Softening the Learning Curve of Software Development Tools

João Fernandes
Instituto Superior Técnico
joao.n.fernandes@ist.utl.pt

Prof. Dr. Mário Rui Gomes
Instituto Superior Técnico
mario.gomes@ist.utl.pt

Prof. Dr. Rui Prada
Instituto Superior Técnico
rui.prada@ist.utl.pt

Eng. Lúcio Ferrão
OutSystems
lucio.ferrao@outsystems.com

ABSTRACT

In the software industry, albeit wide spread 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 within the OutSystems IDE presents low retention rates, which makes users perform poorly when a transfer test is presented. It is discussed how this embedded tutorial system was extended with the developed framework. The challenge lies in combining the simplicity of interactive tutorial systems, with the tailoring offered by Intelligent Tutoring Systems.

Categories and Subject Descriptors

K.3.1 [**Computer Uses in Education**]: Computer-assisted instruction (CAI)

Keywords

Learning, Embedded Tutorials, Improving Knowledge Retention

1. INTRODUCTION

Image editing software, 3D modelling tools and Integrated Development Environments (IDE's), are examples of tools devised to assist in solving complex problems. These tools often deploy cutting edge technology, and are used by experts in their fields. The problem is the time and effort needed for a novice user to master these tools, and became productive. The learning curve for this products tends to be steep, meaning that will take months of practice and frustration, in order for a user to evolve in expertise.

This document presents as a case study the learning curve of the OutSystems IDE. This platform supports full life-cycle development of enterprise Web applications, and presents a steep learning curve to novice users. This allows illustrating the typical learning tools, and their adequacy according to the expertise of the user. Even though a set of tools are deployed, novice users still take much time to evolve, since these tools present low retention rates. In order to address the problem of steep learning curves, this document presents a framework that intends to: **increase the learners' retention rate**, meaning that learners should be able to perform better given a transfer test, not slowing the performance of advanced users, and not to significantly increase the maintenance costs.

2. THE LEARNING CURVE OF AN IDE

This section describes the learning curve of the OutSystems IDE. So it becomes relevant to understand the several levels of expertise, in order to detail the challenges faced by novice users.

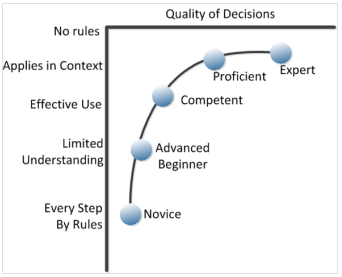


Figure 1: Dreyfus Model Of Skill Acquisition

One can classify the several learning stages, using the Dreyfus model of skill acquisition. This model is detailed on [4, 6]. According to this model, what distinguishes the several expertise levels is the increasingly capability of the expert to execute a task at a subconscious level. It makes him able to act more smoothly and able to perform several tasks at the same time.

The **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. The **Advanced beginner** starts to exploring new ways to do a task, but probably will face difficulties acting on their own, since they give the same level of importance and attention to all the attributes of a task. A **Compe-**

tent user is able to understand how the performed actions fit on the long-term plan, since they are starting to build the correct mental models. The **Proficient** user is able to observe deviations from the normal patterns and understands which attributes of a task deserve more attention. Finally, the **Expert** has developed gut feelings and can understand the difference between irrelevant details and crucial ones.

In order to better illustrate the problem posed by software training, will be described the learning tools devised by OutSystems, to assist users in the learning process. As part of the development team, one had the opportunity to attend several hours of usability sessions, on which participants were exposed to tutorials and then performed transfer tests. This provided insight of the frequent problems when interacting with the platform, and contributed to understanding users'expectations and mental models.

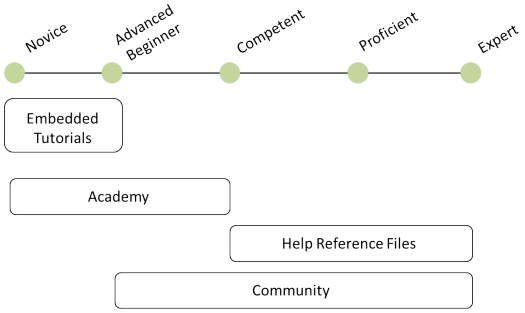


Figure 2: OutSystems Learning Path

Figure 2 shows that, in order to assist users, there are **interactive tutorials**, which are embedded on the IDE, the **academy**, an e-learning system, **help reference files**, and a **community**, on which users trade their knowledge.

