journal
creative technologies • creative gaming

# The Ethos Framework: Generating Contextually Linked Dynamic Quests

Jacob Dodunski and Dr. Imran Khaliq

## Abstract

In today's video game market there are hundreds of massively multiplayer online games, many of which boast dynamic content—the content that responds to player's actions and changes accordingly in runtime. Such content is not truly dynamic and is usually attributed to events on a timer. This presents a major flaw as it heavily affects the re-playability of a game. In this paper we examine four major frameworks: the TRUE STORY, Context Aware Petri Net, Hierarchical Generation, and The Grail Frameworks. These frameworks presented some very good ideas but they also suffered from drawbacks. We propose a new framework for creating dynamic content and called it the Ethos Framework. The *Ethos Framework* is based on four major components: Character, World State, Story, and Library, that will contribute towards the creation of dynamic content. The Ethos Framework takes the best pieces of current methods to create a unique practical design so that on every playthrough of a game the player has a genuinely unique experience.

**Keywords:** Video games, quests, generation, dynamic, contextually-linked, ethos

# 1. Introduction

Many games these days follow a strict formula in which you, as the player, play through a level or world by completing tasks to reach the next level. The player progresses through these games by having to master mechanics and learn about the way to play. In many RPG games, quests provide this progression and the games rely heavily on these quests to keep the players interested. Most of these quests are very prescribed in that the pre-scripted events provide similar content each playthrough, which is where many players lose interest. To counter this and provide unique content that differs from player to player a new emerging concept is that of Contextually Linked Dynamic Quests.

## 1.1 What are Contextually Linked Dynamic Quests

We will first define *contextually linked quests*. After that we will define *dynamic quests*. Then using these two definitions we will define *contextually linked dynamic quests*. *Contextually linked quests* are quests that are generated during the game at runtime based on what situation that player is currently in. In other words, a *contextually linked quest* is one in which the problem and solutions are generated based on where the player is, what the player is doing, what the player has done previously, and what is happening in the overarching story, but with a focus on making sure the quest makes logical sense.

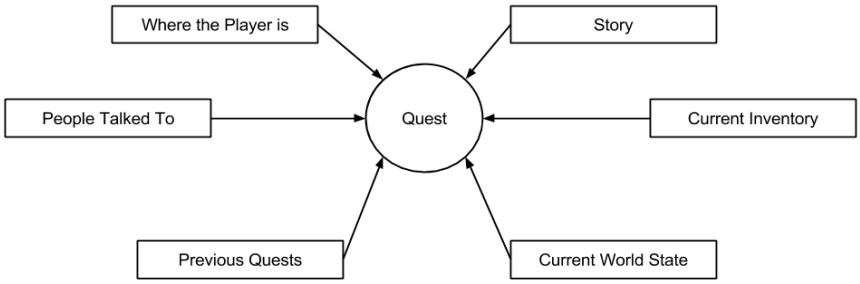


*Figure 1. Contextually Linked Quest Diagram* – Some examples of how a contextually linked quest would be generated based on the state of the current world, the quests the player has done and the people they have met.

*Dynamic quests* are quests which respond to a player's actions and change accordingly. This means that the quest will monitor how the player is doing and change itself based on the player's actions. This can be interpreted in many different ways; our quest has multiple stages to complete with a few different solutions to make different outcomes possible, see Figure 2. This would give the player a feeling that their own actions can change the outcome.

