Vrije Universiteit Brussel

Faculty of Science

Department of Computer Science

A Domain Specific Modeling Language for Specifying Educational Games

Graduation thesis submitted in partial fulfilment of the requirements for the degree of 'Master of Science in Applied Science and Engineering: Computer Science'

Akhila Tirumalai Prasanna

Promoter: Prof. Dr. Olga De Troyer

AUGUST 2012

Preface

This Masters thesis is based on research in the field of Domain Specific Modeling performed at the Vrije Universiteit Brussel in Brussels, Belgium. The topic chosen for this thesis research was "A Domain Specific Modeling Language for Specifying Adaptivity in Educational Games".

The research is mainly about applying domain specific modeling to computer-based games for the development of Mathematical skills of kids. This topic grabbed my attention and interest. The following research will help the educational gaming community while modeling mathematics based educational games.

Acknowledgements

It has been an incredible journey writing this report and making progress. I have had my shares of up's and down's through this process. The initial trouble was getting acquainted with the subject well enough to model an entirely new and domain specific modeling language. This was a new territory for me to cross but it interested me and that was very important to me. It took me around 6 to 7 weeks to gain clarity about the definition of the research question. There were several e-mails sent back and forth with my promoter at Vrije Universiteit Brussel, Prof. Dr. Olga De Troyer. I would like to take this opportunity to thank several thoughtful individuals without whose support this thesis would not have been possible.

Firstly, I would like to thank Prof. Dr. Olga De Troyer for helping me in defining the research question and supporting me throughout the process. Additionally, I would also like to thank her for reading through all my drafts, suggesting corrections where required and helping me define my model. Communication was carried out in the form of regular appointments and through e-mails.

Secondly, I would like to thank my colleagues at the university & my friends who helped motivate me and encouraged me.

Last, but not the least, I would like to thank my parents for their utmost faith in me and for always supporting my decisions. Though we were continents apart (10,280 kms to be precise), they adjusted their timings to mine to be able to talk to me at odd hours. Thank you for everything.

Abstract

This thesis presents a Domain Specific Modeling Language (DSML) that helps in simplifying the process of game development for educators without a programming background to express their storyboards or textual descriptions as formal descriptions. The basic approach was to use a visual modeling language to create an easy to understand description of the overall educational game. The DSML proposed in this thesis helps educators to model various constructs used for the narration of the (mathematics based) educational game's storyflow. Aspects such as player assessment and content adaptation have also been taking into considerations. The detailed model could be used to automate the generation of a playable educational computer game. A comparative analysis of the proposed language with an existing language proposed by Marchiori et al. in their paper titled "A visual language for the creation of narrative educational games" was also made.

Contents

List of Figures

Glossary of terms

DSM : Domain Specific Modeling

DSML : Domain Specific Modeling Language

MDSD : Model-Driven Software Development

UML : Unified Modeling Language

EMF : Eclipse Modeling Framework

GMF : Graphical Modeling Framework

ITS : Intelligent Tutoring System

Chapter 1

Introduction

This first chapter describes the context of this thesis, a description of the problem it solves, the approach that is followed to solve the problem and the structure of the entire thesis.

1.1 Context

Digital games are an important part of the lives of millions of people all over the world. Some games are extremely profitable in the market [75][55]. Games are broadly classified into two different categories, namely, console-based and computer-based games.

Computer games available in today's market are increasing in popularity, as online distribution has become common. Easy access to computer games has allowed people from all walks of life to access and use them to suit different needs. Some people use games purely for entertainment while others use them for learning / educational purposes [20]. Being interactive, engaging and having activities planned for the user to perform have made educational games a huge hit. This has also resulted in making it a very important teaching tool in the education industry [26].The new trend of education through gaming has gained new levels of popularity in recent times resulting in the creation of numerous educational games.

Educational computer games have shown to improve the knowledge set acquired by students in various fields of education, such as arts, science, history and mathematics [58] [47]. Due to the rising popularity of educational games, developers are under a lot of pressure to tailor existing games to the required specifications by not only making the games profitable but by also ensuring that the game can be adapted to different learning situations. This is an important factor to consider, as re-usability of the game is profitable to both the game player and the developer; a win-win situation is achieved without major overhauls of the entire system. Thus, for educational games, not only should the game be a success commercially but it should also be an educational success and motivate the students using it to perform their best [26].

One other important factor that has to be taken into consideration is the academic content provided in such games. Tailoring the content based on the audience and requirements is extremely essential. The viewpoints and the ideas given by the educational community should be considered and should not be ignored. The merging of the academic content along with the storyline of the game is crucial and can either make the game a success or a failure. A successful game embodies all the educative aspects and the gaming aspects in an equal ratio of importance without bias. Nowadays, game design and game-play quality seem to be more important than even graphics and narrative. A lot of research work has been done on these topics of how to make an educational game effective and efficient [26] [53] [50].

The third important factor that has to be taken into consideration is that of the complexity of these games. Complex games require a lot of development time and if this takes too long the game can lose its market value or impact value. To improve the development process, Domain Specific Modeling Languages (DSML) can be considered, as they might help the game development community in general by increasing the level of abstraction used to specify a game and providing code generation functionality thereby reducing the development time and the programming efforts required [29]. Using DSML for developing educational games essentially implies that the research work that has been conducted in the field of educative learning is integrated with game-play to build an effective game in a specific domain of education, such as word games, mathematical games, science games and so on. These game development languages can be re-used in that particular domain with ease and effectiveness.

Domain Specific Modeling (DSM) is also associated with model-based development of software systems, i.e. code is obtained by applying model transformations until code is obtained. This methodology of software development is known as Model-Driven Software Development (MDSD). In this way, by using DSM, it helps to also effectively bridge the gap between a domain expert's view of a system and its implementation [20]. It is important to note here that an expert's view is an integral part of the effectiveness of the system as he alone will be able to give critical information about his particular domain (an insider's scoop of a particular domain) and what requirements are essential to achieve success. The developer then takes the domain expert's view into account and models the system to integrate all the elements of the domain and game-play together in harmony. Domain Specific Modeling uses a Domain Specific Modeling Language (DSML) to specify the unique vocabulary of a particular domain, which aids in making the models more accessible to a domain expert. This unique vocabulary is the well known terminology understood by all the individuals in a specific domain. Integrating this terminology in the modeling process helps the domain expert easily identifying and understanding the usage of the model elements.

In this thesis we shall investigate how we can use Domain Specific Modeling Languages to help domain experts as well as developers to efficiently collaborate and work together during the design phase of the educational games by increasing the level of abstraction. This implies that the domain experts and the developers need to collaborate and work together to achieve a well-balanced system impartial to either education or game-play. By increasing the level of abstraction we provide a solution to decrease the complexity, thereby making the problem easier to understand and solve.

1.2 Problem Description

Motivation is a key aspect of effective learning but motivation needs to be sustained through feedback responses, reflection and active involvement in order for the designed learning to take place [21]. Hence, the key challenge for effective learning integrated with games is to make it engaging, motivating, supporting and interesting for the learner but also more importantly for the learning to be undertaken having clear learning outcomes with relevance to real world contexts of practice. A key challenge for designers is to get the correct balance between delightful play and fulfilling specified learning outcomes [59] [64].

Using Domain Specific Modeling Languages we can better achieve this requirement by supporting the involvement of the domain experts (e.g., pedagogical people, people from the subject domain) in the development process of the educational game. In addition, using a high level of abstraction, it can be specified as to how to adapt the system for the user, for different styles of teaching and also for different modes of game-play. High-level modeling concepts for specifying the game aspects, pedagogical aspects, adaptive aspects and domain (learning) aspects should be provided by the modeling language. There should be a seamless integration of all the previously mentioned elements to be effective and easily understandable by both the domain expert and the developer. For the final product (educational game) to be successful, the learning aspect and the gaming aspect should be well merged and one aspect should not be just an add-on of the other aspect (i.e., some games added on top of a learning tool or some learning exercises added on top of a game). Hence it is important that we analyse how we can handle these two aspects and their integration in the design of the modeling language - in parallel or separately. The main goal of this thesis is to build a DSML for educational games integrating these two aspects. This DSML can be used in an authoring tool for educational games. In the context of this thesis, we will not consider the aspect of code generation, but this can be considered in further work.

The usage of authoring tools [5] to build an application help in making the development process more widely accessible. Since these tools try to avoid programming, which is instead replaced by an interactive user interface, the semantic gap between the designers (authors) and the developers (implementer) is drastically reduced [8].By introducing a non-programming language (such as, an interactive user interface with graphical modeling elements) there will be a greater involvement of the non-technical people in the development process. This helps in a clearer understanding of requirements and a faster development without numerous iterations.

However, only reducing the semantic gap between domain experts and developers is not enough. We also need to consider the learning curve, this curve should not be too steep. The time required to master the DSML should not be too large, neither for the domain experts nor for the developers involved in the development process. This is an important aspect of the creation of the DSML and must be taken into consideration. Game developers rely on game technology platforms, game engines, and other custom tools to produce games quickly, affordably and reliably. However, usage of these tools present a very steep learning curve and may require a substantial amount of knowledge of the game technology which is a huge challenge and often a challenge impossible to overcome for most teachers who have little knowledge about game development. Therefore, for the domain experts, it will be much easier to learn, apply and extend DSMLs, as the domain vocabulary will be used in the DSML which makes the DSML easily understandable to this category of people. Some of the concerns associated with the Domain Specific Modeling Languages are that they are too restrictive when it comes to developing games. Game developers may have the feeling that it will be much faster to implement the game directly. To convince them of the benefits of a DMSL, it will be necessary to show them the advantages and this is only possible by building such languages and elaborate case studies and set up experiments to illustrate and actually demonstrate the advantages. This thesis is a first step in this direction.

We shall see in this thesis how we can build our own Domain Specific Modeling Language for the domain of educational games. The main goal of the language is to allow the domain experts and educators to specify an educational game with ease. We aim to achieve a smooth learning curve for domain-oriented people.

1.3 Approach

The common approach taken to develop a DSML is given by Arie van Deursen et al. in their paper "Domain-Specific Languages: An Annotated Bibliography" [70]. A clear distinction is made between the analysis and design of the language and the implementation of the language. *"The steps are as follows: (1) Identify the problem domain; (2) Gather all relevant knowledge in this domain; (3) Cluster this knowledge in a handful of semantic notions and operations on them; (4) Design a DSL that concisely describes applications in the domain; (5) Construct a library that implements the semantic notions; (6) Design and implement a compiler that translates DSL programs to a sequence of library calls. The analysis is covered by step (1) to (4), the implementation by step (5) and (6)."* We shall follow a similar approach to understand the domain and design the DSML. The implementation aspect shall not be covered in this thesis and can be considered for future work.

A lot of research went into understanding of the domain and collecting all knowledge regarding the domain. We were quickly faced with the problem of the size and coverage of the domain. Creating a Domain Specific Modeling Language to cover the entire domain of educational games would be a monstrous effort as many different types of games and learning approaches can be used, and many different application domains are possible. Hence, we decided to restrict the domain of the language to (1) one particular genre of game and (2) to the context of a particular learning topic (such as Mathematics for 10-12 year kids) and (3) to one particular instructional method (game play)[73].

The game genre we have opted for is *adventure games* , the learning topic we decided to use was *Mathematics* and the instructional method was through *game play*. One of the reasons for selecting this particular combination was the fact that we have some education games available that belong to this type of educational games and which could be used as a point of comparison. In addition, learning through game play has proven to be advantageous [64] [63] and this was exactly what we were trying to achieve. The game player will be motivated by the challenges of the game and will also benefit by learning mathematics along the way.

Acquaintance to the domain and domain terminologies was a major step of the development process of the DSML as mentioned previously. During this phase also a detailed study of the use of domain specific modeling for educational games was performed. DSMLs and the different tools that could be used for this purpose were also evaluated. The advantages and disadvantages of using these methods and tools were noted. The current trends in educational games were also examined and some educational games were tested to see how learning and playing games were integrated.

Finally, a Domain Specific Modeling Language was created by the means of a diagrammatic representation (conceptual model), which encapsulated all the key features of the domain and their relations. The approach of visual programming benefits from the ease with which people can employ visual languages to convey information. This is essential as many people think, remember and recall things in terms of pictures [6]. Therefore, icons and graphical symbols are used to represent the different concepts in the domain. Thus, the idea is to achieve higher levels of abstraction at the end of the modeling process by just using the visual models.

1.4 Thesis Structure

This **first chapter** explained the context, the problem and the approach followed in this thesis.

The **second chapter** provides background information related to this thesis. It has two main sections which describe the concepts of 'Domain Specific Modeling (DSM)' and 'Educational Games'. The sub-sections elaborate on Domain Specific Modeling Languages (DSML), introduction to educational games, educational game types and the sources of motivation. It also illustrates the current educational games in the market.

The **third chapter** will focus on the proposed modeling language. The main sections of this chapter are 'Principles of the language', 'Educational elements', 'Modeling concepts and purpose' and 'Examples'. We shall elaborate on the graphical representations, high-level concepts (such as the storyline), abstraction mechanisms, customization ... We shall also link the representations to a specific storyflow and provide a specific example to better understand the concepts used.

The **fourth chapter** discusses related work and provides a comparison of our language to that proposed by Marchiori et al. in their paper titled "A visual language for the creation of narrative educational games" [41] by modeling their storyflow using our constructs. It also explains the example Marchiori used to provide background to the comparison. This language is close to our purpose but was known to us only in a later stage of the research. Therefore, we decided to take this work into account by making a more detailed comparison with our language. This chapter also deals with the limitations of the modeling language proposed in this thesis.

The **fifth chapter** is the final chapter of this thesis. It summarizes the work done in the sections above, conclusions drawn and also describes the scope of future work.

Chapter 2

Background

The sections and sub-sections of this chapter will discuss the terminologies / concepts and definitions which are required to understand the future chapters of this thesis. We shall discuss in detail and explain the ideology behind Domain specific Modeling, Domain Specific Modeling Languages, Educational games types and concepts.

We shall start by elaborating on Domain Specific Modeling (DSM), which is the methodology used in this thesis for modeling the educational games.

2.1 Domain Specific Modeling (Languages)

In this section, we will describe Domain Specific Modeling (DSM) which is a software engineering methodology used to achieve a higher level of abstraction for representing various aspects / features of a system. In general, a visual diagramming language is used for the purpose of Domain Specific Modeling. This is the modeling approach that we will use in this thesis to design and model educational games. Next, we shall discuss the ideology behind Domain Specific Modeling Languages. Finally, we shall discuss how Domain Specific Modeling is beneficial for educational games and why we chose to use it.

2.1.1 Introduction to Domain Specific Modeling

Images and diagrams have been used by mankind since prehistoric times to communicate. Domain Specific Modeling as stated above usually applies a visual diagramming language which uses a consistent visual scheme to represent ideas and logic. Using visual diagrams, comprised of pictures, symbols and basic geometric shapes to represent ideas and plans, is a valuable strategic skill that can make the ideas more persuasive, help improve team communication and also speed up decision-making on complex business issues [3]. Visual diagrams are powerful communication tools. This is proven to be true as they effectively convey both meaning and context simultaneously. In simpler words, they show how different elements interact with each other, how the ideas relate to each other and to the overall concept and the logic behind the entire system. Using visual diagrams wisely can help clearly demonstrate techniques such as, creative thinking and problem solving, and will reduce the dependency on words for business communications. Ideas, concepts, logic and structure can be visualized and meaning can be derived more easily through visuals with much lesser effort when compared to reading elaborate manuals and analysing huge amounts of data [23].

DSM can be formally defined as a combination of a domain-specific modeling language, a domain-specific code generator and a domain framework to raise the level of abstraction beyond manual coding by specifying programs using domain concepts to generate a final product in a chosen programming language from the high-level specifications [34] [66]. In simpler terminology, DSM is a way of representing the complex ideas behind a specific domain, which are used to build a software language with the help of a set of consistent visual diagrams used to represent the domain concepts and which can be translated into a software program with the help of a code generator. The focus is the representation of the domain concepts (based upon the semantics of that particular domain) which can be used by people with little / no knowledge of programming. DSM can raise the level of abstraction beyond coding by specifying programs directly using domain concepts. DSML is used to raise the abstraction level higher by supporting languages that are custom designed to a given domain[69] [39]. If the language is well written high expressivity can be achieved, domain constraints can be assumed and translation to analysable language can be easily made [66]. DSM aims to achieve a transparent path from high-level requirements to executable code.

Using Domain Specific Modeling we have an ease of automation [66]. Automation can be done easily as the language and the generators are built specifically (uniquely) for one company and one particular domain. The domain concepts are modelled in a very abstract language. Thus, they can be easily defined. The code generation is mostly based upon the relationship between the elements / objects. However, one has to be careful enough to analyse if the domain is capable of handling this approach of code development [16]. The modeling language applies the user end concepts, such as, UI widgets and behaviour as modeling constructs. Let us consider UML, for example, which uses programming concepts (classes, return values etc.) as its central modeling constructs whereas DSML uses the domain concepts (concepts, rules, constraints etc.). All the above mentioned factors lead to the use of Domain Specific Modeling being advantageous when compared to other generic modeling methodologies. The goal of DSM is to raise the level of abstraction coupled with automated code generation [9].

Some of the advantages that can be gained upon the use of Domain Specific Modeling are increase mainly in productivity, reliability, maintainability, portability, re-usability and quality along with use of expertise to share knowledge within the development team [70][54]. The main concept that is to be noted here is that the domain experts should be able to understand, validate, modify and even develop a large part of the domain specific language programs. This helps in gaining greater levels of interaction between the developer and the domain expert to help create a well-rounded and effective system. In this thesis, to better represent the educational game it was important to entwine the knowledge of the game developer and the domain experts (educators). The successful collaboration of the developer and the expert could result in an outstanding educational game representation.

Domain Specific Modeling helps improve productivity when we know how and when to apply it [34]. A major influence to the rise in productivity is the raised level of abstraction [66]. The increase in abstraction leads to less modeling work which can be done by people having little or no programming experience. This increase in performance is one of the main reasons for the success and the wide spread use of Domain Specific modeling over generic modeling. The domain knowledge and the code skill of expert developer when embedded into a tool enable the developers to achieve a higher level of productivity. Such a tool is called a Domain Specific Modeling tool for model oriented technology [33]. Reuse of models is also facilitated due to the model driven nature of development. Furthermore, the quality of the product is enhanced as the software system is derived from the high-level DSM.