The interactive tutorials were developed to assist novice users, but was found that these delivered low retention rates. After being exposed to a tutorial, participants performed poorly on transfer tests.

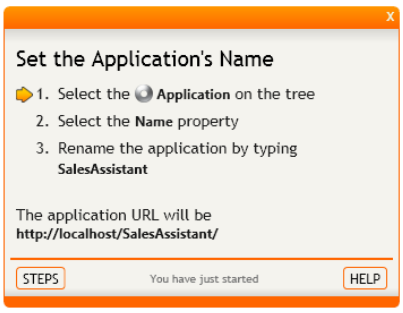


Figure 3: Tutorial Embedded on the IDE

A tutorial is comprised of window embedded on the IDE, which is depicted in figure 3, and an arrow, that points to the elements on the interface, that the user must interact with. A tutorial contains several steps and each step contains several tasks to perform. 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. Since there was a constant guidance, users would follow the arrow, not generating mental models. They were able to successfully finish the tutorial, but performed poorly when presented with a similar problem.

The embedded tutorials are adequate to assist novice users in their learning process, nonetheless, users still perform poorly given a transfer test. Since improvements in the assistance can be made, it is relevant to see what other tools and techniques are deployed, and their adequacy to novice users.

3. RELATED WORK: LEARNING TOOLS

This section surveys approaches to soften the learning curve, and increase software training retention levels. Since this document focus on computer assisted learning, only tools and technology that fall in this category will be discussed.

3.1 Non-Interactive Media

Books, instructionals and other non-interactive media can provide help when learning a new subject or evolving previous 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**, since they allow searching for content. One example is the help reference from TortoiseSVN [1], a subversion client. **Linked with the product**, presenting some degree of coupling with the product. An example can be pressing the "F1" key makes a context aware page to open. This is found on Microsoft Office Excel[2]. Writing a function and pressing "F1" will provide help on that particular function. And **detailed**, meaning that they are highly accurate and extensive, providing information almost about every feature. One example is Microsoft's web page for development reference[3].

One can divide non-interactive media in two distinct categories: **continuous** and **discrete**. The first is developed to incrementally support the learner, making a soft transition between the topics discussed and allowing the learner to get a sense of where a particular element fits in the overall process. An example can be the documents to learn Blender(a 3D modelling tool), that can be downloaded [4]. 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. An example is the help reference files shipped with any product.

3.2 Video Tutorials

Video tutorials are a mean devised to provide context to the learner. This brings the learner closer to a one-on-one learning experience. 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

[1]http://tortoisesvn.net/
[2]http://office.microsoft.com/
[3]http://msdn.microsoft.com/en-us/library/
[4]http://gryllus.net/Blender/3D.html

site[5]. With higher bandwidth this learning method has been spreading and is probably one of the most used within novice users, because it is **direct** since the video provides context and almost no previous knowledge is required. 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. One can also **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 [7]. The main problem with these tools is that they do not imply generating mental models. 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.

3.3 Discussion Forums

From the beginning of the Internet and more recently with the boom of Web 2.0, groups of people have been uniting 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[6], and communities built to perform questioning and answering (Q&A) about software development[7].

These tools proved fit to address *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. On the downside, it becomes difficult and costly to maintain the redundant content and they tend to be avoided by the novice user. These 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.

3.4 "Gamification"

"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 [3]. Some of the mechanics introduced in these systems are: achievements, levels, points, rewards and so on. 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. 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. On the downside is the fact that the learning content will be expensive to develop since it must be closely tied with the game mechanics.

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.

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 [2].

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. 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.

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. 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 } $\Leftrightarrow$ { changing font to arial, changing a text to bold}

3.5 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 experts. 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. Succinctly, successful abstraction layers are those who stop being used as the user evolves in expertise.

[5]http://download.autodesk.com/us/3dsmax/skillmoviesv2011
[6]http://www.javaranch.com/
[7]http://stackoverflow.com/

These tools teach 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, 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.

AutoCAD

AutoCAD is a software application for both 2D and 3D design. 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 a wider spectrum 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. To draw a line, a novice user might: select the line tool, click on the (0,0) point on canvas and click again on point (100,100) on canvas. Instead, the expert user will use the command line to achieve the same result, using: LINE 0,0 100,100 on a command line.

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.