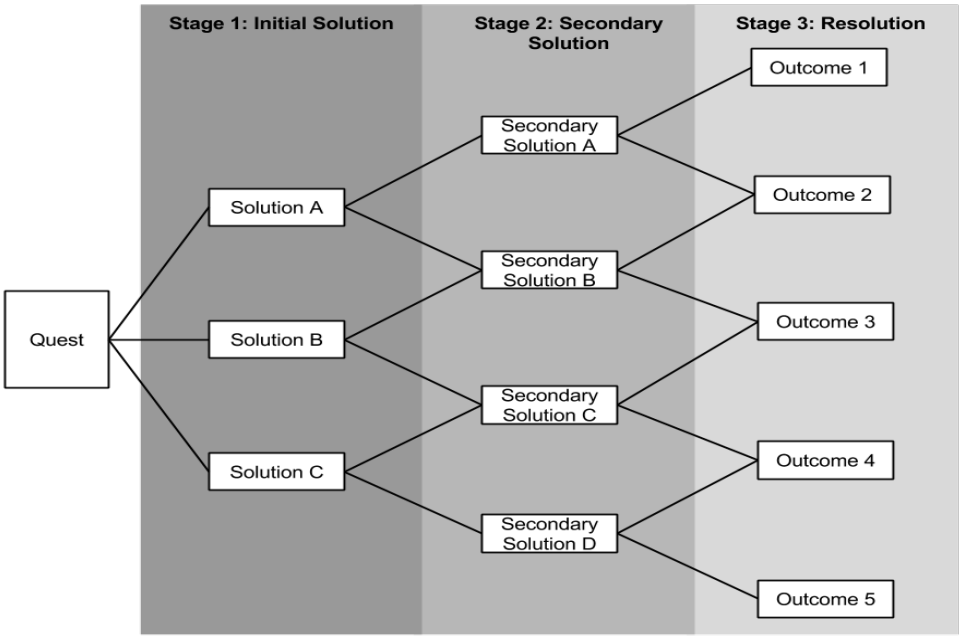


*Figure 2. Dynamic Quest Diagram* – An initial quest with the three stages. The initial solution leads to a choice of secondary solutions which again leads to multiple outcomes in stage 3, the resolutions.

*Contextually linked dynamic quests* are a combination of the above two definitions with the quest being generated at runtime within the game based on the context of the game world (what the player has done, where the player is, the overarching story, etc.), and being dynamic in the sense that the way the player completes the quest changes the outcome of that quest. An example is described in the figure below.

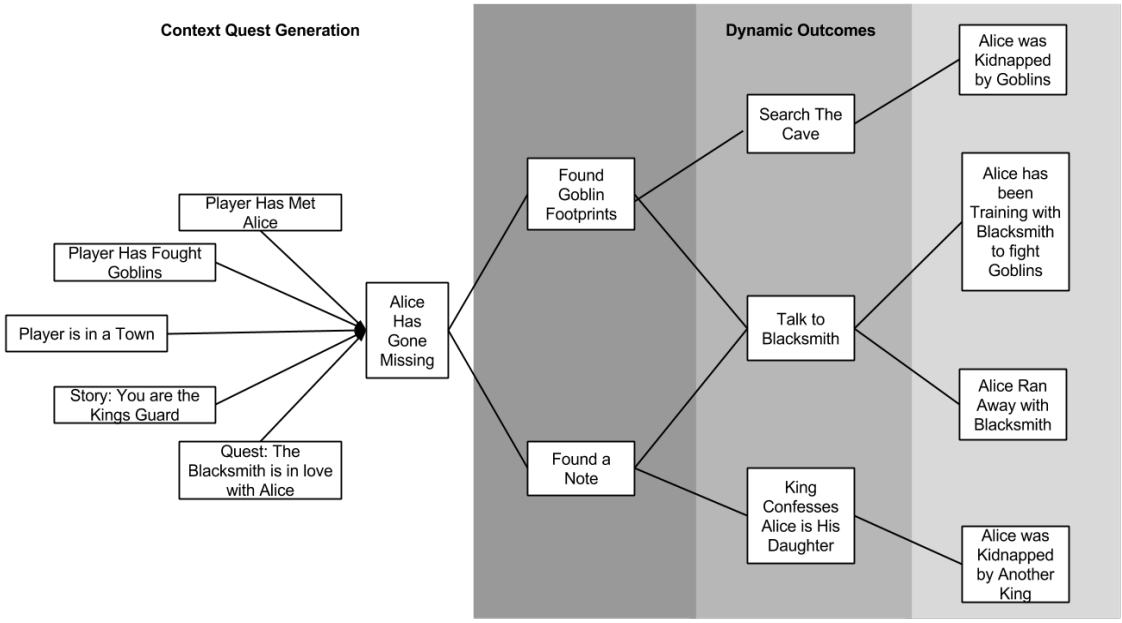


*Figure 3. Contextually Linked Dynamic Quests* – An example of the quest being generated from the contextual cues and its pathway or dynamic outcomes also generated based on the actions the player takes.

In the figure above we can see that the player generates the quest “Alice has Gone Missing” from the context of the current world, such as, the player is within a town, they have met Alice, and the player has fought goblins before. This information is used to create the problem of the quests and various ways to solve them, thus the dynamic outcomes.