To summarize, DSM mainly intends to do two specific things. First, it helps to raise the level of abstraction beyond programming by specifying the solution in a design language that directly uses concepts and rules from a specific problem domain. And the second being, it helps generate final products in a chosen programming language using these high-level specifications. This standard of automation can be achieved as the language and generators are domain specific. In other words, they fit the requirements of only one specific company's domain.

2.1.2 Domain Specific Modeling Language

Domain Specific Modeling Languages (DSML) are visual diagrammatic languages used as special purpose programming language to help building a Domain Specific Model for an application. These languages are exclusive to a specific domain (for example: a business domain, such as, life insurance, salary calculation, billing etc.) [65]. In general, when using a Domain Specific Modeling Language, a matching code generator is also used to generate code based on the model representation. This code generator is also built specifically for the particular domain and the particular Domain Specific Modeling Language. The models are used to interpret the problem at run-time and this is an advantage of using this language. This results in comprehensive analysis, through visualization, of the domain problem. A higher level of semantics can be featured by the usage of this language.

Domain Specific Modeling Languages are mostly used in situations where continuous development of the same or similar systems or configuration of a system to customer-specific requirements is needed. Such a language not only helps in increasing the productivity, but it also helps the expert to formalize the development practices for the other developers [32]. Once the rules and guidelines are embedded in a tool, they can be enforced and can also guide or warn developers appropriately.

Defining a Domain Specific Modeling Language requires the developer to have a good knowledge of the problem domain and also know the code that should be written for it. The quality of a DSML hence, largely depends on the designer's domain experience and language development expertise. However, a DSML is easy to learn.

One way to define a DSML is the use of a generic modeling tool. In that case, the modeling-language definition is put in as data through graphical models and forms, and the generic modeling tool configures itself automatically from that data [1]. This has the significant benefit that any changes to the modeling language can be instantly reflected in the modeling tool - even updating existing models thus evolving the modeling language becomes a simple process.

In existing general modeling languages such as Unified Modeling Language (UML), the language has been fixed by a committee whereas in Domain Specific Modeling Languages the language is designed by the very best domain expert that the company has [24]. In UML, the code generator is designed by the vendor of the UML tool, which is not the case in Domain Specific Modeling Languages as the company's best developer crafts the generator specifically for the problem on hand. When using a Domain Specific Modeling environment for implementing the language and its code generator, further changes can be done instantaneously and automatically by just updating the existing models in this environment. Thus reuse is facilitated by using the existing meta-models for recurring domain problems.

DSMLs help in raising the level of abstraction beyond what today's third generation languages have to offer. The symbols (tailor made for a specific domain) and rules about how they can be connected and used in a DSML are directly taken from the real world in which the application runs. UML models offer a high level of abstraction however DSML (4th generation language) offers even a higher level of abstraction than that provided by UML. This is possible as a DSML is domain specific and has been tailor-made for this particular domain. It is an expressive design language, with boundaries, that uses concepts from the problem domain as first-class modeling concepts. Hence it might be very precise and concise to one domain but completely useless in another.

Tools play a crucial part in this process as the software-development method is of limited value if it has no tool support. The increasing popularity of DSML has resulted in DSM frameworks being added to existing IDEs, e.g. Eclipse Modeling Project (EMP) with Eclipse Modeling Framework (EMF) and Graphical Modeling Framework (GMF).

Now that we have understood what DSML is, we can now apply the concepts of DSML to our domain, i.e. educational games, and create a DSML for educational games. To do so, we have to understand what an educational game is; the various types of educational games that exist; the concepts used in these games, and so on. To obtain this kind of information, we shall investigate the details of some educational games currently available in the market. This will be the topic of discussion in the next chapter of the thesis.

2.2 Educational Games

Game based learning to aid the learning process has been used for decades. With today's advanced technologies, digital educational games have emerged as the latest teaching tool to aid the educational community. Educational games can be broadly defined as games that are designed specifically to teach people about a specific subject, help expand their concepts, reinforce their development or assist them in learning a skill as they play. Educational games have been documented since the late 70's and were then classified into distinct types [57].

The educational applicability of games has been a topic of discussion for a long time [36] [55] [22] [46] [51]. Today, however, it has been established that educational gaming is a relevant business and discussion is now more focused on different aspects such as game design, student tracking and assessment, integration aspects so on [71] [67] [12]. The educational community is not yet massively open to accepting educational games as a system of learning but are becoming more open to this idea.

In this thesis, we are interested in learning with educational games and we shall therefore further classify these educational games into categories. We will learn in detail about the types and concepts behind educational games. We will also look into the current educational games available in the market and choose a game style to build a DSML for.

2.2.1 Introduction to Educational Games

Game playing can aid in social and mental development. Computer games are a newly emerging topic in educational research. Though there has been a lot of progress in this area of research with numerous publications, conferences and federal grants; it has yet being termed as a "technology in search of a paradigm" [25]. With new conceptualizations of games in relation to education some crucial aspects of games and gaming culture were addressed in the Proceedings of the 7th international conference on learning sciences ICLS '06 [27]. Educational game can be defined as a game designed to teach humans about a specific subject and/or to teach them a specific skill. Games are interactive plays that can help teach concepts such as goals, rules, adaptation, problem solving, and interaction, all represented as a story. They give the player the fundamental needs of learning by providing - enjoyment, passionate involvement, structure, motivation, ego gratification, adrenaline, creativity, social interaction and emotion [36].

Learning is a complex cognitive task, as it requires tremendous effort from the students. Educational software that is built to aid and stimulate the students is beneficial, but in today's technologically advanced world, kids and adults like to play games and therefore learning via computer games could be more motivating and challenging. This can be shown to be true by researchers who have conducted empirical studies [44].

The main idea behind educational games is that they help in teaching specific learning topic and/or skills to the game player (children in our case). The relation between ’fun’ and ’learning’ is what gave rise to "educational games". Games by themselves are fun. People enjoy doing things that are fun. Education is about learning. However, ’play’ is also a way of learning. By combining these concepts, a new way of learning through fun and exploring was developed. This method of learning has been proven to be effective and productive [36]. "Play has a deep biological, evolutionarily important, function, which has to do specifically with learning" as stated by Prensky [36]. It has been scientifically proven that it is easier to learn when the body is relaxed, which is exactly what the educational games try to achieve. In general, playing a game is having fun, hence the player is relaxed, and thereby the learning ability will increase.

Creativity is important. Harnessing a player’s creativity to achieve the goal of a game is important [11]. Every game, educational game being no exception, has boundaries (restrictions), players, artifacts, and a winning condition. The goal motivates the players to play to their strengths and increase their chances of winning.

Note that educational games exist in a wide variety, such as board games, card games, and computer games. The focus of this thesis is computer games. A computer game usually has different levels; each with an increased level of difficulty, to ensure the player can play it for a long period of time. In educational games, the levels can also be used to increase the complexity of learning as the levels increase.

Many researchers have developed computer games for educational purposes [10]. Creators of these games only take full advantage of the power of the medium when the educational content is integrated effectively into the structure of the game. This is a consideration that must be taken into account while creating effective interactive games [18]. Several other issues exist concerning the educational effectiveness, appeal and scope of educational software games. Through an evaluation study of an Intelligent Tutoring System (ITS) that operates as a virtual reality educational game it was found that educational virtual reality games can be very motivating while retaining or even improving the educational effects on students [72].

In today's digital age with the rate at which technology is advancing, it is no surprise that educational games are being recommended to aid classroom teaching and to make learning more attractive since traditional methods of teaching alone are no longer successful in pushing the students to their full potential. By using technology in the classroom environment teaching and learning can be enhanced. It gives a new dimension to classroom teaching.

Even though there has been a lot of advancement in technology, educational games are yet not used widely in schools. This can be the result of many factors, such as, not all educators and parents are completely convinced that educational games can be beneficial to learners; there is criticisms about the quality of the existing educational games; and also games need to be tailored to its audience, it cannot be a one-model-fits-all structure. It is hence very important to recognize what motivates an individual learner and helps in improving his performance [7].

Some people in the educational community are against the idea of introducing technology, as they believe that this will cause the 'Death of the Professor' [76]. Therefore, technological benefits to both the students and the educators are essential. It is difficult to balance one without the other. The effectiveness of computer assisted teaching was proven to be successful while using it as a supplement to the regular lecture session [4].

On the other hand, game players and related communities state that games help to develop their social and mental skills. But using computer games as an educational tool requires understanding the game skill sets and game category an individual likes [2].

In this thesis, the focus is on computer based education games for improving the mathematical skills of kids of the age range 10-12 years.

2.2.2 Educational Game Types and Sources of Motivation

In this section, we will examine different sources of motivation for playing games, as motivation to play must be the key characteristic of any educational game. Next, the varieties of educational games will be inspected and analysed.

Sources of Motivation:

Games that disguise the educational content are hard to design and build. But if the educational content is hidden effectively, these games can help encourage the kids to play them voluntarily as they would do with recreational games. The goal is to achieve teaching through playing the educational game.

Numerous publications examine the motivation behind video games in entirety without being specific to educational games. However, even with this broad examination of gaming, there is no unanimous agreement amongst the researchers as to the source of motivation.

The compelling nature of games are attributed to numerous factors such as their narrative context [14][15][18][74], goals and rewards within the game [2][13][30]. An effective game design would hence consider both the intrinsic and extrinsic rewards of game play. The distinction between intrinsic and extrinsic rewards was well explained by Dennis and Jouvelot (2005) [13] where they put across strong arguments that competence, autonomy, and relatedness are the factors that affect motivation. Here, the authors presume motivation as the interplay between desire and pleasure - the desire to be competent and the pleasure one feels while competing.

On the other hand, Dickey(2006) argues that a narrative context that promotes "challenge, fantasy, and curiosity" and provides feedback for players is the factor that promotes intrinsic motivation for play [15]. To add reinforcement to her argument she states that, "Strategies of design that lead to engagement may include role-playing, narrative arcs, challenges, and interactive choices within the game as well as interaction with other players". Waraich (2004) agrees that narrative is essential to motivation. However, he warned that for educational games "intrinsic rewards are based on a high congruence between the material being taught and the motivational techniques used" and thus any dissonance between the two can decrease learning.

The above mentioned are some of the views on the different reasons of motivation for gaming in general. Narration, goals and rewards of game play are considered as the motivation of game play for this thesis. We shall show the usage of these factors in further chapters when we illustrate the DSML built. Furthermore, games that disguise the educational content can help encourage the kids to play them voluntarily as they would do with recreational games. In such games, the goal is to achieve teaching through playing. This is also known as implicit learning, i.e. learning without being aware of it. However, these games are hard to design and build. In this thesis, we shall also illustrate how learning can be hidden behind the main game plot and thus help to make the game more compelling for the player.

Types of educational games:

Some games are referred to as "Serious games". Their root can be traced back to "The Serious Games Initiative (2002)", which focuses on video games usage in public sector administration and management. There are numerous definitions of "Serious games", one of the definition is as follows "the genre of Serious Games includes various types of games where the main intention is not to entertain, but where interactivity and entertainment are a means of achieving other types of goal". "Serious Games" share many similarities with educational games ("play and learn"). However, they are mainly directed towards a public outside the school sector unlike educational games.

Educational games mainly exploit the benefits of using a computer and technology to aid the players through the learning content (information). The basic skills, analytical skills and certain facts can all be integrated into a game in many ways. But typically they are all hidden behind the main game plot. However, the technology behind these games varies, resulting in different types of educational games.

From the educational point of view, the choice of game is an important factor since it is used as a learning material and using any material may not be appropriate for the teacher's objective or to achieve the goal under consideration. Thus to enable ease of selection it is important to classify games into various categories. Many studies have focused on this objective. Some have concluded that there is no specific classification and choice [35] whereas others agree upon the eight categories of games, namely, action, adventure, fighting, puzzle, role-playing, simulations, sports and strategy games [45] [28]. Some researchers have extended this classification to the educational purposes of video games and have categorized various types of video games [48][42]. These classifications are varied and their abilities to differentiate games are numerous. The common idea to all these classifications is that "All video games are not equal, especially so in an educational environment". Kirriemuir and McFarlane request for the involvement of the game development industry to better enable all multiple constrains of educational context.

Teachers and educational researchers need to differentiate games on the basis of their potentials for learning. Some of the educational game types developed are for example: action, adventure, simulation, role-playing, shoot-em up, simulation, puzzle and strategy. Amongst these educational researchers have concentrated mainly on two types, namely, simulation and adventure games [49] [52] [31]. Amory [2] also examined four kinds of game types and analysed the elements the players liked the most. In the research conducted, students rated a number of game qualities such as "the fun aspect, sounds and graphics, type of game, game story and use of technology"; "the importance of some skills [logic, memory, visualisation, and mathematics, reflexes, and problem solving]"; "whether the game was easy to play, addictive, too long, challenging, confusing, too difficult, illogical, difficult to play or manoeuvre and if their performance increased with continuous play".

Based on the researches conducted the most stimulating and highest ratings were awarded to adventure and strategy games. Thus, this finding implies that players preferred or were more motivated to play games with objectives requiring higher order thinking skills, including visualization strategies that nurture creative problem solving and decision-making [2]. Therefore, in this thesis, the categories of adventure and simulation will be considered. We shall apply this to the DSML to be developed. This will be shown in the later chapters of this thesis.

2.2.3 Current educational games in the market

The rising popularity of educational games as the means of educative material has facilitated an invasion of the market with numerous types of games [17] [2] [68]. This implies that there is an implicit increase of revenue generation [55]. Companies, such as Google, are also now actively looking at educational games by stepping up their efforts to generate revenue from education related applications. For example: Placespotting[1] is a geography-based game that uses Google Maps. Here, the player must find a specific location after initially being shown a picture with clues or description of events. It is a non-collaborative game that can be played by individual players without background subject knowledge.

[1]http://www.placespotting.com/

As mentioned previously, we have chosen to investigate games under the learning topic 'Mathematics' and instructional method 'game play'. Therefore, let us have a closer look into digital educational games that fall under these categories, are currently available and which enhance the mathematical skills of the player (specifically, children between the age range of 9 and 12) along with his/her gaming skills.

Some games can be easily played online, for example, Arcademic Skill Builders[2] allows the player to compete with other random players online to finish mathematics based games. Some online games are available by either registering and/or by paying a nominal fee. For example, Coolmath[3] gives the player a series of topics to choose from and then various games to choose from under a chosen topic, DimensionU Games[4] is built for teaching middle school kids pre-algebra , GlobalScholar Games[5] allows the player to choose by grade or by subject or by test preparation which he/she requires. Other games can either be downloaded (for example, Kidspot Learning Games[6] which provides printable worksheets) or bought (for example, Monkey Tales[7], Nintendo 3DS Games[8].

[2]http://www.arcademicskillbuilders.com/

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

[4]http://educator.dimensionu.com/dimu/home/dimugames.aspx

[5]http://www.globalscholar.com/tutoring/search.aspx?Q=math

[6]http://www.kidspot.com.au/schoolzone/section+297+Maths-&-science.htm

[7]http://www.monkeytalesgames.com/

[8]http://www.nintendo.com/games

2.2.3.1 Monkey Tales

The base concept for our modeling language was inspired by the method of game play used in the game 'Monkey Tales'. We bought two specific games from Larian Vault Studios (developer of Monkey Games), namely, 'The Princess of Sundara'[62] and 'The Castle of Draconion'[61].

'The Princess of Sundara' was the first in a series of European Award winning educational games and this inspired us to further investigate it. It is built specifically for 2nd graders (Ages 7+) and is a part of a series of educational games developed at Larian Studios. This is an adventure game with mini math games (for example, shooting at flying targets to gather numbers to solve an equation) and logic puzzles (for example, players must block lasers, push floating blocks into waterways to create safe paths, jump on moving platforms to get where they need to go). The player's goal is to rescue the princess who was captured by Rinjin the dragon who dislikes her for being smart and intelligent. The player travels through fantasy lands avoiding environmental traps to reach the castle of Rinjin. Each level is considered as a single room in which the player not only beats the monkey at a math game but also collects bananas to free the monkey into their very own zoo. The logic game of each level is not timed but the mini math games are timed and can be replayed from the start menu. However, the time pressure is added in a very motivating manner (i.e. play against the monkey) to encourage the player to finish the game without being stressed by the timer. Eventually the player reaches the castle of Rinjin, beats the dragon at a math game and rescues the princess. The game might be a bit scary for the 7 year old kids when they have to dodge shambling mummies and spectres, and avoid running into lasers that blast them to ash (but their avatar reforms immediately). On the positive note, this game has algorithmic software which calculates the players level of math skill and adjusts each mini game to the skill set , thus ensuring that the player progresses easily without stressing about winning the game.

’The Castle of Draconion’ is also a part of the series of educational games which is designed for 5th graders (Ages 10+). It has similar concepts to that of ’The Princess of Sundara’ with changes modified to fit the theme where the player has to explore the castle, find the vampire Valdimir and eventually beat him in a math challenge to win the game. The mini math games and logic games are a bit more advanced to engage the audience.

Some of the concepts of our modeling language are derived from the themes seen in the above games, mini math concepts used and the logic games. However, changes have been made to increase the positive role models, ease of play, positive messages. We shall see this in detail in the following chapter of this thesis.

Chapter 3

Proposed Modeling Language

In this chapter, we introduce and explain the modeling language we designed to support the development of educational games. Within the scope of this thesis, the language is restricted to mathematics as learning domain and instructional method used is gameplay. In this language, we need to provide modeling concepts to specify the gameplay, but also to specify the learning elements.

We will first discuss the principles of our language: the principles of the graphical representation used, the high-level modeling concept used in our modeling language being the storyline, and the principle of customization. Next, we discuss the mechanisms used for expressing the story flow. We also provide an overview of the different modeling concepts and how they relate to the different domains involved in educational games. Finally, we shall give some examples of the usage of the proposed modeling language, including a general example and a specific example of the educational game.

3.1 Principles of the Language

The main idea of the language is to use visual (graphical) notations to describe the details of the game-play. This should help educators to specify a game from scratch, or translate a storyboard or a text-based description easily into a game. Our DSML allows expressing the story flow of the game by explicitly modeling the possible actions that can be done by a player and the consequences of those actions. In this way, the game is represented as a kind of graph. However, the usage of basic graphs in the representation of games can result in extremely complex and difficult to understand models. It also makes modification of such models a nightmare. Researches conducted in the past by Lindley have identified the problem associated with basic graphs which shows that it is capable of rendering an explicit graphical representation of games completely useless due to practical reasons [38]. However, by using a hierarchical representation, the structural components can be broken down into various tiers / levels and each component can be addressed individually. This also helps in reducing the complexity of the entire description [41].

