Design Smart Games with context, generate them with a Click, and revise them with a GUI

Vincenza Cofini[a], Fernando De La Prieta[b], Tania di Mascio[c], Rosella Gennari[d], Pierpaolo Vittorini[a]

[a] UnivAQ. U. of L'Aquila, 67100 Coppito, L'Aquila, IT, {vicenza.cofini, pierpaolo.vittorini@univaq.it}

[b] USAL. University of Salamanca. Department of Computer Science and Automation Control. Plaza de la Merced s/n, 37007, Salamanca (Spain), fer@usal.es

[c] DIEI, U. of l'Aquila, V.le Gronchi 30, 67100, L'Aquila, IT, tania.dimascio@univaq.it

[d] KRDB, Free U. of Bozen-Bolzano, Piazza Domenicani 3, 39100, Bolzano, IT, gennari@inf.unibz.it

KEYWORD

adaptive learning system
user-centered design
game design
game framework
temporal constraint problems
automated reasoning
natural language generation
natural language processing
pedagogy.

ABSTRACT

TERENCE is an FP7 ICT European project that is developing an adaptive learning system for supporting poor comprehenders and their educators. Its learning material are books of stories and games. The games are specialised into smart games, which stimulate inference-making for story comprehension, and relaxing games, which stimulate visual perception and not story comprehension. The paper focuses on smart games. It first describes the TERENCE system architecture, thus delves into the design of smart games starting from the requirements and their automated generation, by highlighting the role of the reasoning module therein. Finally, it outlines the manual revision of the generated smart games, and ends with short conclusions about the planned improvements on the automated generation process.

1 Introduction

Nowadays, circa 10% of young children are estimated to be poor text comprehenders. They are proficient in word decoding and other low-level cognitive skills, but they show problems in deep text comprehension. TERENCE[1] is a European ICT multidisciplinary project. The project is placed in the area of Technology Enhanced Learning (TEL) and its main objective is to develop the first adaptive learning systems (ALS) for improving the reading comprehension of 8–10 year old poor comprehenders, building upon effective pencil-and-paper reading strategies, and framing them into a playful and stimulating pedagogy-driven environment. Learners of the system are primary school poor comprehenders, hearing and deaf, older than 7. They are the main end users of the system. Secondary end users of the system are the learners' educators, and the experts sitting in the consortium, who design and develop the learning material or system.

The learning material of TERENCE is made of stories, collected into books, and games. Games are

[1] http://www.terenceprojecte.eu

specialised into *smart games*, for reasoning about stories, and into *relaxing games*, for motivating and relaxing the learners after playing with the cognitively demanding smart games. In a session, the learner is asked to first read a story and then to play the associated smart games concerning the events of the story; at the end, the learner can relax by playing relaxing games for that story.

Books, stories and games are written in the two languages of the project, Italian and English, and illustrated. The learning material was designed by following the user centered and evidence based design, see [Cofini,, V. *et al.*, 2012]. The models for the learning material and learners of the system are in [Alrifai, M., *et al.*, 2012a], and the first adaptation rules are in [Alrifai, M., *et al.*, 2012b], whereas [Alrifai, M., *et al.*, 2012c] explains how the models and learner model, in particular, stem from an extensive context of use and requirement analysis [Slegers, K. et Gennari, R., 2012] [Di Mascio, T. *et al.*, 2012a].

This paper focuses on smart games. The first goal of this paper is to explain how the smart games of TERENCE are designed on top of an extensive analysis of the context of use of the system. The second and main goal of the paper is to explain the automated procedure that enables the generation of the textual components of smart games. The third goal is to explain how we assessed what is working and what needs manual fixes in the automated generation procedure of textual components of smart games.

The structure of the paper is as follows. First, in the following section, we explain the overall system architecture so as to guide the reader through the relevant components of the TERENCE system for generating smart games, and their roles. Stronger with such background, then we can delve into the design and generation of textual components of smart games. As the design is rooted in the analysis of the context of use, the analysis of the context of use is recapped in Section 3. The design of the smart games from the context of use analysis starts with the presentation of the TERENCE game framework in Section 4. It then continues explaining the data structure of smart games for the generation process of textual components of smart games. The generation process itself is the focus of Section 5. Section 6 concludes the paper by explaining the manual interventions required by the generation process. Their impact on the improvements to the automated generation process is outlined in the conclusive section.

2 The System Architecture

The generation of smart games is a complex process and is divided into a series of steps, the majority of which are automated. The full process is shown in the next sections. In this process, many software components interact, as Figure 1 shows. Each of these components has a specific responsibility in the process. The communication among them is done through RESTful web services, following a SOA approach.

Such a design of the architectures makes possible the reutilisation of each individual component, e.g., by substituting the natural language module for a language with another for another language.

