UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

UPCommons

Portal del coneixement obert de la UPC

http://upcommons.upc.edu/e-prints

Aquesta és una còpia de la versió *author's final draft* d'un article publicat a la revista *Computers in human behavior.*

URL d'aquest document a UPCommons E-prints:

https://upcommons.upc.edu/handle/2117/130803

Article publicat / *Published paper*:

Amato, F.; Moscato, F.; Xhafa, F. Generation of game contents by social media analysis and MAS planning. "Computers in human behavior", 2019. DOI: 10.1016/j.chb.2019.02.030

© 2019. Aquesta versió està disponible sota la llicència CC-BY-NCND 3.0 http://creativecommons.org/licenses/by-nc-nd/3.0/es/

*Title Page with Author details

Generation of Game Contents by Social Media Analysis and MAS Planning

Flora Amato[1]

DIETI, Univ. of Naples Federico II, ITALY

Francesco Moscato[2]

DiSciPol, University of Campania Luigi Vanvitelli, ITALY

Fatos Xhafa[3,*]

Universitat Politècnica de Catalunya, Department of Computer Science, Barcelona, Spain

---

[*]Corresponding author
[1]email: flora.amato@unina.it
[2]email: francesco.moscato@unicampania.it
[3]email: fatos@cs.upc.edu

Preprint submitted to Journal of LaTeX Templates

Generation of Game contents by Social Media Analysis and MAS Planning

Abstract

In the age of pervasive computing and social networks, it has become commonplace to retrieve opinions about digital contents in games. In the case of multiplayer, *open world* gaming, in fact even in "old-school" single players games, it is evident the need for adding new features in a game depending on users comments and needs. However this is a challenging task that usually requires considerable design and programming efforts, and more and more patches to games, with the inevitable consequence of loosing interest in the game by players over years. This is particularly a hard problem for all games that do not intend to be designed as *interactive novels*. Process Content Generation of new contents could be a solution to this problem, but usually such techniques are used to design new maps or graphical contents. Here propose a novel PCG technique able to introduce new contents in games by means of new story-lines and quests. We introduce new intelligent agents and events in the world: their attitudes and behaviors will promote new actions in the game, leading to the involvement of players in new gaming content. The whole methodology is driven by Social Media Analysis contents about the game, and by the use of formal planning techniques based on Multi-Agents models.

Keywords: Procedural Content Generation, Formal Planning, Social Media Analysis, Multi-Agent Systems

1. Introduction

In the past few years, the huge amount of information from social networks and Internet of Things , as well as the increasing interest in Big Data analy-

sis, have promoted the research of intelligent systems able to understand and forecast human preferences, attitudes and behaviours.

The number of people playing games is growing day by day. Types of games include of course casual computer games like Candy Crash Saga or Clash of Clans, but also more complex (non casual) computer games like Overwatch, League of Legends, Call of Duty, The Sims or World of Warcraft. In addition, last years saw a substantial increase of interest even in tabletop games.

The Entertainment Software Association (ESA)[1] states that 63% of U.S. households are home to at least one person who plays video games regularly (3 hours or more per week).

It should be noted that, for all games that are not in the category of Role Play Game, in particular for Massive Multiplayer Games, one of the great issue they face is that static contents bore players. Even if the game is a *interactive novel* the repetitiveness of game mechanics lead gamers to abandon next episodes of the game. Sometimes players leave games even when they have not played all game contents: this often happens in MMORPG (Massively Multiplayer Online Role-Playing Game) when players have paid their access to the game and leave it before the end of their subscription.

Renewing of game content and mechanics is an important factor in keeping players engaged in games. Obviously, manual production of game content is generally really expensive[2][1], requiring a considerable workforce and costs. On the other hand, several techniques already exist to dynamically create game content through Procedural Content Generation for Games (PCG). Typically, PCG techniques create specific content instance from a given set of information; the generation process is often partially random. In addition, PCG offers the ability of generating elements *on the fly* working similarly to compression algorithms, expanding new contents only when needed.

Virtually, PCG addresses many kind of contents, from map and level com-

[1] `http://essentialfacts.theesa.com/`
[2] `https://mmos.com/editorials/most-expensive-mmorpgs-ever-developed`

ponents like textures, agents behaviours, storytelling elements etc. (for surveys see [2, 3]). Basically, existing taxonomies address both the *content* to create and the *way* it is created. PCG is becoming a hot topic in game development that several works in literature focused on the definition of effective Design Patterns to easy the development of PCG techniques[4, 5]. However, despite the plethora of PCG techniques developed over the last years, their application in commercial products is very rare. In addition, most of the efforts in PCG are related to the generation of low level game content like texture, sounds and game objects[6], or to maps, levels and ecosystems[7]. Only few works exploit the application of PCG to narrative and storytelling contents. In fact, they mostly focus on *narrative*[8, 9, 10] dimension and they are not usually applied to MMO (Masisvely Multiplayer Online Game).

This work proposes a methodology (**PAGE**: Pcg-based on Agents for Generation of new Elements) based on formal models for PCG of (secondary) storylines of games, to generate content for quests and story of MMO-RPGs. Here agents in the game act as pro-active elements that involve both players and other artificial agents in the game (as well as game resources) in the creation of new quests and storylines in the game. The approach is based of formal, multi-agent based, planning. The planning strategy does not use classical inference-based approaches, but exploit model checking and refinement in order to improve performances and to face with state explosion problems related to classical search-based strategies[2].

We aim to prove how formal methods and model checking can be used in order to solve planning problem and to provide a new way to introduce *intelligent agents*, able to produce new content inside games. We show how this approach adds more features to classical artificial intelligence techniques used in games, first of all the ability of producing fast results when planning goals are not reachable in the current environment.

The novelty of our proposal relies at generating new contents where both artificial agents and players works all together in the creation of new storylines. They compete for or act as resources and eventually their competition affects

other storylines preventing their completions. PCG Planning start from abstract plans, and it produces actual plans by refinement. The creation of new storylines depends on external events occurring when analysis of social networks and web resources identify conditions like complains from players, a decrease in the interests of some (or all) parts of the game etc. Finally refined plans elements depend on a knowledge base organized as an OWL Ontology, where fast reasoning activities allow for the choice of elements to include in the storyline (like motivation, the creation of new resources in the game etc.).

