work was Microsoft Office Assistant (also known as Clippy), which implemented several components presented on the Lumière Project.

The approach taken was to empirically evaluate how an expert could, without directly interacting with a novice, understand the novices’ goals and difficulties. To accomplish this, the authors used a Wizard of Oz approach. The novice user had to perform a task in Microsoft Office Excel, and an expert would try to understand and help the novice user. The expert subject had only a live feed of what the novice was seeing on screen, and could only interact with him through a one way chat system, in a second computer. From this the authors understood some heuristics used by the experts to analyse the novice’s state. Below some of the described heuristics are presented:

• **Search** - Continuous search for an element in the User Interface (UI). This could be measured by the quantity of actions produced and if these actions were relevant to solve the problem;

• **Introspection** - A sudden stop of interaction with the system. This meant that the user was thinking or analysing the UI, trying to find new ways to approach the problem.

• **Undesired effects** - Undoing a set of commands, after realizing that they had not produced the desired result.

From these observations, the authors built a neural network that, based on the interaction with the UI, predicted the probability of the user wanting a certain kind of help. To accomplish this, the authors found helpful to develop a query language that abstracted low level interactions with the UI, turning them into significant high level sets of actions. Some of these were:

• **Rate**(\(x, t\)) - number of times action \(x\) occurred within \(t\) period

\(^{16}\text{US Patent 6262730, Filled 1998-20-11, Issued 2001-17-7}\)
4.7. INTELLIGENT TUTORING SYSTEMS

Figure 4.6: The Lumière Project. Reproduced from [20]

- **OneOf**($\{x_1, x_2, ..., x_n\}, t$) - if one of the actions as produced during $t$ period return true; otherwise false

- **All**($\{x_1, x_2, ..., x_n\}, t$) - all actions were produced within $t$ period (in a loose sequence) return true

- **Seq**($\{x_1, x_2, ..., x_n\}, t$) - if a tight sequence of actions was produced during $t$ period return true

- **Dwell**($t$) - if the user did not interact with the system for at least $t$ period return true

Another feature presented in this project was the ability to interpret user queries. These were also used as input to the neural network. A revision on the probability distribution was performed having this data in consideration, in order to re-evaluate the user modelling.

As observed, these systems provides several benefits:

- High level of adaptation to the learner. This is probably the closest approach to one on one learning, since the system can predict the user doubts and incorrect mental models;

- Structure and content that will allow the learner to be assisted in the learning process. This will minimize frustration, since the system can predict what challenges should be presented to the learner, at a given time;

- Is able to soften the learning curve, even for very complex topics. This means that it is adequate to novice users;

As previously stated, one of the main concerns is avoiding the introduction of additional complexity and costs while developing and maintaining content to support the learning path of novice users. Analysing the Intelligent Tutoring Systems, one observes that all the complexity of those systems does not only impact their development cycle but also makes them costly to maintain. In order to understand their cost of maintenance, one can review the process involved in the creation of new content.

Suppose that one develops a system that is able to assist the learner in a new programming language. As expected this system is devised to pose questions to the learner and adapt not only the content, but also the tips given in case of a wrong answer is provided. It should also generate sets of new questions. This can be done by either choosing from a set of questions or procedurally building new ones. Nonetheless, new questions must be presented to the learning according to his profile. If implementation details and technology related issues are abstracted, adding a new question or tutorial to this kind of system, would imply the following process:
1. Developing content for the questions. This implies research to guarantee that this particular question, or a similar question does not exist in the system;

2. Create an appropriate test that can validate the quality of the solution inputted by the user. Depending on the system this means testing a final condition or testing a set of steps taken to solve the problem;

3. Use some mechanism to classify the question difficulty. This usually means presenting this question to several learners, and based on their classification and the correctness of their response, classifying the question according to some scale;

4. Branching content must be created. One needs to create \( \sum_{i=0}^{m} (n^m) \) steps in a tutorial, \( n \) being the number of user classification levels and \( m \) the number of steps needed to correctly solve the problem within that tutorial. With such an high number of steps, the common solution is to develop inference rules that are able to make an automatic generation. It is generally accepted that the alternative solution (authoring by hand) would be too costly.

Another issue with this approach is that depending on the implementation, new content creation can only be performed by experts in that field, which means that they will be a scarce resource. As one can observe this might become a costly solution, especially as the system increases in complexity and size.

4.8 e-Learning

In recent years efforts have been made to develop what was coined as e-Learning. e-Learning is the integration of several technologies (most of them presented above) in order to deliver computer-assisted learning. These can be discussions, videos, tests and quizzes, among others. These approaches deliver computer assisted learning that tries to mimic the one on one approach. For an in depth discussion about this subject, please refer to [7].

It is difficult to find examples of companies that focus on developing a product and also deliver e-Learning. Probably because companies find that the cost of maintenance is too high. Microsoft Office has implemented this concept in order to soften the learning curve from one product version to another. While the companies that develop the product leave this gap open, there seems to be a growing business model around delivering software training. Companies like BlackBelt Factory and Code School are only examples of solutions that deliver e-Learning.

The learning materials are structured to let the user evolve. They are typically devised in the following way:

- Courses are divided by expertise level;
- Each course is divided in several lessons;
- Each lesson can be composed of a mixture of forum discussions, videos, interactive tutorials, screenshots;
- A way to test the learner knowledge. Generally this is a multiple question exam.

---

18 http://www.blackbeltfactory.com/
19 http://www.codeschool.com/
4.9 DISCUSSION

In some implementations, at the end of each lesson the learner can do a test about the topic learnt in that section. Typically one can take a test even without watching the videos or doing the tutorials, but one cannot take a test of the lesson \( n + 1 \) without passing the test of the lesson \( n \). This framework was devised to assist users through their evolution process.

Let's review several features that make this technology suited for instructional delivery [21]:

- **Allows for simulations** - such as driving or other tasks that will imply some level of risk. This is a good way to deliver procedural knowledge since the method provides training on the task to be learnt;

- **Can provide in-context feedback** - the system supports knowledge transmission providing some context of the task to be learnt;

- **Support asynchronous and synchronous learning** - the learner can access the learning materials virtually at any time (asynchronous learning). If synchronous learning is supported, the learner must be present at a given time, usually to receive a lesson from someone more experienced meaning that the lesson is time constrained;

- **Dynamic content delivery** - the learning material can change over time (even learners can introduce changes), so that the learning experience can change to best fit the learner's needs;

On the downside, these tools present an high content management cost. As these systems deliver great amounts of content, as the system grows it becomes increasingly difficult to keep it updated. Since e-Learning integrate several components, it can be difficult to extend functionalities as the content grows. Also, as these systems often integrate question and answering, this implies having allocated resources to perform those tasks.

**Khan Academy**

This is an educational example of a platform that delivers e-Learning. This platform has a collection of 100 exercises and more than 2000 videos. It also introduces some game mechanics to make the presented subjects more accessible. Subjects are all organized as a visual constellation like a graph. This turns an abstract concept such as learning progress, into a well-defined physical representation where the learner can measure his evolution.

When performing exercises, the game rewards the learner if a problem is solved quickly. Several correct answers in a row are also encouraged by a bar that shows the best “streak” of correct answers so far. If the learner is able to solve ten randomly generated problems, he will pass that subject. But if the learner does not know the answer he can sacrifice the “streak” count and get a hint, or watch an instructional video at no penalty.

In order to reward the learner, there are badges that can be earned. These badges cover several categories and learner levels. Some of them are presented in annex A.2.

**4.9 Discussion**

After reviewing some of the technology used in the industry to deliver training to the users, can summarize the several techniques and tools.

Figures 4.7 and 4.8 summarize the main issues discussed in this chapter.

- **Non-interactive** media has a low coupling with the product development. This means that the development of these materials will not be on the critical path of the product development. Also the learner needs to perform a high amount of context switching;
CHAPTER 4. RELATED WORK: LEARNING TOOLS

- **Video tutorials** provide context to the learner and reduce the context switching. They also are not coupled with the product development cycle;

- **Discussion forums** are excellent in dealing with issues or questions not previously identified by the development team, but are not the most adequate for the novice user;

- “**Gamification**” gives time and extra motivation for the learner to improve, but its development is highly coupled with the product development cycle;

- **Abstraction layers** provide little content, but they teach a novice to master the platform, increasing their productivity. They are also highly coupled with the product development, since they can be seen as a feature of the product.

- **Interactive tutorials** provide a good balance of procedural and declarative knowledge, and provide a continuous transition between the discussed topics. But they are difficult to change;

- **Intelligent Tutoring Systems** are highly adaptive to the learner, but as they need experts in the domain, their development and maintenance costs are high;

- **e-Learning** provides a smooth introduction to the topics, provides both procedural and declarative knowledge and their development can occur in parallel with the product development cycle.