Also other researchers have used such a graph-based visual approach for modeling educational games. In particular the work of Marchiori et al. [41] is interesting with regards to this aspect. As we only came across their work at a later stage of our own research, we could not base our work on theirs to a greater extent. However, we will compare our language with their language in Chapter 5. Our model has similar features to theirs but helps in further simplifying the complexity levels.

3.1.1 Graphical Representation

Our DSML is using graphical symbols for the different modeling concepts. Rather than using abstract graphical symbols, we have opted for relevant images and pictorial representations embedded in simple geometric shapes to denote different modeling elements. Such a graphical representation helps in enabling a faster understanding of the concepts used in the language. The language has been designed not only to model the story flow but also to accommodate player's choices, the consequences that follow, and can also be adapted easily to different learning subjects and future changes.

3.1.2 High-level modeling concept: Storyline

The main modeling concept in our language is the storyline. The storyline describes the main plot and / or the sub-plot of the game. For example, the storyline of the example game, used further on in this thesis, is based on the storyline of MonkeyTales (mentioned in Chapter 2). The '*main goal*' of this game is to rescue the princess from the dungeons where she is being held captive by the evil sorcerer, Badin. To do so, the player has to complete several '*missions*'. Each *mission* requires the player to be involved in one or more '*game plays*' and also to complete '*mini-math games*'. For example, the player collects certain numbers as he roams around discovering the passages in a mission (i.e., game play) and using the numbers collected he/she answers specific mini-math questions designed for his particular grade/age (i.e., mini-math games). At the end of each mission, upon having answered the mini-math questions correctly the player will acquire a specific '*gift*'(e.g., a blue diamond, glass shoes, a magic red apple, etc.), and finally after completed all missions the player acquires the last gift being the Princess's Tiara (which Badin had snatched). The '*gifts*' collected at the end of each mission will be saved in the players vault (treasure chest). Once the player completes the '*main goal*' all the '*gifts*' will be handed over to the princess who has been successfully rescued. This indicates the end of the game. The main modeling elements needed to model such a storyline are already indicated in the text in italic: '*main goal*', '*mission*', '*game play*', '*mini-math games*', '*gift*'. To model this storyline, we start with a start state denoted by the '*start element*' that indicates where the game starts and in turn also points out the beginning state / point of the player. The start state can also be used as a point where the player can choose the game complexity level. Hence, the start state is available in 4 colors, namely, green (easy mode), yellow (medium mode), orange (hard mode) and red (extreme mode). This is followed by a series of mission states denoted by the '*mission elements*', which represent the states the player needs to reach, i.e., the missions that the player has to complete to reach the final goal / end state. The end of the game play is indicated with an end state denoted by the '*stop element*', which indicate the successful completion of the game and the final state of the player. The graphical representation of these modeling elements is shown in Figure 3.1. Some modeling elements can be customized. This is explained in detail under the subsection 3.1.3 "Customization" of this chapter.

Furthermore, different decomposition levels are used to express all details of the storyline. In the top level, we specify the start of the game by means of a '*start element*', then the different '*mission steps*' that need to be completed indicating the '*mission*' that needs to be achieved (e.g., the '*gift*'s to be collected). The end is indicated by the '*stop element*'. Each mission step can further be broken down into several sub-steps. The next level can show the '*game plays*' and the '*mini-math games*' that the player needs to perform. A game play is represented by the symbol of a '*joystick*' while a mini-math game is represented by the symbol of a '*calculator*'. The game plays and mini-math games can be time-controlled (represented by an '*Objective element with a Timer*') or without any time restrictions (represented by a simple '*Objective element*'. The '*Objective element*' encompasses both mini-math game and game play. This means that both tasks (mini-math game and game play) of the objective task must be performed obligatorily (without any choice, in any order). The graphical representations of these different modeling concepts are given in figure 3.1.

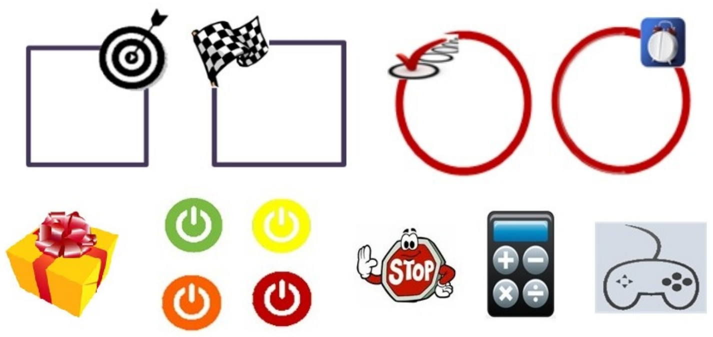

Figure 3.1: Storyline Representation

From L to R Line 1: Mission element, Main Goal element, Objective element without time restrictions and Objective element with Timer. From L to R Line 2: Gifts, Start Element, Stop Element, Mini Math game and Game Play.

3.1.3 Customization

Customization is used in our language to allow a designer adapting the graphical representation of a modeling element to the different scenarios of the game under consideration. Not all modeling elements are customizable. The customizable elements used in the language are outlined with a customizable area. For example, the '*mission state*' is represented by the dartboard shape on the top right-hand corner of a purple square area. The purple colored border of the square area indicates that it is '*customizable*'. Using this customizable area, the modeler can adapt the modeling elements to the scenario at hand. For example, we can customize the modeling element representing a mission state to show the concrete mission goal. This is illustrated in Figure 3.2. The figure on the left shows the generic graphical representation of a mission state; the middle figure show a mission state where the mission goal is achieved by the collection of the gift of the "Glass shoes" which is represented by an icon of the glass shoes, and the right figure shows a mission state where the goal is achieved by the collection of the gift of the "Blue Diamond" which is represented by a blue diamond icon. The elements should be customized based on the story flow of the game.

It should be noted that not all modeling elements are customizable. For instance, in figure 3.3 a non-customizable modeling element is shown. The red colored boundary of the circular area of the '*Objective element*' denotes that the element cannot be customized. Since the element cannot be customized and shows the game play and mini-math game icons in the square area this implies that the player has to compulsorily perform both the tasks.

Note, that the color-coding can be applied to other elements of the domain spe-

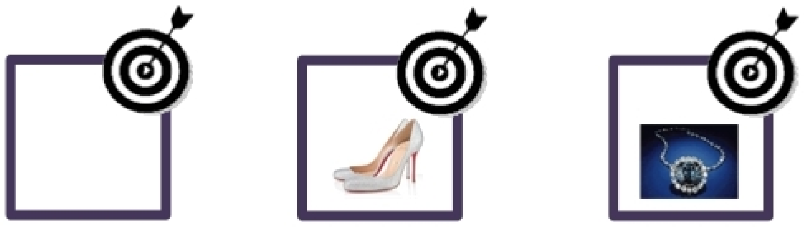

Figure 3.2: Customization of elements.

Figure on the left represents the generic graphical representation of a mission state with customization denoted by the purple colored border, figure in the middle represents the mission goal for the mission 'Glass Shoes' and one on the right represents the mission goal for the mission 'Blue Diamond'. The two latter figures represent the element representing 'mission state' in the customizable area.

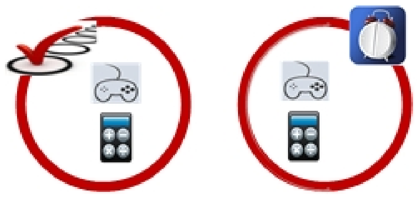

Figure 3.3: Non-Customizable elements.

Figure on the left represents the objective element without time constraints and one on the right represents the objective element with timer. Both represent the objective element which cannot be customised since it has a red border.

cific modeling language to denote if customization can be applied. For example, if a purple colored boundary is used for an element it indicates the *area of customization* where tasks to be performed by the player are shown but can be customized. If an element has a red colored boundary encompassing certain tasks to be performed by the player, this indicates that all the tasks represented within that element should be *performed compulsorily*.

Also note, that the symbols introduced here can also be re-used and customized in other (related) DSMLs. For instance, the representation used for the *mission element* could be used in other languages as a level / goal by replacing the bullseye with another domain relevant symbol. In our language the *mission element* is shown with a bullseye on the right hand top corner of the customizable area whereas the same customizable area with a chequered flag on the top left corner is used to represent a *main goal* element.

The following subsections will elaborate on how to express the actual flow of activities in the storyline. First we will explain how to express choices, next we show how to model sequential flow, and finally how to express loops in a storyline.

3.2 Modeling the Storyline

In this section, we explain in more details how the storyline can be modeled with our DSML. We first discuss how choices can be modeled, next we deal with the modeling the sequential flow, and then we discuss the use of loops. Finally, we will explain the modeling of the details of a game by means of hierarchical decomposition.

3.2.1 Choice in the Storyline

In most games, at certain stages, the player is given the flexibility to choose from a series of options, which eventually lead to the same result through branching at different sections in the storyline. This facility provides a certain level of control to the gamer, which is highly appreciated by players. In educational games, this flexibility can be used to make the games more challenging and thus encouraging exploration and facilitating the decision-making process [40] [55]. However, this branching at different sections to reach the same end result increases the complexity of game design. Let us consider, for the sake of an example, a situation where a player has to make a choice about the order of performing 2 activities, namely, collecting items and solving a mathematical exercise, to achieve the mission goal (reference: Monkey Tales[62][61]). Irrespective of the order in which these activities are performed the goal achieved will be the same. Such branching in the storyline increase the complexity of the design, since the number of possible action combinations increase exponentially [41]. To avoid this repetition, we introduced a modeling element '*order-choice*' that indicate that the order of activities (branches) can be chosen and the branches have the same result. This result is represented once rather than representing the branches with two separate, identical results. This order-choice symbol is shown in the (Figure 3.4). The double-arrowed circular symbol represents the '*order-choice*' modeling element. The two options are on the left and the right side of the symbol. Both activities need to be performed but the order in which these activities are performed will not affect the result and is up to the player. In the example given in figure 3.2, the options for the player are to chose an '*avatar*' (represented on the left, '*avatar element*') and to chose a '*buddy*' (represented on the right, '*buddy element*'). Irrespective of whether the player chooses first his avatar and then a buddy or vice versa, the result will always be that once he completes both, the player moves to the objective element to perform a game play and a mini-math game.

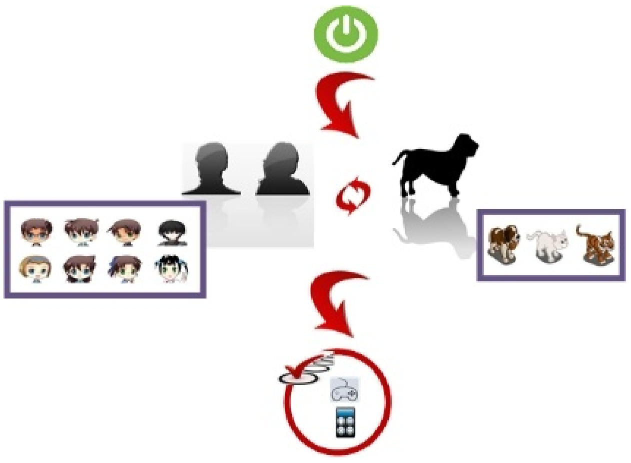

Figure 3.4: order-choice Representation

*The double-arrowed circular symbol represents the '**order-choice element**'. It indicates that irrespective of the order in which the activities are performed the player moves from the '**start element**' to the '**mission element**'.*

By using this principle, we can eliminate repetition of transitions; exponential branching is also reduced which results in a more straightforward and direct representation of the game design. Furthermore, we need a *choice element* to allow the player to select one option out of a number of options. In the previous example, the *order-choice element* allowed to do a number of activities in any order and it led to the same outcome irrespective of the order of choice. With the *choice element*, the options could led to different outcomes. This new model element is represented by a blue-colored split arrow where each arrow indicates a different choice that can be selected by the player. Figure 3.5 shows the '*choice element*'. It shows that when the player, at this point in the game, can choose between game play or completing a mini-math game.

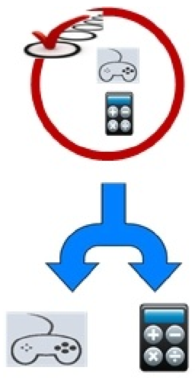

Figure 3.5: Choice element

'Choice element' leads to two possible different outcomes based on the player's choice and is represented by a blue-colored split arrow.

3.2.2 Sequence of the Storyline

Figure 3.6 shows the generic graphical representation used to represent the sequence of steps to be performed by the player in the storyline.

The concepts discussed previously in the sub-section 'storyline' are used in a particular sequence to express the entire game / story flow. The thick red semi circular arrow is used to links the steps in the sequence. In figure 3.5, we notice that the storyline starts with the '*start element*', followed by the '*order-choice*' modeling element where the user chooses an avatar and a buddy, followed by a set of '*mis-*

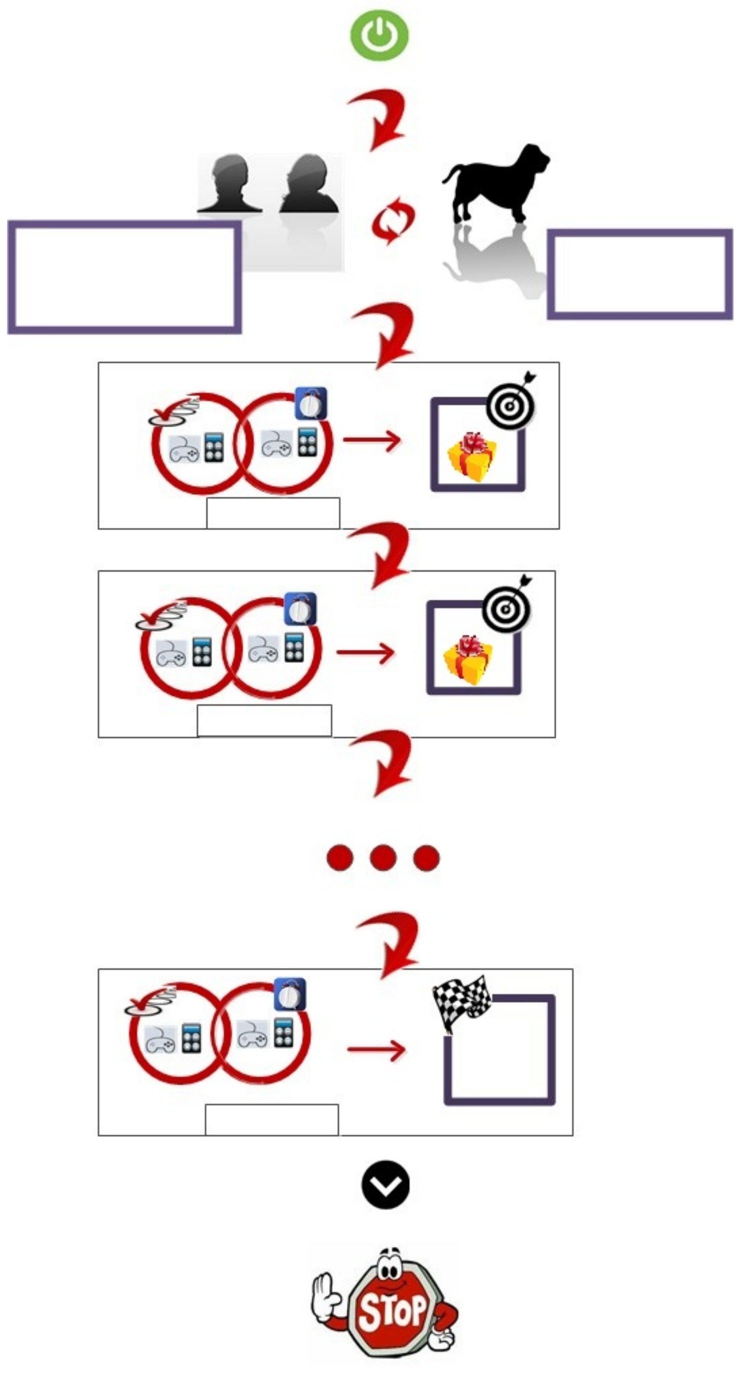

Figure 3.6: Storyline: sequential flow

Generic graphical representation of the sequence in the storyline.

sion steps' that the player needs to complete to finally reach the '*main goal*'. Once he completes this main goal the game ends, indicated with the '*stop element*'.

Notice that in the generic model we have paired '*objective elements*' and the consecutive '*mission element*' using a black box. A *red straight arrow* shows the association of the '*objective elements*' and the consecutive '*mission element*'. This is to show explicitly that each objective has a mission attached to it. Later on, we will see that such a objective-mission pair, called a *mission step*, can be further decomposed to show the sub-steps that need to be taken to reach the '*mission element*' from the *objective element*'. The (empty) box below each of the '*objective elements*' is to specify the unique name of the objective of each mission. This will ease the "reading" of the diagram. Once the mission is accomplished the user will move to the next 'objective element' and the associated mission. **Note:** Notice the overlapping of the objective element without time constraints and the objective element with a timer in figure 3.5. This indicates that the game can be designed with mini-math games with a time constraint and without time constraint. In other words, an objective can be timed or untimed.

As already indicated, the 'thick half-curved red arrows' show the flow from each objective-mission pair to the following. The player might fulfill numerous objectives and accomplish a similar number of missions. The representation of numerous objective-mission set usage in the storyline is represented by the '*three red dots*'. Once the various missions are completed the player must eventually finish the 'main goal' element to finish the game. The black circle containing the white arrow used between the '*main goal*' and the '*stop element*' is used to indicate that a final save of the entire game will be automatically done at this point.

3.2.3 Loops in the Storyline

Loops were introduced as modelling concepts to indicate the tasks that had to be done repeteadly by the player until a certain condition is fulfilled or till the goal is achieved. There are several kinds of loops, such as, simple loop, nested loops and so on. We use the simple loop which satisfies the conditions of either 'Do while' or 'Repeat until'. Loop is represented as shown in the Figure 3.7. The customisable area represented by the purple box can consist of several decomposition steps, each with individual step objectives. An example of a loop can be seen in Chapter 4, Figure 4.5 for the event FireAlert.