### 1.2 Why do players want Contextually Linked Dynamic Quests

Players want to have a personalised experience when they play games, especially if they play them more than once. Players want to have events and quests happen that are a result of their previous interactions so they feel a sense of accomplishment as their actions have shaped the world around them, unlike many games in which this is based very heavily on storytelling.

### 1.3 Current research aim

Our aim is to develop a framework to generate contextually linked dynamic quests so that on every playthrough of a game the player has a unique experience. We examine in detail four major frameworks: the TRUE STORY architecture (Pita, Magerko, & Brodie, 2007), the Context Aware Petri Net (Lee & Cho, 2011), Hierarchical Generation (Lima, Feijo, & Furtado, 2014), and The Grail Framework (Sullivan, 2012). These frameworks presented some very good ideas but they also suffered from drawbacks. We propose a new framework called the Ethos Framework that has the best pieces of the examined frameworks and is free from their drawbacks. Before we explain our own framework, we will discuss the four frameworks in the next section.

## 2. Previous Designs

### 2.1 Case Study: TRUE STORY architecture

The TRUE STORY architecture system presented by Pita et al. (2007) aims to dynamically create contextual quests for the player based on past experiences. The system draws from relationships with objects and characters that have been previously interacted with within the world to try and draw the user into the story and quest by using these as a starting point. These experiences are stored in a list unique to the player and called ‘memories’. Their system works by having every player and NPC (Non-Player Character – an entity within the game that the player can interact with) within a world having a set of memories which are formed based on quests the player has done in the past. Depending on the perspective of the participant involved their memory of the event contains a differing amount of information. The example they used was Character A stealing an item from Character B. Character A retains the full memory of *“Character A stole item x from Character B”* while Character B retains “*Character Unknown stole item x from Character B*”. These memories are used to create quests in the future, such as another example they give that Character C is a lawman and known to Character B, so Character C generates a quests to find the thief of the item which was stolen. The TRUE STORY framework seems very dependent on the relationships between characters and the interactions between these characters to create memories.

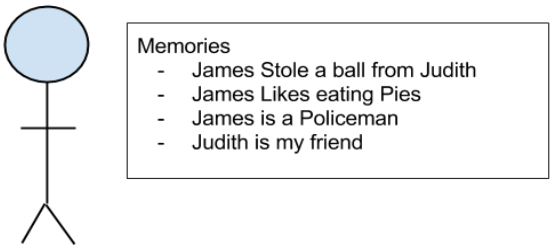


*Figure 4. Memories Example* – An example of how some memories are represented within a person in the TRUE STORY framework.

This design provided what seemed to be a solid base but fell short in implementation as the generation of quests based off experiences only appeared to work on a very rudimentary level, and seems to be due to the design of the implementation rather than the idea itself. The research did find some success in terms of *dynamic quests* as these were providing unique experiences for the player, but could not find a way to contextually link them to create a seamless experience. It also seemed that the quest generation was very limiting, requiring interactions between characters to generate memories for quest down the line.

Pita et al. (2007) provide some very good ideas in terms of creating a unique system. The idea of creating 'memories' unique to the player and using these to generate quests will be examined more in the current research paper when looking at the creation of a new design.

### 2.2 Case Study: Context Aware Petri Net

In the Context Aware Petri Net research, presented by Lee and Cho (2011), we see the use of Bayesian networks to classify the player based on their past actions and playstyle to generate content unique to that player that they will enjoy. Their system worked by figuring out the type of player (Casual or Hardcore) then selecting a quest based on that information. The quest was then generated via a series of events, with characters and locations added to fill in each event. These quests were varied in content and offered many branching opportunities.

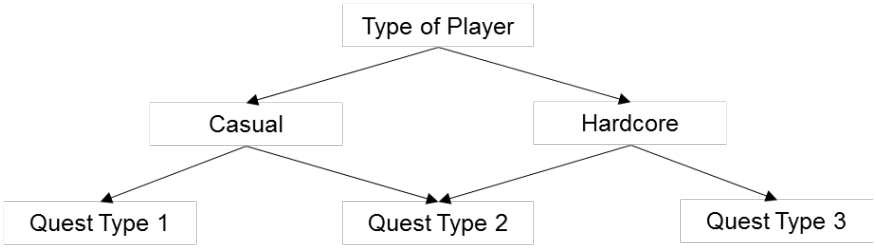


*Figure 5. Petri Net Tree* – An example of how their system works, collecting analytics to classify the player and generating quests based off that classification.