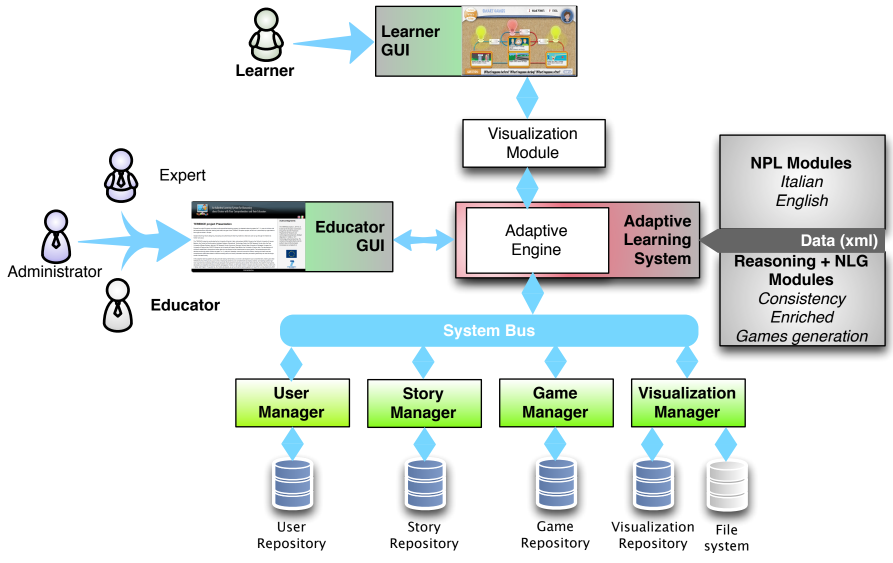


Fig. 1. System architecture Overview

The lowest level layer is the persistence layer whose storage engine is openRDF. This engine is based on ontologies that make possible to store structured information and to modify dynamically the structure of the data. Within this layer are located the different repositories of TERENCE (User, History, Games and Visualisation), each of them supported by a management component.

The highest level layer is the Graphical User Interface (GUI) which is divided itself into two main components.

1. A RIA (Rich Internet Application) web application developed with the Vaadin Framework (http://www.vaadincom). It is specifically designed for educators, experts and administrators, and allows them to manage all the information of the TERENCE system (books, games, avatars, etc.) as well as the tracking of the learner progression.
2. A web application for learners, specially designed for tablets and based on Flash. This application is supported in its tasks by the Visualisation module that preprocesses the data. And finally, as core of the system, there are three independent modules that communicate among them through web services.

The NPL (Natural Processing Language) modules, in Italian and in English the Reasoning module work together to annotate semantically relevant information in flat story texts. The Natural Language Generation (NLG) modules, in English and in Italian, and again the Reasoning Module work together to generate automatically textual components for smart games. Such generation process is described in details later on in this paper.

The Visualisation module and the Learner GUI assemble the textual components and the visual components for the entire Learner GUI, e.g., for stories, and display them in the Learner GUI. In the case of smart games, the Visualisation module takes as input the textual smart games of the Reasoning module and retrieves the visual components for the generation of visual smart games from the visualisation repository.

Finally, the ALS (*Adaptive Learning System*) module of the learning system, whose main internal component is the adaptive engine, is in charge of the selection of the learning resources per each learner according with the learner's interaction and progress with TERENCE.

3 Context of Use

For designing and evaluating the TERENCE system, we adopt the user centered design (UCD) methodology [Norman, D., 2012]. The analysis of the context of use is a mandatory first step in UCD, which means, in TERENCE, analysing and specifying the following in relation to the TERENCE users.

1. The characteristics of the TERENCE learners and educators.
2. The learners’ tasks, that is, the learners’ activities in relation to reading comprehension.

3. The environment constraints, divided into the physical environment constraints in which educators and learners read, and the satisfaction associated with it (e.g., school, house), the instructional environment in which educators and learners do their activities, and the analysis of devices (e.g., software) for such activities.

The TERENCE context of use is so articulated that its analysis required first a comprehensive and long preparatory study, and then field studies. The preparatory study involved ICT researchers, cognitive and educational psychologists of the consortium, and educational stakeholders. The field studies, performed in Italy and the in the UK from the beginning of 2011 to May of 2012, involved the experimenters, learners as users of the system, teachers and parents as users of the system, and teachers as domain experts. The adopted methods were:

- for the preparatory studies: brainstorming meetings, the study of the state of the art, and the study of the bureaucratic documentation;
- for the field studies: diaries, observations and contextual inquiries.

Field studies are standard in the UCD context, whereas preparatory studies are a need of the TERENCE project. In fact, the latter studies were supposed (and demonstrated to be) necessary for building the knowledge base of the consortium team, which is highly cross-disciplinary, and hence for gathering information relevant for the field studies, like the characteristics of the TERENCE learners known in the literature and the different administrative, legal and ethical issues in UK and Italy.

The preparatory studies dealt with the learners' characteristics, the reading comprehension task, and the organisational environment constraints. The field studies dealt with the learners' and the educators' characteristics, the reading comprehension tasks, and the physical environment constraints.

Thanks to the preparatory studies and the field studies we could define the user classes and the requirements for the TERENCE ALS system [Di Mascio, T., 2012a] [Di Mascio, T., 2012b]. In what follows, we only report the results of the overall studies performed for the context of use, divided into the analysis of the characteristics of users, tasks and environment constraints for the design of smart games.

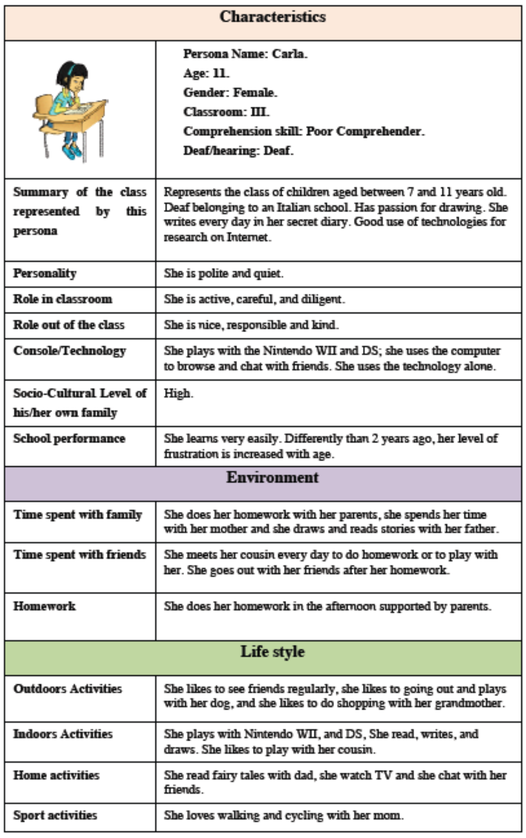
3.1 Characteristics of the users

The types of learners in TERENCE are deaf and hearing learners, distinguished according to their knowledge in relation to the specific learning goal at the start of the project. The TERENCE classes of users refine the types of users on the basis of the results of the analysis of data for the context of use. Such data have been gathered via a mix of expert-based method inquiries (e.g., interviews with primary school educators) and user-based method inquiries (e.g., field studies with primary school children by making them play). The learners involved were about 300 in Italy and about 300 in the UK; the educators involved were about 50 in Italy and about 30 in the UK.

