

# Automatic Generation of Game Level Solutions as Storyboards

David Pizzi<sup>1</sup>, Marc Cavazza<sup>1</sup>, Alex Whittaker<sup>2</sup> and Jean-Luc Lugin<sup>1</sup>

<sup>1</sup> School of Computing, University of Teesside, Middlesbrough, TS1 3BA, United Kingdom.  
{d.pizzi, m.o.cavazza, j-l.lugin}@tees.ac.uk

<sup>2</sup> Eidos - Beautiful Game Studios, London, N7 9DP, United Kingdom.  
alexw@bgstudios.co.uk

## Abstract

Interactive Storytelling techniques are attracting much interest for their potential to develop new game genres but also as another form of procedural content generation, specifically dedicated to game events rather than objects or characters. However, one issue constantly raised by game developers, when discussing gameplay implications of Interactive Storytelling techniques, is the possible loss of designer control over the dynamically generated storyline. Joint research with industry has suggested a new potential use for Interactive Storytelling technologies, which stands precisely as an assistance to game design. Its basic philosophy is to generate various/all possible solutions to a given game level using the player character as the main agent, and gameplay actions as the basic elements of solution generation. We present a fully-implemented prototype which uses the blockbuster game *Hitman*<sup>TM</sup> as an application. This system uses Heuristic Search Planning to generate level solutions, each legal game action being described as a planning operator. The description of the initial state, the level's objective as well as the level layout, constitute the input data. Other parameters for the simulation include the *Hitman*'s style, which influences the choice of certain actions and privileges a certain style of solution (e.g. stealth versus violent). As a design tool, it seemed appropriate to generate visual output which would be consistent with the current design process. In order to achieve this, we have adapted original *Hitman*<sup>TM</sup> storyboards for their use with a generated solution: we attach elements of storyboards to the planning operators so that a complete solution generates a comic strip similar to an instantiated storyboard for the solution generated. We illustrate system behaviour with specific examples of solution generation.

## Introduction

Despite the growing interest in Interactive Storytelling (IS) techniques, their actual relation to traditional gameplay remains to be investigated. Many game designers have expressed concerns about the incorporation of such generative techniques in traditional game titles, mainly because of the lack of control they will have upon dynamically generated contents.

However, in the course of joint research with a major European publisher (Eidos Interactive), a novel use for IS techniques was suggested, precisely as a support to the game design phase. It consists in generating all the possible solutions for a given game level. In this approach, the player character is represented as the main planning agent, making use of all gameplay actions to produce a solution to the game level. This relies on the dual nature of Planning technologies, which are able to generate sequences or narrative actions as well as solutions to a game level problem. The generated solutions, which help assess the final gameplay for a given level design, can be visualised using comic strips similar to the original storyboard. We present in this paper the first version of a fully implemented prototype of such a design tool.

## Related Work

Formal approaches have been previously proposed to model the design process of computer games (Natkin and Vega 2003; Natkin *et al.* 2004; Collé *et al.* 2005; Brom and Abonyi 2006). The underlying idea is to model the spatio-temporal relationships which occur within the game universe. These techniques thus allow describing the logical structure of the level missions in the game by modelling the ordering of action sequences using graphical models such as Petri Nets. They allow a dynamic visualisation of game scenarios. However, for industrial use, the designers would have to learn the adequate formalism. Moreover, they are essentially designed for analysing and validating pre-existing scenarios rather than assisting in their creation.

In the field of IS itself, several authors have described tools facilitating the construction of story using some visualisation support. For instance, story graphs have been used in different authoring systems to explicitly represent all the possible story paths in INSCAPE (Zagalo *et al.* 2006), U-Create (Sauer *et al.* 2006) and SceneMaker (Gebhard *et al.* 2003). These tools present intuitive methods of visualisation which can assist authors in their creation process. However, they are based on non-generative formalisms (Skorupski *et al.* 2007) constraining authors to manually encode possible plan variations.