3.6 Interactive Tutorials

Another approach to soften the learning curve for novice users 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. On the downside, these systems are not adequate for more advanced users, since it will not be able to explain in detail the topics covered. 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.

Try Ruby

Try Ruby[8] is a web site that provides an interactive Ruby[9] tutorial. In this site the user can find a console, where all the features and built-in methods of Ruby 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>.

For every tutorial step, the user is presented with an explanatory text and a new expression to be inserted. The command or method that is relevant to the step is highlighted. This way, the learner can scan the text for these keywords. Also highlighted, are the commands that the learner must insert to advance in the tutorial.

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. Even when inside the tutorial, the user can explore and develop software with no restrictions.

3.7 Intelligent Tutoring Systems

The Intelligent Tutorial Systems (ITS) community addresses some of the problems that this document does, providing adaptive learning experiences. Examples in the industry are hard to find, since most of these systems are used within the academic community.

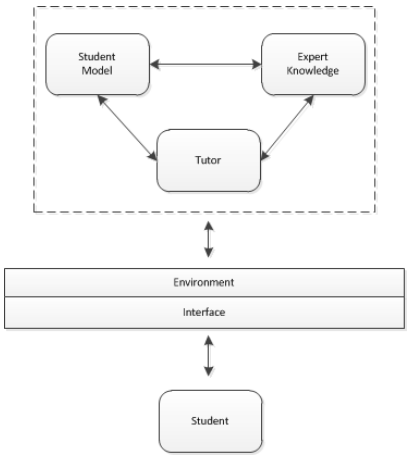


Figure 4: Intelligent Tutoring System Architecture

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 [10]: **Expert module**, where the domain knowledge is coded and can be inferred upon. It can be a simple input-output system (known as black-box), or can be a system that encodes knowledge which can be inferred upon (known as glass-box). **Student module**, which allows to develop a model of the students' knowledge. Finally the **Tutor module**, which will select how to act to deliver knowledge or knowledge building opportunities to the learner. It

[8]http://tryruby.org/
[9]http://ruby-lang.org/about

can present new exercises, provide corrective feedback or other means that it finds fit to help the learner.

One example that is interesting to analyse is the Lumière Project. While this system is not a tutoring system, it tries to assess the user difficulties and provide assistance. What makes this example valuable is the fact that is was integrated on a product. This system also comprises a student module that performs user classification, and uses Bayesian belief networks to implement functionalities similar to the expert module described above.

Lumière Project

Lumière Project [5] was an approach that intended to assess the user goals and difficulties using Bayesian Belief Networks. The most known contribution of this work was Microsoft Office Assistant[10] (also known as Clippy), which implemented several components presented on the Lumière Project.

This system used instrumentation in order to detect user needs. Some of the heuristics indicative of this were: continuous **search** search for an element in the User Interface **introspection**, where a sudden stop of interaction with the system happens. And producing **undesired effects** and undoing a set of commands.

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: counting the number of times an action had occurred in a given period, testing if an action had occurred, and testing the time the user spent dwelling. 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.

On the downside, Intelligent Tutoring Systems are very difficult to develop and maintain, and 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.

3.8 Discussion

After surveying some of the tools and technologies that address the problem posed by software training, figure 5 summarizes their adequacy according to the Dreyfus Model.

Interactive Tutorials, Intelligent Tutoring Systems and e-Learning are able to address the entire spectrum of expertise, because the content delivered in these systems can be adjusted to each of the several stages of the learning process.

When learning with **Non-interactive**, learners perform a high amount of context switching. **Video tutorials** provide context to the learner and reduce the context switching.