Learners are grouped into 5 classes in Italy and 4 classes in UK, see [Di Mascio, T., 2012a] for details. The most significant features related to the characteristics of the user across classes are:

a. biographical information such as the level of reading comprehension (RC), the level of deafness, and the gender;
b. personality traits such as the management of frustration;
c. usage of technology, like the preference for certain types of avatars.

All the classes and the features used for deriving the TERENCE classes were then specified using personas, which are explained in details in [Di Mascio, T., 2012a] and outlined in [Alrifai, M., *et al*., 2012c]. A persona per user class was created and allowed us to share the information concerning the analysis among all the members of the TERENCE heterogeneous consortium, and pass on the relevant information to the designers for the definition of the use cases of the ALS. Figure 2 is an excerpt of the persona for the deaf female class, Carla. All the other personas are structured in the same manner.



Characteristics	
	Persona Name: Carla. **Age: 11.** **Gender: Female.** **Classroom: III.** **Comprehension skill: Poor Comprehender.** **Deaf/hearing: Deaf.**
Summary of the class represented by this persona	Represents the class of children aged between 7 and 11 years old. Deaf belonging to an Italian school. Has passion for drawing. She writes every day in her secret diary. Good use of technologies for research on Internet.
Personality	She is polite and quiet.
Role in classroom	She is active, careful, and diligent.
Role out of the class	She is nice, responsible and kind.
Console/Technology	She plays with the Nintendo WII and DS; she uses the computer to browse and chat with friends. She uses the technology alone.
Socio-Cultural Level of his/her own family	High.
School performance	She learns very easily. Differently than 2 years ago, her level of frustration is increased with age.
Environment	
Time spent with family	She does her homework with her parents, she spends her time with her mother and she draws and reads stories with her father.
Time spent with friends	She meets her cousin every day to do homework or to play with her. She goes out with her friends after her homework.
Homework	She does her homework in the afternoon supported by parents.
Life style	
Outdoors Activities	She likes to see friends regularly, she likes to going out and plays with her dog, and she likes to do shopping with her grandmother.
Indoors Activities	She plays with Nintendo WII, and DS, She read, writes, and draws. She likes to play with her cousin.
Home activities	She read fairy tales with dad, she watch TV and she chat with her friends.
Sport activities	She loves walking and cycling with her mom.

Fig. 2. An example of personas: Carla.

3.2 User Tasks

The evidence-based practice of the experts responsible for the pedagogical plan requires three main learning tasks in relation to the learning material of the system: (i) reading stories, (ii) playing for stimulating inference-making about stories, and (iii) relaxing activities for relaxing and motivating the learners.

In particular, (ii) became the source for the design of smart games, whereas (iii) became the source for the design of relaxing games.

In particular, the data for relaxing games are popular causal video-games, such as memo, which the TERENCE learners are likely to be familiar with. A casual game is a video-game meant for casual gamers who come across the game and can get into the gameplay almost immediately. This means that a causal game has usually simple rules that are easy to master, and usually it can be played everywhere, anytime and with any device. The data for smart games are mainly diverse reading interventions by pedagogy experts working as therapists with poor comprehenders, by cognitive psychologists or by educators. More precisely, the main data collected were:

a. paper-and-pencil inference-making question-answering interventions, with or without picture aids, by cognitive psychologists working on the diagnosis of poor comprehension;
b. paper-and-pencil puzzle-like games, much relying on visual stimuli, by therapy and pedagogy experts;
c. diverse interventions of educators, divided into interventions for the analysis of texts in class like question-answering, and interventions like drama exercises for stimulating the empathy of the learners with the characters of the story.

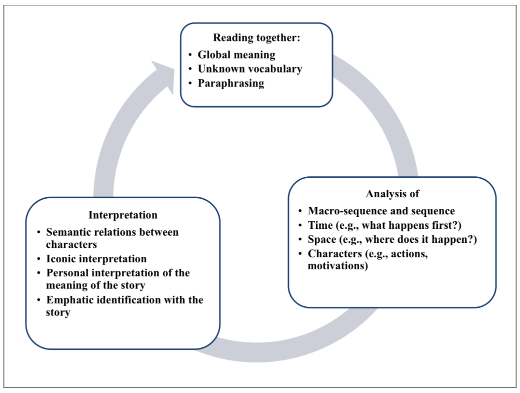

Reading together:
• Global meaning
• Unknown vocabulary
• Paraphrasing
Analysis of
• Macro-sequence and sequence
• Time (e.g., what happens first?)
• Space (e.g., where does it happen?)
• Characters (e.g., actions, motivations)
Interpretation
• Semantic relations between characters
• Iconic interpretation
• Personal interpretation of the meaning of the story
• Emphatic identification with the story

Fig. 3. The pedagogical hermeneutic cycle.

The interventions of the educators can be framed in the three stages of the hermeneutic cycle explained in [Valeriani, A., 1986] and outlined in Figure 3. In particular, the explanatory analysis stage can be broken down into the following reading interventions, done in class, mainly using question-answering and drawing:

1. the story is broken down into a sequence of episodes, if possible referring to the story grammar, that is, the story setting, the initiating episode, the culminating episode, the resolving episode, and the final episode;
2. finally, the time, the space and the characters of the story episodes are analysed together.