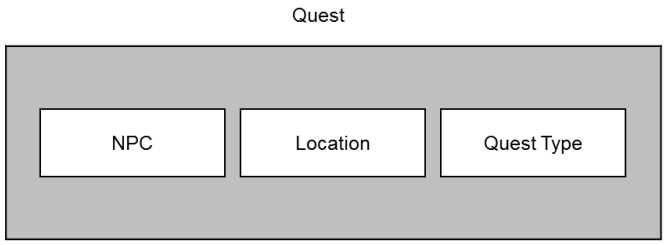


*Figure 6. Quest Library* – A generic dictionary with exchangeable slots which are used to generate the quest.

The idea of this method of procedurally generated quests based on playstyle offered an excellent insight into a design that worked well. Though, due to the fact that no contextual information was saved, the quests act almost as random events. This design fell short in the lack of diversity within the generated quests as there was only a very short list to choose from with very little customisation. Also, within the paper no implementation was proven to work as the game they decided to test this system on did not provide the source code to test their solution. In terms of the design using a Bayesian network to figure out what kind of player the player is, this limits the degree of procedural generation as certain players will only ever generate the same type of quests again and again, whereas removing this classification allows for far greater diversity.

Lee and Cho (2011) provide an excellent design for quest generation through a series of linked events and the current research paper will examine this particular aspect and the benefits of its application.

### 2.3 Case Study: Hierarchical Generation

Hierarchical Generation by Lima et al. (2014) proposes a method of generating *dynamic quests* based on planning, execution and monitoring to handle non-deterministic events and support quests with multiple endings that affect the game's narrative. That is, having a quest planner that selects a quest from a library, generates the events within that quest, and monitors it. That quest has multiple outcomes and can either lead to a following quest or an end in that chain of quests. On the completion or failure of a quest the world state can change. This world state is monitored by the quest monitor which communicates with the quest planner so a new quest can be generated for the situation.

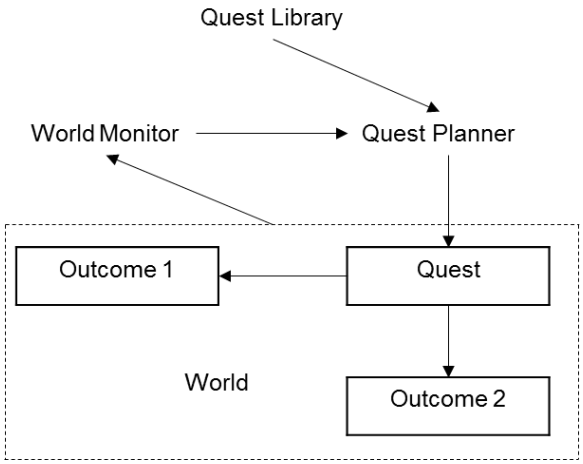


*Figure 7. Hierarchical Generation Planner* – The Quest Library contains a dictionary of quests that are passed to the quest planner. The quest planner receives information from the world monitor about the state of the world and uses the dictionary to generate a new quest which is placed in the world which modifies the world state. The player then interacts with the quest to generate a new world state.

Lima et al.'s (2014) system seems to be a good solution but, as seen previously, does not provide enough customisation of the quests. Although they are geared to having multiple outcomes, with one action leading to another smaller quest in which the outcome affects the larger quest, they are still predetermined and offer no real customisation, which, as mentioned above in the Aware Petri Net, limits the amount of things the player can do and thus provides repeated quests. The world state monitor seems like an excellent idea as it can respond not only to the completion of other quests but also overarching events on a grand scale, such as things that only happen at certain times of the day or respond to an overabundance of items within a shop or auction house; the possibilities are endless.

We will be expanding the idea of world state monitoring in our own framework in Section 3.

### 2.4 Case Study: The Grail Framework

The Grail Framework by Sullivan (2012) introduces a concept that games should develop by providing four different levels of storytelling: player, quest, game, and world. Each of these four levels work together in different ways to change the outcome of the story within the game. The framework in the generation of stories is built up around social mechanics. These social mechanics are very ingrained in how you interact with NPC's (Non-Playable Characters). They each have their own personalities and unique lines of dialogue, that is, each character has an internal set of different values affecting how they interact with the player. These values can increase or decrease depending on the player's interactions with the character, other characters, the world or really anything. Increasing or decreasing these values may open up new dialogues, quests or events. Also, depending on these values or specific things the player has done, the characters can gain or lose 'traits', specific words that can act as flags for future interactions and other quest, such as 'is dating', 'alcoholic' and 'angry'. These interactions are stored in a database and are used in

other quests, either as triggers to unlock certain things or in main plot points which is what I will look at now.

| | |
|---|---|
| Name: | El Rupertos |
| Traits: | Angry, Alcoholic |
| Relationships: | Dating Mary |
| Levels: | Friendship: 20 |
| Special: | Becomes 'Annoyed' if player talks to Mary |

*Figure 8.* Example of a NPC internal values.