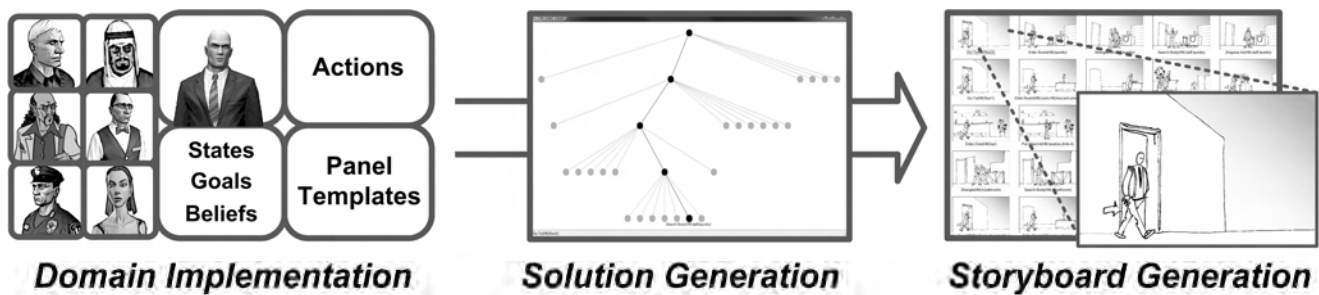


Figure 1: Overview of the Authoring Process

## Overview

Our system enables game designers to generate and explore complex game scenarios without relevant expertise in AI Planning technologies, and immediately visualise them as storyboards. This approach differs from previous generative systems such as ScriptEase (Cutumisu *et al.* 2006), which assists designers in producing character behaviour scripts within the boundaries of previously validated game scenarios, in that it produces level solutions from first principles.

As illustrated by Figure 1, the exploratory approach supported by our system follows three main stages: *domain implementation*, *solution generation* and *storyboard generation*.

- The first step corresponds to the elicitation of all knowledge required to describe a game level, such as the various states (e.g. the level goal is to kill three different targets, different initial states, etc.) and the various game actions described through their pre-conditions and their consequences. In terms of Planning, this corresponds to *domain implementation* where each part of the planning domain is created (i.e. propositions, operators, states and goal). The domain description also includes a formalisation of the initial state and the goal state, which corresponds to the level's objective.

The (Drawing) Panel Templates are attached to planning domain elements; therefore, they need to be defined at this stage. Their composition will be detailed in the section *storyboard generation*.

- The second step, or *solution generation*, consists in generating a possible solution to the Hitman<sup>TM1</sup> level under consideration, seen as a planning problem whose operators are the game actions. The solution is generated through Heuristic Search Planning (HSP) (Bonnet and Geffner 2000) planning the shortest solution from initial state to level goal. We implemented a real-time version of the HSP in C#, in which heuristics are used to guide action selection towards the level solution. The heuristics calculation is based on the simple Value Iteration (VI) method (Liu *et al.* 2002). We have opted for a “real-time” version of HSP by implementing RTA\* (Korf 1990) as it allows backtracking

to avoid deadlocks and also anytime world state variations. Computation of the heuristic accounts for a significant fraction of the total CPU time for the planner, as is classically described in the HSP literature (Bonnet and Geffner 1999). The planner still produces on average a complete solution in approx. one second on a 2 GHz Intel processor, which is fully compatible with its use within a design/authoring environment. It generates solutions to the game level as a sequence of actions leading to the level objective, represented as the plan's goal.