Figure 4.8 summarizes the costs associated with the development of the several tools discussed in this chapter. The coupling level of the tool with the product development is crucial to understand if the development of the tool is in the critical path of the product development. If so, the product backlog will be impacted.
### Figure 4.8: Comparison Between Learning Tools and Maintenance Costs

<table>
<thead>
<tr>
<th>Non-interactive media</th>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video tutorial</td>
<td>No</td>
<td>Easy</td>
</tr>
<tr>
<td>Discussion Forum</td>
<td>No</td>
<td>Easy</td>
</tr>
<tr>
<td>“Gamification”</td>
<td>No</td>
<td>Medium</td>
</tr>
<tr>
<td>Abstraction Layer</td>
<td>No</td>
<td>Easy</td>
</tr>
<tr>
<td>Interactive tutorial</td>
<td>Yes</td>
<td>Hard</td>
</tr>
<tr>
<td>Intelligent TutoringSystem</td>
<td>Yes</td>
<td>Hard</td>
</tr>
<tr>
<td>e-Learning</td>
<td>No</td>
<td>Very Hard</td>
</tr>
<tr>
<td>Coupled With Product</td>
<td>Change</td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Medium</td>
<td></td>
</tr>
</tbody>
</table>

This table summarizes the comparison between various learning tools and their associated maintenance costs.
Why Interactive Tutorials Are Suited To Address The Problem

As discussed throughout this document, companies struggle to support the wide spectrum of the Dreyfus model. As novice and expert users have opposite needs, developed solutions only tend to address a particular stage of the learning curve. The goal of this dissertation is presenting a framework that will assist the novice user providing continuous support. This framework should balance procedural and declarative knowledge in order to guarantee that the learner is active in the learning process and should not increase the frustration levels. This chapter presents the factors that lead to choose an interactive tutorial in order to address the problem.

Trying to apply an Intelligent Tutoring System approach, would not solve the problem given the high maintenance costs, but one can agree that the level of assistance provided by those systems is adequate to novice users. So instead of trying to apply that approach to the problem, was relevant to develop a solution that was easier to integrate with a product and cheaper to maintain, while retaining some of the features that made Intelligent Tutoring System fitted for novice users. One approach was to extend an interactive tutorial in order to provide some degree of adaptation to the learner. This way one could provide a platform for computer-assisted learning with an increased subject matter retention rate, while keeping the maintenance costs and frustration levels low. Some of the factors that lead to choose an interactive tutorial system as the core of the solution, are described below.

5.1 Balancing Procedural and Declarative Knowledge

If one observes the learning curve presented by a complex software system, such as an IDE, can be observed that there is a combination of procedural and declarative knowledge that must be mastered. Only this way a user can successfully accomplish his goal. In the case of a programming language this can be for example implementing a feature or developing some application. As in any other procedural knowledge learning process, the key is to deliver the correct information at the right time and give the opportunity for the learner to experiment and practice within the relevant context of the task. This is discussed by Schank in [21].

Also, Sweller et al has demonstrated in [8] that the learner should be active in the learning process in order to improve the performance when problem solving. In this study two groups of participants were formed. The first group was passive in the learning process, only receiving instructions. While the second group was active, solving problems during the learning process. When presented with a transfer test, participants from the second group, on average performed significantly better than participants who were passive in the learning process. This gap was even more noticeable when the learners were faced with unknown problems and situations. Since the correct scripts were developed these participants could infer new knowledge and correctly address the new problem.

Effective procedural knowledge acquisition requires real experience. An interactive tutorial has the capabilities of delivering and stimulating this knowledge. Also it provides means that will force the learner to be active in that process.

Exploratory learning methods, also called discovery learning might not be the most adequate to address the problem. This kind of approach is extensively used in games and simulations. Part of this learning process is based on understanding how one can modify the surrounding environment [22]. One could let the users explore at will, thus learning through trial and error, but it has been shown that
pure-discovery learning has its shortcomings. These can provide incorrect mental representations to the
learner, and doing so would also increase the user frustration levels. These topics were discussed on
section 3.2.

So, even if the learner while exploring is able to learn how to solve a problem, literature shows that
incorrect mental representations are build [23], being difficult for the learner to apply the new knowledge
to other instances of the same problem. It was shown that although students performed well when asked
about the instance of the problem that they solved during exploration, they performed rather bad when
trying to generalize that knowledge and took more time than students who experienced guided learning,
when solving a different instance of the same problem.

The interactive tutorial system provides a framework that delivers procedural knowledge, teaching
the learner how to accomplish certain tasks. One could extend it in a way to provide declarative knowledge
while keeping the learners engaged.

5.2 Worked Examples and Solving Exercises

As previously stated, while developing this framework one had the opportunity to watch several
hours of usability tests, where the Agile Platform was tested. Typically participants were asked to
perform tasks such as building an application to list and edit data. They were told to install the Agile
Platform and use whatever resources they found in order to learn how to accomplish the task given
to them. After finding some tutorial which explained how to build a similar application, users would
always try to mimic the steps of the lesson (learning environment) in order to solve their problem (real
environment). It was common observing participants executing actions that made little sense in the
context of the problem they were solving. As an example there was a situation of a learner that named
a variable “Company”, because the tutorial was teaching how to show a list of different companies. This
made no sense for the problem solution since the learner had to list several employees during the transfer
test.

These situation shows that users are being mentally overloaded, thus not being able to successfully
perform a transfer test. Interactive tutorials are able to provide mechanisms to present a layered approach
to the learning process, since they can incorporate:

- The ability to navigate back and forward in the tutorial;
- The ability to give users time to read and structure the problem before handing a solution;

The first is crucial for the learner to go back and forward in the lesson. This allows him to take
the lesson at his own pace and review prior steps. This ability also serves as a “worked example”. A
worked example is explaining step by step how a given problem was solved, giving time for the learner
to analyse. This allows the learner to start dissecting the problem and acquire some basic schemata. If
the learner is able to freely navigate within the several steps in the tutorial, he can analyse the problem
being solved and try to understand how the several components interact. Kalyuga et al have shown in
[14], that when problems are inherently complex students will perform better if they are first exposed to
worked examples, prior to problem solving. By doing so, the student will be less overloaded when trying
to solve a problem. This allows to increase their performance.

The second factor is relevant because as discussed in chapter 3, learning needs time and repetition.
From usability tests, one has observed users advancing in the tutorial without mastering the previous
step. This will make them unable to problem solve if later presented with a test task. But interactive
tutorials can provide mechanisms that foster repetition and give time for the learner to structure the
problem so that users can address it. One example is inserting some “gamification” elements into the
interactive tutorials, making it more enjoyable to repeat actions.
5.3 Usability

As discussed in section 3.1, misleading “signifiers” lead to poor usability which is one of the main reasons for low performance and steep learning curves on software tools. Jakob Nielsen discusses this issue on his “alertbox”\(^1\). Misleading “signifiers” can confuse the novice learner and introduce extraneous cognitive load. As an interactive tutorial can be easily subjected to usability testing, one can iteratively eliminate these kinds of problems. Thus, the tutorial authors and developers can trace and understand the impact that a change had, after it has been introduced. Having the ability to perform A/B testing is crucial to improve the learning materials. Also, instrumentation systems can be deployed in order to evaluate the steps in the tutorial that learners find more difficult. This is harder to accomplish in other approaches such as non-interactive media or video tutorials, since the learning environment and the problem solving environment are not coupled.

5.4 Controlled Failure

Another feature that makes the interactive tutorials a good opportunity for improving the learning experience is the contained failure [21] and the minimized stress [24]. Failure is an important part of the learning process, because sometimes one learns more from mistakes than from successful experiences. This is mainly because one will always remember situations on which one has. Two failure related subjects are relevant when discussing the interactive tutorials:

- **Controlled failure** - In a learning environment the failure should be seen as learning opportunities and as such, the learner should not be severely punished. Instead of what happens in the real world, either directly through instructor’s call for attention or through peer pressure. These technologies provide a learning path with reduced pressure, which incentives the learner to fail and learn with those experiences;

- **Privacy** - Computer assisted learning provides the capability of private failure. Peer pressure is not critical since there is no one to observe and judge the mistake, reassuring users’ feelings.

5.5 Provide Scripts to the Learner

Interactive tutorials can deliver both declarative and procedural knowledge. This way, the learner can, at his own rhythm, acquire the schemata needed. Because a tutorial can teach how to solve a problem and explain patterns instead of steps to solve a problem, this will also lead to the development of scripts. As discussed in section 3.3, this will allow the learner to solve any instance of that particular problem. So, if the user is able to retain most of the information presented in the tutorial, he will be able to implement that pattern without further need of assistance.

\(^1\)http://www.useit.com/alertbox/20030825.html
Increasing Interactive Tutorials Retention

As observed in the previous chapter, an interactive tutorial could be extended in order to integrate some level of adaptation to the learner’s experience. As was observed in the previous chapter, an Intelligent Tutoring System provided a great level of adaptation to the learner, but introduced costs that jeopardized some of the goals of this dissertation. Although a full implementation of this system did not solve the problem, because of its costs of development and maintenance, one agrees that tailoring a tutorial to the user needs would prove to be a very interesting approach. So, this dissertation proposes a carefully balanced system that combines the core of an interactive tutoring system with the tailored assistance provided by Intelligent Tutoring Systems, to address the problem. This way the costs of integration and maintenance will be balanced, with all the benefits that the adaptation provides.