The rest of the paper is structured as follows. Some related works are overviewed in Section 2. Section 3 presents a methodology to retrieve sentiments about game contents from social media. We present the architecture of PAGE in Section 4 where we show how results from social media analyses can be used within a multi-agent based framework in order to generate new content in games. Then we follow in Section 5 with planning with multi-agents and its phases. There we discuss the model we use to translate generation of new content in a model checking problem. We finish the paper in Section 7 with some conclusions and directions for future works.

2. Related Works

The field of Procedural Content Generation in games is very wide and include several aspects: from map and level, to graphical elements generation; from the adaptivity of levels difficulties to users, to on-demand content generation in order to save space [11, 12, 13].

Our approach focuses on generation of quests and game elements in the storyline, thanks to the activity of artificial intelligence that act like pro-active elements in the game.

A tentative to approach the problem by using Machine Learning is in [14]. Authors aims at overcoming all search-based issues, their result is that Machine Learning is good to generate functional game contents, like game maps, and platform levels. In particular, it is clear that with this approach it is difficult to

generate complex organized storylines.

Another approach that uses pattern and Objectives to introduce new contents in games is discussed in [15], but again the objective-based approach (similar to the Goal-Based proposed in this work) was able to generate only level contents.

The work of Togelius et al.[2] contains a wide survey on search based techniques to procedural content generation. In short, they have been applied only to generation of levels and contents in platform and action games, where the "living" presence of intelligent elements that creates contents is not needed.

At the best of our knowledge the first attempt to create an *experience driven* generation of contents in games, which exploit data on external (web) sources, is in [16], but it lacks of all the benefits of sentiment analysis and of formal planning. Hence, our approach introduces innovative aspects as it takes information on what to generate from the semantic analysis of social media, and it generates contents by using a formal planning approach, based on multi-agent systems, model checking counterexamples and classical state-based search.

3. Social Media Analysis

We define a methodology to retrieve sentiments about game contents from social media. Data are collected automatically from existent media, without direct intervention of users. We start with a set of known models of gamers behaviours and then we analyse textual information from social media (tags, comments etc.) to measure how much a gaming community is deviating from required behaviour. Hence we classify contents from social media in order to :

- collect *bad* sentiments about the game
- select best game contents for users.

Social analysis from both web and game resources helps to enable the generation of new contents in the game. As previously discussed, pro-active agents enact

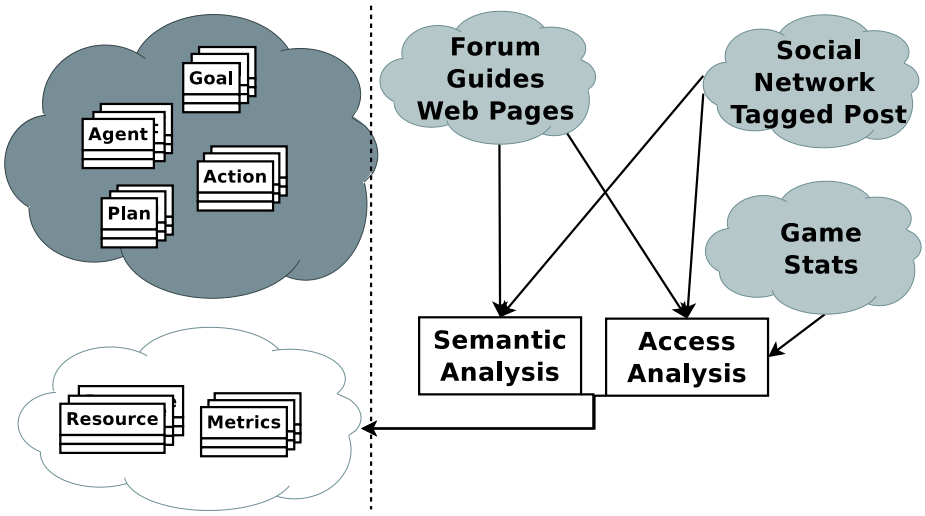

Figure 1: Analysis and generation of new contents

planning actions when something in their beliefs changes and they evaluate the possibility of improving their attitudes and desires.

Fig.1 resumes the analysis process.

The information retrieved for the analysis is fetched from web contents and from game data. The analysis is eased by the highly structured contents of web resources. We consider three kinds of elements: web forum, tagged posts from social networks and web guides. Forum topics and tags, as well as guide titles helps in identifying the topic we are analysing such as when players are talking about the whole mechanics, a particular game zone or agent, a static or dynamic storyline etc. Then, we perform simple sentiment analyses based on posts comments and access rate to resources and game elements like zones, agents and resources. A decrease in the access to some web or game elements, as well as bad comments on them, mark given topic, agent, resource or storyline as obsolete. This modifies metrics related to agents involved in the obsolete storyline and this yields to the creation of new storylines in next planning activities. In addition we force generation of new storylines in the game by updating beliefs of some agents in the game, by creating new resources and eventually by introducing new pro-active agents. The choice of events and resources generating

new contents depends on a domain ontology and by reasoning, (more can be found in [17]).

Moreover, depending on social media content and type, and on frequencies of interventions we classify users into different categories:

- Expert game fan: this category includes gamers that are expert of the game mechanics, that leave lot of contents in social media, mainly regarding game mechanics (like quests or massive multiplayer strategies, farming, acquiring legendary objects and so on)
- Game Content Fan: here we have gamers that discuss about game story lines in the game. These are characterized by high frequency mentioning of named characters, zones, as well as main story lines names.
- Normal Players: these are the users that have not so high frequencies of intervention in a given category to be placed inside it.
- Need an Help gamer: these users mainly requests for "how-to"s to complete quests, gaining objects etc. They are also frequent users of guides on forums and usually, in these kind of media, they *thanks* authors for their works.
- Haters: these users mainly have adverse sentiments with game contents and mechanics. The main problem of these category is to understand if users have ever been haters or if they were other kind of users before becoming haters.

Notice that this classification follows structures of game forums according on topics, which we can analyse in order to facilitate our understanding and processing. Of course, other social medias can be considered, in particular Twitter posts on tagged arguments about the game.

We also analyse frequencies of comments of users in different categories and, by sentiment analysis, we understand the topic and if they are positive (or neutral) or negative comments.