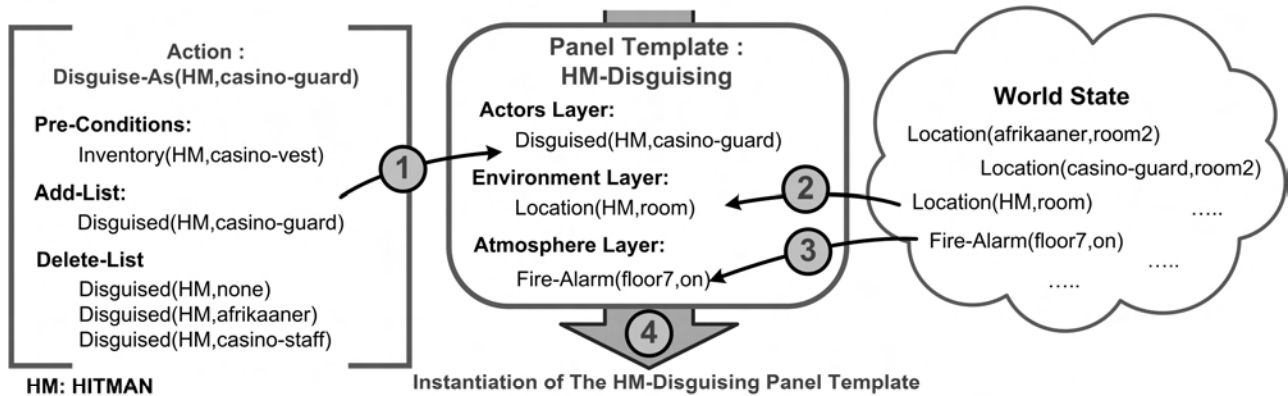
As an alternative to offline generation of a complete solution, an interactive mode allows step-by-step generation of a solution, which includes the visualisation of all possible outcomes. Starting from the initial state, the user can expand the plan at each step using a tree representation until the goal state is reached. After each action is selected by the user, the system automatically offers a list of possible subsequent actions.

At this point, the solution plan only exists in the form of a list of operators. These are difficult to read for non-programmers or anyone not familiar with Planning formalisms. We thus looked for a more user-friendly way of displaying level solutions which could also be integrated in the standard design process. One natural idea would be to automatically generate solutions as comic-like storyboards.

Comics constitute a highly expressive medium as they allow presenting a story using a limited number of panels (McCloud 1993). However, even if comic strips have been described as part of IS system output in previous work (Alves *et al.* 2008) and seem promising, the creation of comics is quite a complex process and obeys a large set of rules (e.g. on transitions, panel shapes, etc.) (Eisner 1985; Alves *et al.* 2007). On the other hand, storyboards, which use simplified conventions, are commonly used by designers because they are both expressive and simple to produce. This is why we have decided to follow simplified storyboard conventions for the generation of visuals corresponding to the level solutions produced.

- The last stage, *storyboard generation*, uses templates to construct a storyboard panel from data corresponding to the selected action (Figure 2). In turn, the various panels will be assembled sequentially into a storyboard presenting the complete level solution generated.

<sup>1</sup> Hitman<sup>TM</sup> is a trademark of IO Interactive Ltd and Eidos PLC.



**Figure 2: Example of a Panel Template Instantiation from a selected Action and current world state**

- In addition, designers are allowed to interact with the solution generation process at any time using a *dynamic environment simulation* feature. It allows them to reproduce the dynamic state variations that will normally occur within the game (e.g. simulating NPC movements from a room to another), so as to explore opportunities for various player actions.

In the next sections, we present in detail each of these stages. First, we discuss the authoring methods for the domain implementation. We then describe the solution generation process, the dynamic environment simulation process and finally the storyboard generation.

### Domain Implementation

The first step consists in providing a complete propositional representation of the world (e.g. `Disguised(HM,afrikaaner)`, `Location(HM,room)`, etc.). The initial state and the level goal are then simply represented by conjuncts of propositions. The planning operators are represented using a STRIPS-like formalism (i.e. a set of propositions as pre-conditions and effects) (Fikes and Nilsson 1971) and correspond to game actions that can be performed by the player character *Hitman*. Each operator is associated with a panel template which will construct a storyboard's individual panel according to the current world state.

During generation, the selection of certain critical actions (e.g. various killing or neutralising methods) is made using a categorisation of operators according to the different *Hitman* styles. For instance, the stealth style will favour discreet actions such as `Strangle` or `Put-Poison-In`, over noisier executions such as `Shoot` or `Trigger-Bomb`.