This chapter presents the general guidelines that can be implemented on any software system that has an inherently steep learning curve. Such systems can be IDE’s, image editing software, 3D modelling tools and others that present the problem that this dissertation addresses. For a better illustration, in the next chapter will be discussed how these guidelines were implemented into a real product, the OutSystems IDE, and how it impacted that platform. In order to increase the retention rate of the learning materials, this framework implements:

- An increasing assistance level;
- Increasing challenge as the learner evolves in expertise;
- Allows the delivery of inquisitive and exploratory tasks;

6.1 Basis of an Adaptive Tutorial

Similarly to the Intelligent Tutoring System, the developed architecture is composed of two modules. The first is intended to evaluate the user while experiencing the tutorial, and the second to react according to that evaluation:

- **User modelling** - it monitors user events and classifies the user according to them. This module will monitor the user, in order to understand if he is performing below or above the expected. It will capture metrics such as number of interactions with the system, and the time to perform a given action;
- **Adaptation** - adapts the experience according to a set of predefined rules, on which the input would be given by the user modelling module. This module will define the level of assistance that the user will receive.

More concretely, the user modelling will monitor if the user is performing below the expected and notify the adaptation module so that it increases the level of assistance. On the other hand, if the user is performing above the average, the adaptation module can start to provide less assistance.

To implement this, one needs:
1. **Baseline** - a set of metrics that allow specify the average learner. If the user performance, while experiencing the tutorials is above these metrics, one considers it to be positive. Otherwise, it is considered negative. This baseline is set according to an average user, but will be adapted to the user at run-time, when the tutorial is started;

2. **User history** - How many tutorials has the user experienced before, and in each of those tutorials his performance was below or above the baseline. This information can be saved in a settings file each time the user ends a tutorial;

3. **Initializer** - even though the baseline provides information about the average user, one needs to adapt the experience of the tutorial according to the past performance of the user. If the user has performed several tutorials with success, then the next tutorial’s baseline should be changed expecting more of him. On the other hand, a user that never performed a tutorial should have a baseline that provides assistance earlier;

4. **Recorder** - also part of the user modelling, this module will record every interaction of the user with the system. This will allows identifying if the user needs assistance. This information can also be stored for later use if necessary.

Figure 6.1: User Modelling and Adaptation Modules

Figure 6.1 shows how the described elements compose the user modelling and adaptation. One of the goals of this framework is to balance the level of assistance provided to the users. If the user is performing below the baseline, he should receive support sooner. If performing above the metrics specified in the baseline, one should decrease the level of assistance, until a point that allows the user to solve the tutorials almost without additional assistance. But one needs to have in consideration that:

1. If scarce support is provided, the learner can become frustrated, thus not continuing the learning process. This would increase the frustration levels of the users experiencing the tutorial;

2. If too much support is given, the learner will easily complete the tutorial, but since he did not actively process the subject matter, no schemata will be formed. This will lead to a bad performance in a transfer test. This is what is happening in current version of the tutorials. As discussed in section 2.3 because too much assistance was given to the users, they were able to perform a tutorial, but perform badly when problem solving.

This is the typical scenario observed in software training. As can be seen from figure 6.2, each tutorial has its how baseline, that is specified by the tutorial author. At the beginning of the tutorial, the “Tutorial Initializer” component, from that baseline and the user history recorded, will generate an adapted baseline.

The “User History” is the condensed data about the performance of the user while experiencing a given tutorial. Figure 6.3 shows how this process is performed. While the user experiences the tutorial,
6.1. BASIS OF AN ADAPTIVE TUTORIAL

Figure 6.2: Generation of an Adapted Baseline from the User History

Figure 6.3: Saving the User Performance

metrics are stored per task. Metrics were collected about: number of interactions with the system, time taken to solve the task, number of undos performed. Was also stored the timestamp of the first interaction in that task, how much time did the user had to read the text in that task and expected number of interactions with the system, in order to solve the task.

\[ f(n) = \sum_{i=0}^{n} (\text{Time To Solve}_n - \text{Expected Time to Solve}_n) \]

Although simplistic, it provides a good metric about the user expertise, as discussed in section 4.7. Also, this was done having in consideration the limitations of the platform on which this framework was implemented. These limitations will be discussed in chapter 7.

\[
\text{SuccessfulTutorial} = \begin{cases} 
True & \text{if } f(n) > 0 \\
False & \text{otherwise}
\end{cases}
\]

This framework will only consider tutorials who have been successfully finished. Although it is also saved information about tutorials finished unsuccessfully, that will not affect the “Tutorial Initializer”. As
an example, if a user performs two tutorials, finishing the first with success and the latter unsuccessfully, the “Tutorial Initializer” will consider that only one tutorial was successfully finished, and will adjust the baselines to that fact.

6.2 State Machine With Fuzzy Transitions

As discussed, a state machine was implemented, in order to model and assist the user. This architecture comprises a state machine with fuzzy transitions, as described by Schwab [25]. It is important to notice that the implementation of the fuzzy transitions can be accomplished by implementing a higher abstraction layer. This allows for a simpler representation of the user interactions with the system. An example of that implementation is the one described in the Lumière Project (section 4.7). There are three states, and each state is responsible to produce changes either on the UI or the tutorial system itself, providing an increasing assistance level.

The three states depicted on figure 6.4 are:

- **Reading State** - It is intended so that the learner is focused on reading the presented text and starts to address the problem without any help;

- **Cue State** - This state is responsible to introducing a small clue on how to solve the problem or what element in the UI the learner needs to interact with. This should be discrete enough to not disturb the learner if he is still reading;

- **Guided State** - This state should introduce changes that are noticeable by the peripheral vision of the learner. Even if he is focused on another part of the screen, his attention should be directed to the UI elements that need interaction to solve the problem.

This was developed in order to increase the level of assistance given to the learner, according to the state of the modelling component. This way, the learner will receive more assistance in case the modelling system detects that the learner is having difficulties in solving a task.

It is important to notice that this is a cascading system. As soon as a task is solved, a new one is presented in the tutorial. The state machine will go back to the Reading State whatever the current state was. This is developed so that the learner only has to read a certain amount of text that is related to the task he is trying to perform, avoiding extraneous processing. In this implementation, which is described on chapter 7, one found useful for the fuzzy transitions to consider the events produced by the user. As such, if the user is in a given state and a transition is activated that signals that the state machine should change to the next one, two things can happen:

1. The user is producing events on the interface. This means that he is working to solve the problem, and as such the state machine will not change state;

2. The user has not produced events on the interface for some time. This could mean that he is thinking or needs some help, so the state machine should go to the next state in order to provide more assistance.

Although there are only three states, the Guided state can fire two different events, which will increase the level of assistance. As the level of assistance remains the same during the Guided state, was found that there was no need to inserted them as new states in the state machine. Please note that this is a generic framework, and as such one must consider the details of the platform on which it would be implemented in order to assess the introduction of more states. As will be discussed in chapter 7, was found that implementing only three states provided a simple implementation.

This was done having in consideration to keep frustration levels low, thus not being too intrusive. These two events are:
6.2. STATE MACHINE WITH FUZZY TRANSITIONS

• **Help** - Each task in the tutorial should be associated with a help text that the learner can consult. This allows the learner to become independent and understand how to solve a task if for some reason they cannot do it with the instructions provided. If this event is triggered, this help message will automatically be shown to the user. The first time this event is triggered, it will show a standard message that explains how the learner can navigate in the tutorial. After that, the help message associated with the current task is displayed;

• **Lost** - If the modelling system identifies that the user is lost (great number of interactions with elements not relevant to the task and a great amount of time is spent trying solving the task), a message will be displayed asking if the user wants to search for help outside the tutorial system. This makes it possible to integrate the interactive tutorial system with other instructional delivery methods.

This approach is intended to increase the level of assistance if the user is performing below the baseline, or provide less assistance in the case he is performing above the metrics defined in the baseline. Also, this framework will not make more experienced users (advanced beginner - competent) less efficient. This is an important factor, since as discussed, the interactive tutorials are able to assist users in this level of expertise. This set of users might start a tutorial to replicate a certain pattern. It was observed that more advanced learners would know that a certain tutorial contained a pattern or a solution to a given problem. So instead of performing the tutorial from the beginning, they would search for the step that taught how to implement that pattern, and replicated the logic in a new application. This architecture is able to maintain the support for this kind of behavior without introducing frustration in the process. At the same time, it provides a good level of adaptation to the novice learner.
6.3 Provide Time For Reading

Having the problems described in chapter 2 in consideration, and more concretely the fact that users were scanning for words such as: click, double-click and drag and following the arrow. This made a tutorial easy to finish but with low retention rates. This is a problem since no schemata is acquired and these users will perform poorly on a transfer test. To address this, the developed framework implements an automatic mechanism in order to only give assistance after the user had time to read the presented text. In the state machine this is the Reading state.

If the user is experienced enough, he should be able to perform a task almost without assistance. This means that the user can keep searching for keywords that allow him to advance in the tutorial. On the other hand, novice users will be encouraged to read the text, because no assistance will be given during this period.