Constraints of the project triggered a first prioritisation of the requirements. This first sieve left out, for instance, drama exercises or other interventions meant at stimulating the empathy of the learners with the story characters. The remaining interventions refer to the explanatory analysis stage of the hermeneutic cycle, with visual aids. They were selected mainly for their expected efficacy for the pedagogy plan, according to the available empirical evidence: they should guide the child to better recall and correlate the information, acquired by reading the story, via adequate visual representations. More precisely, the TERENCE smart games should:

1. propose to reason about the *characters* and their participation in the stories;
2. other types of game should propose to reason about *time*, namely, temporal relations between events;
3. more demanding games should propose to reason about *causality* and, more precisely, causal-temporal relations between events.

3.3 Environment

Considering the environments in TERENCE means taking into consideration: the physical environment, the instructional environment, and devices. Among the environment constraints, those that mostly affect the design of the smart games are the organisational constraints set by the stimulation plan of TERENCE, which makes the smart games the main means for stimulating the learners' reasoning about stories. Environment constraints of TERENCE are all described in details in [Di Mascio, T. *et al*., 2012] and mainly derive from the suggestions given by teachers both in the first and in the second field studies as well as the expertise of the stimulation plan experts. Of relevance for this paper are the OC3 and OC4 constraints, reported as follows.

OC3. During the field studies, both schools' principals and experts asked us to stay in one classroom for no more than 45 minutes/1 hour, so to preserve the normal lesson's flow, and ensure a proper level of attention (that decreases after that period of time), therefore the duration of all the TERENCE smart games cannot last more than 45 minutes/1 hour.

OC4. School principals suggested to adequately weight the number of interventions, so to preserve the regularity of the standard school program. Thus, the intervention should be though as an external activity (as an extra-school lab) of varying difficulty.

4 Design of the TERENCE Framework

The effective interventions, relevant for the TERENCE smart games according to the context of use analysis, were hierarchically organised in levels according to their main pedagogical goal and the expected difficulty for the TERENCE classes of learners. The levels, from the easiest to the most difficult, are as follows:

- *characters*: games concerning characters, namely, who the agent of a story event is (who), what events a character in the story participates in (what);
- *time*: games for reasoning about temporal relations between events of the story, purely sequential (before-after) or not;
- *causality*: games concerning causal-temporal relations between events of the story, namely, the cause of a given event (cause), the effect (effect), or the cause-effect relations between two events (cause-effect).

According to the game design guidelines presented in [Alrifai, M. et Genari, R., 2012], the gameplay should detail the following data: the instructions and the overall goal of the game, the initial state of the game, the termination state, the legal actions of the players, and the maximal duration time per action, if foreseen. For specifying the gameplay of the TERENCE games we also analysed the organisational constraints resulting from the context of use analysis. Then we abstracted the common characteristics of the TERENCE smart games in the TERENCE game framework presented in Table 1 and described below. The framework serves to specify, in a structured manner, the above data for the gameplay of the TERENCE smart games, essentially, through a timed transition system, with states of the system, and transitions labelled by the player's actions and time constraints.

In the following, we first present the framework specialised for a specific level of games, namely, before-after time game concerning the sequencing of three events in time. Then, in the next section, we sketch how this framework is used for designing and populating the data structures of TERENCE before-after game instances.

Identifier	**Identifier of the game**
Goal	A textual message specific to the game: "You can win N points if you resolve this game. To solve the game, look at the central event in the circuit. Which of the below events comes before the central event? Which comes after?".
Choices	Available choices: events available for resolving the game
	Fixed choice: fixed event
Solutions	Which available event occurs before the fixed event, which comes after and which comes neither after nor before the fixed event
Feedback	Consistency feedback: a yes message if the learner choose the correct before and after solutions; a no message otherwise
	Explanatory feedback for wrong solutions
	Interaction feedback for how to interact with the game
Points	N points of the before-after game as a function of the performance of the learner in previous sessions with before-after games

Table 1. Before-After Game

Given its aim, the TERENCE framework is less general than other frameworks like [EMAPPS, 2012] and, clearly, less general purpose than game patterns like [Kelle, S. et al. 2011]. Therefore it better lends itself to the implementation of pedagogy-driven single-player casual and puzzle games, where the organisational constraints of the pedagogy plan set restraints on the gameplay. Table 1 presents the fields of the framework for before-after games that are relevant for this paper, e.g., we do not discuss the rules of the game that are instead outlined in [De la Prieta, F., *et al*., 2012]. The fields of the framework are self-explicative except for the fields labeled "Choices" and "Solutions" that deserve some comments. In who games, a fixed choice correspond to a story's event of which the learner has to find the agent, e.g., the subject; the available choices are then characters of the story; a correct solution is then a character that is an agent of the story's event, else it is a wrong solution. In all the other games, like in before-after games, a fixed choice is an event that occurs in the story; available choices are other events that, in case of before-after games, are correct solutions if they happen before or after the fixed event, else they are wrong solutions.

To better understand the intended semantics of such fields, see Figure 6, which is an instance of a visual before-after game. In this game, the learner is asked to focus his or her attention on the central event, which is both illustrated and described with a simple sentence; this is the fixed event of the before-after framework. Then the learner is asked to choose 2 out of the 3 events, depicted and described with a simple sentence, that are in the bottom part of the visual before-after game; these are the available choices in the before-after framework. One of the 2 events should be placed to the left of the fixed central event, if it happens before the fixed event in the story, and the other should be to the right of the fixed event, if it happens after the fixed event in the story. These are the correct "before" and "after" solutions in the before-after framework.

The data of before-after game instances are structured according to the before-after game framework, as explained in the following section.