The formalisation of gameplay actions and states as a Planning domain is not without impact on the types of solutions that will be generated. Common pitfalls consist in representing the domain at a level of abstraction too high or conversely using too specific actions, which will limit the generative power of the system. Knowledge elicitation consists in the description of a level's actions (e.g. `Search-Body`, `Disguise-As`, `Unlock-Door`, or `Shoot`).

This is a manual process which becomes error-prone when the size of the planning domain and the number of operators increase. It is well-known that maintaining consistency between the predicates used by the various operators can become challenging when describing operators manually. The calculation of the HSP heuristic introduces further constraints, as it requires that each proposition appearing once as a pre-condition of an operator should at least also be present in one add-list (this could otherwise lead to the calculation of spurious heuristic values inducing potential action selection errors). Inconsistencies in predicates' labels could also be responsible for errors in the content of operators with other detrimental side-effects. There is thus a need to check consistency of pre-conditions and effects every time changes to the planning domain are introduced as part of the knowledge elicitation process. Our authoring interface ensures the development of a coherent and consistent planning domain.

This stage requires to be encoded by a member of the development team who is familiar with logical formalisms. The designers will from there produce and validate the game solutions at a higher level of abstraction (i.e. ignoring failures that can be caused by timing issues).

### Solution Generation Process

In *Hitman*<sup>TM</sup>, where solutions rely on a sophisticated plan, the various causal dependencies generated by HSP planning may be difficult to recognise. For instance, when the level goal involves killing a target, *Hitman* will first kill a casino staff member in order to take his clothes. Then, he will need to pick up poison situated in his hotel room before being able to taint and serve a toxic drink to his target.

Therefore, we have imagined that the set of possible plans could be visually represented in order to control the unfolding of the generated content. Consequently, our system also proposes a step-by-step plan simulation, in which the user can visualise the results using a tree-like hierarchy (See Figure 4). This corresponds to an expert mode, which is of interest when enquiring about the

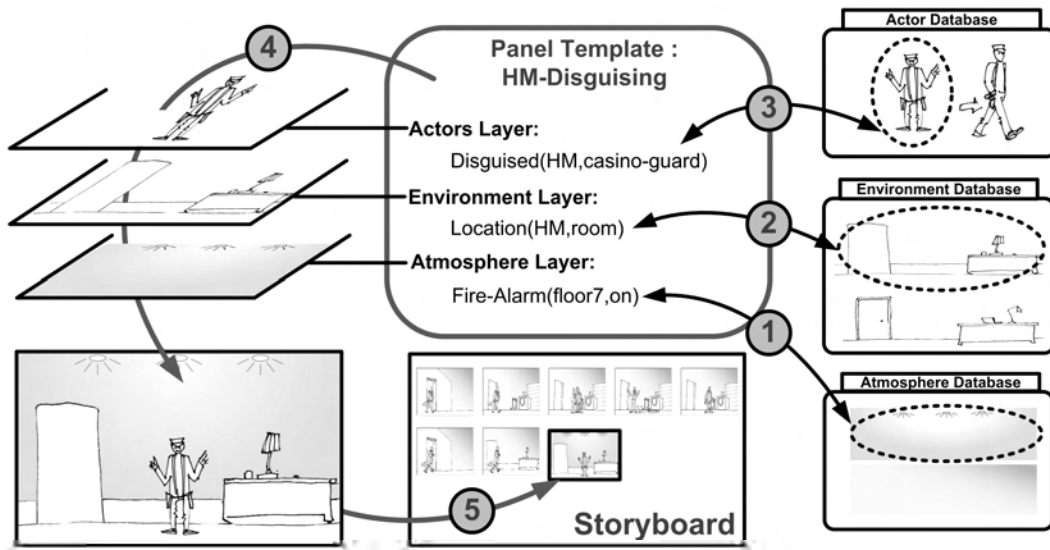


Figure 3: Example of Panel Generation from an Instantiated Panel Template

explanation behind certain solutions produced. Every operator in the planning domain is tested for applicability. Whenever an operator pre-conditions are satisfied, it is applied to create a new state that could further be extended. For efficiency purposes and in order to avoid redundancy, we only develop new states that diverge from their parent. The whole tree can be automatically scanned so that all the possible solutions can be listed and visualised. When a node is selected, the entire plan that leads to it can then be rendered as a storyboard using the *storyboard generation* process.