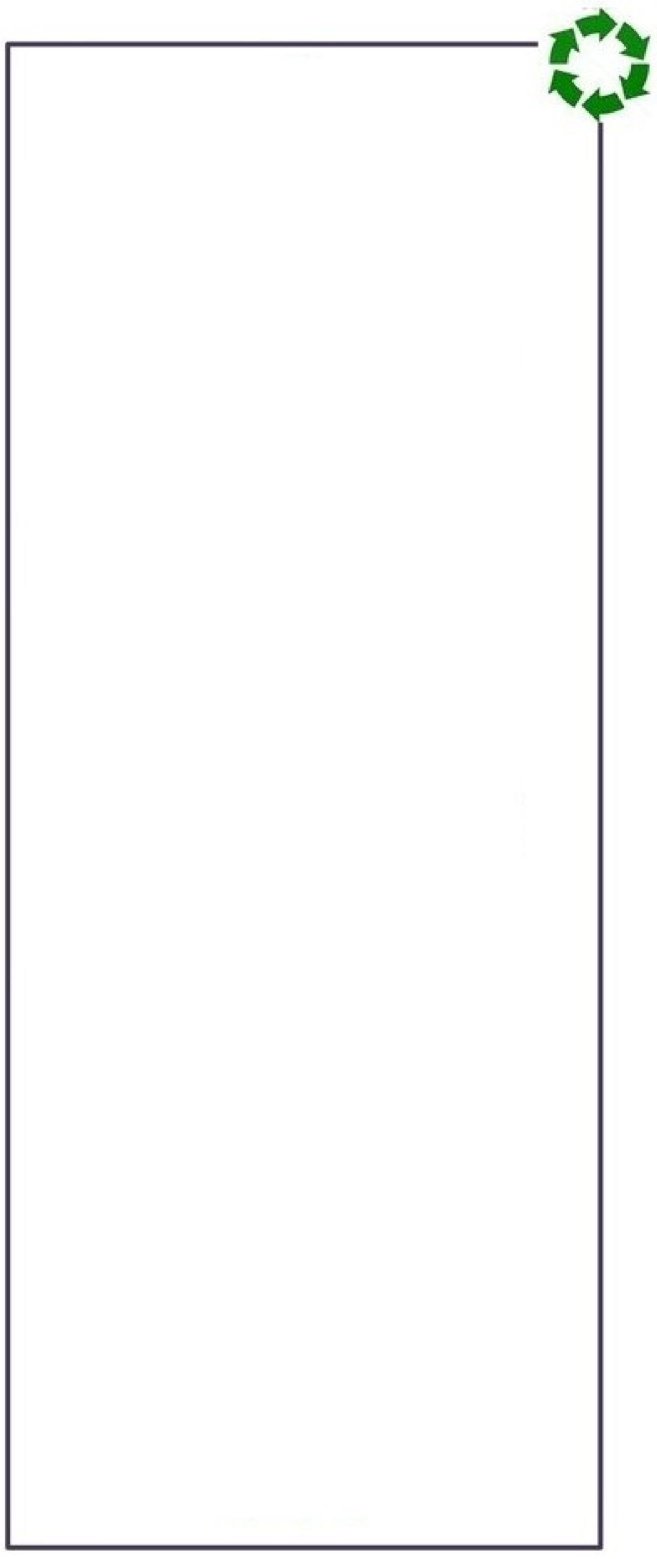

Figure 3.7: Loop representation

Generic graphical representation of a loop in the storyline.

3.3 Educational elements

In this section, we shall discuss the educational modeling elements that we provide in the language. The educational elements used in the modeling languages aid to cover various educational aspects of the educational games. We shall consider some of the characteristics of educational games that contribute to learning, such as, assessment, guidance and adaptation [56] [37]. We shall show how our modeling language provides support for each of these characteristics in a simple but straightforward manner. This will also help to show how the modeling language is conceived to fit various requirements (for example, user's level of gaming, such as expert or novice) or can model games for specific fields (example, math or biology). We shall also see the indirect ways of assessment such as modifying the game automatically to suit the player's level or through instructional adjustments based on the results of the assessment.

3.3.1 Guidance

In-game hints are a methodology used in *student (player) guidance* that allow the player to make progress regardless of his knowledge background / level / understanding. Monkey Tales games used this specific method to aid the player. Since some of our concepts are derived from this commercial game, we have chosen to provide in-game hints as modeling concept in the modeling language. This is done by providing the *buddy* element as shown in Figure 3.8. The purple area is used to denote that customisation is possible. The various choices for a buddy that the player can choose from are depicted within the purple area.

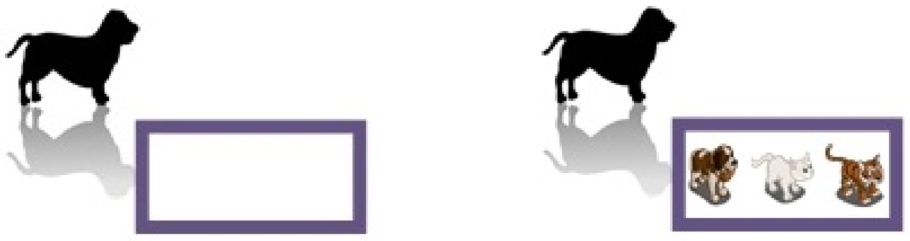

Figure 3.8: Buddy element representation

Left: Graphical representation of 'buddy element' in the storyline. Right: Buddy element with customization. The player can chose a buddy from the three given choices, namely, a puppy, a kitten or a tiger cub.

The basic purpose of in-game hints is to help the players who require this additional form of support / guidance to proceed. But it also ensures that the players who do not seek help / guidance can play without any disturbance. One of the ways to implement this is to provide *breadcrumbs* along the game. Breadcrumbs are defined as a set of navigational aids provided to the player / user to help in navigation (in this case, it aids the player to navigate through the game overcoming the obstacles faced along the way). It is important to note that providing too many hints can be a bane instead of a boon. Instead of helping the player learning the concepts being taught through game play and encouraging learning through exploration, it might hinder the exploratory nature of the game. Therefore, to prevent that this occurs, the player can be provided the hint / aid along with a penalty. The penalty could be in the form of time taken away from the timer of the mini-math game or a reduction of the overall game score. This would encourage the player to try to overcome his obstacles on his own and to seek help wisely, i.e., only when necessary.

Monkey Tales games has implemented in-game hints using an adaptive pedagogical agent in the form of a *buddy* chosen by the player at the start of the game. A similar solution is also seen in *Prime Club/Clime* game [10]. The mentor can be created explicitly or with modifications to the system by creating new characters each time the player asks for a hint (provided in the form of a conversation line). Therefore, in the example, we have chosen to create a *buddy* explicitly at the start of the game to help guide the user in times of need, and who also can talk to the player at different times and provide him with useful information. In our game, conversations with the buddy have not been explicitly mentioned or denoted in the storyline. This is because the need for the buddy to communicate with the player depends on various factors, such as, player's skillsets, proficiency and so on. Depending on the player and his/her needs, the buddy can be called upon at any point of the gameflow once the player has selected the buddy and his/her avatar.

The choice between these two methods (namely, breadcrumbs and buddy element) of creation of an adaptive pedagogical agent is up to the discretion of the educator since these mentors will have an influence on the player and can have different consequences based on their presentation method.

3.3.2 Assessment

An educational game can aid in tracking the student's moves to assess the outcome based on choices made to help educators better understand the learning progress of each individual player / student. The outcomes of the educational games could be compared to those of less interactive educational tools to track the learning process, to modify the instructional environment to make it more appropriate for the player's requirements. In-game assessment aids educators to notice, highlight and score actions of the player according to the relevance to the learning methodology. Based on these assessments, feedback can be generated to help identifying and also rectifying inaccurate assumptions. This is an integral part of educational games, to continually strive for betterment [21] [13].

Stealth assessment [56] is also gaining popularity through serious games as a methodology of seamlessly integrating assessment and gaming. Stealth assessments are embedded assessments in the learning environment of a serious game such that these assessments are nearly invisible. This can be accomplished by automated scoring and machine-based reasoning to infer things that might be hard for humans (for example, estimating the competencies of a player based on a set of skills). Such assessments can also help in improving the gaming concepts based on a better understanding of the needs and requirements of a player based on occurring patterns and responses.

In our approach, we want to use assessment by adapting the game automatically based on the user's choices throughout the game. For example, if the player is able to solve each of the mini-math games in a short period of time this implies that he has learnt / understood this particular mathematical concept well and the mini-math game could increase in complexity slowly to keep the player engaged in the learning process and to ensure that the complexity of the math game is not too tough for the player to give up playing the game. The direct approach to assessment is by means of collecting the information of the player's choices and adjusting the instructional environment to better suit the player. Communication with the buddy at certain predefined points along the game play can also be used to determine the progress. For instance, if the player fails to finish a min-math game with time constraints within the given time period, then the buddy could talk to the player and informs him that the timer timed out and that the mini-math game has to be played again to complete the mission.

This integration of an '*assessment element*' (Figure 3.9) as a part of the modeling concepts also helps in including evaluation as an integral part of the game. In our language, the *assessment element* is actual the *objective element*. Previously two types of objective elements were mentioned, namely an objective element with time constraint and an objective element without time constraint. Both types of objective elements can be used for assessment. This double use of a single modeling concepts can be perceived as overloading the modeling concept, however in this case this is not a problem as attaining the objective of the objective element can always be considered as a kind of assessment, and as such there is no need to have a separated modeling element for assessment.

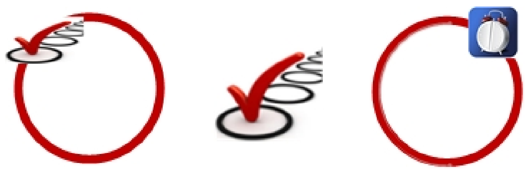

Figure 3.9: Assessment element.

The figure on the left denotes the assessment element without time constraint. The figure in middle highlights the assessment representation. The one on the right denotes the assessment element with time constraint.

Educators might find it hard to make decisions on where to introduce these *assessment elements* to help evaluate the player's progress. However, many educators tend to choose multiple-choice questions to achieve the evaluation. Again, instead of introducing a dedicated modeling element for multiple choice questions, the modeler can realize multiple choice by means of the *choice element*. The Figure 3.10 contains an example. It shows that when the player chooses between game play and playing a mini-math game. If the player chooses, game play then he continues playing the game and if mini-math game is the choice made the player plays the mini-math game and answers the questions asked. The mission is accomplished when the player performs both mini-math game and game-play and also receives the gift.

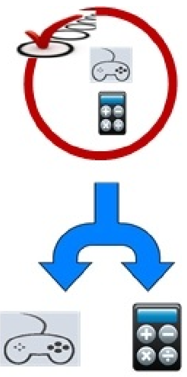

Figure 3.10: Realizing multiple choice

'Choice element' leads to two possible different outcomes based on the player's choice and is represented by a blue-colored split arrow.

A visual representation of mathematical questions with numerical answers also enables the simplification of the process. For instance, the player collects some numbers as a part of game play and then performs the mini-math game using the numbers he has collected. Based on the order in which the numbers were collected and the mini-math game was performed, assessment can be made on the player's strengths and attention can be paid to areas which need improvement.

To implement this, we should re-define the 'objective element' to have an order of tasks as follows, game play (collection of numerical values), mini-math game and continuation of game play. This can be implemented simply by re-definition (proves the ease of modification) without adding new elements or changing the existing layout of the element. This also helps the player visualize better and the integration of the learning aspect would be seamless.

3.3.3 Hierarchical Decomposition

Hierarchical decomposition is a process of decomposing a given system in a top down manner, namely, the system is divided into sub-systems, the sub-systems are divided into different steps, each step consists of various elements. The purpose of using the hierarchical decomposition is to help provide a logical understanding of the various steps in the decomposition model. Elaborating on the various steps involved in each decomposition enables a better understanding of the storyline and the varied concepts used in the storyline.

In this thesis, the generic hierarchy (i.e., the system) is represented as shown in Figure 3.6 which clearly denotes the various sub-systems. Each of these sub-systems can be further decomposed into various steps and the steps are composed of various elements. An example can be found later on as shown in Figures 3.16 to 3.23.

The steps are mainly composed of sets of '*action elements*' and '*event elements*' which the player has to fulfil to move onto the next step of the decomposition. An *action element* denotes the action that has to be performed by the player. It is used in a decomposition step to specify more details. An action can be associated with an object. The *event element* is used to denote an event that could happen along the game. For example, fire in the building is an event. It is used in a decomposition step to specify more details. It can be combined with an object. The representation of the action element and the event element can be seen in Figure 3.12 which lists the various modeling concepts used.

3.3.4 Adaptation

To ensure that a player is not bored and his attention stays with the game, the concept of *dynamic adaptation* or *adaptivity* can be used. This is a process where the game adapts itself to the player's needs based on his choices and game play. This kind of dynamic adaptation will not be considered in the modeling language. We think that it is not useful to overload an educator by asking him to specify this kind of dynamic adaptation; rather we envision some artificial intelligence built in the game runtime environment that performs this automatically.

The differentiation between different types of players such as 'beginners', 'intermediate' and 'expert' also helps modifying the game to grab the attention of each type of players without overwhelming them or causing boredom. This is a crucial aspect as the player must not only be involved in the game play but must also absorb the learning content of the educational game. This adaptation method can be specified using our proposed language.

By default, the player's level of game play will be set as '*default*' where the adaptation is gradual from *easy mode* to *extreme mode* as the levels increase. However, the modeler can also choose to allow the player to choose its level by providing the option of *mode selection* to the player where the player can choose between 'easy mode', 'medium mode', 'hard mode' and 'extreme mode'. This can be given as a '*mode-selection element*' which is realized by the customization of the *start element*. Figure 3.11 illustrates the easy mode, medium mode, hard mode, and extreme mode by altering the color of the start element. Green colored *start element* for '*default / easy mode*', yellow colored *start element* for '*medium mode*', orange colored *start element* for '*hard mode*' and red colored *start element* for '*extreme mode*'.

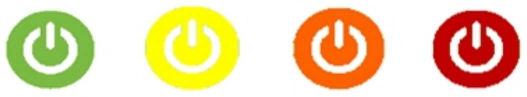

Figure 3.11: Mode selection: representation.

The customization of the start element by changing the color represents the different modes the player can choose from.

NOTE: The player should choose the mode at the start of the game. This is a kind of '*choice*' but which does not result in any further changes to the definition of the storyline, i.e. the steps that follow the 'start element' in the storyline remain the same irrespective of the mode but the *complexity* of the mini-math games and game play should be adjusted (at runtime) to suit the mode selected.

Another adaptation method could be to present the initial game with instructions and guidelines that enable the player to learn the techniques and then from the second level of the game could force the player to pay a penalty for hints already explained previously. It is important to note that these adaptations alter only small, specific sections of the story line and not the entire concept. Thus, introducing this should not be cumbersome.

3.4 Overview of the Modeling concepts and Purpose

Domain specific modeling languages, as the name itself implies, derive their modeling concepts directly from the domain [34] [66] [39]. We have seen earlier that these languages are unique to a specific domain and this helps to raise the level of abstraction. In the following section, we shall review the different domains that are involved in the modeling of an educational mathematics based game, the domain concepts that we have used to build our modeling language, and indicate how the domain concepts are specific to each domain. Lastly, we shall explain the purposes of each of these concepts.

The following is a list of the domains [60] that are involved and their purposes :

Social / Motivational Domain :
This domain influences the way a player interacts or the choices he makes due to social influence. The social influences such as the actions of parents / teachers can influence the player's motivational levels. **Purpose :** A DSML should motivate the player to indulge in the game and learn the educational concepts being offered. *Example:* The '*gifts*' that the player receives on the completion of each mission as a reward is in generally seen as motivating the player to continue.

Intellectual / Educational Domain :

This domain includes cognitive concepts such as the knowledge of the learning domain, existing skills, skills achievable through gaming, perception of failure and success. **Purpose :** The focus of this domain is to assess the player's mathematical skills achieved through gaming and grade them. *Example:* The option of 'game play' task in the 'objective element' helps to enhance the player's game domain knowledge through gaming. The option of 'mini-math game' task in the 'objective element' helps to assess the player's mathematical skills. Thus we are able to promote education through gaming. The 'assessment element' is a part of this domain as well.

Psychological / Emotional Domain :

This domain comprises of the players behavioral and emotional reaction to the gaming states. It is key to focus on this domain since the players feelings decide if he continues gaming and thus learning. **Purpose :** The goal here is to boost the confidence levels of the player and to avoid situations of anxiety. *Example:* 'Buddy element' acts as the motivator for the player by motivating them throughout the game. Also having specific goals at the end of each mission and the final game goal help motivating and encouraging the player psychologically.

The Figure 3.12 depicts a list of the basic modeling elements used, Figure 3.11 depicts a list of the connectors used to connect the different modeling elements and Figure 3.14 depicts a list of the different possible ways of combining the basic modeling elements into different steps.