5 From the TERENCE Framework to Smart Games, Automatically

One of the main technological advances of TERENCE is that the TERENCE system enables the generation of smart games starting from the TERENCE stories in an automated manner. To this end, all the smart games are similarly structured in XML and, independently of their level, share the same persistence schema. The data structures are designed on top of the TERENCE framework. Section 5.1 explains the generation process for populating the related smart game XML data structure, and Section 5.2 elaborates on the performances of the automated generation.

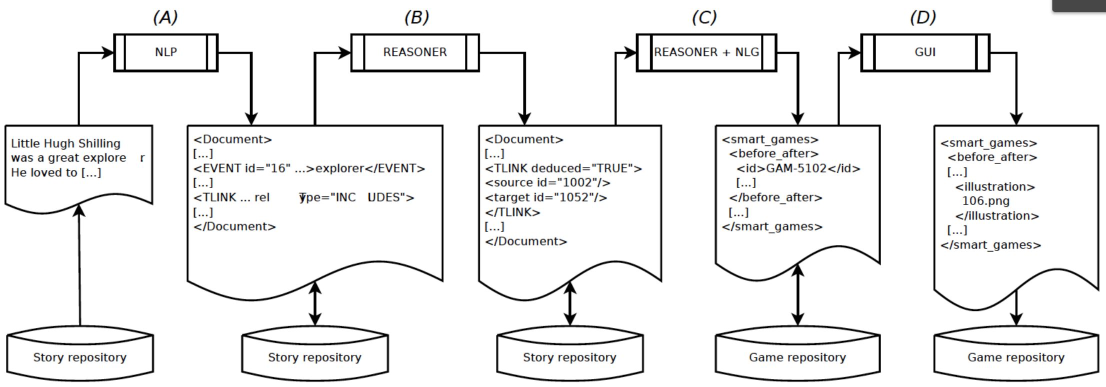
5.1 From the game framework to the data structure

Figure 4 shows an overview of the entire textual smart games generation process.

Phase A. Firstly, from a story text contained in the story repository, an NLP module generates a story annotated with a variant of the TimeML language that was extended in [Moens, S., 2012] with tags for information that is relevant for the TERENCE smart games, e.g., the ENTITY and CLINK tags, that aim, respectively, to represent the entity related to an event, and the causal-temporal relations between two events. The annotated story is then stored in the same repository.

Phase B. Then the Reasoner module maps the temporal relations into Allen-like temporal relations, giving them a semantics over the real line. Thanks to such a semantics, the Reasoner detects the eventual temporal inconsistencies in the temporal relations and, in case none is present, enriches the annotations by adding deduced temporal relations as further TLINK tags. This new consistent and enriched story is also stored in the story repository.

.



```
<Document>
[...]
<EVENT id="16" ...>explorer</EVENT>
[...]
<TLINK ... rel   Ϋpe="INC  UDES">
[...]
</Document>
```

```
<Document>
[...]
<TLINK deduced="TRUE">
<source id="1002"/>
<target id="1052"/>
</TLINK>
[...]
</Document>
```

```
<smart_games>
  <before_after>
    <id>GAM-5102</id>
    [...]
  </before_after>
  [...]
</smart_games>
```

```
<smart_games>
  <before_after>
  [...]
    <illustration>
     106.png
    </illustration>
  [...]
</smart_games>
```

Fig.4. Data and processes for the automated generation of smart games.

Phase C. Afterwards, starting from the enriched story, the Reasoner module and two NLG modules, one for Italian and the other for English, generate automatically instances of smart games that are stored in the game repository. In particular, the fixed event, choices and solutions (see Table 1) are produced by the Reasoner module via dedicated algorithms that query the enriched stories returned in Part B. Instead, human readable description of events and choices are populated by the NLG modules that generate who questions for the fixed choice of who games and simple sentences for the choices of all the other smart games. See [Gennari, R., 2012].

Phase D. Finally, a manual revision of the generated smart game instances takes place, where the related visuals (e.g. background illustrations, buttons) are also illustrated. All such components, textual and visual, are then assembled by the visualisation module and the learner GUI.

In the following, we delve into phase C of the process and in the work of the Reasoner module. Starting from a story s, annotated and then enriched as explained above, the smart games generation goes as in Algorithm 1

Algorithm 1 initially iterates among all events, tagged with the EVENT tag in story s. Iteratively, an event e is selected as the fixed event for the generation process. Then, the algorithm generates instances of smart games with e and other events (lines 1–3). For example, let us consider a time before-after game, shown in Figure 6. The fixed event is displayed as the central even in the figure. Then the algorithm, using specific heuristics, finds an event that happens before the fixed event, and one that happens after the fixed event, and a further event that does not happen before or after the fixed event in the story s or in the story s0.

Algorithm 1: generate smart game instances, with a fixed event, for a story s

```
Require: story s, number of events n, number of games k
1: foreach event e in s do
2: generate all types of games with e as fixed event
```

```
3: od
4: sort events
5: keep the games for the first n events
6: reduce the total number of games to k
7: generates the texts.
```

Algorithm 2 shows how all possible before-after games are generated for a fixed event e. In brief, the algorithm selects all couple of temporal links t1, t2, so that t1 has e as target and t2 has e as source, and produces a temporal game with:

- e as fixed event;
- the source event of t1 as the right before choice;
- the target event of t2 as the right after choice;
- a random event taken from another story as wrong choice.