### 2.4.1 Story Quest Generation

It is explained that story based plot points and quests are created from two elements, content and rules, and the framework combines these with the relationships that the player has built up with NPC's. This way you have the overarching story point which is changed dynamically based on the NPCs the player has interacted with. This leads to different situations emerging with the player, such as if the player has annoyed all of the NPC's he might not receive any help when fighting the dragon in contrast with if he had helped out. This also leads to different completion states of the quests and plot points which can be taken forward in the story. Plot point events are also not placed in fixed locations or triggered by the player accomplishing events, but rather chosen at runtime and are revealed during the dialogue as part of social integration.

The Grail system also locks the player into performing only one quest at a time which allows the player to focus on the single quest rather than getting swamped with hundreds of quests at once.

### 2.4.2 Contextual Quest Generation

In terms of the actual quest generation, the framework uses a 'brainstormer' to generate relationships between the quest concepts, environment, problem, reason, and solution. For example, the environment 'forest' is related to 'birds' and these create 'music' which is a 'song' and we have a quest which is 'the birds within the forest have stopped singing'. Then, via the brainstormer through association, we find that birds are afraid of cats, providing a reason. Through the same process you find that cats are afraid of water and love catnip so the framework can generate multiple solutions, for example *kill x number of cats*, *create a moat around tree and trap them*, *create catnip to befriend cats*, etc.

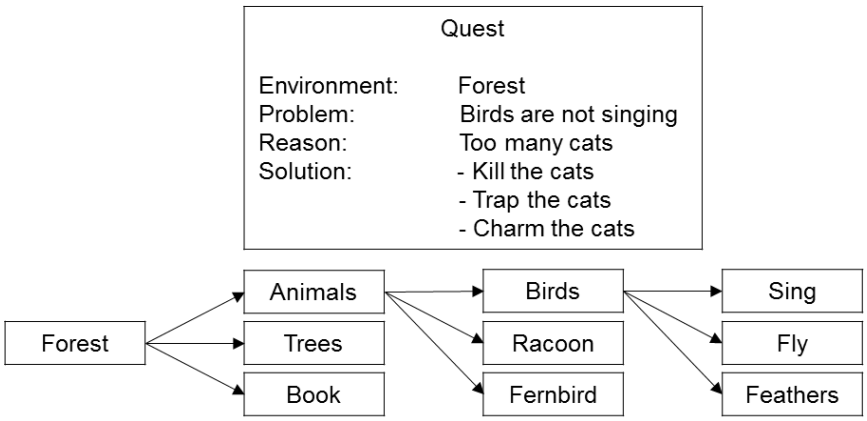


*Figure 9.* An example of the contextual quest generation tool in the Grail Framework. For generating a quest within the forest, the forest is contextually linked to animals which is linked to birds which is linked to singing.

The paper Sullivan, A. (2012) explores some complicated algorithms to create *dynamic quests* with an incredible social system to help drive the story forward.

## 3. The Ethos Framework

Now we will explain our own framework, the *Ethos Framework*. The *Ethos Framework* aims to generate *contextually linked dynamic quests*, quests that are unique depending on the situation and can be solved in a number of different ways. It aims to take inspiration from the four reviewed papers and take the parts that worked the best and combine them to create a framework which surpasses the rest.

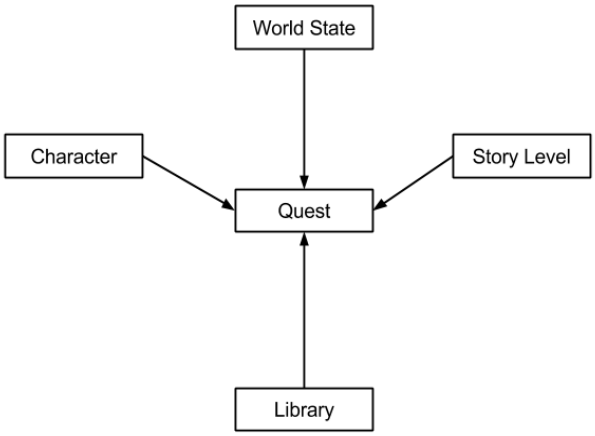


*Figure 10. The Ethos Framework Components* – The four major components of the Ethos Framework funnelling into the quest for generation.