Figure 4: Step-by-Step Solution Generation proposing all possible alternative actions.

### Dynamic Environment Simulation

Here the planning agent represents the actions of the player character. However, the real game environment is dynamic (i.e. NPC, including *Hitman*'s targets, have autonomous behaviour). For instance, the Afrikaaner character (i.e. one of the *Hitman*'s main targets) keeps walking cyclically from the bar to his room passing by the bathroom. *Hitman* has thus several options to kill his target and each of these options should also be represented in the scenarios. Consequently, spatio-temporal variations have also to be simulated within the solution generation process. We have introduced the possibility for the designer to modify the world state at any stage in order to make the appropriate state variations. Plan generation is synchronised with plan

interruptions reproducing characters movements (e.g. Update-Position) or world events (e.g. Character-Level-Arrival).

### Storyboard Generation

We use, as a way of creating graphical content, a method called *automatic composition*. Each game action, described by a Planning operator, is associated with a panel template. Such template supports the generation of the final panel from a set of elementary images<sup>2</sup>.

As illustrated in Figure 3, our storyboard template is composed of three main layers: i) An *Atmosphere Layer* (background), ii) an *Environment Layer* (mid-ground) and iii) an *Actors Layer* (foreground). For each layer, a template will define a position for an image (jpg, png, etc.). In addition, selection rules can be attached to these layers to associate a domain proposition to a specific pre-drawn image.

The *Atmosphere Layer* is used as a background image to emphasise a particular ambience (e.g. glows of fire or emergency lighting (i.e. darker background), etc.) or game events (e.g. fire alarm on) while the *Environment Layer* represents a given room or place such as a kitchen or laundry. The *Actors Layer* is used to draw elements of the scene that could be rendered at different positions (e.g. characters, objects, etc.). The objects in the scene can be rendered at different pre-set positions (left, centre, right, etc.) and also different heights (above ground, ground level, etc.).

The panel generation process relies on two operations: panel template Instantiation and Aggregation. The first phase happens once an action is selected. The generic template, previously associated with the action is then activated; consequently the *Actor Layer* is matched to the action add-list predicates (Figure 2-1). The current

<sup>2</sup> All design documents have been provided by IO Interactive Ltd.

*Environment* and *Atmosphere* layers are then retrieved from the current world state (Figures 2-2 and 3). Once instantiated with the current story context (Figure 2-4), the panel template layers are now able to determine the image corresponding to the proposition found (Figure 3). Here,