Algorithm 2: generate a before-after game instance for the fixed event e of story s and not s0

```
Require: event e, story s, story s0 is different than s
1: foreach TLINK t1 in s, so that t1 has e as target do
2: foreach TLINK t2 in s, so that has e as source do
3: select a random event w from s0;
4: create a before-after game so that
    e is the fixed event
    the source of t1 is the correct before choice
    the target ef t2 is the correct after choice
    w is the wrong choice
5: end foreach
6:end foreach
```

After all possible games are generated, Algorithm 1 produces an ordered list of fixed events (line 4) according to the following heuristic. Given two fixed events e1 and e2, in order to decide if e1 > e2 , we compare the related number of generated games, weighting these according to their difficulty level, established by the stimulation plan. In other words, e1 > e2 if the number of causality games for e1 is higher than for e2. If equal, we compare the number of time games, and so on.

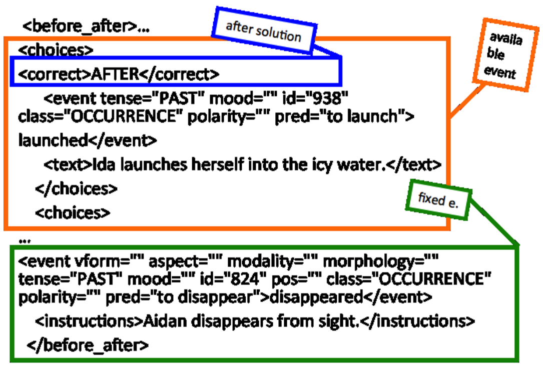
After the ordering, two types of filtering take place.

- The first keeps only the games for the first n fixed events (line 5) in the ordered list;
- The second filter is concerned with the total number of smart game instances per level, reduced to a fixed number k. For each game level, the algorithm selects k game instances with different reasoning complexity, e.g., before-after games with both "deduced" events, implicit in the text, as well as who-games and what-games with both "protagonist" and "secondary" characters as participants (line 6).

For further details on how games are removed from the list, please refer to [Gennari, R., 2012].

Finally, for all games, the instructions and texts of each choice are generated by using the NLG modules (line 7).

Figure 5 is a snapshot of the XML data of a before-after game instance, and is used to sketch the data structure of smart games generated as above, that comply with the games framework described in Section 4.


after solution
availa ble event
fixed e.

```
<before_after>...
<choices>
<correct>AFTER</correct>
    <event tense="PAST" mood="" id="938"
class="OCCURRENCE" polarity="" pred="to launch">
launched</event>
    <text>Ida launches herself into the icy water.</text>
   </choices>
   <choices>
...
<event vform="" aspect="" modality="" morphology=""
tense="PAST" mood="" id="824" pos="" class="OCCURRENCE"
polarity="" pred="to disappear">disappeared</event>
   <instructions>Aidan disappears from sight.</instructions>
  </before_after>
```

Fig. 5. A snapshot of the data structure of a before-after game instance. In green, the fixed event and the field of instructions. In orange, an available event that is also a correct after solution.

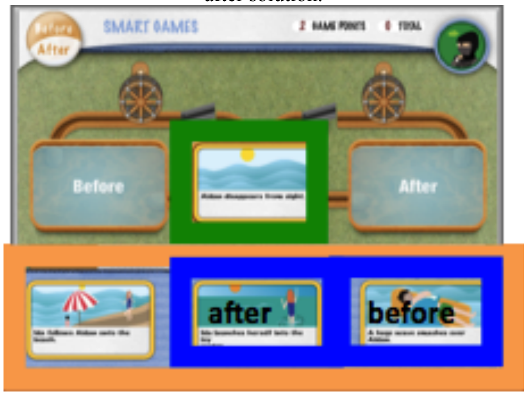


Fig. 6. A before-after game instance. In green, the fixed event. In orange, all the available events. In blue, the correct before and after solutions.

As can be noticed, the portion on XML code enclosed in the green box contains:

- the fixed event (selected as Algorithm 1, in the extended TimeML language [Moens, S., 2012]);

- the instructions acting as description of the fixed event (generated by the NLG module).

Furthermore, the piece of XML code in the orange box refers to a choice, and contains:

- the related event (selected as of Algorithm 2);
- its correctness as "AFTER" event (in blue),
- the text acting as explanation of the event (generated by the NLG module).

Figure 6 illustrates how the data structure of a before-after game instance is used in the corresponding visual smart game instance, when illustrations are paired to all textual events.

5.2 Performances

Table 2 summarises the average performances in generating the smart games, divided by activity and language.

Differences are remarkable with respect to the different activities.

The generation of sentences is faster in English language than for Italian language. However, the comparison is almost uninteresting, since we use a web service in the Italian case, and a local library for the English case.

Language	Activity	Time (msec)
IT	Game generation	2.09
	Filtering	0.53
	Simple sentence generation	20981.1
EN	Game generation	24.08
	Filtering	26.5
	Simple sentence generation	297.22

Table 2. Performances

More interesting is the comparison of the other two activities, i.e., smart game generation by the Reasoning module and filtering, that are faster for Italian language. The reason for these differences is that the number of generated games, satisfying the constraints of the Reasoner's algorithms, is higher for English that for Italian (86 vs. 62 on average). The explanation for this is summarised in Table 3, which reports the number of ENTITIES, EVENTS and T-LINKS, deduced or not, and if the difference is statistically significant or not. As can be noticed, the Italian NLP module performs better for entity recognition, while the English service recognises more EVENTS and TLINKs, these being increased to 35% more after deduction.

Type	N		S.D.
	EN	IT	
ENTITIES	14.1	28.2	*
EVENTS	67.6	56.6	*
T-LINKS	66.6	53.5	*
Deduced T-LINKS	2274.5	335.6	*

Table 3. Summary of annotated files

6 Manual Revision

The automated generation of games explained in Section 5 may be affected by errors that are introduced during the annotation, e.g.,

- a TLINK between two events is uncorrectly recognised, and thus the automatically generated BEFORE-AFTER game contains by mistake a wrong event as correct solution;
- co-references are not properly resolved and the actor of an event is wrongly recognised, thus the corresponding WHO game asking for the actor of the event contains a wrong character as correct solution.