[10]US Patent 6262730, Filled 1998-20-11, Issued 2001-17-7

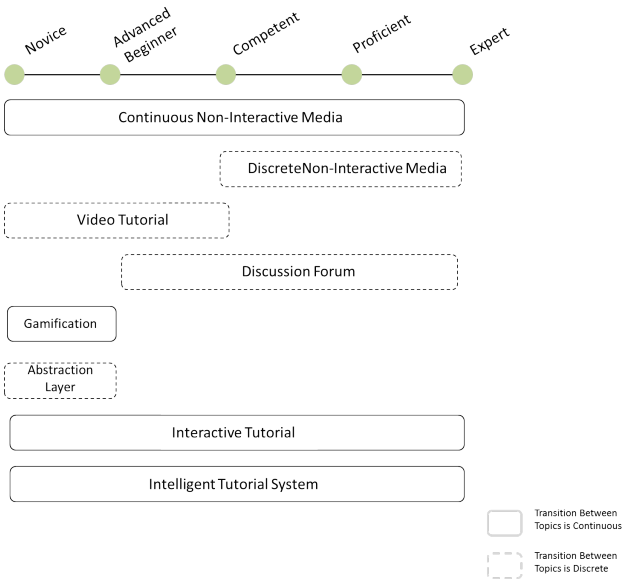


Figure 5: Comparison Between Learning Tools and Their Dreyfus Model Range

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

Trying to apply an Intelligent Tutoring System approach, would not solve the problem given the high maintenance costs presented by those systems. But the level of assistance provided is adequate to novice users. One approach is extending an Interactive Tutorial in order to provide some degree of adaptation to the learner.

4. INCREASING RETENTION

In order to provide several guidance levels, the embedded tutorials of Outsystems were extended with an adaptive guidance system. And given that users were not reading most of the content, there is a period without guidance (Reading State), but also Inquisitive Tasks that will stimulate users to read and address the problem without assistance, making users build mental models.

Similarly to the Intelligent Tutoring System, the developed architecture is composed of two modules: a **user modelling**, that monitors user events and classifies the user accordingly. The second, **adaptation**, which will define the

level of assistance that the user will receive.

To achieve this, this framework implements: a **baseline** which are a set of metrics that allow specify the performance of an average user in a given tutorial. A **user history**, that saves information about how many tutorials has the user performed and their success levels. An **Initializer**, that will allow to adapt the general baseline to the particular experience of a user. And finally a **Recorder**, which will record every interaction of the user with the system. This allows to identify if the user needs assistance. This information can also be stored for later use if necessary. Figure 6 shows how this process is performed.

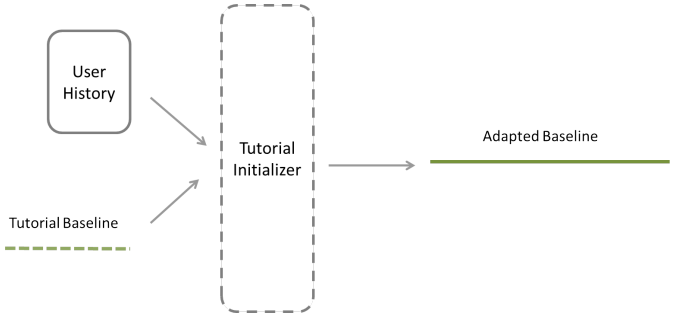


Figure 6: Generation of an Adapted Baseline from the User History

In order to summarize the data taken from all the tasks, was opted to only use the time component. The expected time to complete the task was evaluated, and the time it took for the participant to solve that 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 subsection 3.7. Also, this was done having in consideration the limitations of the platform on which this framework was implemented.

The tutorial system in Service Studio validates 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 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. On the other hand, it is not possible to understand if a performed action is relevant to solving a given problem.

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

This framework will only consider tutorials who have been successfully finished. Although 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.

4.1 Increasing Guidance System

Given that users where provided with too much guidance in the embedded tutorials, making them unable to build solid mental models, an increasing guidance system was implemented. This system is devised to provide the minimal guidance possible, while not increasing frustration levels. 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 [13]. 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 are: **Reading State**, which is intended so that the learner is focused on reading the presented text and starts to address the problem without any help; the **Cue state**, which 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. And finally, the **Guided State**, which 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.

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 insert them as new states in the state machine. These events are: **Help**, which will show a help message associated with the current task; and the **Lost** event, which after a great number of interactions with elements not relevant to the task and a great amount of time is spent trying solving the task, will ask if the user wants to search for help outside the tutorial system.

The tutorial system embedded in Service Studio was extended with the state machine depicted on figure 7.

In the embedded tutorial comprised on Service Studio, the three levels of assistance were implemented as depicted on figure 8. When in Reading State, the learner will not be guided. When in Cue State a dimmed arrow is displayed, that will not be noticeable by the users' peripheral vision. Finally when in Guided State, a moving arrow will point to the relevant UI element, and being noticeable by the peripheral vision.