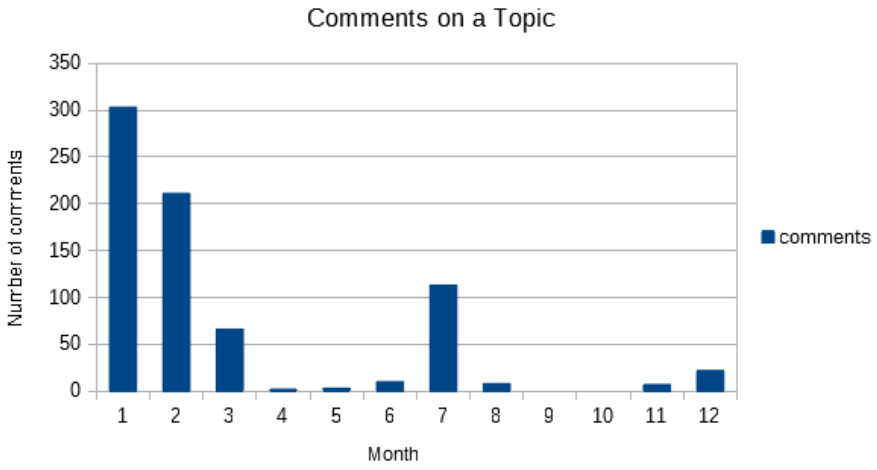

Figure 2: Comments on a topic in a month

With respect to one given topic (e.g. a quest, a raid etc.) we usually find is the following behaviour while taking into account only the number of comments, depicted in Fig.2

If each bar contains the number of comments reported on a topic on a given month, we usually have an high number of comments in the first months, then the interest falls, except for some months where new users joins the game or developers introduces new patches. Obviously, the third month in this example should be used as an alarm condition on the topic: if this was a "quest" topic, and majority of topics have reached the same "falling" behaviour, it is time to develop new content. In addition, we analyse sentiment about mechanics and quest contents, trying to understand the best quest elements and the best mechanics for different kind of users. This is done by analysing tags and textual information of comments user by user, evaluating the most common terms about game elements inside social media. In particular, for expert game fan, we retrieve the best and worst mechanics in the game (e.g. the majority of gamers would like multiplayer raids and player vs. player etc.) and for game content fans, we retrieve the elements in a storyline that are preferred by users

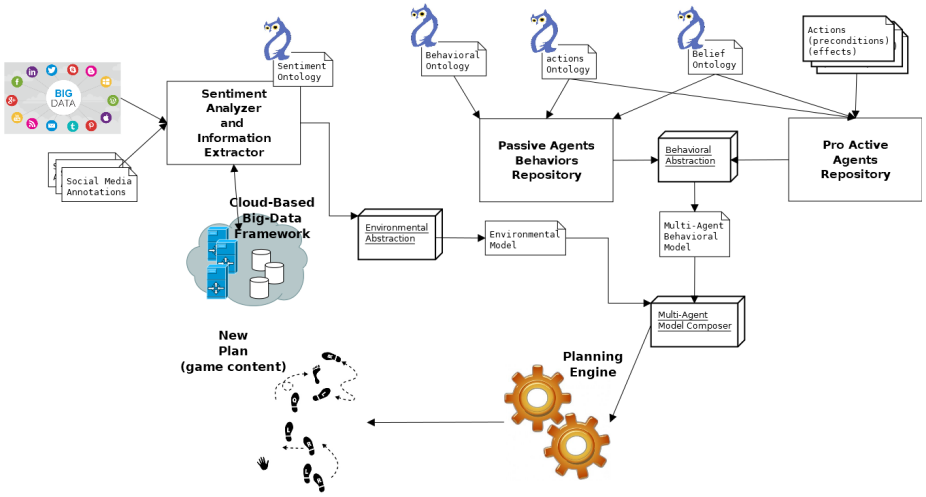

Figure 3: PAGE Architecture

(e.g. investigative vs. kill and loot quests, quests that are related to the main story, contents that explain something new on the virtual world etc.). Obviously, haters that were previously fans have a higher impact on determining bad content.

In this way we can determine *when* and *what* to introduce in the game to renew its contents. The next step is to provide the proper events to intelligent agents in the game, so that they can eventually produce new contents by acting as "normal" players.

4. System Architecture

The PAGE architecture aims at improving the state of the art in identifying sentiments about good and bad contents in game, as well as to generate new content. Generation is possible thanks to an innovative Procedural Content Generation technique which is based on formal planning of multi-agent systems. The overall architecture is in Fig.3.

We first provide a tool for analysing and storing data from social media. We focus here on textual data, but any other kind of media can be stored. The

platform is based on Big Data, Cloud-based components to store high volumes of data (for more details the reader is referred to [18].

The *Sentiment Analyzer and Information Extractor* analyses textual information from social media (post and comments contents, tags, emoticons etc.) in order to cope each element with an abstract sentiment, which in turn is defined in the *Sentiment Ontology*. Here we characterise in an abstract way main users sentiments and behaviours when dealing with social media. Eventually, social media are previously annotated by the elements of the same ontology in order to easy the analysis process. As it will discussed in Sec.3, we identify and extract from this analysis the elements that users prefer to insert in the game: a new quest, a new object, a new investigation etc. Such elements are defined as abstract elements to be introduced in the game environment (Environmental Abstraction and Model) This information will help in create elements that will trigger pro-active actions of intelligent agents in the game.

In addition, as we will explain in Sec.5, intelligent agents in the game are classified as passive and pro-active. All of them are able to perform actions, but passive agents await orders by pro-active agents to implement an orchestrated plan. Actions that agents can perform (with prerequisites and effects), as well as their behavioural patterns and the description of the state of the environment the agents sense (called beliefs), are modelled in an abstract form by three ontologies: the behavioural, the actions and the belief ontology. Both passive and pro-active agents repositories share this information. Furthermore, pro-active agents have the descriptions of the actions they can enact deliberately. Agents behaviours are abstracted in a MultiAgent Model. This model is then associated to the required environmental requests in order to produce a composite MAS for planning. The output of Planning is a series of steps orchestrated by one intelligent agent in the game that will generate new contents.