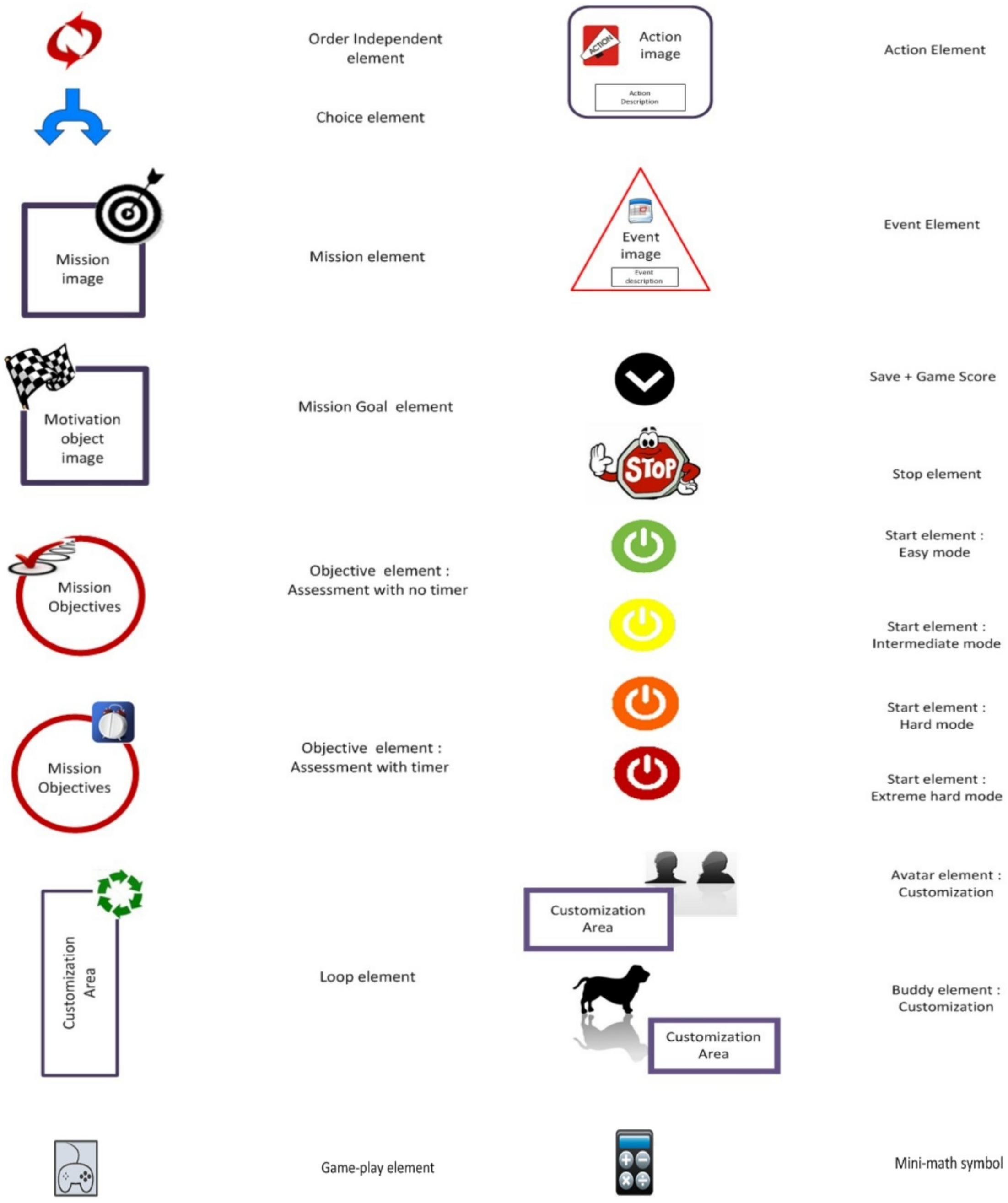

Figure 3.12: Basic Modeling elements

This is a list of all the basic elements used in the modeling language.

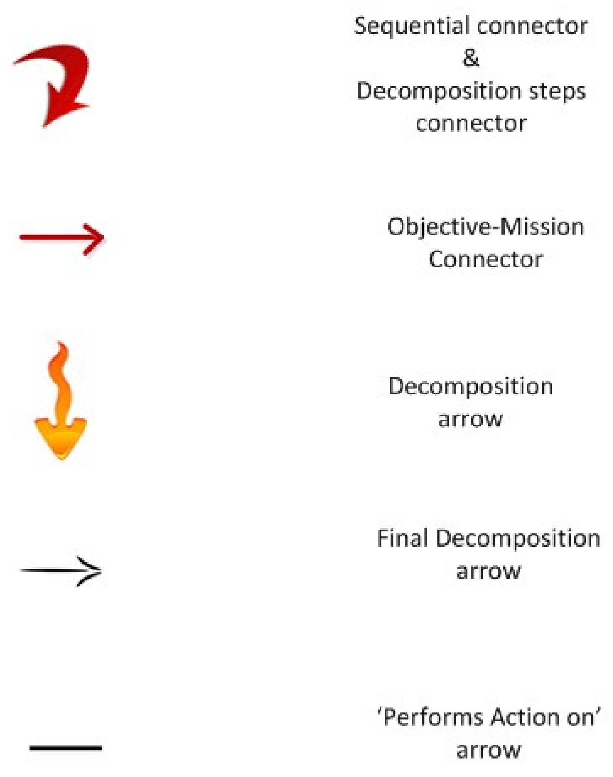

Figure 3.13: Modeling concepts: Connectors

The connectors allow expressing the connections between the modeling concepts used in the model.

Below is a list of the various elements used and their purposes:

Start element:

Start element indicates the start of the game.

Mission element:

Mission element is used to denote the mission that the player should complete. Customization of this element can be used to indicate the gift/reward that the

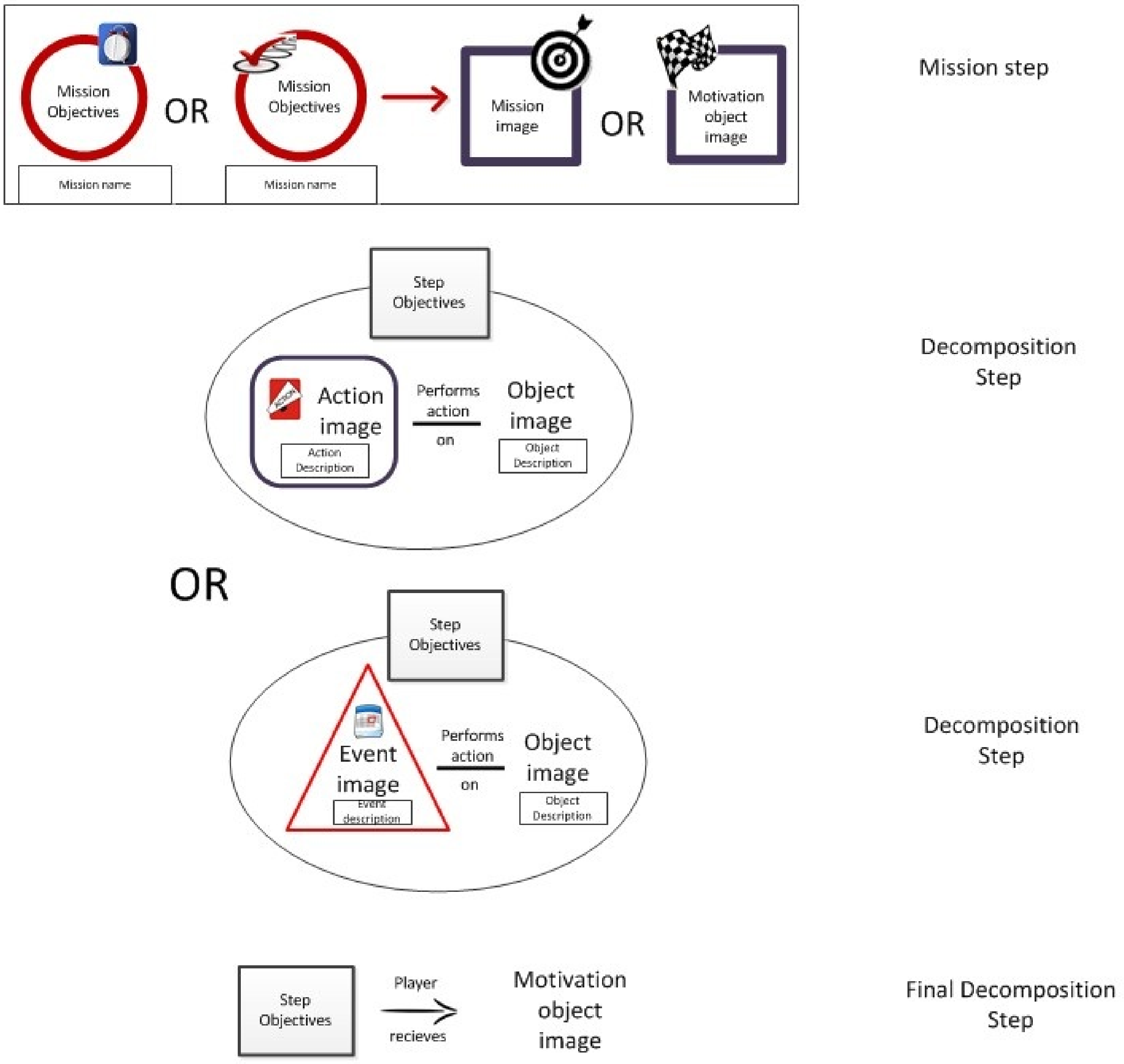

Figure 3.14: Modeling concept: Steps

The different steps are expressed by grouping modeling elements, to facilitate the modeling.

player receives upon completion of the mission. Missions can also indicate the level of complexity of the game play (e.g., indicated by a number (1, 2, ...)).

Objective element:
The objective element is used to indicate the objective that needs to be achieved by the player at the moment in the storyline were it is used. As our domain is mathematical games, the objective is to perform successfully a game play or a mini-math game or both. In our language, we have chosen that game play and mini-math game to be order independent, i.e. the order in which these tasks are performed does not change the outcome of the objective element. Furthermore, the objective element is non-customizable. But there exist two version of the objective element, one with timer and one without timer.

Game play:
Game play is a task to be performed as part of the objective element. This is used to indicate that the player should play a game.

Mini-math game:
Mini-math game is also a task to be performed as part of the objective element. Mini-math games are mathematical-based games. The player has to play and win in-order to successfully reach the objective. This element is used to indicate a learning aspect of game.

Choice element:
Choice element indicates the choices the player has to make during game play. The player has to choose from different options. The player's choices could lead to different consequences and can also aid in assessing his progress.

Order-choice element:

The player has to perform a set of tasks and he is given a choice to select which task he performs first and which one next. The player has to perform all the tasks compulsorily but the order in which he/she performs the task is not important.

Assessment element:

Assessment element is denoted as an objective element. Assessment can be done for different reasons: to assess the progress of the learner or to collect meta-information about the game to improve it (dynamically or manually later on).

Goal element:

The goal element denotes the main goal of the game. It can be considered as a special type of objective element. It indicates the final mission to complete the entire game.

Stop element:

This element denotes the end of the game.

Save element:

This element indicates that the status of the game will be saved. The game can be saved by the player at different points, known as 'save-points'. The player can recover a game from a previous save when required. Save-points are not predefined and can vary from player to player, hence this has not been explicitly depicted. However, in our storyline we have explicitly depicted only the save used at the end of the game to perform a 'final save'. This denotes not just the 'final save' of the game but it also indicates the point of score generation.

Object element:

This element indicates an object in the game. This element is customizable to be able to use a meaningful symbol or icon.

Event element:

The event element is used to denote an event that could happen along the game. For example, fire in the building is an event. It is used in a decomposition step to specify more details. It can be combined with an object.

Action element:

An action element denotes the action that has to be performed by the player on the object. It is used in a decomposition step to specify more details. An action can be associated with an object.

Loop element:

The loop element denotes a looping of actions. For reasons of simplicity, it is not indicated how often the looping has to be repeated or the condition governing the looping.

3.5 Example

In this section we shall provide an example of a mathematical-based game, modeled with our DSML. The example borrows ideas from the Monkey Tales game [61] [62]. The main idea is to illustrate how the language can be used and tailored to the specific needs of the game. Hence, customization will also be depicted to show how the elements were adapted to fit this particular example. The following subsections will deal with the main storyline and the different decompositions.

3.5.1 The Storyline

The example that we consider for this particular example is based on the Monkey Tales game concept as mentioned previously. Figure 3.15 illustrates the tailoring (i.e. customization) of the modeling concepts to illustrate the story line described in section 3.1.2.

At the start, the player can chose to play the game in one of the four different modes. This is denoted by the four differently colored start buttons. We notice that the story line is the same for those 4 options. As mentioned earlier, the mode does not affect the storyline. Next, the *order-choice element* is used to indicate that the player has to choose an avatar to represent the player in the game and a buddy. This choice of avatar and buddy that will be presented to the player is depicted by the different avatars and buddies represented in the purple colored customization area of the avatar and buddy elements respectively. As indicated by the order-choice element, the choice of an avatar and a buddy is order-independent. The player can choose an avatar from the eight given avatars, four representing male avatars and four female avatars and a buddy can be chosen from the three given choices, namely, a puppy, a kitten or a tiger cub. Once the player selects both an avatar and a buddy he can begin with the kernel of the game. The game consists of eight different mission steps. Each mission step is given a name. Each mission step also indicates the type of the objective, as well as the difficulty level of the objective and the mission to be reached.

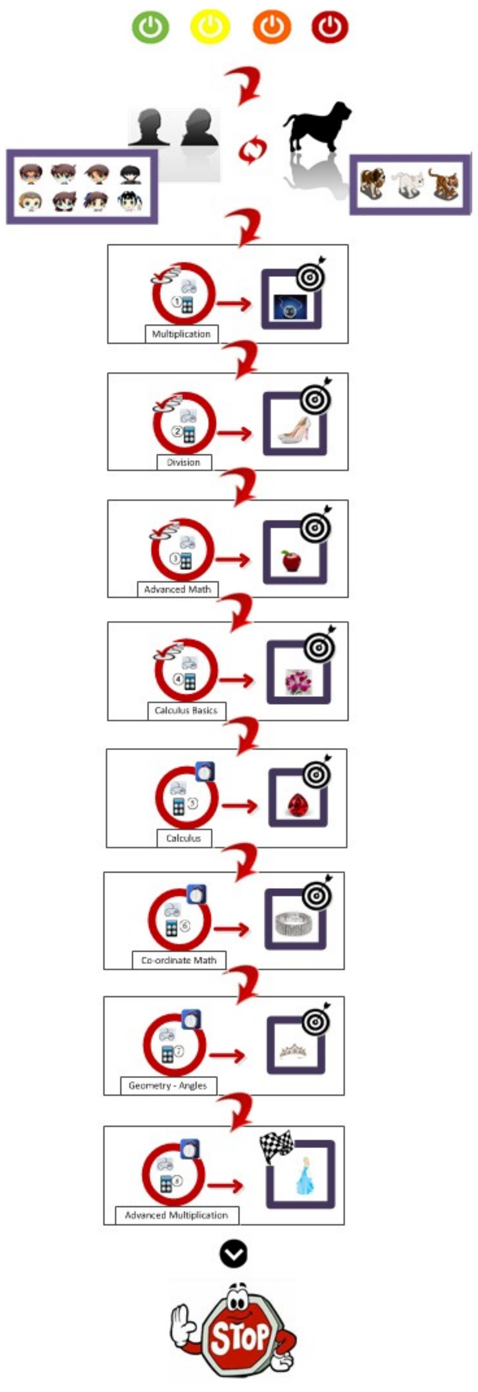

Figure 3.15: Storyline: Rescue the princess.

Illustrates the use of our language using a specific example.

In this storyline, the player performs first the tasks of the *Multiplication* step, namely, indulging in game play and solving mini-math games to complete the first mission, i.e., *mission blue diamond*. This is denoted by the representation of a *blue diamond* in the purple colored customization area of the *mission element*. We notice that there is an assessment of the players' actions being conducted without any time constraints. After successfully completion of this step, the player moves on to the next steps, Division, Advanced Math, Calculus Basic, Calculus, Coordinate Math, Geometry - Angles, and finally Multiplication. Note that each step needs to be completed successfully before the player can move to the next step. Note that the steps are increasing in the level of difficulty and each step has a different mission represented by a different symbol: *mission glass shoes* represented by customizing the mission element to depict glass shoes; *mission magic red apple* depicted by a red apple; *mission purple corsage* depicted by a purple corsage, and so on. Note that the objectives in the four first steps are without any time constraints. Upon the completion of the *mission purple corsage* the player has to complete a time restricted mini-math game which is represented by an objective element with timer. It also shows that the complexity of the mini-math games have been increased, i.e. the player is now playing mini-math games which are more complex that those in the previous missions. The complexity has been increased gradually so that the player does not feel discouraged by a sudden change in complexity. The missions with higher complexity of mini-math games and time constraints are as follows: *mission ruby red* represented by customizing the mission element to depict a red colored ruby stone, *mission princess's tiara* depicted by a tiara, *mission diamond bracelet* depicted by a diamond bracelet, and finally the *mission freedom* represented by customizing the mission element to depict the princess. The final mission is also the goal of the game, hence the use of a *goal element*(with a chequered flag on the top, left hand side of the customizable area). Once the player reached this mission it implies that he has successfully completed all the previous missions of the game and upon completion of this final goal the kernel of the game is finished. The storyline indicates that the game state will be automatically saved along with the players timing and assessment. The player receives a game score. The *stop element* indicates the end of the game.

The model in figure 3.15 provides the main storyline of the game. To provide more details, each step in the storyline can be decomposed into more details. Figure 3.16 provides the decomposition of the first step, Multiplication. In this decomposition, the player has to perform a mini-math game based on 'Multiplication'.

In a similar way, the other steps can also be decomposed. The 8 steps of decompositions in this specific example are given in Figures 3.16 to 3.23. This completes the modeling of the example game.

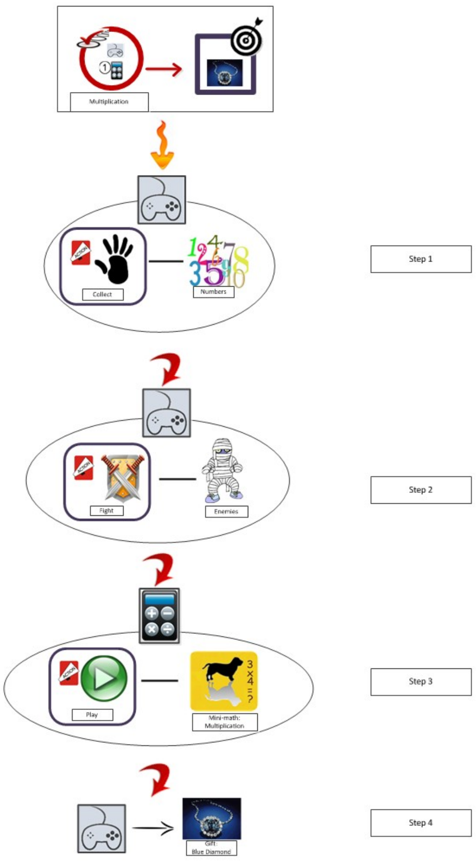

Figure 3.16: Decomposition Step 1: Multiplication

Decomposition model of the 'Mission: Blue Diamond' is detailed showing the steps to be performed by the player to accomplish this mission.

The decomposition model details the following steps:

a) The player has to *collect* (action) the numbers (object) [step 1],

b) Player hears the *fight* (action) with the enemies (object) [step 2],

c) The player has to *play* (action) mini-math games based on Multiplication (object) [step 3] and

d) The player receives a gift, namely, Blue Diamond (object) [step 4].