Having in consideration the fact that users were scanning for words such as: **click**, **double-click** and **drag** and following the arrow, which makes a tutorial easy to finish but with low retention rates, the Reading State is crucial to provide users with the opportunity to develop mental models. Since one of

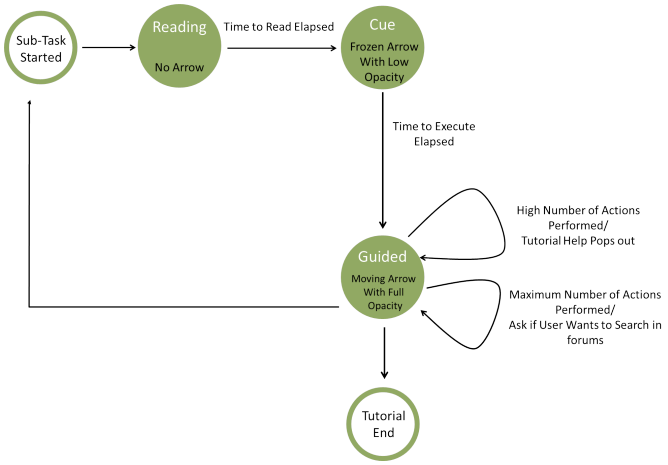
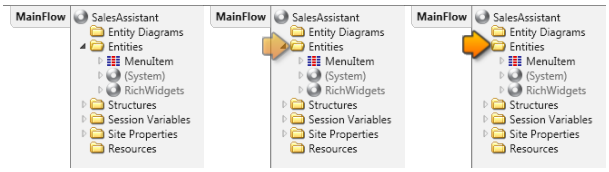


Figure 7: Implementation of the User Evaluation States



(a) Reading State (b) Cue State (c) Guided State

Figure 8: Increasing Help System

the goals of this framework is not increasing the maintenance costs of the tutorial system, only at run-time the amount of text should be calculated, and the time given to the Reading state should be set.

Given that on Service Studio an event is raised when the user is entering a new task, one could easily implement a lazy evaluation system to compute the amount of time that the user should be given in the Reading State. 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 happens, 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.

4.2 Framework For Building Tutorials

In order to captivate users' attention, and making them able to build mental models, the provided guidance must be adequate, as also as the content should be presented in an engaging way.

One came to understand that there was support for two task types in the embedded tutorials of Service Studio: Action and Informative tasks. Instead a tutorial should comprise four different task types: **Actions tasks**, where the learner must interact with the system, introducing changes. These tasks stimulate procedural learning; **Informative Tasks** intended for the learner to absorb some information. These tasks deliver declarative learning. **Inquisitive Tasks**,where the learner will be encouraged to stop and review the subject matter; and **Exploration 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 framework proposes the following combination of tasks, as a guideline to constructing a tutorial: Start the tutorial with an informative task. It should explain what the goal of the tutorial is, and give an overview of the process. The tutorial author should guarantee that even if the learner does not read this step, the leaner will be able to perform the tutorial. Deliver a set of action tasks. The correct number is dependent on the complexity of the task, but from our experience the learner should experience around five of these tasks. This is also supported by [8]. 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 relationships between them and forming schemata. Then 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. Near the end of the tutorial, present a review question, that asks about the core idea presented in the tutorial. And 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 our experience, we believe that delivering 80% of procedural knowledge and 20% of declarative one, is a good balance. This allows keeping the learner motivated and engaged in the learning process. According to Csikszentmihaly [2], 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 our framework, the stress periods will be provided by the Action and Inquisitive Tasks, and the relaxation periods by the Informative and Exploration tasks.

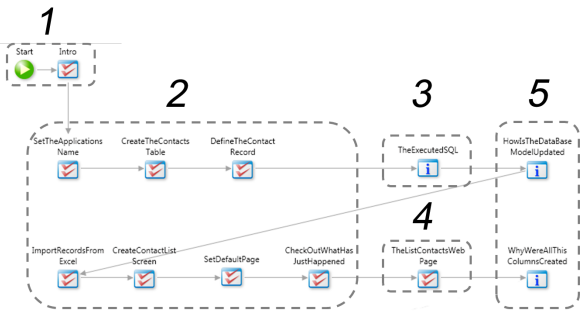


Figure 9: Tutorial Re-authored Using the Proposed Framework

In figure 9 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 is incorporated: an **Introduction (1)** that presents a summary of what the user

will learn in this tutorial. Several **Action Tasks (2)**, being that each of them teaches the user how to perform certain tasks to achieve the goal of the tutorial. Two **Informative Tasks (3)** which provide declarative knowledge. One **Exploration Task (4)**, where 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. And two **Inquisitive Tasks (5)** where the user will experience inquisitive tasks after an informative or exploration task.