Let us consider this brief example. The sentiment analyser understand that new elements in the game are needed and that the most interesting for users is the retrieval of a new legendary object. Assume that in the belief ontology we find that an object can be created or retrieved. Since the object is not in the

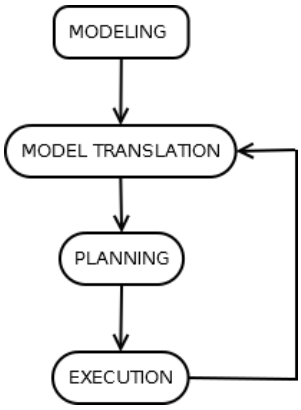

Figure 4: Methodology phases

game, it must be created. Consider also that there is a King in the game that knows (beliefs) how to create a new object. He needs a blacksmith with a sacred forge, and someone that can get prime materials from somewhere. In order to spread the voice, he uses chatting agents that are able to spread rumours where they are, with their action: *chat(rumour)*. Then, gathering of materials will be done by players. The only thing to do is the creation of the new material in the world, and the creation of some kind of guardian. The sequences of plans in terms of actions consist the new storyline. Notice that other behaviours may include resource gathering to declare a war, or something else, which radically modify the world environment.

5. Planning with Multi-Agents

The Framework that enacts planning in various phases is summarized in Fig.4

In the *Modelling* phase, we provide a representation of systems that is compliant with the meta-model described in previous sections.

The environment ($World$) is modeled in terms of Beliefs defined by means of $\langle n, d, v \rangle$ triples.

In addition, Agents are modeled in terms of Beliefs and Actions. Actions require the definition of $(name, Precondition, Postconditions)$ triples, but we

introduce in this phase an extension of the model since we need some additional information during planning. Hence we extend Actions definition to the quadruple:

$$(name, Precondition, Postconditions, VarInfo)$$

where $VarInfo$ set is necessary to specify variable domains, quantification(universal or existential), and eventually other properties.

For every modelled agent it is necessary to specify:

- The *type* of the agent:every agent can be modelled as a *Resource*, the simplest agent type composed only by a set of beliefs, a *Resource Agent* described by its set of Beliefs, Actions and a single Goal, or a *Proactive Agent*, defined by its Beliefs, Actions and a set of Goals with different priorities.

- *Beliefs*: they represent the vision of the environment of the agent and as for the world facts, a belief is a triple *name,domain,value*

- *Actions*: they represent what the agent can do and how it can modify the state of the environment. The formalism defined for the actions modelling is an evolution of the STRIPS that offers a more expressive power; every action is a triple *Pre,Post,Vars*, where Pre is the set of *Preconditions* that express a conjunction of logical formulas defining the values that the agent beliefs must assume to execute the action; the Postconditions define the values of the agent beliefs and of the world facts after the execution of the action; if in the preconditions or postconditions variables are present, in the Vars set is necessary to specify their domain, quantification(universal or existential),and optionally specific properties.

- *Goals*: a set of logical formulas defining the beliefs the agent has to achieve(every goal is modelled in the same way as a precondition).

During *Model Translation* and *Planning Phase*, a *Planner Engine* processes the model of the system. This produces a scheduling of actions representing

the plan to achieve the requested goal. In the last phase, Agents perform plans action by action at run time. The *Execution Environment* monitors correct behaviours of agents: if for some reasons the current goal is no longer reachable, or if scheduled actions cannot be executed, e.g. because preconditions no longer meet or because Beliefs were different from $World$ real configuration, the Execution Environment enacts a *Replanning* action that tries to retrieve new plans for the same goals (if they exist).

Interactions among agents follow a *hierarchical orchestration* model as summarized in Fig.5.

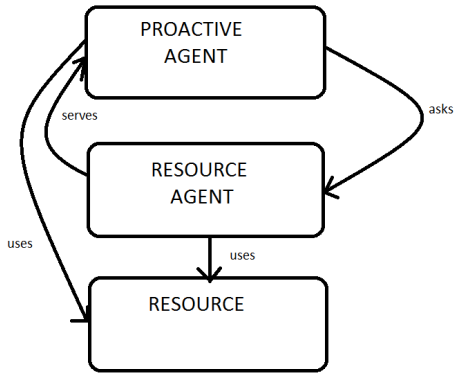

Figure 5: Agents interaction model

In this work we consider communication among agents only at different layers. The interaction model from Proactive Agents and Resource Agents to Resources is very simple: They can use a resource if it is *free* (i.e. if it is not used by other high level agents). However, interactions between Proactive and Resource Agents is more complex: a Proactive Agent can *ask for a service* to a Resource Agent (and this latter *serves* this kind of request). When a proactive agents asks for a service, it assigns a schedule of actions to a resource agent, providing a scheduling for a plan it wants to execute.

Planning actions execute with the following steps: (1) A Proactive Agent finds a plan in an environment with one ore more Resource Agents and resources. Actions in the plan are related to Resource Agents actions which in turn may use

simple Resources. (2) The proactive Agent with the new plan, assigns sub-goals to Resource Agents that, in turn, executes planning actions at a different layer of abstraction to reach sub-goals. (3) Resource Agents executes their sub-plans to reach sub-goals assigned by proactive agents. This usually involves the use of simple Resources.

Agents are grouped into *clusters*. A Cluster contains Resource Agents that are currently serving a proactive agent. Proactive Agents are able to *ask* for a service only to Resource Agents in their clusters. Communication between agents is guaranteed by a very simple messages exchange protocol. It uses four kinds of messages: *Join*, used by Proactive Agents to ask for a Resource to join its cluster;*Leave* used by a Proactive to notify a Resource that it is no longer in its cluster;*SendGoal* to ask for a service; *OKJoin* as acknowledgement message.

The methodology described above is fully implemented by a framework whose architecture diagram is in Fig.6.

The framework uses the UPPAAL engine to perform counter-example searching. Two kinds of planners are implemented: a *Classical Planner* that is a simple implementation of the Breadth First and A* state space search algorithms, and *Counter Example Planner*. The presence of different planning strategies enables the implementation of a *Multi Expert* system. Multiple planners can start in parallel processes improving performances or providing solutions at different layer of abstraction. The planning algorithm in the Counter Example Planner works in four steps: (1) The TA (Timed Automata) translator implements an algorithm for Agent models to Timed Automata; (2) The TA is passed to the UPPAAL engine in order to produce a counter example (if any) to the following formula: "**A [] !'GoalCondition'**" that is *"is it true that from initial state, only states where the goal condition is not satisfied are reachable?"*; (3) If the previous property is satisfied, then no plan exists for the current goal; otherwise, UPPAAL returns a *counter example* that is a path from the initial state to a state where the goal condition is satisfied, in form of an *Uppaal trace*; (4) The Uppaal trace translator perform the translation from trace to a sequence of actions that is the agent plan.