While humans typically read at a 200-350 words per minute rate (considering a word composed of 6 letters), it has been recognized that computer screens lower this ability. Also both the platform and the tutorial system are very rich in information and comprise several UI elements. A review on reading speed for printed paper is done by Bell on [26]. Jakob Nielsen presents a study indicating that reading on other materials tends to be slower [27]. Since the platform conveys much information, this makes the attention dissipate from the text displayed in the tutorial. For this reason, the user modeling should be iteratively tested, in order to find the most suited value for the reading speed. Was found that a value between 150-200 words per minute yielded positive results. Within this range, the user was encouraged to read the explanatory text and start addressing the problem.

Since one of the goals of this framework is not increasing the maintenance costs of the tutorial system, the Reading state should be developed in such a way that it becomes transparent to the tutorial author. When authoring the tutorial, the author should not be constrained or have to deal with more complexity. For this reason, only at run-time the amount of text should be calculated, and the time given to the Reading state should be set. There are at least two ways of implementing it in this architecture. The “Initializer” module could check the amount of text per task and set the tutorial baseline according to this value. The second, which was chosen in this implementation, was to delegate this task to be performed when the user enters a new task. This way, the processing of the amount of text and the inherent reading time that will be given to the user are only done when the user enters a new task.

So only at run-time the reading speed for each of the tasks developed is calculated and the system adapted when this state is activated.

6.4 Increasing Challenge

A user that has successfully performed several tutorials, in which he was classified above the average, is clearly not on the same learning stage that the user that only performed one tutorial and finished it below the average. In order to reflect this, a system was developed so that the baseline according to which the user will be classified expects more from an expert user (providing less assistance) and expects far less from a user that performed badly, providing assistance sooner.

Figure 6.5 shows a basic system that is able to increase the challenge according to the user history. With the components described in section 6.1 (Baseline, User History, Initializer and Recorder) it becomes possible to keep track of the evolution of the user over time. Using that information one can take a general baseline and adapt it to the particular history of the learner. This framework allows this technique so that:

1. If the learner has performed badly or did not perform any tutorial, the system can decrease the metrics in the baseline, so that he receives assistance sooner;
2. If the learner is performing several tutorials with success, the system can increase the metrics in the baseline. This way he will be challenged by loosening the assistance that he receives.

This functionality is important to stimulate schemata acquisition. By giving the least amount of necessary assistance, the user will be encouraged to stop, read and analyse the environment and start to address the problems posed. It will avoid the problems described that were in part aggravated by the excess of assistance. It is important to highlight that the adjustment on the baseline should be soft, so that it does not expect too much from the user, leading to high frustration levels. The acquisition, retention, retrieval cycle will be improved, because with the correct level of assistance, the user will retain more of the learning materials.

This system could also be used during usability tests in order to collect metrics of the user interaction with the tutorial, which can later be analysed.

6.5 Framework For Building Tutorials

Also from the experience gathered, one came to understand that there were four different kinds of tasks that should be supported by the tutorial system. But only two of them were supported in the embedded tutorials of the Agile Platform: Action and Informative tasks (depicted in figure 2.4). This framework suggests the careful equilibrium between four task types:

- **Actions tasks** - tasks where the learner must interact with the system, introducing changes. These tasks stimulate procedural learning;

- **Informative Tasks** - purely informative tasks intended for the learner to absorb some information. These tasks deliver declarative learning;

- **Inquisitive Tasks** - as informative tasks, these deliver declarative knowledge, but are developed to test the learner. This way the learner will be encouraged to stop and review the subject matter;

- **Exploration tasks** - tasks intended for the user to interact with the system, navigating through it, while not introducing any changes in the application that is being developed. This allows the learner to relax for some moments before tackling a new problem, as well as observing the effects produced by a given action.
As discussed, while providing software training, one needs to balance the delivery of procedural and declarative knowledge. As such, action tasks will allow the delivery of procedural knowledge, and informative tasks allow the delivery of declarative knowledge. Inquisitive and exploration tasks will improve the acquisition - retention cycle, since the user is encouraged to mentally review the training materials. Inquisitive tasks also serve to maintain the user focused on the other task types.

One came to understand that finding the correct amount of each task type, in order to develop a tutorial was difficult to achieve. Some learners will instinctively jump over informative tasks, since they will get immersed in solving tasks and interacting with the interface.

The following combination of tasks is proposed:

1. Start the tutorial with an informative task. It should explain what the goal of the tutorial is, and give an overview of the process. Our experience indicates that most learners will not fully read this task. The tutorial author should guarantee that even if the learner does not read this step, the learner will be able to perform the tutorial;

2. Deliver a set of action tasks. The correct number is dependant on the complexity of the task, but from our experience the learner should experience around five of these tasks. This is also supported by [3]. One should break a given action into several tasks (using the 7±2 rule). By doing so, the user will be able to keep these items in the working memory, making it easier to understand the relations among them and forming schemata;

3. Present an exploration task. If the learner spent too much time focused on solving problems, this is a good way to provide declarative knowledge and at the same time reduce the stress levels. This is because with this task the learner does not have to introduce changes in the system. Also this kind of tasks should be used to show what the user has accomplished so far;

4. Present one task that stimulates declarative knowledge. This can be accomplished by:

   (a) Informative task. This should explain at a higher level of abstraction what goals the previous actions achieved. It can be used as a summary as seen on figure 4.4b. Basically this should help the learner building scripts on how to solve similar problems;

   (b) Present an inquisitive task (this issue will be detailed on the next section). This allows creating schemata by reinforcing the knowledge learnt in the course of the tutorial.

5. Near the end of the tutorial, present a review question, that asks about the core idea presented in the tutorial;

6. The last step in the tutorial should congratulate the learner and encourage him to perform other tutorials. this will form the bridge to other topics;

From this experience, delivering 80% of procedural knowledge and 20% of declarative one, is a good balance. This allows to keep the learner motivated and engaged in the learning process while performing tasks, and getting some time relaxing and building the correct schemata by experiencing declarative knowledge. As discussed in Flow [16] and depicted in figure 4.1, to keep the user engaged, one needs to introduce periods of stress followed by relaxation. Continuous stress will lead to frustration and continuous relaxation will lead to boredom.

On this framework, the stress periods will be provided by the Action and Inquisitive Tasks. The relaxation periods by the Informative and Exploration tasks.

### 6.6 Framework For Inquisitive Tasks

The developed framework allows the introduction of Inquisitive Tasks inside a tutorial. This feature is developed to:

- Facilitate deeper understanding of the material by prompting learners to think critically about the content.
- Encourage active engagement and interaction with the tutorial.
- Provide opportunities for learners to apply their knowledge in new contexts.
- Promote the development of problem-solving skills.
- Enhance retention through the incorporation of declarative knowledge into procedural tasks.
6.6. FRAMEWORK FOR INQUISITIVE TASKS

- Present tasks that stimulate declarative knowledge;
- Provide the correct schemata. This way the user will not only be able to solve a particular problem, but will generalize the knowledge acquired. It will allow to increase performance when new problems are presented;
- Provide a “barrier” that will not allow the learner to advance in the tutorial without actively processing the materials. This is crucial to solving the problem posed by users following the clues given, without actively processing the information.

Was realized that providing inquisitive tasks to a tutorial could yield a positive impact on the learning retention rate. This decision is supported by Jakob Nielsen in [28]. This study points to the fact that surveys and questions can be a good way to address the low retention levels on the web. Even though the study focuses on the web, the author discuss one of the problems this document addresses: users scanned through the text without absorbing the content.

But it is not enough to structure the tutorial in a given way, as discussed in the previous section. It is also important to know how to correctly present the Inquisitive Tasks to the user in order to maximize their efficiency.

The implemented solution provides the learners the chance to choose between options at a given step within a tutorial, presenting questions such as “How do you think one can change the application’s header?” and showing a message congratulating and explaining the reason for that, or in case of being unsuccessful, showing another message that explains what is incorrect and why. It was observed that humans try to fit new knowledge in previous built schemata, and with these questions at the core of a tutorial, one ensures that a learner does not maintain a faulty mental model for a long time, therefore not letting a snowball effect of wrong ideas to be build and at the end would be harder for the learner to change his wrong schemata.

Even if the given answer is incorrect, the schemata will start to be built as the subject matter was processed actively in order to answer to the posed questions. This implies that the authoring of the inquisitive tasks must also follow a set of rules to ensure that generative processing is made. This dissertation presents a set of guidelines, based on the psychology background presented at chapter 3, but also guidelines provided by Mayer et al in [7] and by Schank in [21]. Is also important noticing that these guidelines were developed in an iterative way.

1. A question should be placed to revise a set of steps in the tutorial. This will allow stimulating the acquisition-retention cycle and can be accomplished in two ways:
   
   (a) After several Action Tasks present an Informative Task with a summary and then an Inquisitive Task;
   
   (b) After several action tasks present and Inquisitive Task followed by an Informative one.