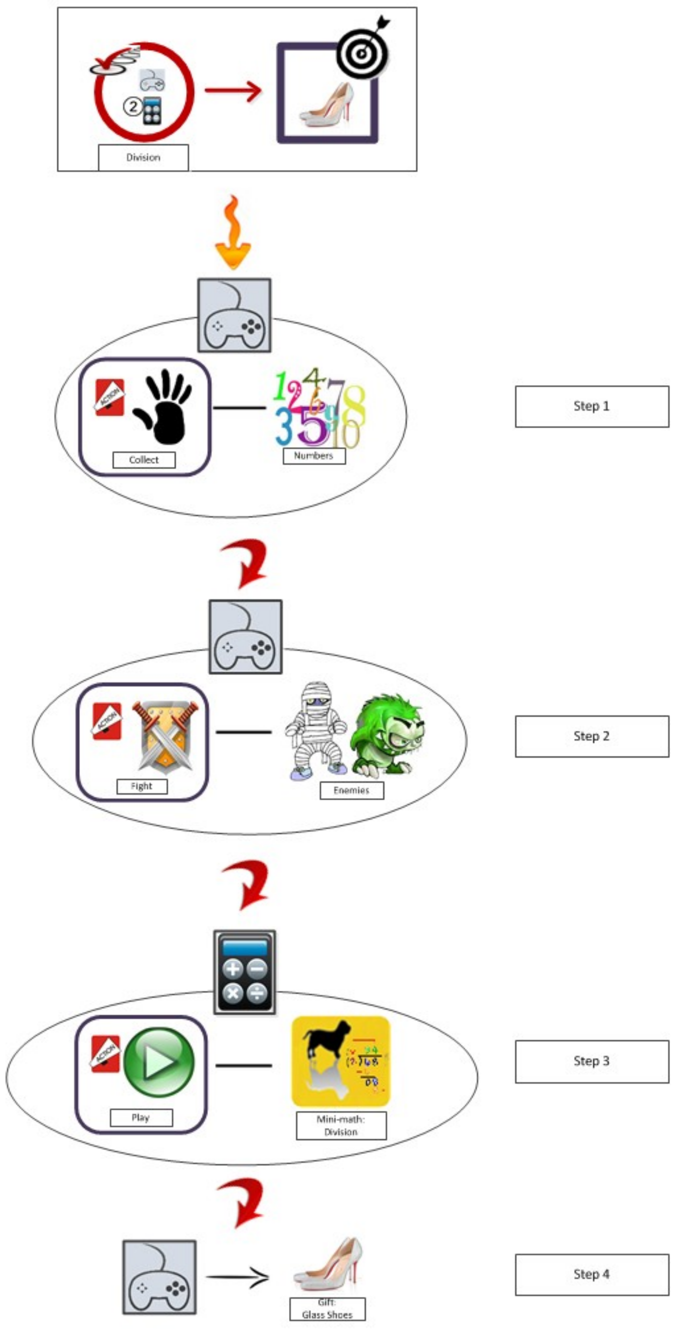

Figure 3.17: Decomposition Step 2: Division

Decomposition model of the 'Mission: Glass Shoes' is detailed showing the steps to be performed by the player to accomplish this mission.

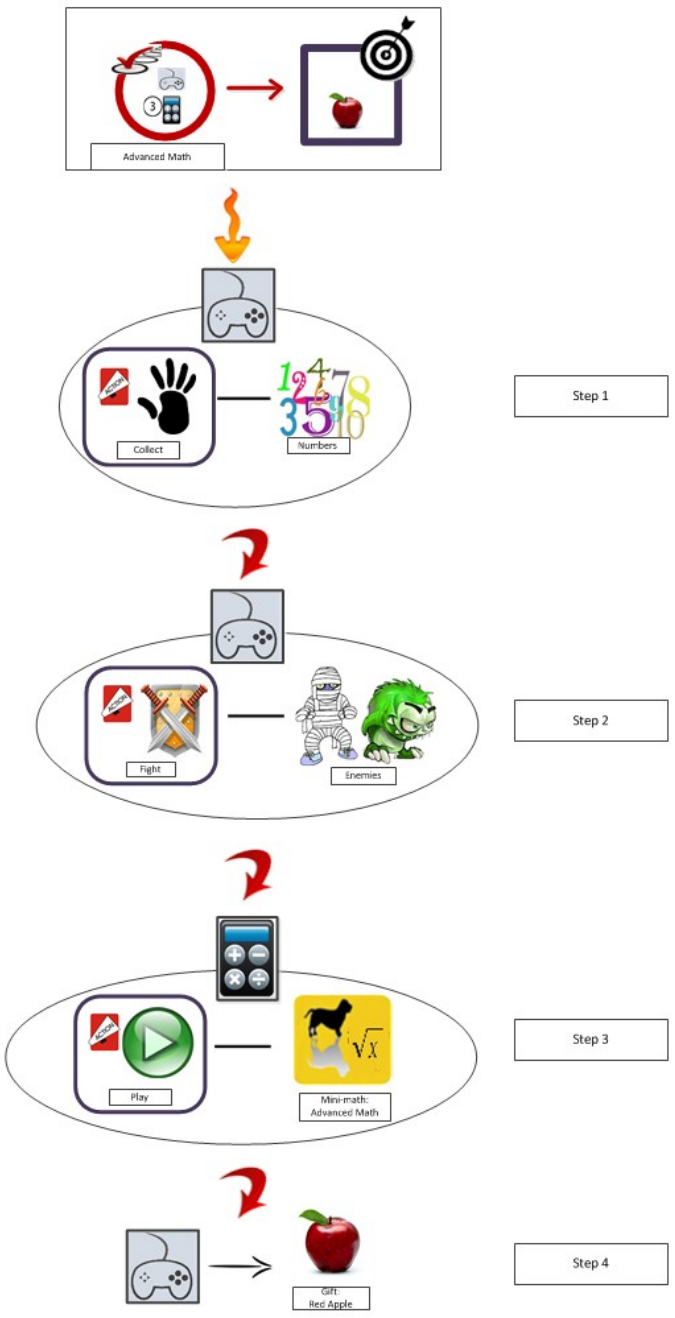

Figure 3.18: Decomposition Step 3: Advanced Math

Decomposition model of the 'Mission: Red Apple' is detailed showing the steps to be performed by the player to accomplish this mission.

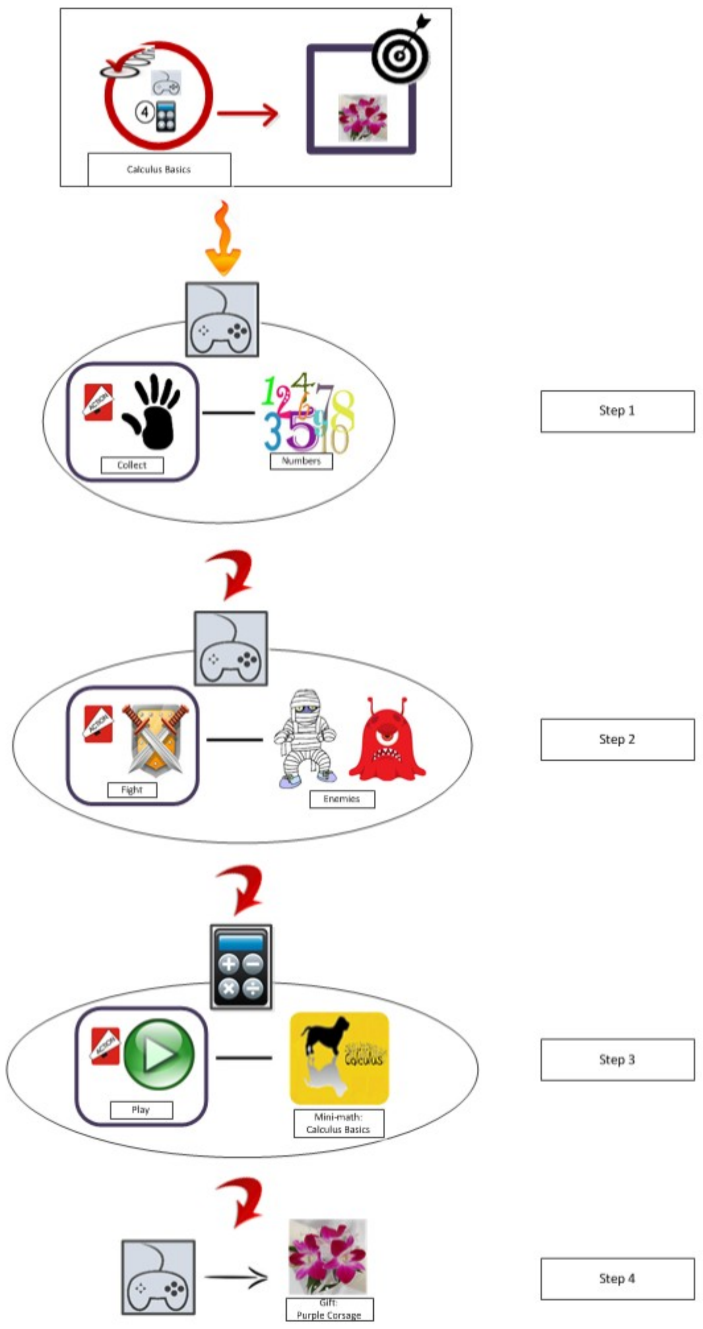

Figure 3.19: Decomposition Step 4: Calculus Basic

Decomposition model of the 'Mission: Purple Corsage' is detailed showing the steps to be performed by the player to accomplish this mission.

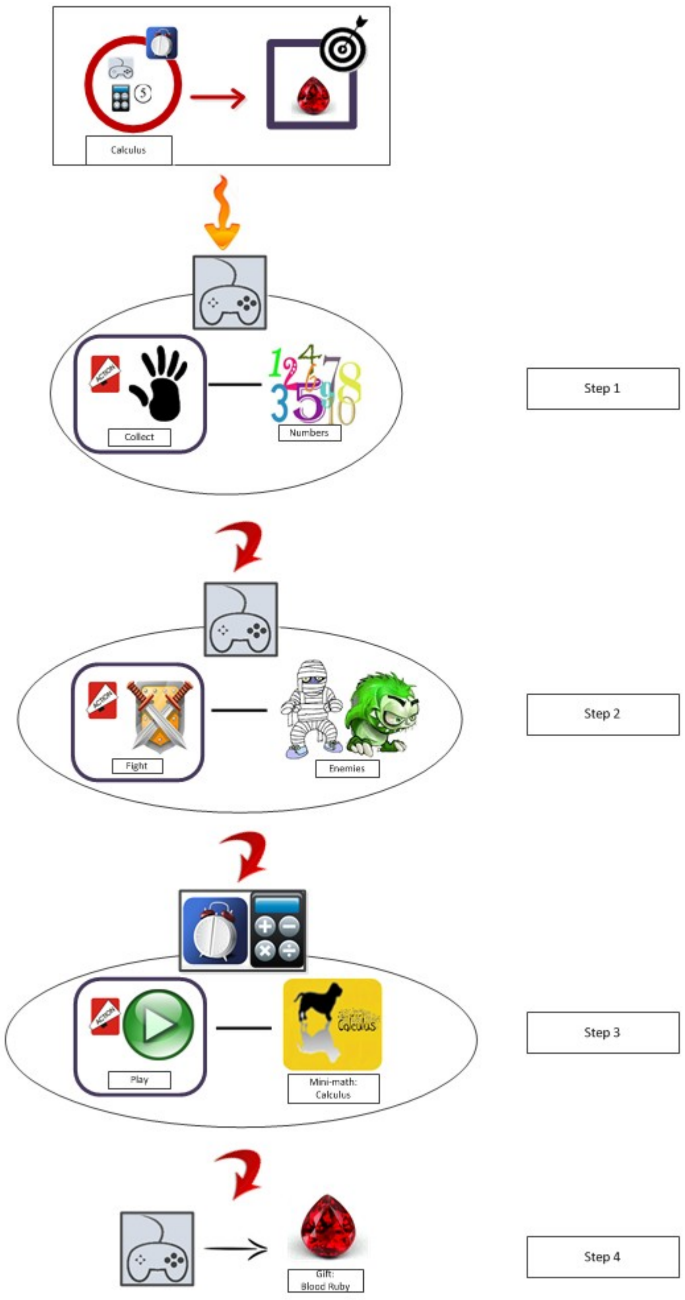

Figure 3.20: Decomposition Step 5: Calculus

Decomposition model of the 'Mission: Blood Ruby' is detailed showing the steps to be performed by the player to accomplish this mission.

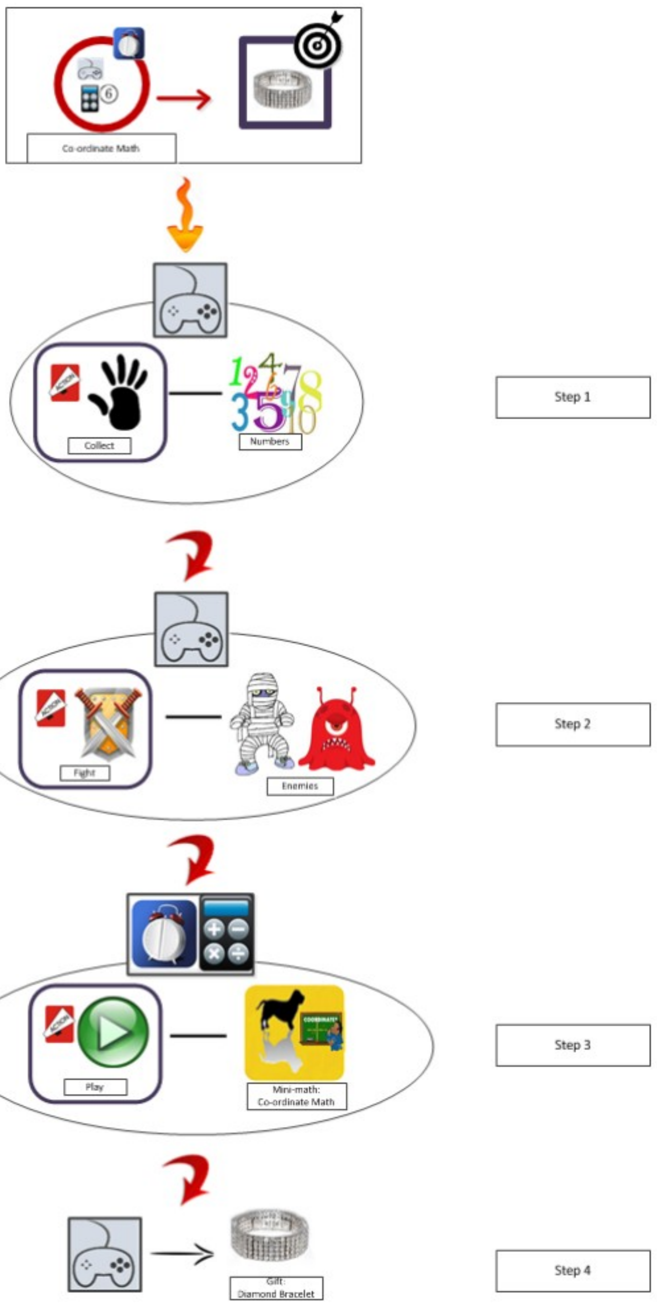

Figure 3.21: Decomposition Step 6: Co-ordinate Math

Decomposition model of the 'Mission: Diamond Bracelet' is detailed showing the steps to be performed by the player to accomplish this mission.

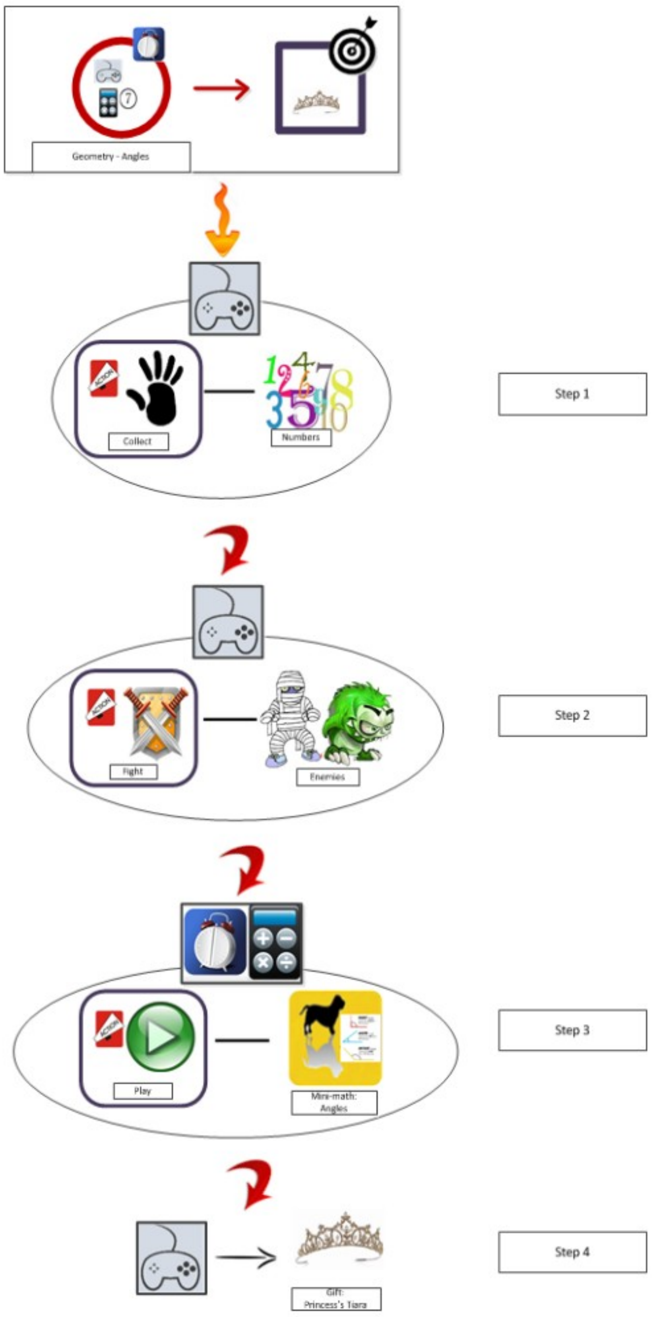

Figure 3.22: Decomposition Step 7: Geometry - Angles

Decomposition model of the 'Mission: Princess's Tiara' is detailed showing the steps to be performed by the player to accomplish this mission.