Several factors led to choosing to re-author this tutorial: 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. And 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.

4.3 Framework For Inquisitive Tasks

In order to increase the challenge, and help users build mental models, Inquisitive Tasks were introduced in the tutorial system. This decision is supported by Jakob Nielsen in [9]. 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 we are addressing: users scanned through the text without absorbing the content. But it is not enough to introduce Inquisitive Tasks, it is also important to know how to correctly curate their content. This document presents a set of guidelines, based on a psychology background provided by Mayer *et al* in [1] and by Schank in [11].

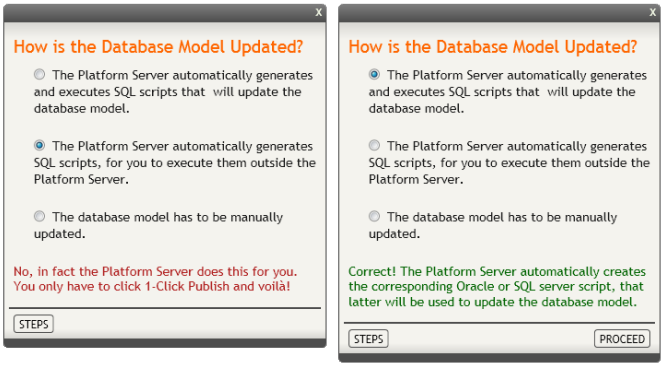
Do not present options such as "all of the above"/"none of the above". These kinds of options do not help a novice learner building mental models since they are too vague. Questions should be written in an informal tone. This will ease the pressure on the user, not introducing additional stress to the tutorial. Also, every hypothesis should have an associated explanation. It should state why that hypothesis 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. Finally, 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.

Figure 10 depicts how Inquisitive Tasks were implemented on the embedded tutorial system of Service Studio. As can be seen, the learner can only advance in the tutorial, if the correct answer is selected.

5. EVALUATION METHODOLOGY

Participants



(a) Wrong Answer (b) Correct Answer

Figure 10: Questions Inside The Tutorial

After concluding the implementation was devised a testing scenario so that one could compare the developed solution against the original tutoring system. 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 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 asked to pinpoint at most two emotions towards the tutorial system using the Geneva Emotion Wheel [12]. Then, participants were wasked to solve a transfer test, which presented a different instance of the same problem taught in the tutorial. The time that took each participant to solve the transfer test was evaluated in order to observe participants performance. 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: **Start a new Application** which can be achieved in three distinct UI paths. This task will mark the beginning of the test. **Create and populate records** where participants had to create entities to save data about employees. Also he had to populate the records by importing data contained in an spreadsheet. **Create a list screen**, on which participants needed to create a

web page to show the imported data. **Publish application** where participants would deploy and test the application. **Create an edit screen**, on which participants had to create a web page that allowed to edit the information of a given employee. And finally **Publish** the final application.

5.1 Observed Results

All participants from both groups were able to successfully finish the tutorial. The minimum time for the control group was 11:41 minutes, while the maximum was 22:59 minutes. The average was 16:33 with a standard deviation of 3:22 minutes. For the test group, the minimum was 12:15 minutes, while the maximum was 26:26 minutes. The average was 18:21 with a deviation of 4:10 minutes. As can be seen from figure 11, 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.

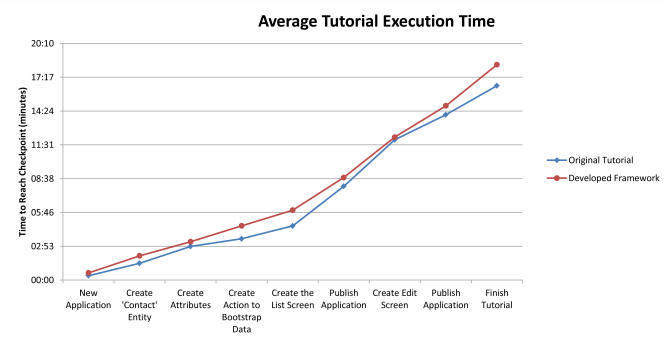


Figure 11: Average Tutorial Execution, in Minutes

Was observed that only a small minority of participants felt negative emotions towards the tutorial system. That data is summarized in figure 12.