Both approaches will allow reviewing the concepts presented in the Action Tasks. In this implementation, after around five Action Tasks, an Informative Task is presented, with a summary and then presented the Inquisitive Task. But this is dependant on the complexity of the lesson and system;

2. A question should be provided before the last step in the tutorial. This should be a review question. It should be about the most important topic discussed in that tutorial. This is intended to provide an overview of the lesson;

3. Inquisitive tasks should be made after a significant Action Task. An example can be “You just dragged an entity to the canvas, what will happen now?”. When guided the user will be told to perform an action and explained about the consequent reaction. With Inquisitive Tasks, one is testing the ability of the user to understand the link between the action and reaction, allowing to improve schemata;
CHAPTER 6. INCREASING INTERACTIVE TUTORIALS RETENTION

4. Every hypothesis should have an associated explanation. It should state why that hypotheses is correct or incorrect and what is the correct way to address the problem. The answers to the Inquisitive Tasks are as important as the questions themselves. This way the user will be understand why he was correct/wrong which improves schemata acquisition;

5. The wrong alternatives provided in the Inquisitive Tasks should be typically observed questions from previous learners. These questions should be collected during classes, frequently asked in forums, or others. As the tutorial author can be an expert on the system, he might not correctly identify the faulty mental models of novice users. One should ensure that Inquisitive Tasks are used to diagnose and revise some of the incorrect schemata of the users;

6. Users should be stimulated to read all the answers on the Inquisitive Task. This improves schemata acquisition since the user not only will know why a certain answer was correct, but also will have the opportunity to improve his mental models by understanding why given options are not valid;

7. Do not present options such as “all of the above”/“none of the above”. These kind of options do not help a novice learner building schemata since they are too vague;

8. Questions should be written in an informal tone. This will ease the pressure on the user, not introducing additional stress to the tutorial;

9. Use forgiving language to explain the wrong alternatives. Wrong answers should be replied in a way that alleviates the pressure on the learner.

Providing Inquisitive Tasks and opportunities to stimulate declarative knowledge does not necessarily imply helping users build the correct schemata. So one must ensure that the questions and answers correctly address the user faulty mental models, in order to build a consistent learning path.
Concrete Implementation
In OutSystems Platform

As stated in section 2.2, the framework described in the previous chapter was implemented on Service Studio, an IDE for a domain specific language. This IDE had already an embedded tutorial system that could be extended with the proposed solution.

7.1 Context

Before discussing how the framework was implemented, on the tutorial system embedded in Service Studio, it is relevant to discuss the platform architecture, to provide a better insight on some of the implementation options.

Basic Architecture

Service Studio was developed using a Model View Presenter (MVP) architecture\(^1\). This architecture not only proved to be fit for the integration of an embedded tutorial system, but also provided a good starting point for the development of the framework on top of it.

Furthermore, the basis of an Intelligent Tutoring System, as described in section 6.1, could be implemented since:

- The user modelling could be implemented either on the Model or Presenter side. Those modules already stored the internal state of the program, so one could also use this information to assess the user state;
- The adaptation module could be implemented on the Presenter side, since it provided mechanisms to change the View. With this architecture one was not limited to change and adapt the tutorials, but could modify anything in the User Interface in order to provide guidance to the user.

Validation Mechanism

As discussed in section 2.2, the embedded tutorials are comprised of steps and tasks. A step contains several tasks and each task contains several validations. Once the user starts a step, the validations of the first task will be performed, to check if the tutorial can load the next one. If not that first task is displayed to the user. It is important to notice that the validation system tests discrete states.

This means that if the first task asks the user to rename the application to “SalesAssistant”, then the logic performed will be: every time that the application model is changed, check if the application name is set to “SalesAssistant”. If this happens, load the next task and its validation stack.

This validation by states is very useful, because one can abstract the way a user arrived to a given state. If in this example, the user could change the application name in five different UI elements, it

\(^1\)http://msdn.microsoft.com/en-us/magazine/cc188690.aspx#S1
CHAPTER 7. CONCRETE IMPLEMENTATION IN OUTSYSTEMS PLATFORM

would not matter where the user did so, because once the name was as expected, the tutorial would validate it and would fire an event signalling that this task in the tutorial was completed. This provides a good abstraction layer to handle validations in the tutorial system.

### 7.2 State Machine with Fuzzy States

Having the constrains of this particular platform in consideration, was implemented the several levels of assistance as described in 6.2. In figure 7.1, one can observe that was chosen the arrow of the tutorial system in order to provide the increasing assistance.

![Figure 7.1: Implementation of the User Evaluation States](image)

The tutorial system embedded in Service Studio was extended, in order to incorporate several levels of assistance. While implementing this framework, three options were considered in order to provide guidance to the user. Two of these were discarded:

1. Certain features of the UI could be disabled, and as the user progressed they would be made available. This would be an iterative disclosure of the interface. Although interesting, this would impact the performance of advanced beginners and competent users;

2. Whenever an UI element should be used in order to advance in the tutorial, that element could be highlighted in the UI. This technique is also know as dim focus. Some possibilities were increasing the brightness of that icon, or reducing the brightness of all but that icon. This hypothesis was discarded, because the tutorial system already had an arrow that points to the place that needs attention, and so one would only be implementing another mechanism to focus the user attention.
7.3. PROVIDE TIME FOR READING

Figure 7.2: Increasing Help System

The modifications introduced in the level of assistance can be observed on figure 7.2:

- **Reading state** - As can be seen from figure 7.2a, in this state the arrow in the tutorial does not appear. It is intended so that the learner is focused on reading the presented text and starts to address the problem without any help;

- **Cue state** - A static arrow with 50% opacity is displayed next to the UI element that the learner must interact in order to solve the task. This is tuned in order to be almost unnoticed by the peripheral vision, forcing the learner to scan the UI after reading the problem. Only when focusing attention near the relevant UI element the arrow will be noticed. Figure 7.2b depicts this;

- **Guided State** - A moving and fully opaque arrow is displayed next to the UI element that needs attention. This is tuned in a way to be noticeable by the peripheral vision, so that even if the learner is focused on another part of the screen, the moving arrow will alert his attention. This can be seen on picture 7.2c.

As specified on section 6.2, the “Guided State” can fire up to two events. The first signalling that the user needs further help and the latter that the user is lost and should find help outside the tutorial.

In figure 7.3 is depicted the implementation of the event that is raised if the system finds that the user needs additional help. In this implementation, when the event is fired, the tutorial window will be centred on the screen and the help pane will pop-up. This way, even if the user has pushed the tutorial window near the screen edge, he will be able to see the help pane. The first time the help event is raised, a generic message will be displayed explaining how the user can advance in the tutorial. This is depicted on figure 7.3.

If this event is fired more times, the message displayed will be related to the particular task that the user is trying to solve. This can be seen on figure 7.5.

One important feature is that in this implementation, the Lost event will only be fired once in the course of a tutorial. This was made to ensure that the system would not interrupt the user, leading to high frustration levels. If the user chooses to go out, by pressing the “Search Community” button, the tutorial will open the browser in the OutSystems Community (forums) page, with a query already made containing the name of the tutorial that the user was performing.

7.3 Provide Time For Reading

As observed in section 6.3, was chosen to only calculate the amount of time that will be given to the Reading State when the user enters a new task. This was done having two aspects in consideration:
1. If an eager evaluation was made, one would be introducing computational costs which would delay the start of the tutorial;

2. A lazy approach would distribute the computational costs evenly throughout the tutorial.

Since an event is raised when the user is entering a new task, one could easily implement a lazy evaluation system.

When the tutorial validates the user action, if there are no more validations to perform, a new task will be loaded and should raise an event. When this happen, the reaction to that event should be to get the amount of text associated with the new task, calculate the necessary time to read, set the Reading state with that value and change the state machine to it. From now on, the state machine will fire another event when the time for reading has ended and one should react accordingly. One more feature had to be implemented because of the singularities of this system. As shown on figure 2.5, every task has some introductory text, and this too should be added to the time that will be given to the “Reading” state.
On figure 7.6 one can observe that in this implementation, the time for reading for every first task in a tutorial step, has into account both the introductory text as well as the text contained in the task itself.

### 7.4 Framework For Building a Tutorial

In figure 7.7 is depicted a part of the “Build an App in 5 minutes” tutorial, which was re-authored in order to test the developed framework. For clarity purposes is only presented a part of the tutorial. In this tutorial was incorporated:

- **Introduction (1)** - this presents a summary of what the user will learn in this tutorial;
- **Action Tasks (2)** - each of them teaches the user how to perform certain tasks to achieve the goal of the tutorial;
- **Informative Task (3)** - this task will explain how SQL queries are handled by the platform. This provides declarative knowledge;
- **Exploration Task (4)** - in this task the user can explore what he built during the action tasks. During these tasks the user modelling is disabled, and the user can take the necessary time to explore;
- **Inquisitive Tasks (5)** - the user will experience inquisitive tasks after an informative or exploration task.

In order to keep the maintenance costs low, the tutorial authoring tool was extended to support authoring of Inquisitive Tasks as well as Exploration Tasks. This way, the author can develop a tutorial that comprises the four task types described in 6.5 in a What You See Is What You Get way.
Support for Inquisitive Tasks

Picture 7.8 depicts the extended version of the tutorial authoring tool, while performing the edition of an inquisitive task.

By extending the authoring tool, was reduced the cost of new content creation, because the author can create, test and deploy the inquisitive tasks in a visual way. Figure 7.9 shows a user experiencing an inquisitive task in the course of the re-authored tutorial.