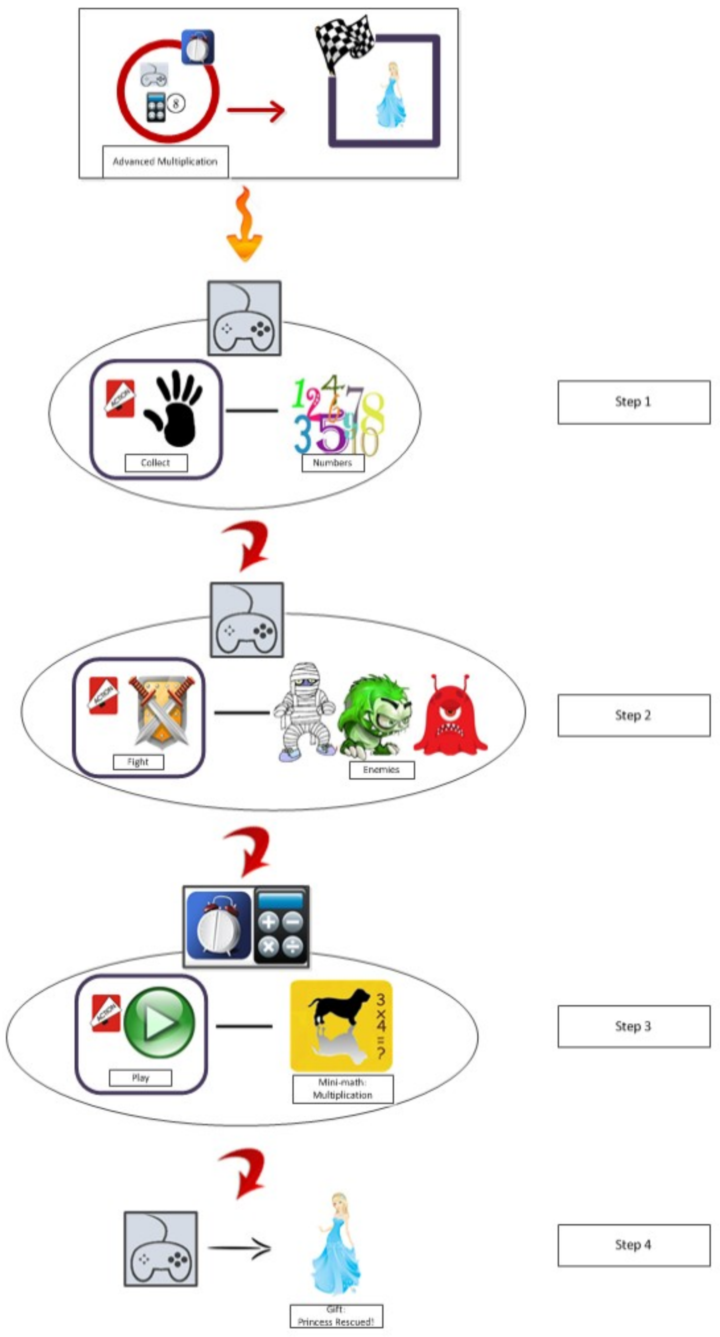

Figure 3.23: Decomposition Step 8: Advanced Multiplication

Decomposition model of the 'Goal: Princess Rescued' is detailed showing the steps to be performed by the player to accomplish this mission.

Chapter 4

Related Work

In this chapter, we discuss related work and we provide a comparison of our Domain Specific Modeling Language with the Domain specific visual language provide by Marchiori et al. in the paper titled "A visual language for the creation of narrative educational games" [41]. This work is very close to ours, therefore we will compare their language with ours. For this we will elaborated their example used in the paper using our language.

4.1 Overview of Related Work

A paper that is closely related to this thesis is named "A visual language for the creation of narrative educational games" by Marchoiri et al. [41]. The abstract of this paper reads as follows "*This paper presents a DSVL that simplifies educational video game development for educators who do not have programming backgrounds. Other solutions that reduce the cost and complexity of educational video game development have been proposed, but simple to use approaches tailored to the specific needs of educators are still needed. We use a multidisciplinary approach based on visual language and narrative theory concepts to create an easy to understand and maintain description of games. This language specifically targets games of the adventure point-and-click genre. The resulting DVSL uses an explicit flow representation to help educational game authors (i.e. educators) to design the story-flow of adventure games, while providing specific features for the integration of educational characteristics (e.g. student assessment and content adaptation). These highly visual descriptions can then be automatically transformed into playable educational video games.*"

Focus is on concepts that have been considered in this particular thesis as well, such as, hierarchical representation notation, assessment evaluation and so on was observed. This paper helped in making a valid comparison of the language proposed with an existing language. It highlighted the drawbacks of the existing language as well as the limitations of the proposed DSML. The following section describes the comparative analysis of these two modeling languages.

The paper "The use of computer games as an educational tool: identification of appropriate game types and game elements" by Amory et al. [2] helped in the identification of game types and game elements. This paper is considered as a part of related work since it helped understand and specify an important aspect of this thesis, which is game element identification. UML was a language that was initially studied as a modelling language, but various other visual domain oriented modeling languages (such as MetaEdit and metaCASE tools) were analysed to better understand the scope of this thesis. "Visual domain-specific modelling: Benefits and experiences of using metaCASE tools" by Kelly et al. [34] helped in detailing the benefits of using these tools from the creation of domain meta-models to code generation from these visual models created. This paper helped in setting the rules and standards of element creation used in this thesis. The paper by Daniel L Moody named "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering" [43] was extremely helpful in the creation of visual notations of the DSML used in our thesis. This paper helped immensely to evaluate and design the visual representations of game elements and constructs effectively.

4.2 Comparison with the Visual Modeling Language of Marchiori et al.

Marchiori et al. present a Domain Specific Visual Language (DSVL) in their paper "A visual language for the creation of narrative educational games" [41]. According to the authors, this DSVL simplifies educational development for educators who do not have programming backgrounds. They used a multidisciplinary approach based on visual language and narrative theory concepts (such as pop-up blocks with text to help guiding the user through gameplay) to create easy to understand games. Their language specifically targets games of the adventure *point-and-click* genre.

The main gameflow and storyline is visualized and depicted as shown in Figure

4.1. To illustrate their language, they used a Fire protocol game, in which the player must carry out an evacuation protocol during a fire. They created a DVSL which used an explicit flow representation to help educators to design the storyflow of adventure games, while providing specific features for the integration of educational characteristics (e.g. student assessment and content adaptation). These resulted in a series of highly visual descriptions.

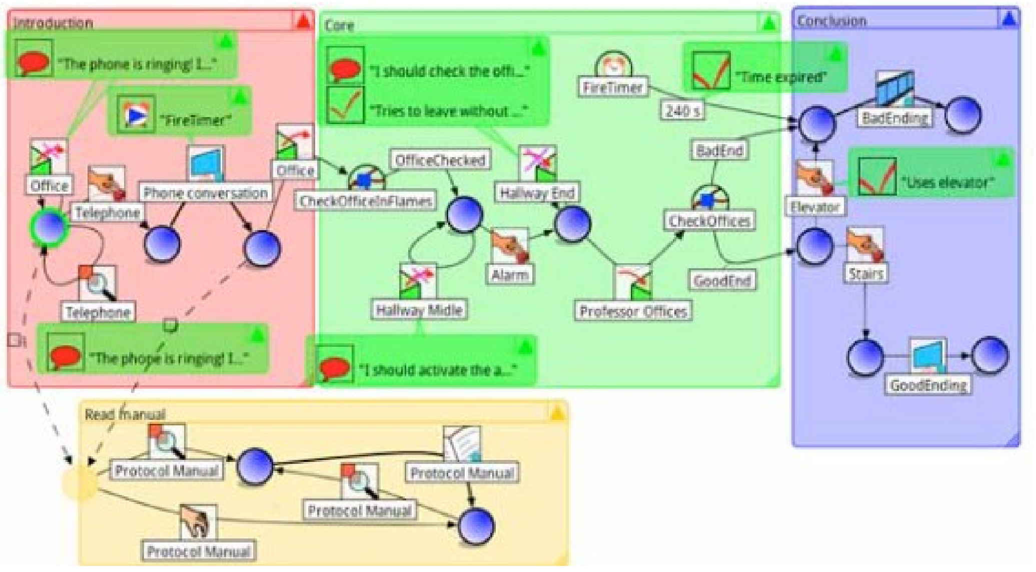

Figure 4.1: Storyflow: Fire Alert (Marchiori et al.)

Illustrates the story flow of Marchiori's fire alert game depicted using the modeling constructs from the <e-adventure> game.

The DSVL is using *state diagram* elements such as the 'start state', 'intermediate states', 'transition lines' from one state to another associated with an 'action' or loops to a state. Figure 4.1 shows these elements and the main sections of the storyflow, namely 'Introduction', 'Core, and 'Conclusion'. These sections are intended to guide the user through the gameplay using concepts of story narration, namely, introduction to the storyline, core content and conclusion of the story. The chat bubbles with text are the pop-up dialogue boxes used to guide the player and provide narration to a section of the game when required. Actions which have to be performed by the player are depicted using action symbols (such as, hand symbol or office symbol) with associated action description in a text box below the symbol.

Though the description has been elaborated in a descriptive manner, upon comparison we realized adaptation of the storyline based on our modeling language was not always feasible as many constructs had been modeled only to fit their particular language. The following section elaborates the application of our DSML to the example shown above.

4.2.1 Application of our DSML to Marchiori's example(Fire Alert)

In this section, we will model the example as described by Marchiori et al. [41], being a fire alert game, using our modeling language. By doing this, on the one hand we are able to compare the expressive power of the two languages and on the other hand it allows to better understand the shortcomings and advantages of our modeling concepts and language.

The storyline for the fire alert game is given in Figure 4.2. It shows the different missions that a player has to successfully complete to move ahead in the game and reach the final goal. The storyline is pure sequential. The player can choose between three difficulty levels and then four mission steps needs to be performed. The objectives only contain game play and are not timed constrained.

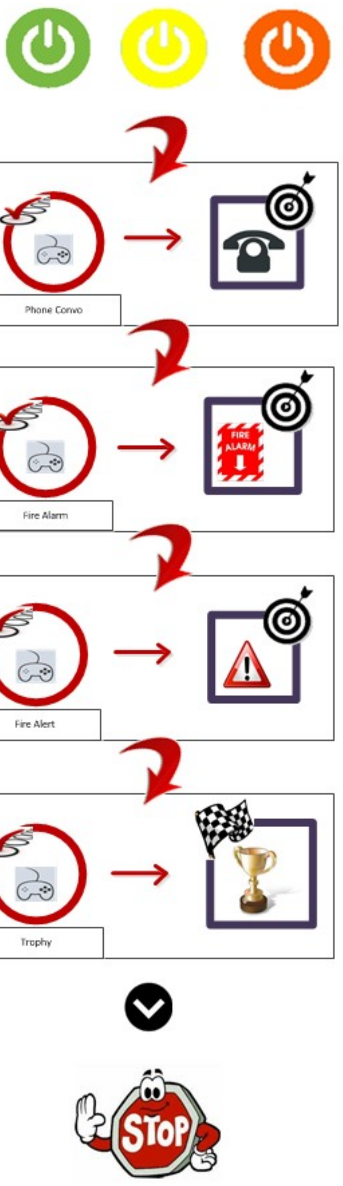

Figure 4.2: Storyline: Fire Alert

Models the storyline of Marchiori's fire alert game using the modeling concepts of our language.

To achieve a mission several other steps need to be taken by the player. The steps are further specified by means of decomposition models. Hence, each mission step has a decomposition model. We shall now look into the decomposition models of the fire alert example. Figure 4.3, Figure 4.4, Figure 4.5 and Figure 4.6 are the decomposition models for this particular example, corresponding to the missions '*phone*', '*fire alarm*', '*fire alert*' and '*trophy*' respectively.

Each of these decomposition models elaborates on the steps or choices that can be made by the player during the game play to accomplish the mission. As indicated in figure 4.2, the player needs to achieve the missions one after the other. This continues till the player reached the goal and completes the entire game.

The decomposition models are detailed alone with the models.

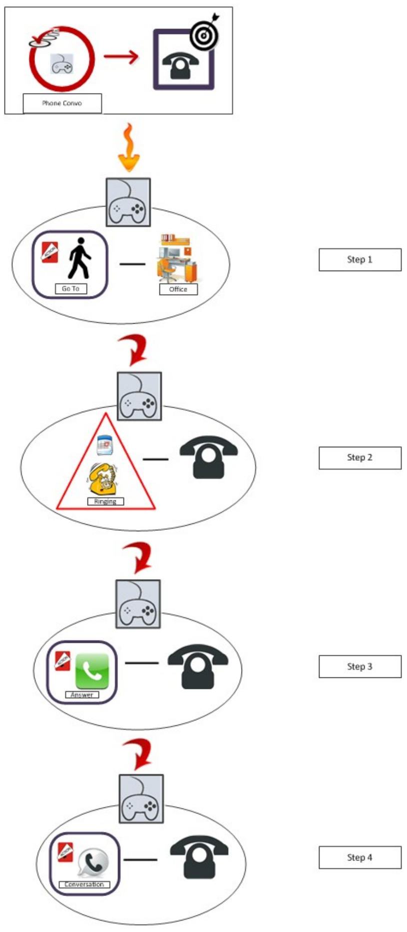

Figure 4.3: Decomposition model: Phone

Decomposition model of the 'mission : phone' is detailed showing the steps to be performed by the player to accomplish this mission.

In Figure 4.3, the player has to make a phone conversation. The decomposition model details the following steps:

a) The player has to *go to* (action) the office (object) [step 1],

b) Player hears the *ringing* (event) of the phone (object) [step 2],

c) The player has to *answer* (action) the phone (object) [step 3] and

d) The player has a *conversation* (action) on the phone (object) [step 4].

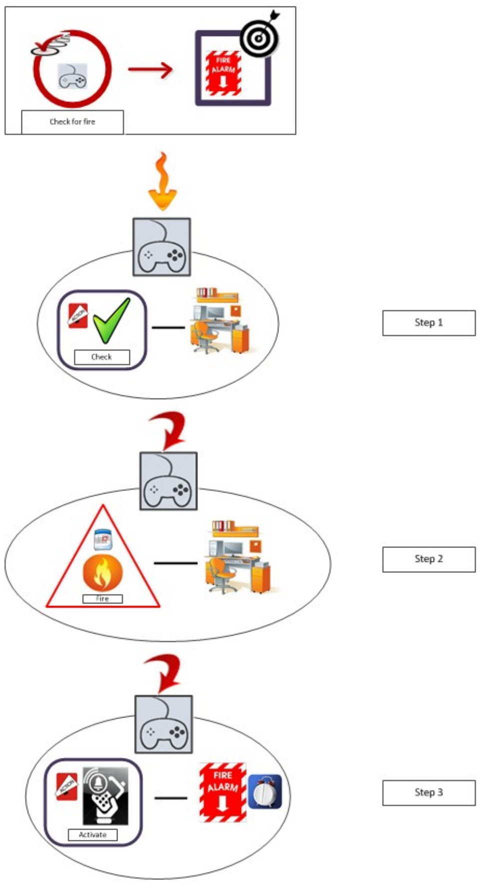

Figure 4.4: Decomposition model: Fire Alarm

Decomposition model of the 'mission : fire alarm' is detailed showing the steps to be performed by the player.

In Figure 4.4, the player activates the fire alarm. The decomposition models details the following steps:

a) The player has to *check* (action) the office (object) [step 1],

b) If there is a *fire* (event) in the office (object) [step 2]

c) The player has to *activate* (action) the fire alarm (object) [step 3].

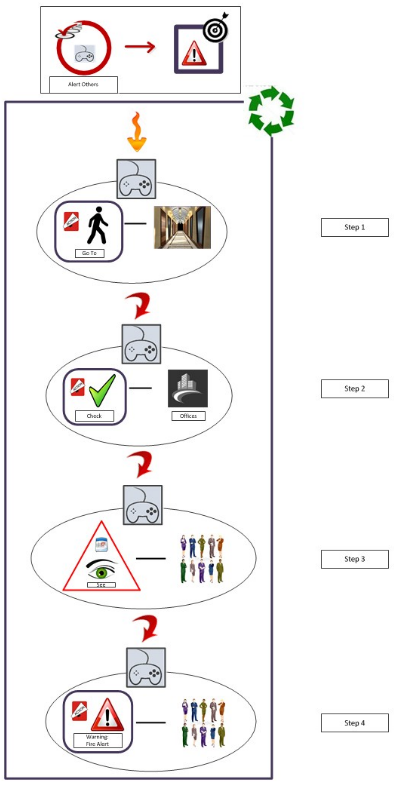

Figure 4.5: Decomposition model: Fire Alert

Decomposition model of the 'mission : fire alert' is detailed showing the steps to be performed by the player.

In Figure 4.5, the player has to alert the people in the different rooms by issuing a fire warning alarm. The decomposition models details the following steps:

a) The player has to *go to* (action) the hallway (/corridor) (object) [step 1],

b) Player has to *check* (action) an office (object) [step 2],

c) When the player *sees* (event) people (object) [step 3] then

d) The player issues a *fire warning* (action) to the people (object) [step 3].

Rule: Once the player finishes step (d) he goes back to step (a) and loops (until all rooms are checked). Note that the condition "until all rooms are checked" is not expressed in the model.

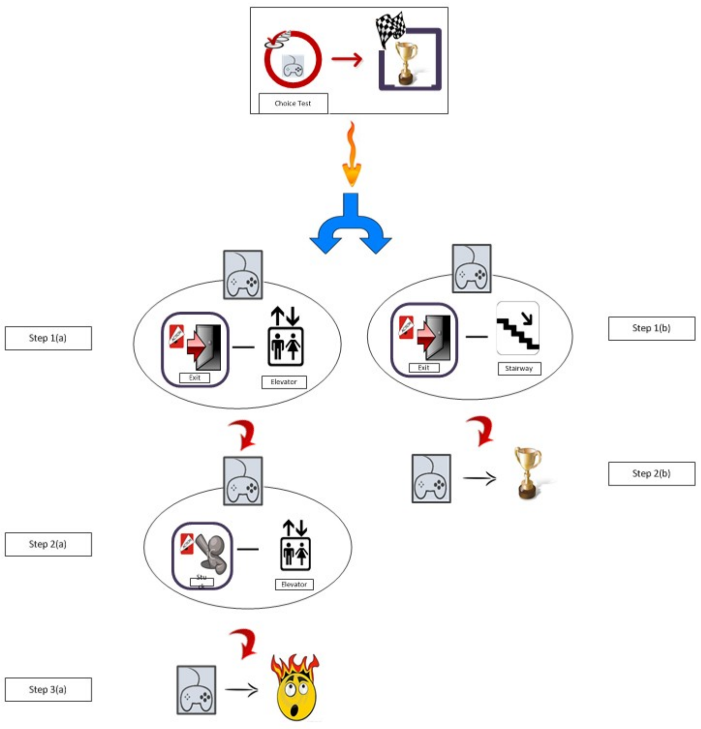

Figure 4.6: Decomposition model: Trophy

Decomposition model of the 'mission : trophy' is detailed showing the steps to be performed by the player.