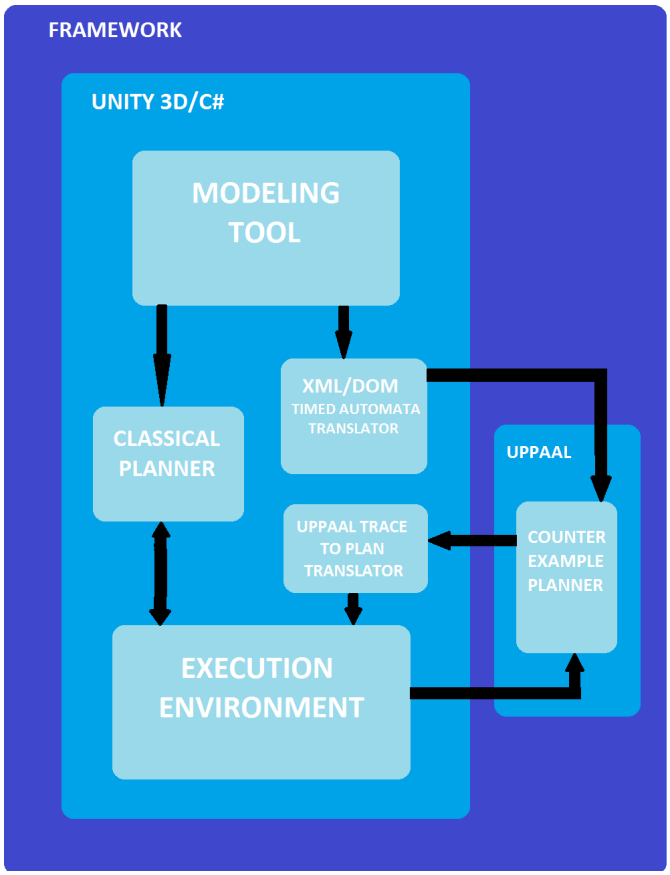

Figure 6: Framework architecture

The automata generation follows a *Self Learning* approach: the planner expands current state of the agent considering only possible performable actions. The planning automata is updated only with new state during expansion. This considerably reduces the number of state to analyze during planning and monitoring phases.

Thus, the planner forces pro-active elements in the game to produce new storylines. This involves interaction of other pro-active elements and players to fulfil some newly generated goals. We call pro-active elements *Agents*. Main agents in the game have some basic behavioral attitudes that describes abstract goals they want reach during their "lives" and the way they prefer to follow

while attempting to reach their goals.

When the new state space "sensed" by an agents changes, it tries to understand if it is possible to reach one of his attitudinal goal. This is done by reasoning in abstract terms, without considering any possible state space on the environment agent is living in. Notice that the representation of the state space of an agent should contain information about other agents and about the whole external environment, including any data about information held by all other agents.

In a MMO, this require a huge amount of data, and resulting planning and reasoning activities will be too much consuming in terms of time and resources[19, 20, 21].

This is the reason why each agent considers a limited amount of information about the environment and what the agent knows about other agents (notice that this information may be misleading). For the same reason, first attempt of planning is executed on abstract actions.

5.1. The Meta-Model

Formal models here enable formal planning: Fig.7 describes (an abstract version of) the metamodel used for intelligent definition in games. More details can be found in [22].

The meta-model extends the "Beliefs, Desires, Intentions" (BDI[23]) logics and in figures it is sketched by using a basic UML diagram. The main elements for **Agents** description are **Beliefs**, **Actions**, **Plans** and **Desires**. *Beliefs* contain data representing information the agents know about their state in the environment, about the environment itself and about beliefs they have about other agent. Particular types of Beliefs are the *Goals* they want to reach: a goal is represented hence by a particular assignment of variables in the beliefs of the agents. *Desires* represent combination of goals agents want to reach depending on their *Attitudes*. *Actions* describe the steps the agents are able to perform in the environment. Abstract Actions can be refined by using lists of refined sub-actions. Beside their functional descriptions, actions are related

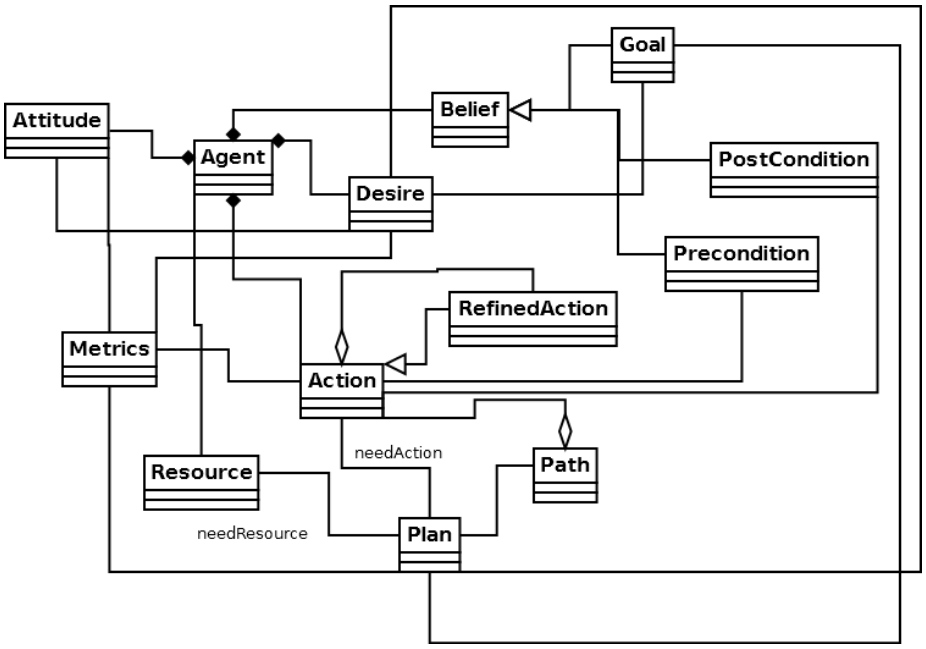

Figure 7: Multi-Agent Meta-Model

to two particular types of beliefs: *Preconditions* and *Postconditions* are beliefs combined by logical formulas that respectively indicate what must be verified to enable action execution, and what are its effects. *Attitudes* describe preferences of agents about their available actions, goals, desires etc. Preferences evaluation involve definition and use of proper *Metrics*. *Resources* define all the non pro-active elements in the game. Finally, *Plans* are proper lists of actions that lead to a Goal.