Support for Exploration Tasks

To provide support for exploration tasks, the authoring tool was extended, in order to support it. This will deactivate the modelling system for a given task. By doing so, no assistance will be provided to the user, because the goal of such task is for the learner to explore the environment. A typical use case would be to show to the user the elements of a screen that was automatically created.

The tutorial author could let the user explore, click, interact and observe the logic behind a screen that was created during actions tasks. Albeit the tutorial author might want that the user interact with
the application being developed, he might not wish to provide any assistance to the learner, since it was not the objective of that particular step in the tutorial.

The implemented system allows the tutorial author to specify Exploratory Tasks. In figure 7.10a can be observed that the first task implies an action on the user interface, while the second should give time for the learner to explore, since he will be testing the application built so far. With the implemented solution, the tutorial author sets a property (depicted in figure 7.10b) to true, in the authoring environment. By doing so, the “SearchForRita” task, will be an exploratory task. So in this case, during the first task the user modelling is active, but during the second it will not. The user will not be interrupted by the adaptation module while navigation on the browser and exploring the application.

It is also important noticing that in the situation depicted in figure 7.10a, it makes no sense to break the task in two. Doing so would only introduce extraneous processing, since the learner would have to remember the context of the previous task. While performing the “RunTheApplication” task, the user will be classified and assisted. In the second task, the user modelling will be disabled, thus not receiving any kind of assistance.

### 7.5 Framework For Inquisitive Tasks

In section 6.6 was discussed how to develop Inquisitive Tasks. In figure 7.11 one can observe one example of the implementation of the proposed framework.

Figure 7.11 depicts an Inquisitive Task inside the “Build An App in 5 Minutes” tutorial. This tutorial was re-authored in order to accommodate these changes. This tutorial was used to test the developed framework, and the reasons that lead us to choose this particular tutorial are discussed in section 8.1. As can be seen in figure 7.11c, the learner can only advance in the tutorial, if the correct answer is selected.
7.6 Platform-specific Implementations

From the observations made, was understood that the undo command could be used with two different intents:

- Performing an undo command when the learner understood that the performed action did not contribute to solving the problem;
- Performing an undo command in order to understand the changes that an action had produced.

**Undo Support**

Since it was possible to undo actions (that affected both the application and the tutorial), one had to maintain the consistency of the implemented state machine. In the way the system was implemented, two different kinds of information were stored when the learner enters a new task:
7.6. PLATFORM-SPECIFIC IMPLEMENTATIONS

- **Current task to solve** - this is the task that the user will be solving;
- **State machine task** - which is the task that the user is being modelled.

Most of the time these two tasks are the same, but if the user performs undo commands, the current task to solve will be task $n - m$, being $m$ the number of undo actions that affected the tutorial, and the state machine task will still be on task $n$. This means that even though the learner went back several times, the consistency of the state machine is kept. This way, the user can rollback several times, but the state machine will only go back to the Reading State when he solves that task from which he rollback from.
8.1 Evaluation Methodology

As discussed, the developed framework was iteratively tested in order to fine-tune some parameters but also to find critical bugs. Some of the parameters that needed tuning were the average reading time, and the time taken in the fuzzy transitions. Three participants were selected that had no previous experience with the platform and asked them to complete the re-authored tutorial. These users did not participate in the test sessions described on section 8.1. This process allowed to find and correct bugs, on the implemented solution, that might have had negatively impacted the tests. Some bugs led to crashing the tutorial system, and if this had happened during the tests those participants would have to be discarded from the sample. After concluding the implementation was devised a testing scenario so that one could compare the developed solution against the original tutoring system.

Below the testing procedure is described.

Participants

The sample was composed of $N = 19$ participants. 6 participants were female, and the ages of the participants were comprised between 22 and 25, with an average of 23.21 years old.

All of the participants had at least a BSc degree in computer science. While one can consider these participants at least at a competent level in C++, Java and other technologies, none of them had previous contact with the Agile Platform. Thus no participant had ever seen the IDE or the tutorial system, making them novices in the learning curve of the Agile Platform.

Materials and Procedure

Participants were randomly assigned to one of two Service Studio versions, where one had the original tutorial system while other contained the implemented solution described in this document. A basic A/B test was performed. The participants were exposed to a particular tutorial entitled “Build an App in 5 minutes”. Several factors led to choosing this particular tutorial over others:

- This was the tutorial subjected to more usability tests thus, usability related problems were minimized;
- It taught how to develop a full application while most of the other tutorials taught how to extend an application;
- It is a fairly complex project for novice users to try without any assistance.

The possibility of developing a new tutorial was also considered, but was intended to test the system as closest to the production environment as possible. Some metrics had been previously collected about this tutorial, so a better analysis could be performed. The tutorials were authored in a minimal way, in
order to test the developed framework. Inquisitive and Exploration Tasks were inserted, while everything else remained constant.

After finishing the tutorial, a transfer test was presented to the learners.

The transfer test consisted of a task that was devised in order to create a different instance of the problems addressed in the tutorial. The tutorial taught how to build an application to list and edit contacts where the fields Name, Job Title, Phone and Email were stored (displayed in figure 8.1). The transfer test on the other side, asked the learners to build an application to list and edit employees, where the fields were: Name, Phone, Birthday, Email and Department would be stored (shown on figure 8.2).

![Figure 8.1: List Screen Built in Tutorial](image1)

![Figure 8.2: List Screen Built In the Transfer Test](image2)

The script used for the transfer test can be consulted in Annex A.1. It is important noticing that was used a mock-up instead of a screen shot of a similar application, in order to keep the script flexible, since the focus was not on the graphical interface design, but on building the logic behind it. Before starting the tutorial, participants were asked to navigate in the browser, for example searching a flight from Lisbon to New York. This was only intended to let the participants interact with the laptop, outside the platform that was being tested. This was done in order to get the users comfortable with the hardware, reducing errors that could be caused by usability or ergonomic difficulties.

The sessions were performed on a laptop with Internet connection, and both the Agile Platform of Outsystems and the Mozilla Firefox browser were installed. Also all the interaction with the platform was recorded using a screen recording software.

**Metrics**

After being exposed to the tutorial, each participant was given a transfer test. The time that took each participant to solve the transfer test was evaluated in order to observe how did the participant performed. The transfer test was divided into six checkpoints. For each participant was measured the time to complete each checkpoint both while performing the tutorial and while performing the transfer test.

The checkpoints were:
8.1. EVALUATION METHODOLOGY

- **Start a new Application** - This can be achieved in three distinct UI paths. This task will mark the beginning of the test;

- **Create and populate records** - The learner had to create entities to save data about employees. Also he had to populate the records by importing data contained in an spreadsheet. This could be accomplished in one of two ways:
  
  - Create a new entity, create the appropriate attributes, bootstrap an excel file. This was the process taught in the tutorial;
  
  - Import an Excel file. This will automatically create the records and the logic to populate them.

  We consider both being a valid solution to complete the checkpoint.

- **Create a list screen** - Create a web page to show the imported data;

- **Publish application** - Deploy and test the application;

- **Create an edit screen** - Create a web page that allows to edit the information of a given employee;

- **Publish application** - Deploy and test the application;

  It is important noticing that the “Create and populate records” and the “Create a list screen”, are interchangeable actions, so that:

  \[
  \{ \text{Create and populate records, Create a list screen} \} \leftrightarrow \{ \text{Create a list screen, Create and populate records} \}
  \]

  This means that it will not be relevant the order that the user reached one of these two checkpoints.

  Also, as one of the goals was not increasing the frustration levels of users, one had to assess the level of frustration introduced by the developed solution. To measure the frustration levels, was used the Geneva Emotion Wheel [29]. After completing the tutorial, participants were asked to pinpoint at most two emotions felt towards the tutorial system. In Annex A.3 can be found an example of the Portuguese adaptation used in this study. This instrument spreads twenty emotions across two axes: control and pleasantness. A third dimension is used to represent the intensity of those emotions. Closest to the circle center represents minimum intensity, and near the circumference represents maximum intensity.

![Geneva Emotion Wheel quadrants](image1)

![Geneva Emotion Wheel](image2)

Figure 8.3: Assessing the User Frustration Using the Geneva Emotion Wheel
8.2 Observed Results

Tutorial Execution

As stated, the time that each participant took to complete each checkpoint, was recorded.

All participants from both groups were able to successfully finish the tutorial. Figure 8.4 displays the time that each participant in the control group took to complete each of the checkpoints. The minimum time for this group was 11:41 minutes, while the maximum was 22:59 minutes. The standard deviation for this group was 3:22 minutes.

![Figure 8.4: Tutorial Execution Times with the Original Tutorial, in Minutes](image)

Since the developed framework provided an increasing guiding system, as well as inquisitive tasks, the users spent more time reading, searching for information and answering the posed questions. As can be seen from figure 8.6, participants who experienced the extended tutorial, on average took more time to complete it, than users from the control group. It is interesting to see that in the checkpoint “Create an Edit Screen”, the averages of the two groups almost converge.