Furthermore, some of the heuristics for selecting the wrong choices can select events that are not plausible, the learner could easily find them as wrong solutions, thus affecting the overall quality of the game. Finally, the sentences generated by the NLG modules are not always grammatically correct and thus they must be revised both in their grammar and to improve their clarity.

Therefore, in order to have the games tidied up for being played by learners, a manual revision is mandatory. Such a manual revision was conducted by a team of trained operators. The revision was divided into 3 steps:

1. Formal review: Correction of grammatical and syntactic errors in the text; correction of punctuation; check of the verb (present tense, active form); correction of referential expressions, e.g., substitute "Ernesta" for "the little girl scout"; check of sentence length and structure; check of game identification number (ID).
2. Substantial revision: Correction of the automatically generated questions with the aim to identify unambiguously the event in the text story; correc-

tion of the solutions keeping fixed the main event; choice of new fixed events for solutions

3. Construction of cause/effect games: Text proposal; check out of proposals; loading of games.

Game	**n**	**%**
BEFORE/AFTER	70	31.96
BEFORE/WHILE	33	15.07
BEFORE/WHILE/AFTER	29	13.24
WHAT	32	14.61
WHILE/AFTER	33	15.07
WHO	22	10.05
Total	219	100.00

Table 4. Details about the revised games

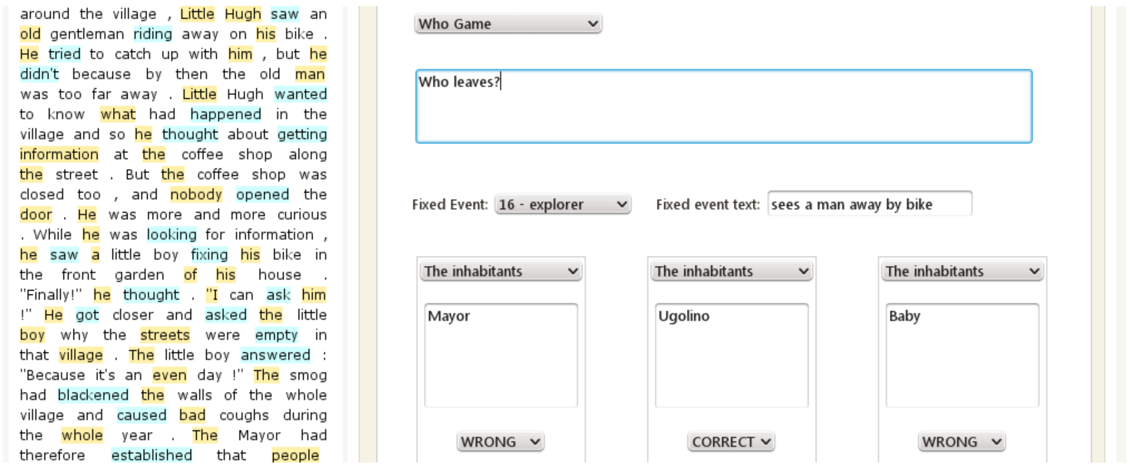
Each operator studied the text of the story and reviewed all the games associated with it. For the creation of cause-effect game, the work was submitted to internal validation by a working group leader and an external blinded validation with the help of an expert who knows neither TERENCE nor the revision work of smart games.

In summary, 219 games have been revised for 22 stories, according to the distribution in Table 4.

Each operator had the task of filling in a diary in excel format composed of 33 fields where they had to record all changes made game by game and depending on the levels of games.

A quantitative analysis of the revision shows a good quality of the automated game generation: in only 5 cases it was necessary to change the fixed event that was automatically generated.

Most of the wrong solutions were changed (70 changes). The main effort has been in the revision of the text automatically generated for the games by the NLG modules, which were in alpha version at the time in which the revision process took place: the simple sentences generated were incomplete or inconsistent with the criteria set for the revision of the first version of the story, so it was necessary to continue to work on accents, the verb tenses and sentence length.


around the village , Little Hugh saw an old gentleman riding away on his bike . He tried to catch up with him , but he didn't because by then the old man was too far away . Little Hugh wanted to know what had happened in the village and so he thought about getting information at the coffee shop along the street . But the coffee shop was closed too , and nobody opened the door . He was more and more curious . While he was looking for information , he saw a little boy fixing his bike in the front garden of his house . "Finally!" he thought . "I can ask him !" He got closer and asked the little boy why the streets were empty in that village . The little boy answered : "Because it's an even day !" The smog had blackened the walls of the whole village and caused bad coughs during the whole year . The Mayor had therefore established that people

Who Game

Who leaves?

Fixed Event: 16 - explorer

Fixed event text: sees a man away by bike

The inhabitants	The inhabitants	The inhabitants
Mayor	Ugolino	Baby
WRONG	CORRECT	WRONG

Fig.7: Part of the expert GUI supporting the games manual revision process

Figure 7 shows a portion of the expert GUI that supports the manual revision of the textual smart games. The expert can read the story and revise the textual components of the associated smart games. The expert decides whether it is necessary to correct the text of who-questions or sentences that describe events of the story, or the solutions. In the example, the expert can change to correct the question “Who leaves?” because it is ambiguous and can opt for the following question “Who leaves right after packing?”

The expert can decide for a new "who game" and choose, for example the following question “Who is curious?”, with the help of the interface. Then the expert works on the solutions. For all kind of solutions (correct, wrong, wrong) he or she can correct the text or search a new proposal by menu.

In this example, a typical “who game”, solutions have been changed according to the following criteria:

- always work on subjects (preferably looking for the proper names), distributed over the text with a certain distance,
- with particular attention to gender so as not to guide the reader to the correct choice.