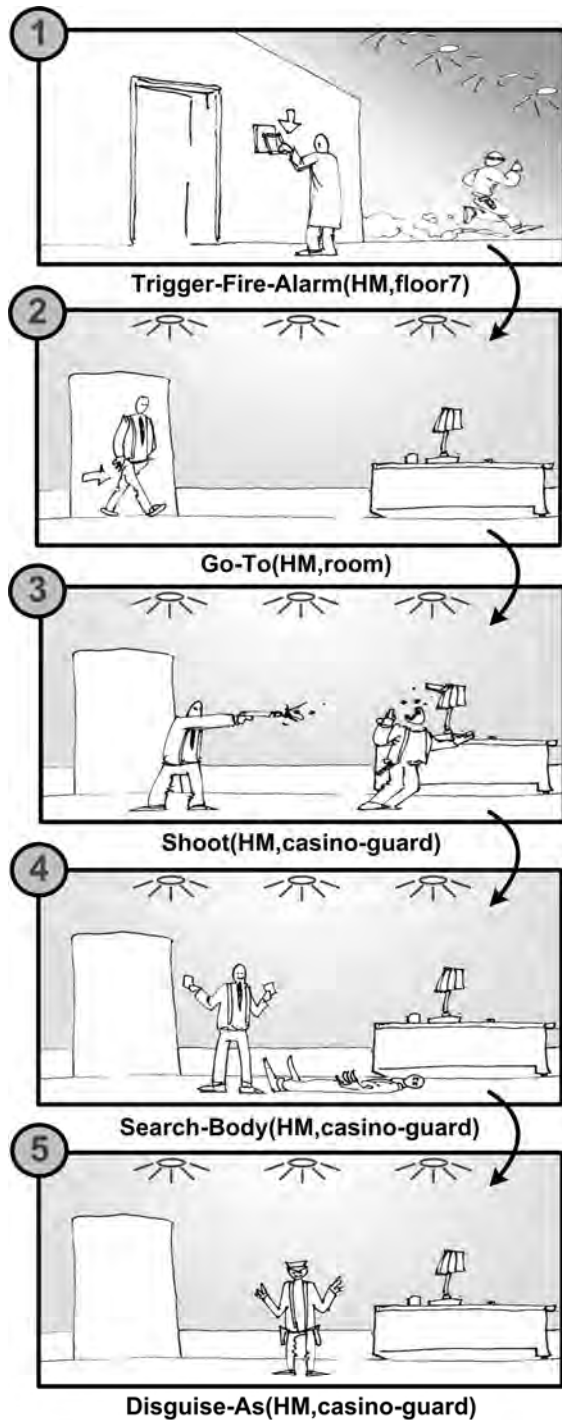


Figure 5: Extract of a storyboard produced

*Hitman* is actually located in a room, the proposition  $Location(HM,room)$  is part of the world state, so the corresponding picture “Room.png” is selected (Figure 3-2). During this stage, referred to as Panel Aggregation, the template browses the Actor, Environment and Atmosphere picture databases. Additional selection rules exist at the template level to distinguish different images which correspond to the same proposition: for instance  $Disguised(HM,casino-guard)$  could have two pictorial representations in our databases (in one of these, *Hitman* is standing (Figure 3-3) while he is walking in the other). Therefore, each panel template layer could contain “rules” stating the correct picture to use. These rules are specified by the designer during the template creation in the case of an ambiguous situation.

Finally, once each image layer has been selected, we render the final panel by drawing each layer from the furthest ones (i.e. the background and the location) to the closest ones (i.e. all the characters and objects) (Figure 3-4). Lastly, the rendered panel is added to the storyboard (figure 3-5).

## Results and Validation

The final planning domain contains 88 operators and 95 propositions. The length of the solution generated varies from 25 up to 40 actions, depending on the *Hitman*'s style (i.e. from stealth to violent). The generated solutions were assessed against known published solutions as well as presented to the game designers. Among the generated solutions certain reproduce faithfully the original solutions initially storyboarded by the designers. The alternative level solutions produced were also successfully tested within the game environment itself.

Figure 5 shows an example of a generated storyboard: *Hitman* triggers the fire-alarm in order to clear the door by forcing the guard to leave (Figure 5-1). Once the entrance is deserted, he can enter the room (Figure 5-2) and kill the only casino guard who remains inside (Figure 5-3). Then, he is able to take the casino guard's clothes (Figure 5-4) and disguises himself as a casino guard, which will grant him future access to new areas (Figure 5-5).

In terms of storyboard generation, the total number of panel templates is 24. However, we must take into account that certain templates cover a significant number of actions. For instance, the “HM-Walking” template is used by 43 actions. This generative approach could represent a considerable time saver knowing that it can generate panels for any possible situation where traditional approaches would have requested a 2D artist's intervention (i.e. drawing a completely new situation). This is particularly relevant when generated plans create novel situations. Furthermore, the total number of required drawings being significantly reduced, their quality could become even more elaborate.

## Conclusions

While most of the AI techniques used in computer games are still dedicated to character behaviour generation, we have presented an AI system assisting the game design process. This approach, derived from Interactive Storytelling technologies, supports the generation of level solutions, while validating their narrative content. Our first prototype has been tested using data from the released title Hitman™ Blood Money. This made it possible to validate the solutions generated by the system, either experimentally by playing the game or by matching them to published strategy data or on-line spoilers. Using AI in support to game design appears as a promising research topic which can have an impact on development costs and could facilitate collaboration between AI Programmers and Game Designers (an example of the latter would be the “expert mode” of our level solution generation system).