For what the planning methodology concerns, one of the most used approach in classical planning is to find a sequence of actions achieving a goal state, is the State Space Informed Search [24]. In the last years many works tried to develop new strategies to optimize the planning problem proposing different approaches [25, 26, 27]. It should be noted that planning in the literature usually neglects a common problem: the control of the existence of a plan. Approaches based on State Space Search are not efficient in detecting if a Goal is not reachable in any way and they result in excessive time and resources

consumption[28, 29, 30]. This problem is very important in Multi-Agent Systems, where concurrent execution of multiple agents complicate the state search problem because of resource sharing and of agents interactions. In our work we propose a Framework composed by a Modelling Tool, that provides the means to model Multi Agents Systems, and a Planner based on Counter Example of a Model Checking procedure.

The extended BDI metamodel also exploits First Order Logic (FOL) STRIPS [31] [32] for pre and post condition definition of actions. The Counter Example search is performed by the UPPAAL[33] Model Checker. This need the application of proper model translation techniques in order to produce a timed automata representation of the multi-agent system. Counter examples are returned in form of UPPAAL traces and then translated in sequences of actions.

If the number of beliefs is large, generation of the whole automaton in a single run could be infeasible; for this reason, automata generation follows an iterative refinement approach: each time an agent finds a new feasible plan, refined actions and beliefs are used in the next step of the methodology.

Action refinement is a complex problem and it is out of the scope of this work (similar refinement and composition techniques are found in [34, 35, 36, 17]).

5.2. Agent Based Planning Problem

In this section we provide a simplified version of the model we use to define our planning problem. The model is based on a Multi-Agent system representation of systems. We consider a variant of Beliefs, Desires, Intentions (BDI) logics[23] for our Multi-Agent System model that is a quadruple:

$$(Agents, World, \mathcal{TS}, \mathcal{F})$$

Where $Agents$ is the set of all agents in the system; $World$ represent the environment where agents executes; $\mathcal{TS}$ is a Transition System that resumes possible state transitions of agents in the environment and, finally, $\mathcal{F}$ is a a set of formulas expressed in first order logics that characterize each state in $World$.

We use a triple $\langle n, d, v \rangle$ to define variables evaluations that describes states in the World, where n is the name of a variable, d represents its domain and v a value assigned to the variable. A State $s \in World$ is a set of variable evaluation. In addition, we call *State Conditions* of a state, the set of all formulas in $\mathcal{F}$ that holds in a state s:

$$StateCondition(s) = \{\phi \in \mathcal{F}, s \in World : s \models \phi\}$$

In addition, ϕ cannot be a subformula of other formulas holding in s.

In this work we consider $\mathcal{TS}$s and States with only one State Condition per state. If $s \models \psi$; $s \models \phi$ and $s \models \psi \wedge \phi$; then we consider only the last formula as State Condition in s.

An Agent is in turn a triple:

$$(Actions, Beliefs, Goals)$$

Actions is a set of possible actions an agent is able to perform. Actions modify the environment changing $World$ representation. They can also require the intervention of other agents in order to achieve common goals and in general, they include *communication* and *execution* actions. In addition, an action can be *reactive* if its execution depends on external events or messages; or *proactive* if its execution is decided directly by the agent. We call *Proactive Agent* an agent with at least one proactive action; an agent with no proactive actions and with at least one reactive action is a *Resource Agent*; an agent with no reactive or proactive actions is classified simply as a *Resource*.

$Beliefs$ include the knowledges the agents have about: the $World$; the Agent itself; other Agents.

Notice that an agent may have a belief about the $World$ which in turn is *not* true in the environment: in general, beliefs of each agents may not be exact. $Goals$ is a set of states in $\mathcal{TS}$ that represent goals an agent want to reach. Since $\mathcal{TS}$ is not available when agents are defined, abusing notation we identify Goals with formulas in $\mathcal{F}$ that are satisfied in goal states. We call these formulas:

Goal Conditions. Notice that a goal condition is a State Condition for a goal state.

Beliefs are manages as World variables and states (they are practically local World representation in each agent). Agents define the $\mathcal{TS}$ Transition System on $World$ states by means of $Actions$. An action $\alpha \in Actions$ is a triple:

$$(name, Precondition, Postconditions)$$

where *name* is trivially the name of the action;$Postconditions$ is a set of formulas that hold in the new state; *Precondition* is a formula that *evaluates* true in a state s in order to *apply*(execute) the action and to produce a state transition:

$$s \overrightarrow{\alpha} s'$$

produces a transition from the state s to s'. If Precondition of α evaluates true in s, s' will be the same of s, except for variables involved in Postconditions evaluation. The values of these variables have to change in order to satisfy *all* effects in s':

$$\forall \phi \in Postconditions; s' \models \phi$$

Furthermore we must consider that an agent executing an action can access only to its local representation of $World$, i.e. to its beliefs. Hence, if agent's beliefs and World State are not synchronized (i.e., if the agents has a wrong belief about the world), it is possible that Precondition is evaluated true on beliefs, but *not* on $World$ state.

In order to apply an action, we must execute the following two steps: (1) an agent tries to apply Postconditions in a state s producing a transition from s to s' if precondition is evaluated true on its *beliefs*; (2) if Precondition evaluates true in *World too*, then $s \overrightarrow{\alpha} s'$ both in agent's Beliefs and in $World$ too.

$\mathcal{TS}$ is then the Transition System defined by the application of all actions

in any state of $World$, performed by *all* Agents in the model. The execution of an action to build $\mathcal{TS}$ must follow the two steps previously defined. State transitions apply both to agents $Belief$ and to $World$. Anyway Precondition control is enacted on Beliefs first. If evaluation fails on Beliefs, the action is not applied even if Precondition would evaluate true on $World$. In this model, a **Plan** to reach a Goal $\mathcal{G}$ with given Goal Condition is a *path* from a starting state to a state where the Goal Condition holds. Notice that in a Multi Agent System, actions in a transition System can be executed by different agents, even concurrently. We consider here a path as a linear scheduling of concurrent applications of actions.

A Planning problem hence, is the problem of finding such a path or to state that the requested goal is unreachable in the current environment.