![Figure 8.5: Tutorial Execution Times with the Developed Framework, in Minutes](image)

Since the developed framework provided an increasing guiding system, as well as inquisitive tasks, the users spent more time reading, searching for information and answering the posed questions. As can be seen from figure 8.6, participants who experienced the extended tutorial, on average took more time to complete it, than users from the control group. It is interesting to see that in the checkpoint “Create an Edit Screen”, the averages of the two groups almost converge.
8.2. OBSERVED RESULTS

Figure 8.6: Average Tutorial Execution Time

Frustration Levels

As was intended to assess the frustration levels, one could focus on the second and third quadrants of the Emotion Wheel, since they represent negative emotions.

Figure 8.7: Emotional Results

Was observed that only a small minority of participants felt negative emotions towards the tutorial system. That data is summarized in figure 8.7. It is important to notice that while using the Geneva Emotional Wheel, each participant can pinpoint from none to two emotions. In the “other” category, a user described “confusion” and another “sense of accomplishment”. In a follow-up session, participants were asked to explain the emotions that had been pinpointed. Both users that expressed negative feelings, related those emotions with one particular step in the tutorial.

Even though only participants who were exposed to the original version showed frustration, this might not be significant. This is because this particular step provides the same content to the learner, and exactly the same actions have to be performed in both versions of the tutorial. The fact is that throughout the realization of tests one came to understand that users tended to have some difficulties in
this particular step. It is the tenth step to be performed in the tutorial, and it explains how to change
the layout of a contact list, that was created in a previous step.

When inquired, the participant who demonstrated “Sense of accomplishment”, replied that with this
tutorial system he was building a real application. This made him feel good because of the fact that usually
while training one builds “toy” projects, but during this session he had developed a useful application
that later could be used and extended. One can also observe that involvement and wonderment were the
top voted positive emotions. This is relevant to the learning process in the long run, making the learner
engaged in the learning process. This makes it more probable that the learner will spend more time with
the learning materials.

**Transfer Test**

As described above, the time that each participant took to reach a given stage was recorded. As
discussed on sub-section 8.1 there were two valid ways to complete the “Create and populate records”
checkpoint. While most participants did what was taught on the tutorial (create an entity, create the
appropriate attributes, and finally bootstrap an excel file), three participants chose to import the excel
file. For this reason users 5, 10 and 17 do not have a time assigned to the “Create five attributes” and
“Create action to Bootstrap data” checkpoints, on figures 8.8 and 8.9.

<table>
<thead>
<tr>
<th>New Application</th>
<th>Create 'Employee' Entity</th>
<th>Create Five Attributes</th>
<th>Create Action to Bootstrap Data</th>
<th>Create the List Screen</th>
<th>Publish Application</th>
<th>Create Edit Screen</th>
<th>Publish Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>23:06</td>
<td>01:25</td>
<td>02:45</td>
<td>03:49</td>
<td>07:46</td>
<td>07:55</td>
<td>12:54</td>
</tr>
<tr>
<td>User 2</td>
<td>21:28</td>
<td>00:54</td>
<td>03:27</td>
<td>14:50</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>15:28</td>
<td>00:00</td>
<td>01:50</td>
<td>02:22</td>
<td>04:55</td>
<td>05:03</td>
<td>14:45</td>
</tr>
<tr>
<td>User 4</td>
<td>20:41</td>
<td>00:34</td>
<td>02:04</td>
<td>05:00</td>
<td>05:47</td>
<td>09:25</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>17:33</td>
<td>01:25</td>
<td></td>
<td>02:13</td>
<td>02:27</td>
<td>03:09</td>
<td>05:27</td>
</tr>
<tr>
<td>User 6</td>
<td>15:48</td>
<td>00:28</td>
<td>01:27</td>
<td>02:13</td>
<td>02:27</td>
<td>03:09</td>
<td>05:27</td>
</tr>
<tr>
<td>User 7</td>
<td>13:44</td>
<td>00:33</td>
<td>01:51</td>
<td>02:30</td>
<td>06:28</td>
<td>06:43</td>
<td>10:43</td>
</tr>
<tr>
<td>User 8</td>
<td>41:08</td>
<td>41:24</td>
<td>00:48</td>
<td>01:08</td>
<td>01:20</td>
<td>01:45</td>
<td>06:35</td>
</tr>
<tr>
<td>User 9</td>
<td>18:45</td>
<td>01:52</td>
<td>02:47</td>
<td>02:57</td>
<td>06:58</td>
<td>07:07</td>
<td>12:09</td>
</tr>
</tbody>
</table>

**AVERAGE** | 05:24 | 02:07 | 04:21 | 04:48 | 05:30 | 09:43 | 09:59 |

* Denotes that the user was not able to finish because he could not remember
** Denotes that the user was not able to advance further due to platform constrains

Figure 8.8 depicts the time taken for each participant of the control group to solve the transfer test. Two users were not able to finish the transfer test, users 2 and 4. The minimum time for participants who finished the transfer test was 5:38 minutes, while the maximum was 14:50 minutes. The standard deviation was 3:30 minutes.

In figure 8.9 can be observed the time that participants who experienced the developed framework took. Three participants from this group were not able to finish the transfer test (users 10, 13 and 18). The minimum time it took for a participant to complete the transfer test was 5:34 minutes, while the maximum was 9:21 minutes, with a standard deviation of 1:40 minutes.

On figure 8.10 can be seen that for the two first checkpoints the time that took participants to complete was almost the same. From the third checkpoint onwards, there is a noticeable difference in the
8.3 DISCUSSION

Figure 8.9: Transfer Test Execution Times with the Developed Framework, in Minutes

<table>
<thead>
<tr>
<th>New Application</th>
<th>Create 'Employee' Entity</th>
<th>Create Five Attributes</th>
<th>Create Action to Bootstrap Data</th>
<th>Create the List Screen</th>
<th>Publish Application</th>
<th>Create Edit Screen</th>
<th>Publish Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 10</td>
<td>22:21</td>
<td>00:43</td>
<td></td>
<td>02:09</td>
<td>02:22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 11</td>
<td>20:28</td>
<td>00:19</td>
<td>01:45</td>
<td>01:58</td>
<td>02:57</td>
<td>03:02</td>
<td>05:10</td>
</tr>
<tr>
<td>User 12</td>
<td>26:44</td>
<td>00:44</td>
<td>02:24</td>
<td>03:35</td>
<td>05:12</td>
<td>05:52</td>
<td>08:52</td>
</tr>
<tr>
<td>User 13</td>
<td>33:03</td>
<td>00:33</td>
<td>01:18</td>
<td>01:32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User 14</td>
<td>14:05</td>
<td>00:43</td>
<td>01:45</td>
<td>03:17</td>
<td>04:12</td>
<td>04:21</td>
<td>06:37</td>
</tr>
<tr>
<td>User 15</td>
<td>19:24</td>
<td>00:00</td>
<td>01:27</td>
<td>02:13</td>
<td>02:30</td>
<td>03:44</td>
<td>06:31</td>
</tr>
<tr>
<td>User 16</td>
<td>24:45</td>
<td>00:28</td>
<td>01:20</td>
<td>01:57</td>
<td>02:20</td>
<td>03:12</td>
<td>05:28</td>
</tr>
<tr>
<td>User 17</td>
<td>20:36</td>
<td>02:29</td>
<td></td>
<td>03:39</td>
<td>03:47</td>
<td>05:45</td>
<td>06:37</td>
</tr>
<tr>
<td>User 18</td>
<td>15:26</td>
<td>00:26</td>
<td>01:45</td>
<td>02:55</td>
<td>03:23</td>
<td>03:45</td>
<td>**</td>
</tr>
<tr>
<td>User 19</td>
<td>15:58</td>
<td>01:21</td>
<td>02:32</td>
<td>03:26</td>
<td>03:27</td>
<td>03:52</td>
<td>06:17</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>00:47</td>
<td>01:47</td>
<td>02:37</td>
<td>03:19</td>
<td>03:46</td>
<td>06:23</td>
<td>07:06</td>
</tr>
</tbody>
</table>

Figure 8.10: Average Transfer Test Execution in Minutes

As can be observed, this framework allowed for learners to decrease their problem solving times, when compared with the original tutorials. It is interesting to note that even the learner that took more time while experiencing the developed solution, performs below the average of control group.

8.3 Discussion

Participants in the control group were able to finish earlier the tutorial. This was predicted since the re-authored tutorial contained Inquisitive Tasks, that would incentive users to stop and meditate on the lessons learnt. Also the developed framework provided time for reading and during these periods there would be no clues on which elements needed interaction to advance in the tutorial. Since the control group was given early assistance on the elements that regarded attention, these users were able to perform the actions earlier than those on the test group. On average the control group was able to end 1 minute and 48 seconds earlier.

Was validated that the developed framework did not increase the frustration levels of the participants. Of the ten participants, none demonstrated negative emotions towards the tutorial system. On the other
hand, two participants on the control group, described negative emotions. The first “anger” and the second “confusion”. The dominating positive emotions were “involvement”, “wonder” and “astonishment”, being that the control group tended to choose “involvement”, whereas the test group spread more evenly across “involvement” and “wonderment”.