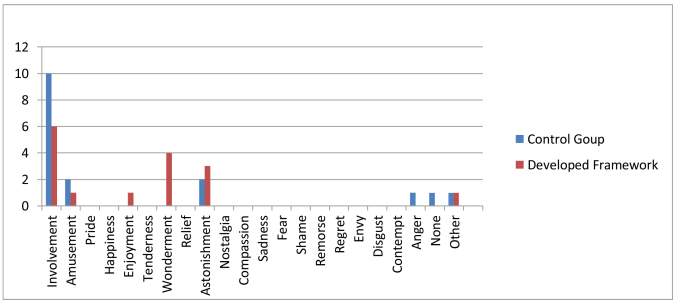


Figure 12: Emotional Results

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.

Two participants from the control group were not able to finish the transfer test. The minimum time for participants who finished the transfer test was 5:38 minutes, while the maximum was 14:50 minutes. An average of 9:59 minutes with a standard deviation was 3:30 minutes.

Three participants from the test group were not able to finish the transfer test. 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 an average of 7:06 minutes and standard deviation of 1:40 minutes.

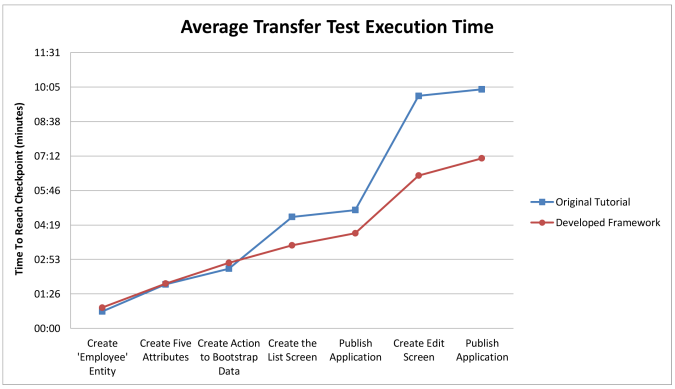


Figure 13: Average Transfer Test Execution in Minutes

On figure 13 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 time taken. As can be observed, the framework allowed for learners to decrease their problem solving times, when compared with the original tutorials.

5.2 Discussion

Participants in the control group were able to finish earlier the tutorial. This was predicted since the re-authored tutorial contained Inquisitive Tasks, which 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.

It 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 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 0.0(7) 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.

6. CONCLUSIONS

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.

Future Work

Because of resources and time constrains, only a relatively small number of participants participated in the study ($N = 19$). Further tests should be made with more participants, in order to validate these findings. For the same reason there was no possibility to evaluate how the learning process progresses over a vast period of time. Currently there is no 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.

Although this document addressed 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. It is not enough to develop tools to assist the learner. These tools should be integrated in the product, and efforts should be made to make the product adequate to novice users, making processes and tools simpler to understand and use by those users.

7. REFERENCES

[1] R. C. Clark and R. E. Mayer. *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning.* Jossey-Bass Inc.,U.S., 2nd revised edition edition, 2007.

[2] M. Csikszentmihaly. *Flow: The Psychology of Optimal Experience.* HarperCollins Publishers, 2008.

[3] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon. Gamification. using game-design elements in non-gaming contexts. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 2425–2428, New York, NY, USA, 2011. ACM.

[4] S. E. Dreyfus and H. L. Dreyfus. A Five-Stage Model of the Mental Activities involved in Directed Skill Acquisition. Technical report, 1980.

[5] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265. Morgan Kaufmann, 1998.

[6] H. Hunt. *Pragmatic Thinking and Learning: Refactor Your Wetware.* 2008.

[7] R. E. Mayer, R. Moreno, M. Boire, and S. Vagge. Maximizing Constructivist Learning From Multimedia Communications by Minimizing Cognitive Load. *Journal of Educational Psychology*, 91(4):638–643, December 1999.

[8] G. A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63:81–97, 1956.

[9] J. Nielsen. Test-taking enhances learning. `http://www.useit.com/alertbox/learning-recall.html`. Accessed July 25, 2011.

[10] M. C. Polson and J. J. Richardson, editors. *Foundations of intelligent tutoring systems.* L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1988.

[11] R. C. Schank. *Designing World Class e-learning.* 2002.

[12] K. R. Scherer. What are emotions? And how can they be measured? *Social Science Information*, 4(44):693–727, 2005.

[13] B. Schwab. *AI Game Engine Programming.* Charles River Media, 2008.