5.3. Evaluation of Counter-example based planner

In order to evaluate how our counter-example planner overcomes some computational problems of classical approaches, we provide a multi-agent based representation of a well known Artificial Intelligence problem: the N Queen positioning (which has a high computational complexity). Given a NxN chess board, the problem consists in positioning N queens in a way they cannot attack each other. With our framework, the system environment is modeled with the following *World* elements, defining initial state,(we consider for simplicity the case of N=5):

(pos(q1),"Position",0); (pos(q2),"Position,0); (pos(q3),"Position",0); (pos(q4),"Position",0); (pos(q5),"Position",0); (index(q1),"Position",1); (index(q2),"Position",2); (index(q3),"Position",3); (index(q4),"Position",4); (index(q5),"Position",5);

Here, *Position* is the name of a domain defining the possible values of the $World$ variables: in this case the set of values $[0, 1, 2, 3, 4, 5]$; pos(q_i) is the row position of the ith queen and index(q_i) is its column position. If $pos(q_i) = 0$ the queen i is not currently positioned. We have here a single Pro-active agent (the

player) and Queens are modelled as resources. Beliefs of the agent are modelled exactly in the same way of $World$ variables. Actions requires the definition of Preconditions and Postconditions. We have here a single action defined as follows:

- Precondition: (pos(X)==0) AND (pos(X)!=Y) AND (Y!=pos(Z)) AND (Y!=(pos(Z)-(index(X)-index(Z)))) AND (Y!=(pos(Z)+(index(X)-index(Z))))
- Postcondition: pos(X)=Y
- Variables:

  (X,"Queen","Existential") (Y,"Position,"Existential") (Z,"Queen","Universal", (index(Z) <index(X)))

So "X" and "Y" are existentially quantified variables, "Z" is a univesally quantified variable with the property that index(Z) <index(X),and "Queens" is a domain,in this case of N=5 is the set of values [q1,q2,q3,q4,q5]. The agent has a single goal,whose condition is, for N=5 : (pos(q1)!=0) AND (pos(q2)!=0) AND (pos(q3)!=0) AND (pos(q4)!=0) AND (pos(q5)!=0).

Fig.8 reports response times of the two planners(for the counter example planner the time shown is the sum of the automata construction and planning times).

In this evaluation we can notice that Counter Example Planner clearly overcomes the other planner: for N=7 the planning time is not shown for the Classical Planner since the solution couldn't be found because it consumed all available hardware resources[3].

6. Example

As example, let us consider the introduction of a new agent in the game that possesses an item that increases magical powers. If this information is

[3]We performed experiments on a Quad Core i7 4800 CPU with 4 Gbytes RAM

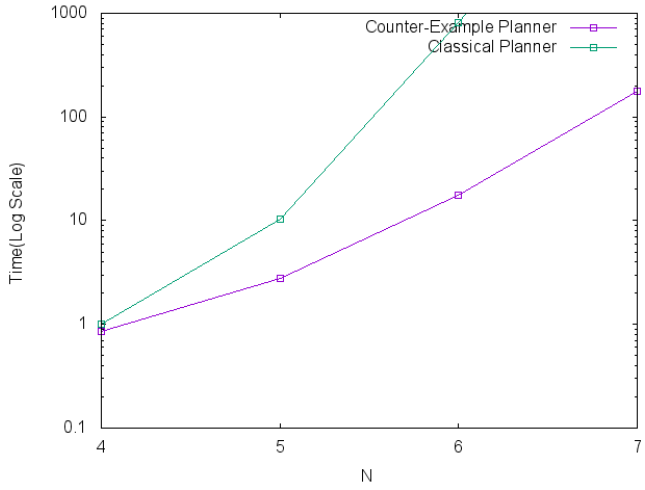

Figure 8: Planning Time comparison for the N-Queens problem

propagated (in any way) to an agent in the game, it can try to understand if a plan exists that can improve his goals. Let us consider its goals include the increase of its magical power. A first abstract reasoning results in the fact that he can *get* the item by *buying* it or by *stealing* the object in any way. If it "prefers" to buy it, he can now refine its plan by understanding how to buy an item to get it. This may require a refinement of the abstract plan: it has to know the price of the item, paying for it and getting the item. In order to know the price and to pay, probably the agent needs that another agent (or a player) will travel to the place where the item owner lives, paying for the item and returning the item to the main agent. In the same way, if it chooses to steal the object, he can hire someone to steal it etc.

Some plans can be unfeasible from the very beginning of planning actions. For example, buying a so powerful item is possible only if the owner is willing to sell it. In other case, the only possible plan is the other one (if stealing is among main agent attitudes). Further refinements define hiring mechanisms etc. Notice that the more fine grained the agents are characterized, the more complex is the planning refinement. Final steps in refining are the arrangement of game mechanics to use to enact more abstract actions. For example, the

hiring of other agents to steal the object can be performed by creating messages on notice boards or by introducing a simple (and not pro-active) chatting agent to spread requests in the environment.

Notice that in the same environment, if a player steals the object to give it to the main agent, it actually possesses the object, and other agents can plan in turn to steal it from the player or from the main agent.

6.1. Discussion

With This creates a network of possible new story-lines that can evolve in many ways, depending on the number of agents and players in the game, their position, their attitudes etc. We can then suggest a sequence of actions to execute by translating a MAS model into a Timed (product) Automaton and exploiting model checking in order to find plan. We identify events to generate, agents to activate and resource to use depending social media analysis.

Even if planning is usually a time consuming operation (since it must navigate into something similar to a search based tree) we have collected many interesting and promising results from the application of the methodology.

7. Conclusions

In this paper we presented a methodology for enabling Procedural Content Generation (PCG) by combining social media analysis and formal methods. In particular, our approach is able to understand when a given content in a game is becoming boring for users, thank to the analysis of the semantics and of the frequencies of posting of elements in social media. Our approach is is based on multi-agent models and on counter-example guided planning procedure. Planning and generation of new contents depends on results of web and social networks resource. We presented a framework able to enact the methodology. The framework allows to define *intelligent* agents in the game, able to evolve and act pro-actively, and able to generate new contents by requiring actions of other passive agents and, of course, of players. Future works include

the improvement of the counter-example planner algorithm and the extension of the methodology to conflicting interactions among agents and to choreographed execution of plans.

7.1. Limitations of the study and directions for future research

Here we want to address some limitations of our work, in order to point where the described approach fails. First of all let us consider that the more pro-active, reactive and resource agents interact, the more complex become the planning problem. Hence, the system provide quests at real-time if we consider few elements in the generation. In order to enhance generation time, we must limit the number of agents with available actions and events. This can be done by implementing proper abstraction policies as discussed before, but proper refinement actions have to be implemented too, in order to force the execution of real steps in plans.