In Figure 4.6, the player has to exit the building by making a choice between the stairs and the elevator. If he makes the right decision (the stars) he receives a trophy (final goal). The decomposition models details the following steps:

a1) The player *exits* (action) using the elevator (object) [step 1a] or

a2) The player *exits* (action) using the stairway (object) [step 1b],

b1) The player is *stuck* (action) in the elevator (object) [step 2a]

b2) The player receives a trophy (object) [step 2b]

c1) The player is engulfed in flames (object) [step 3a]

Note: The player can choose between step(a1), steps (b1) and step (c1) and step (a2), step (b2).

These decomposition models, when combined with the storyline model (Figure 4.2) depict all the details of the game; they model the entire storyflow as described by Marchiori et al. [41]. Therefore, our modeling language could successfully model this example.

In the next subsection, we will discuss some of the advantages and drawbacks of modeling an existing storyline using our notations.

4.2.2 Discussion

- The main advantage was the flexibility of the modeling constructs. When we compare the storyline used in our specific example (Section 3.4.1, Figure 3.13) to the ’fire alert’ story we notice that the *storyline* is very different in both cases but the *modeling constructs* could be applied to both to specify the details. For example: The ’fire alert’ model is based entirely on game play without any educational content being explicitly played. The player learns about the rules of fire alert implicitly through game play. This could also be expressed through the use of an "assessment element" to evaluate the players learning.
- An inconvenience of our model in comparison to Marchiori et al.’s is that they depicted the entire storyflow in one single model with ’introduction, ’core’ and ’conclusion’ sections whereas we have chosen to elaborate the steps in separate decomposition models. This has simplified the understanding process of the entire story but it may be more difficult to get "the complete picture", i.e. a detailed step-by-step view of the game. On the other hand, we think that the approach of Marchiori will not scale well. For large

games, it will be impossible to put everything in one model and still keep an overview. Our approach looks more scalable. In addition, it has added structure to the entire story without losing expressiveness.

4.3 Limitations

This section will deal with the limitation that have been noted in this thesis. Due to time constraints it was not possible to fix all the limitations and it can be handled as a part of future work.

The concepts of *loops* was introduced to show sections over which the player would have to loop and perform the steps until a condition is fulfilled. However, the control over these loops have not been mentioned explicitly. The player loops until a condition unknown to the player is reached. For instance, let us consider the decomposition model of Fire Alert (Figure 4.5), here the condition is that the player loops until there are no more people in the offices and the building is evacuated completely. But this is unknown to the user. It can be mentioned and represented in the storyline through the use of breadcrumbs.

The concept of *save-points* is also implicit and have not been mentioned explicitly. However, it can be modeled into the storyline by fixing the save-points as fixed points for all players.

Buddy and user interaction is also implicit and have not been denoted. The conversation can be pre-defined and denoted in the storyline through the usage of *breadcrumbs* or *pop-up dialogue boxes*.

Chapter 5

Conclusions and Future work

The thesis focuses mainly on presenting a new DSML that aims to create a easy transition from storyboards or text-based description to a playable computer game by explicitly modelling the concepts of the game story flow. To achieve this, the processes of creating an educational math-based game is illustrated. The End-User Development(EUD) guidelines [19] have been followed to ensure that the target users are educators and domain experts. It is important to observe that educators may have limited or no technical knowledge / background but should be involved in the process of developing new games and / or modifying existing ones to tailor them to the needs of the player. Our modelling language ensures that the learning curve of the language is not too steep for the educators.

The language designed can be extended based on future works to include new concepts and elements to be tailored to the needs of different sets of players with different requirements. For example, in our case the game tailors itself to fit to the needs of the player implicitly by means of dynamic adaptation but also explicitly through assessments. This helps in decreasing the effort that is required to build a final educational game as the gaming aspects and the educational aspects are interwoven together in a seamless manner.

To demonstrate the usage of the modelling language built, we modeled a specific game example which shows the advantages and capabilities of this particular DSML. The specific example is chosen to fit the domain we have chosen previously as mathematics-based. It also shows the hierarchy in the language used to represent the story flow in a compact manner and to easily be able to comprehend the visual representation of the story flow based on the high-level modelling concept, namely, story-line.

A comparison of our DSML to an existing DSML is also conducted to show the benefits and the differences between the two modelling languages. In the comparison stage, we considered the possibilities of adapting our language to the existing language's domain or vice versa. By choosing the later, we have demonstrated not only the shortcomings of the existing language in comparison but also the advantages.

The DSML designed can be used for future work to define the transformation into a playable game, i.e. to achieve full code generation. Using the concepts and simplifications enlisted we can provide the mechanisms required to achieve direct transformation of the games' visual representations into fully functional playable games.

Future work can also include the creation of different components in the DSML to include development of language constructs such as representations of questions and tests to be presented to the player to conduct effective assessment. This would help in achieving better evaluations of the players level of game usability, playability, motivational levels and learning abilities. This would help in building models which can cater to the specific needs of the user to motivate and encourage the player to be indulge in game play. In our language, we have implemented this by providing hints to the user and through user communication with a pedagogical agent, namely, the buddy. Focus on translation of the DSML to full code-generation of the model after the identification of specific elements in various fields of education can be taken up as future work as well. The scope of future work can also be extended over the section 4.3 "Limitations" of this thesis.

Bibliography

[1] *DSM '10: Proceedings of the 10th Workshop on Domain-Specific Modeling*, New York, NY, USA, 2010. ACM.

[2] Alan Amory, Kevin Naicker, Jacky Vincent, and Claudia Adams. The use of computer games as an educational tool: identification of appropriate game types and game elements. *British Journal of Educational Technology*, 30(4):311–321, 1999.

[3] Jordan Ayan. Visual diagramming adds impact to ideas, 2002.

[4] Ramazan Basturk and Ph D. The effectiveness of computer-assisted instruction in teaching introductory statistics. *Educational Technology & Society*, 8(2):170–178, 2005.

[5] D. Bickerton, T. Stenton, and M. Temmermann. *Criteria for the evaluation of authoring tools in language education*. Swets & Zeitlinger, 2000.

[6] Marat Boshernitsan and Michael S. Downes. Visual programming languages: A survey. Technical Report UCB/CSD-04-1368, EECS Department, University of California, Berkeley, Dec 2004.

[7] H. Brody. "video games that teach?," technology review. 96:51–57, 1993.

[8] P Brusilovsky. *Developing adaptive educational hypermedia systems: From design models to authoring tools*, pages 377–409. Kluwer Academic Publishers, 2003.

[9] Barrett R. Bryant, Jeff Gray, and Marjan Mernik. Domain-specific software engineering. In *FoSER*, pages 65–68, 2010.

[10] Cristina Conati. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence*, 16(7-8):555–575, 2002.

[11] Paul Coulton, Kate Lund, and Andrew Wilson. Harnessing player creativity to broaden the appeal of location based games. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, BCS '10, pages 143–150, Swinton, UK, UK, 2010. British Computer Society.

[12] Sara de Freitas and Martin Oliver. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Comput. Educ.*, 46(3):249–264, April 2006.

[13] Guillaume Denis and Pierre Jouvelot. Motivation-driven educational game design: applying best practices to music education. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACE '05, pages 462–465, New York, NY, USA, 2005. ACM.

[14] M.D. Dickey. Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education. *British Journal of Educational Technology*, 36(3):439–451, 2005.

[15] Michele D. Dickey. "ninja looting" for instructional design: the design challenges of creating a game-based learning environment. In *ACM SIG-*

GRAPH 2006 Educators program, SIGGRAPH ’06, New York, NY, USA, 2006. ACM.

[16] George Edwards, Yuriy Brun, and Nenad Medvidovic. Automated analysis and code generation for domain-specific models. In *the joint 10th Working IEEE/IFIP Conference on Software Architecture and 6th European Conference on Software Architecture (WICSA/ECSA)*, Helsinki, Finland, August 2012. Extended and revised version of [**?**]Edwards11ase. A previous version appeared as University of Southern California, Center for Software Engineering technical report USC-CSSE-2010-517.

[17] Klopfer Eric, Osterweil Scot, and Salen Katie. Moving learning games forward. Technical report, 2009.

[18] Shalom M. Fisch. Making educational computer games "educational". In *Proceedings of the 2005 conference on Interaction design and children*, IDC ’05, pages 56–61, New York, NY, USA, 2005. ACM.

[19] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev. Meta-design: a manifesto for end-user development. *Commun. ACM*, 47(9):33–37, September 2004.

[20] Robert B. France and Bernhard Rumpe. Domain specific modeling. *Software and System Modeling*, 4(1):1–3, 2005.

[21] R Garris, R Ahlers, and J E Driskell. Games, motivation, and learning: A research and practice model. *Simulation & Gaming*, 33(4):441–467, 2002.

[22] James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, October 2003.

[23] Michael Goodman. System thinking as a language. 2(3), 1991.

[24] Bas Graaf. Model-driven evolution of software architectures. In *Proceedings of the 11th European Conference on Software Maintenance and Reengineering*, CSMR '07, pages 357–360, Washington, DC, USA, 2007. IEEE Computer Society.

[25] Margaret E Gredler. *17 . EDUCATIONAL GAMES AND SIMULATIONS : A TECHNOLOGY IN SEARCH OF A (RESEARCH) PARADIGM*, pages 521–540. Number 39. MacMillan, 1996.

[26] Glenda A Gunter, Robert F Kenny, and Erik H Vick. Taking educational games seriously: using the retain model to design endogenous fantasy into standalone educational games. *Educational Technology Research & Development*, 56(5-6):511–537, 2007.

[27] Richard Halverson, David Shaffer, Kurt Squire, and Constance Steinkuehler. Theorizing games in/and education. In *Proceedings of the 7th international conference on Learning sciences*, ICLS '06, pages 1048–1052. International Society of the Learning Sciences, 2006.

[28] J. C. Herz. *Joystick Nation: How Videogames Ate Our Quarters, Won Our Hearts, and Rewired Our Minds.* Little, Brown & Co. Inc., Boston, MA, USA, 1st edition, 1997.

[29] J. Hight and J. Novak. *Game development essentials: game project management.* Game Development Essentials. Thomson Delmar Learning, 2007.

[30] Henry Jenkins, Eric Klopfer, Kurt Squire, and Philip Tan. Entering the education arcade. *Comput. Entertain.*, 1(1):8:1–8:11, October 2003.

[31] Edward Ju and Christian Wagner. Personal computer adventure games: their structure, principles, and applicability for training. *SIGMIS Database*, 28(2):78–92, April 1997.

[32] Mika Karaila. Evolution of a domain specific language and its engineering environment - lehmanâĂŹs laws revisited. Thesis, 2009.

[33] Gabor Karsai. A configurable visual programming environment: A tool for domain-specific programming. *Computer*, 28:36–44, 1995.

[34] S. Kelly and J.P. Tolvanen. *Domain-specific modeling: enabling full code generation*. Wiley-Interscience publication. Wiley-Interscience, 2008.

[35] J Kirriemuir and A McFarlane. Literature review in games and learning: a report for nesta futurelab. *NESTA Futurelab series report 8*, 5, 2003.

[36] Digital Game-based Learning. Fun , play and games : What makes games engaging. *Scientist*, pages 1–31, 2001.

[37] Detlev Leutner. Guided discovery learning with computer-based simulation games: Effects of adaptive and non-adaptive instructional support. *Learning and Instruction*, 3(2):113–132, 1993.

[38] C A Lindley. Story and narrative structures in computer games. *Bushoff Brunhild ed*, (January):1–27, 2005.

[39] Janne Luoma, Steven Kelly, and Juha-Pekka Tolvanen. Defining domain-specific modeling languages: Collected experiences. In *Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM 04)*, October 2004.

[40] T. W. Malone. Toward a Theory of Intrinsically Motivating Instruction*. *Cognitive Science*, 5(4):333–369, 1981.

[41] Eugenio J Marchiori, Ángel Del Blanco, Javier Torrente, Iván Martinez-Ortiz, and Baltasar Fernández-Manjón. A visual language for the creation

of narrative educational games. *Journal of Visual Languages Computing Computing*, 22(6):443–452, 2011.

[42] A McFarlane, A Sparrowhawk, and Y Heald. Report on the educational use of games: An exploration by teem of the contribution which games can make to the education process. page p26, 2002.

[43] Daniel L. Moody. The x201c;physicsx201d; of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35:756–779, 2009.

[44] Shazia Mumtaz. Children's enjoyment and perception of computer use in the home and the school. *Comput. Educ.*, 36(4):347–362, May 2001.

[45] J. Orwant. Eggg: automated programming for game generation. *IBM Syst. J.*, 39(3-4):782–794, July 2000.

[46] Seymour Papert. Does easy do it? children, games, and learning. *Game Developer magazine*, 1988.

[47] M Paquin. Effects of a museum interactive cd-rom on knowledge and attitude of secondary school students in ontario. *International Journal of Instructional Media*, 56:101–111, 2002.

[48] Marc Prensky. Computer games and learning- digital game-based learning. *Handbook of computer game studies*, 18:97–122, 2005.

[49] Clark N. Quinn. Designing educational computer games. In *Interactive Multimedia in University Education*, pages 45–57, 1994.

[50] Emanuel Montero Reyno and José Á. Carsí Cubel. Model driven game development : 2d platform game prototyping. In *GAMEON*, pages 5–7, 2008.

[51] Lloyd P Rieber. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research & Development*, 44(2):43–58, 1996.

[52] N. Roberts. *Simulation Gaming: a critical review*. Computer Microfilm Internat. Corp, 1978.

[53] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2004.

[54] Shane Sendall and Wojtek Kozaczynski. Model transformation: The heart and soul of model-driven software development. *IEEE Softw.*, 20(5):42–45, September 2003.

[55] D W Shaffer, K R Squire, R Halverson, and J P Gee. Video games and the future of learning. *Phi Delta Kappan*, 87(2):104–111, 2005.

[56] Valerie J. Shute, Matthew Ventura, Malcolm Bauer, and Diego Zapata-Rivera. *Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow*. Routledge/LEA, 2009.

[57] R.A. Smith. *Educational games in today's learning*. ERIC Reports. San Jose State University, Industrial studies department, 1976.

[58] K Squire. Replaying history: Learning world history through playing civilization iii. *Unpublished doctoral dissertation Indiana University*, 56(January):435, 2004.

[59] Vinod Srinivasan, Karen Butler-Purry, and Susan Pedersen. Using video games to enhance learning in digital systems. In *Proceedings of the 2008*

Conference on Future Play: Research, Play, Share, Future Play '08, pages 196–199, New York, NY, USA, 2008. ACM.

[60] V.W. Strawderman. *A Description of Mathematics Anxiety Using an Integrative Model*. Georgia State University, 1985.

[61] Larian Studios. Monkey tales games: The castle of draconion, November 2011.

[62] Larian Studios. Monkey tales games: The princess of sundara, November 2011.

[63] Kaveri Subrahmanyam, Patricia Greenfield, Robert Kraut, and Elisheva Gross. The impact of computer use on childrenÊijs and adolescents' development. *Journal of Applied Developmental Psychology*, 22(1):7–30, 2001.

[64] How Teachers and Can Leverage. using the technology of today , in the classroom today. *The Education Arcade Retrieved*, 10(29):09, 2009.

[65] Juha-Pekka Tolvanen. Domain-specific modeling: no one size fits all. In *Proceedings of the 8th international conference on Model Driven Engineering Languages and Systems*, MoDELS'05, pages 279–279, Berlin, Heidelberg, 2005. Springer-Verlag.

[66] Juha-Pekka Tolvanen and Steven Kelly. Defining domain-specific modeling languages to automate product derivation: Collected experiences. In *Proceedings of the 9th International Conference on Software Product Lines, SPLC 2005*, pages 198–209. Springer, 2005.

[67] Javier Torrente, Pablo Moreno-Ger, and Baltasar Fernandez-Manjon. Learning models for the integration of adaptive educational games in virtual learning environments. In *Proceedings of the 3rd international conference on*

Technologies for E-Learning and Digital Entertainment, Edutainment '08, pages 463–474, Berlin, Heidelberg, 2008. Springer-Verlag.

[68] Mary Ulicsak and Martha Wright. Serious games in education. literature review, futurlab, 2010.

[69] Tamás Vajk, Róbert Kereskényi, Tihamér Levendovszky, and Ákos Lédeczi. Raising the abstraction of domain-specific model translator development. In *Proceedings of the 2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, ECBS '09, pages 31–37, Washington, DC, USA, 2009. IEEE Computer Society.

[70] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, June 2000.

[71] Richard Van Eck. Digital Game-Based learning: It's not just the digital natives who are restless... *EDUCAUSE Review*, 41(2), March 2006.

[72] Maria Virvou, George Katsionis, and Konstantinos Manos. Combining software games with education: Evaluation of its educational effectiveness. *Educational Technology & Society*, 8(2):54–65, 2005.

[73] Richard Wainess, Deirdre Kerr, and Alan Koenig. Improving the way we design games for learning by examining how popular video games teach. cresst report 798. *CRESST*, page 42, 2011.

[74] Atif Waraich. Using narrative as a motivating device to teach binary arithmetic and logic gates. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '04, pages 97–101, New York, NY, USA, 2004. ACM.

[75] Dennis R. Wixon. Are we having fun yet?: computers as entertainment objects. *Interactions*, 13(1):46–60, 2006.

[76] Raymond A. Zepp. Teachers2̆019 perceptions on the roles on educational technology. *Educational Technology & Society*, 8(2):102–106, 2005.