7 Conclusions

TERENCE is a complexy adaptive learning system with several software components that allows its learners to read through books of stories and reason about them with smart games, as prescribed by the TERENCE pedagogy plan. The design of the system is UCD based, starting from the context of use analysis moving to the design of the system, which is then evaluated and iteratively refined. This paper focused on the design of the TERENCE smart games starting from the analysis of the context of use, via the TERENCE game framework. It then delves into the description of how the framework was used to design the data structures of the TERENCE smart games, and how the data structures are populated by the TERENCE system starting from the TERENCE stories, automatically. Such an automated process requires manual revisions, which are explained in the end of this paper.

To the best of our knowledge, it is the first time that games are design and generated in such a manner, through a collaborative work that sees together NLP, automated reasoning and NLG technologies. The manual revision process, moreover, helped to highlight the areas that require improvements, and how these can be carried out. For instance, the revision process purport that the heuristics chosen for filtering game events and selecting the central fixed event seem to work well. The generation process of textual components requires optimisation, and is the focus of the work for the last year of the TERENCE project.

Acknowledgment

This work was supported by the TERENCE project, funded by the EC through FP7 for RTD, ICT-2009.4.2. The contents of the article reflects only the authors' view and the EC is not liable for it. Gennari work was also funded through the CRESCO and DARE projects, financed by LUB and the Province of Bozen-Bolzano. The authors thank MOME for producing the images of events in snapshots of games. Then the authors wish to thank all the school stakeholders, in particular, Saverio Baldasciono, masters, teachers, children and their parents from Italy and UK, experts and volunteers that worked on this project across Europe and USA, in particular, Marc Marschark whose insight, support and contributions go well beyond what can written in a few lines. As usual, we thank our children for being the primary inspiration for our work and for putting up with our absences.

4 References

[Alrifai, M., *et al*., 2012a]	Alrifai, M., Gennari, R., Tifrea, O. and Vittorini, P. *The Domain and User Models of the TERENCE Adaptive Learning System*. In: International Workshop on evidenced-based Technology Enhanced Learning. AISC, Vol. 152. Springer. 2012. pp. 83-90
[Alrifai, M., *et al*., 2012b]	Alrifai, M., Gennari, R. and Vittorini, P.: *Adapting with Evidence: the Adaptive Model and the Stimulation Plan of TERENCE*. In: International Workshop on evidenced-based Technology Enhanced Learning. AISC, Vol. 152. Springer. 2012. Pp. 75-82
[Alrifai, M., *et al*., 2012c]	Alrifai, M., De la Prieta, F., Di Mascio, T., Gennari, R., Melonio, A. and Vittorini, P.: *The Learners' User Classes in the TERENCE Adaptive Learning System*. In: IEEE 12th International Conference on Advanced Learning Technologies (ICALT), 2012. IEEE Press. 2012. pp. 572-576
[Alrifai, M., *et* Genari, R., 2012]	Alrifai, M. and Gennari, R.: *Game Design*. Deliverable 2.3. Technical Report. TERENCE project (2012)
[Cofini,, V. *et al*., 2012]	Cofini, V., Di Giacomo, D., Di Mascio, T., Necozione, S. and Vittorini, P.: *Evaluation plan of terence: when the user-centred design meets the evidence-based approach*. In: International Workshop on evidenced-based Technology Enhanced Learning. AISC, Vol. 152. Springer. 2012. pp. 11-18
[De la Prieta, F., *et al*., 2012]	De la Prieta, F., Di Mascio, T., Gennari, R., Marenzi, I. and Vittorini, P.: *Playing for Improving the Reading Comprehension Skills of Primary School Poor Comprehenders*. In: Proceedings of the 1st International Workshop on Pedagogically-driven Serious Games. CEUR-WS. Vol-898. 2012. Pp. 41-50
[Di Mascio, T. *et al*., 2012]	Di Mascio, T., Gennari, R., Melonio, A. and Vittorini, P.: *The user classes building process in a tel project*. In: International Workshop on evidenced-based Technology Enhanced Learning. AISC, Vol. 152. Springer. 2012. pp. 107-114
[Di Mascio, T., 2012a]	Di Mascio, T.: *First User Classification, User Identification, User Needs, and Usability goals*, Deliverable D1.2.1. Tech. rep., TERENCE project (2012)
[Di Mascio, T., 2012b]	Di Mascio, T.: *Revised user classification, user identification, user needs and usability goals*, Deliverable D1.2.2. Tech. rep., TERENCE project (2012)
[EMAPPS, 2012]	EMAPPS consortium: *EMAPPS Game Framework*. Retrieved January 2012 from http://emapps.info/eng/Games-Toolkit/Teachers-Toolkit/Games-reation/Framework-for-Game-Design
[Gennari, R., 2012]	Gennari, R.: *Second release of Automated Reasoning Module*. Deliverable D4.2.2: Tech. Rep. D4.2, TERENCE project. 2012
[Kelle, S. et al. 2011]	Kelle, S., Klemke, R. and Specht, M.: *Design Patterns for Learning Games*. Journal of Technology Enhanced Learning. 2011. 555–569
[Moens, S., 2012]	Moens, S.: *State of the Art and Design of Novel Annotation Languages and Technologies*. Deliverable D3.1. Technical Report, TERENCE project. 2012
[Norman, D., 2012]	Norman, D.: *The design of everyday things*. Doubleday, NewYork (1998)
[Slegers, K. et Gennari, R., 2012]	Slegers, K. and Gennari, R.: Deliverable 1.1: *State of the Art of Methods for the User Analysis and Description of Context of Use*. Tech. Rep. D1.1, TERENCE project. (2012)
[Valeriani, A., 1986]	Valeriani, A.: *Ermeneutica retorica ed estetica nell'insegnamento verso l'oriente del testo*. Andromeda (1986)