The presence of a domain ontology in our approach is a point of strength as well as a limitation: in order to provide coherent events in planning, to define properly preconditions and postconditions, and to describe actions behaviors for agents, we have to develop ontologies with lot of details and relationships. At the moment, this work is performed manually and we have not yet implemented any technique able to populate the ontology by analysis of social media.

Finally, we have to model behaviors of both intelligent and user agents. This means that we can consider only few actions from real users, hoping that they reply to intelligent agents requests in the way pre-determined by generated plan. We should retrieve possible actions from social media analysis too, but we leave this as future work.

References

References

[1] A. Iosup, Poggi: generating puzzle instances for online games on grid infrastructures, Concurrency and Computation: Practice and Experience 23 (2) (2011) 158–171.

[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, C. Browne, Search-based procedural content generation: A taxonomy and survey, IEEE Transactions on Computational Intelligence and AI in Games 3 (3) (2011) 172–186.

[3] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: A survey, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 9 (1) (2013) 1.

[4] A. Pantaleev, In search of patterns: Disrupting rpg classes through procedural content generation, in: Proceedings of the The third workshop on Procedural Content Generation in Games, ACM, 2012, p. 4.

[5] S. Dahlskog, J. Togelius, Patterns and procedural content generation: revisiting mario in world 1 level 1, in: Proceedings of the First Workshop on Design Patterns in Games, ACM, 2012, p. 1.

[6] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, K. O. Stanley, Combining search-based procedural content generation and social gaming in the petalz video game., in: Aiide, Citeseer, 2012.

[7] M. Stephenson, J. Renz, Procedural generation of levels for angry birds style physics games, in: The AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2016.

[8] M. Mateas, A. Stern, Façade: An experiment in building a fully-realized interactive drama, in: Game developers conference, Vol. 2, 2003.

[9] R. Aylett, J. Dias, A. Paiva, An affectively driven planner for synthetic characters., in: Icaps, 2006, pp. 2–10.

[10] Y.-G. Cheong, M. O. Riedl, B.-C. Bae, M. J. Nelson, Planning with applications to quests and story, in: Procedural Content Generation in Games, Springer, 2016, pp. 123–141.

[11] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: A survey, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 9 (1) (2013) 1.

[12] T. Ishida, T. Ando, N. Uchida, Y. Shibata, The digital contents management system based on position information initiate fusion of AR and sensor technology, International Journal of Space-Based and Situated Computing (IJSSC) 6 (1) (2016) 31–42.

[13] M. Urakawa, M. Miyazaki, I. Yamada, H. Fujisawa, T. Nakagawa, A study about integrating video contents with web services based on the RDF, International Journal of Space-Based and Situated Computing (IJSSC) 6 (2) (2016) 65–73.

[14] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (pcgml), arXiv preprint arXiv:1702.00539.

[15] S. Dahlskog, J. Togelius, Procedural content generation using patterns as objectives, in: European Conference on the Applications of Evolutionary Computation, Springer, 2014, pp. 325–336.

[16] G. N. Yannakakis, J. Togelius, Experience-driven procedural content generation, IEEE Transactions on Affective Computing 2 (3) (2011) 147–161.

[17] F. Moscato, F. Amato, Automatic cloud services composition for big data management, in: Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016, 2016, pp. 46–51.

[18] F. Amato, V. Moscato, A. Picariello, G. Sperli, Modelling multimedia social network for topic ranking, in: Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016, 2016, pp. 81–86.

[19] X. Ye, B. Khoussainov, Fine-grained access control for cloud computing, International Journal of Grid and Utility Computing 4 (2-3) (2013) 160–168.

[20] A. Verma, S. Kaushal, Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud, International Journal of Grid and Utility Computing 5 (2) (2014) 96–106.

[21] G. Feng, R. Buyya, Maximum revenue-oriented resource allocation in cloud, International Journal of Grid and Utility Computing 7 (1) (2016) 12–21.

[22] F. Moscato, F. Amato, Thermal-aware verification and monitoring of service providers in metamorp(h)osy, in: Proceedings - 2014 International Conference on Intelligent Networking and Collaborative Systems, IEEE INCoS 2014, 2014, pp. 551–556.

[23] M. Wooldridge, Agent-based software engineering, in: IEE Proceedings on Software Engineering, 1997, pp. 26–37.

[24] S. Russell, P. Norvig, A. Intelligence, A modern approach, Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25.

[25] J. A. Baier, F. Bacchus, S. A. McIlraith, A heuristic search approach to planning with temporally extended preferences, Artificial Intelligence 173 (5) (2009) 593–618.

[26] J. Hoffmann, B. Nebel, The ff planning system: Fast plan generation through heuristic search, Journal of Artificial Intelligence Research (2001) 253–302.

[27] P. Haslum, H. Geffner, Heuristic planning with time and resources, in: Sixth European Conference on Planning, 2014.

[28] B. Balusamy, P. Krishna, Collective advancements on access control scheme for multi-authority cloud storage system, International Journal of Grid and Utility Computing 6 (3-4) (2015) 133–142.

[29] X. Wang, D. Huang, I. Akturk, M. Balman, G. Allen, T. Kosar, Semantic enabled metadata management in petashare, International Journal of Grid and Utility Computing 1 (4) (2009) 275–286.

[30] T. Xue, S. Ying, Q. Wu, X. Jia, X. Hu, X. Zhai, T. Zhang, Verifying integrity of exception handling in service-oriented software, International Journal of Grid and Utility Computing 8 (1) (2017) 7–21.

[31] R. E. Fikes, N. J. Nilsson, Strips: A new approach to the application of theorem proving to problem solving, Artificial intelligence 2 (3) (1972) 189–208.

[32] D. Burfoot, J. Pineau, G. Dudek, Rrt-plan: A randomized algorithm for strips planning., in: ICAPS, 2006, pp. 362–365.

[33] G. Behrmann, A. David, K. G. Larsen, P. Pettersson, W. Yi, Developing uppaal over 15 years, Software: Practice and Experience 41 (2) (2011) 133–142.

[34] G. Di Lorenzo, F. Moscato, N. Mazzocca, V. Vittorini, Automatic analysis of control flow in web services composition processes, in: Proceedings - 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP 2007, 2007, pp. 299–306.

[35] F. Moscato, Exploiting model profiles in requirements verification of cloud systems, International Journal of High Performance Computing and Networking 8 (3) (2015) 259–274.

[36] F. Amato, F. Moscato, Exploiting cloud and workflow patterns for the analysis of composite cloud services, Future Generation Computer Systems 67 (2017) 255–265.