## Acknowledgements

IO Interactive Ltd is thanked for providing the original storyboards and level design documents. This work has been funded in part by the Department of Trade and Industry, via the Technology Programme BARDS Project, in collaboration with Eidos Interactive Ltd.

## References

- Alves, T., McMichael, A., Simões, A., Vala, M., Paiva, A. and Aylett, R. 2007. Comics2D: Describing and Creating Comics from Story-Based Applications with Autonomous Characters. In *the Proceedings of CASA*, Hasselt, Belgium.
- Alves, T., Simões, A.R., Figueiredo, R., Vala, M., Paiva, A., and Aylett, R. 2008. So tell me what happened: turning agent-based interactive drama into comics. In *the Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, Estoril, Portugal, 1269–1272.
- Bonet, B., and Geffner, H. 1999. Planning as heuristic search: new results. In *the Proceedings of the European Conference on Planning (ECP'99)*, 360–372.
- Bonet, B., and Geffner, H. 2000. HSP: Heuristic Search Planner. In *AI Magazine*, Vol. 21(2).
- Brom, C., and Abonyi, A. 2006. Petri-Nets for Game Plot. In: *the Proceedings of Artificial Intelligence and Simulation Behaviour*, Bristol, UK, Vol. 3:6–13.
- Collé, F., Champagnat, R., and Prigent, A. 2005. Scenario analysis based on linear logic. In *the Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, Valencia, Spain.
- Cutumisu, M., Szafron, D., Schaeffer, J., Waugh, K., Onuczko, C., Siegel, J., and Schumacher, A. 2006. A Demonstration of ScriptEase Ambient and PC-interactive Behavior Generation for Computer Role-Playing Games. In *the Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment International Conference*, Marina Del Rey, California, USA, 141–142.
- Eisner, W. 1985. Comics & Sequential Art. *Poorhouse Press*, Tamarac, Florida, USA.
- Fikes, R., and Nilsson, N. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. In *the Proceedings of the Second International Joint Conference on Artificial Intelligence*, 608–620.
- Gebhard, P., Kipp, M., Klesen, M., and Rist, T. 2003. Authoring Scenes for Adaptive, Interactive Performances. In *the Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Victoria, Australia, 725–732.
- Korf, R.E. 1990. Real-time heuristic search, *Artificial Intelligence*, Vol. 42 No. 2-3:189–211.
- Liu, Y., Koenig, S. and Furcy, D. 2002. Speeding Up the Calculation of Heuristics for Heuristic Search-Based Planning. In *the Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 484–491.
- McCloud, S. 1993. Understanding Comics. *Kitchen Sink Press*. Northampton, MA.
- Natkin, S., and Vega, L. 2003. Petri Net Modelling for the Analysis of the Ordering of Actions in Computer Games. In *the Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, London, UK, 82–92.
- Natkin, S., Vega, L., and Grünvogel, S. 2004. A new Methodology for Spatiotemporal Game Design. In: *Mehdi, Q. and Gough, N., (Eds.). Proceedings of CGAIDE 2004, 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education*, University of Wolverhampton, Reading, UK, 109–113.
- Skorupski, J., Jayapalan, L., Marquez, S., and Mateas, M. 2007. Adding Aspects of Implicit Creation to the Authoring Process in Interactive Storytelling. In *the Proceedings of the Fourth International Conference on Virtual Storytelling (ICVS 2007)*, Saint-Malo, France, 26–37.
- Sauer, S., Osswald, K., Wielemans, X., and Stifter, M. 2005. U-Create: Creative Authoring Tools for Edutainment Applications. In *the Proceedings of the third Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006)*, Darmstadt, Germany, 163–168.
- Zagalo, N., Göbel, S., Torres, A., and Malkewitz, R. 2006. INSCAPE: Emotion Expression and Experience in an Authoring Environment. In *the Proceedings of the third Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2006)*, Darmstadt, Germany, 219–230.