In the transfer test, on average the test group performed better, but two on ten participants could not remember how to advance in order to complete the tutorial. Although slower on average, the control group only contained one of nine participants that was not able to remember how to complete the given task. This is relevant because although the developed framework is able to on average reduce the time of a transfer test, the number of unsuccessful participants is higher.

It is also relevant to discuss the case of the two participants who were not able to finish the transfer test due to platform restrictions.

Both users were not able to finish the task, because when creating the “Employee” entity, they deleted the primary key. Since the Agile Platform performs instant error checking, instead of only presenting them at compile time, these users saw an error indicating that an entity could not be only composed by an attribute with the property “AutoNumber” set to true. These two users noticed that the application had an error, and tried to address it by deleting the primary key. When the primary key was deleted, this error disappeared, reassuring them that the application was correct. The problem was that later, the platform let these users create a list screen but did not allow the creation of an edit screen. This was because, in order to build an edit screen, one needs to uniquely identify the employee that will be edited.

This happened both in the developed framework and the control group, and the tutorial could not predict this behaviour. Since this is an embedded tutorial system, the error presented to the user is triggered on the platform side and not on the tutorial environment. This shows how usability problems on the platform can impact the learning process.

Summarizing, the developed framework was able to increase the average time a learner spent with the learning materials, without increasing the frustration. On the original version of the tutorial, the learner was immediately given clues, on where to interact in order to advance. This made learners able to solve faster the tutorial, but lowering the retention rate on the presented subject matter. Although on average users from the test group took less time to perform a retention test, but fewer were able to complete it.
Conclusions

Although this dissertation addressed the problem of low retention rates presented by software training tools and mechanisms, the problem is far from being solved. The developed framework allowed users to solve a retention test in less time than the control group, with no subjects demonstrating negative emotions. But the fact is that some users were unable to finish the retention test.

This dissertation discussed, as an example, the learning curve steepness of an Integrated Development Environment. It was argued that these tools need to be complex since they are intended to solve complex problems, but they do not need to be complicated, in the sense that they mislead users [1, p. 2]. It was also argued that this is not restricted to IDE’s, but other inherently complex systems such as image editing software, 3D modelling tools, and many others. The problem faced by the product owner is finding the correct balance between the needs of a novice user and the ones of an expert while keeping the development costs low. This fine equilibrium can be the difference between getting satisfied users, or creating a steep learning curve that leads to users giving up in the middle of the learning process.

The platform that was presented as a case study, had an embedded tutorial system on the IDE. This platform was used to develop data-centric applications using a visual paradigm. The Service Studio (the IDE for this domain specific language) was the user’s entry point to the platform, thus the embedded tutorials were able to assist in the first stages of the learning curve. What was observed was that these tutorials, albeit the low frustration levels, provided a low retention rate. This is a problem, because the tutorials were not correctly addressing the acquisition-retention cycle, thus not efficiently helping the users to build schemata.

To address this issue, a framework was developed, not only providing a different adaptation system, but also to author the tutorial to increase the retention rates on the users. With this framework, one is able to assist the user only when he is not able to advance. This is intended to stimulate schemata acquisition, since he is encouraged to actively process the learning materials. To provide this adaptation a state machine with fuzzy transitions was implemented. This is a trade-off between the complexity presented by the Intelligent Tutorial Systems, and the scripted interactive tutorials. The framework implements three states:

1. Reading, where no assistance is provided to the user. This time is automatically calculated based on the amount of text the user is given to read. This is intended to stimulate the user to read the information provided and start addressing the problem;

2. Cue, where a small cue is given, such as a dimmed arrow pointing to the element in the user interface that the user needs to interact with;

3. Guided, where explicit help is provided to the user, when he is not able to complete the task with a lower level of assistance. A constantly moving arrow was used to guide the user to the interface element that needed attention.

Aside from the extension on the assistance provided to the user, was also suggested that a combination of four task types should be used to increase the learning retention rates. These task types were:

1. Action Tasks - these tasks incentive the user to interact with the system to solve a given problem. They provide declarative knowledge;
2. Informative Tasks - provide declarative knowledge;

3. Inquisitive Tasks - test the user knowledge in order to improve the acquisition-retention cycle;

4. Exploration tasks - let the user explore the system, without being assisted. They provide relaxation periods as well as incentive discovery by trial and error.

After implementing the developed framework on the tutorial system of Service Studio, was tested with $N = 10$ participants, while another $N = 9$ participants were part of a control group. Participants were asked to perform a tutorial, after finishing they were asked to pinpoint at most two emotions felt towards the tutorial system, and finally a transfer test was presented.

The observed results were not as expected. While on average the users that experienced the developed framework were able to present a quicker solution on the transfer test, three out of ten were not able to finish it. One was due to performing an action that later would make it impossible to proceed, due to platform restrictions. Another two were not able to successfully solve the transfer test. On the control group two out of nine participants were not able to finish the transfer test. The first user also deleted a primary key, leading to the impossibility to later create a screen to edit data. The second could not remember how to create a list screen and move forward in the transfer test.

As described, the developed framework did not negatively impacted the frustration levels presented by the users, being that all participants of the test group pinpointed positive emotions after completing the tutorial. On the other hand, two participants from the control group demonstrated negative emotions towards the tutorial system. Albeit not impacting the emotions, and on average decreasing the time taken to solve a transfer test, the test group still contains users who are not able to finish it. This means that further work must be done to validate the findings, but also to decrease the number of users not being able to finish a transfer test and further decrease the time taken to perform it.

Although this dissertation addresses the problem by implementing a framework for an assisting tutorial system, the fact is that the problem must be addressed in a wider context. The learning process cannot be evaluated without considering the platform itself. The learning tools described on chapter 4 are devised to soften the learning curve, and so efforts must be made to not only improve these tools, but also to address the root problem: the increasing complexity of platforms and systems are fit to expert users, but overburden the novice one. This in itself is not a problem, the problem arises when complex systems are designed in a complicated way. Usability testing and improvements are only one of the few techniques that are increasingly proving to be effective in addressing these issues.

Future Work

Because of resources and time constrains, only a relatively small number of participants participated in the study ($N = 19$). We believe that further tests should be made with more participants, in order to validate the results. For the same reason there was no possibility to evaluate how the learning process progresses over a vast period of time. Currently we do not have data about evolution of the learners over a large timespan. This could provide interesting insights not only about the evolution of novice users, but also from those who are higher in the Dreyfus Model. Participants in this study only performed one tutorial that taught how to build an application with listing and editing capabilities. But the platform on which the solution was implemented is complex, thus this tutorial is only able to present a superficial approach to the platform. Even though this lesson was considered only superficial by an expert, for a novice user it was very complex, and we believe that a novice user could not build the application without any level of assistance.

Since one participant pinpointed “anger” and another “confusion” we believe that there is still work to be done in the tutorial authoring. This only happened on the control group, but the same content was present on the re-authored tutorial, making us hypothesize that the same issue that negatively impacted the frustration of these users could also arise on the developed framework. This only shows how relevant
is the content of the tutorial, not only in terms of explanation but also in terms of presentation. The same content can be presented in several ways, but not every is able to decrease the extraneous processing of novice users, supporting schemata acquisition.

Also, because of the architecture of the tutorial system deployed on Service Studio, it became difficult to understand what actions performed by the learner, contributed to the goal of the task. As explained on section 7.1, the platform only validated states. This provided a very flexible approach, but did not allow creating a model that classified the relevance of the tasks being performed. We think that it would be very interesting to change the architecture of this system, or build a different system, on which this model was implemented. A comparative study between the costs of implementing and maintaining this system, and the one that we presented could be performed, and would be of great interest.

Another interesting work would be to investigate if there is any correlation between the frustration level that will lead to a user giving up during a tutorial, and the level of the user according to the Dreyfus Model. This can be complex to measure since it cannot be performed in a short period of time or in a controlled environment. To do so, metrics need to be collected over a long period, without introducing changes in the tutorial system or the platform. This work cannot be performed in a closed environment such as a laboratory, since we intend to measure the frustration levels of a software tool in the real world. We believe that this can only be accomplished through instrumentation and remotely sending data that latter could be analysed. Other techniques may be too intrusive for the user, thus introducing errors in the study.
Bibliography


73
Appendices

A.1 Tranfer Test Script
A.2 Khan Academy Badges
A.3 Geneva Emotion Wheel
A.4 Tutorial Comparison
Task 1

You know your boss lately has been thinking about creating an intranet.

Try to develop a web application to manage the list of employees of your company. In your desktop you have a spreadsheet that contains the information about all the company’s employees.

Task 1.1

Build an application where you can view your company’s employees.

Figure A.1: First Task of Transfer Test
Task 1.2

Extend your application so that you can add new employees to the list and edit their information.

Figure A.2: Second Task of Transfer Test
Figure A.3: Khan Academy Badges
Figure A.4: Geneva Emotion Wheel Used
Deprecated Non-Interactive Tutorial

![Non-Interactive Tutorial Image]

Embedded Tutorial System

![Embedded Tutorial System Image]

Figure A.5: Comparison between the Non-Interactive Tutorial and the Embedded Tutorial System