The Ethos Framework is split into four major components that will contribute toward the generation of a *contextually linked dynamic quest*.

**Character:** The character component encompasses all social relationships that the player's character has with other NPC's as well as NPC's relationships with one another. This component places traits, relationships, memories, and triggers on all the characters. That is, the traits will encompass the personality of the person, such as 'cheerful' or 'alcoholic'. These traits are not fixed but will vary with the current

state of the world and interactions with the player. Relationships encompasses any relationship the NPC has with other NPC's or the player, and will be represented as a value, such as a quantifiable friendship level or closeness or hatred. This value will act to open up different dialogues and allows this character to be placed in certain quests (as will be seen later). Memories will be a record of previous interactions with the player and could be used in quest/dialogue generation. Triggers will be certain programmed tricks that will apply to the character based on events within the world or actions the player has done. These could be as small as changing a friendship value or removing/adding a trait to the character. Character is a crucial feature of our framework, as the player progresses through the story of a game these relationships will evolve and, for example, an NPC which hates the player may turn up again and again in the procedural quests and could eventually be seen as the player's nemesis and may be promoted to take a role in more story based events.

**World State:** The world state component refers to the current state of the world the player is in, that is, what area the player is currently in, such as a cave or a forest, what quests the player has done and how they were completed, how many of a certain type of monster the player has killed, how much money the player has spent at stores, and what the player currently has in their inventory, etc. This component is very large and ideally would monitor almost everything so that quests can be generated off of this information. This component is important because it can generate the small details in quests and customise dialogue so that it relates specifically to the player. An example would be: after killing lots of chickens outside a town, the town might have a chicken shortage and a quest may appear to help increase the chicken population within the area.

**Story:** The story component is the overarching designed story of the game itself which will contain big predetermined story events as embedded narratives of Salen and Zimmerman (2004). Basically, at what point the player is in the game will determine many things. This could be encompassed into the generation of quests by determining NPCs' attitudes towards the player. For example, whether the NPCs saved a town in a previous story event, would cause a ripple effect on NPCs initial relationship values or even if the NPCs exist at all.

**Library:** The library component is essentially a blank divergent map that presents a generic problem and ways to solve it. Each problem and its possible solutions will have specific slots they need to fill in the generation of a quest. The library templates will be multi staged, that is, there will be a few different 'mini-quests' to do to complete the quest. This is to provide more diverse content, to make it more dynamic, and to be able to tell a small story within the quest that relates to the larger overarching story so that the player will feel more involved in the game world. Below shows a very basic example.

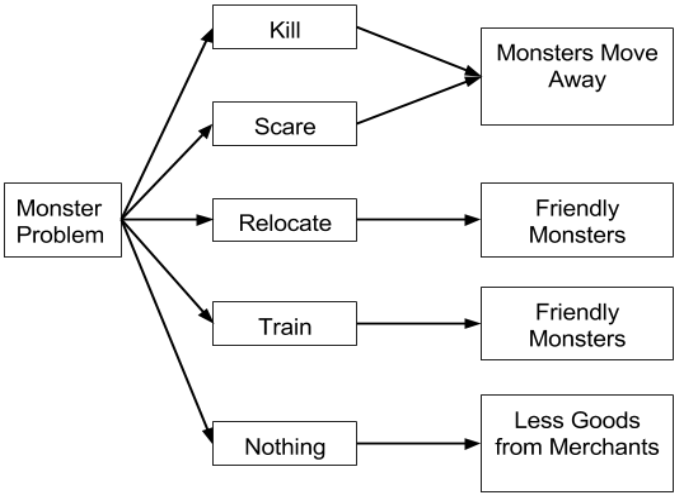


*Figure 11. Basic Library Quest Component* – The 'dynamic' part of the quest generation with a generic problem, a few different ways to solve it, and the outcome of those choices.

As we can see in Figure 11, the component contains a collection of these blank templates for the quest generation. In this one template alone there are five different options and four very different outcomes. This has increased the diversity of simple quests immensely already. The figure below shows the specific slots that are within the library templates.

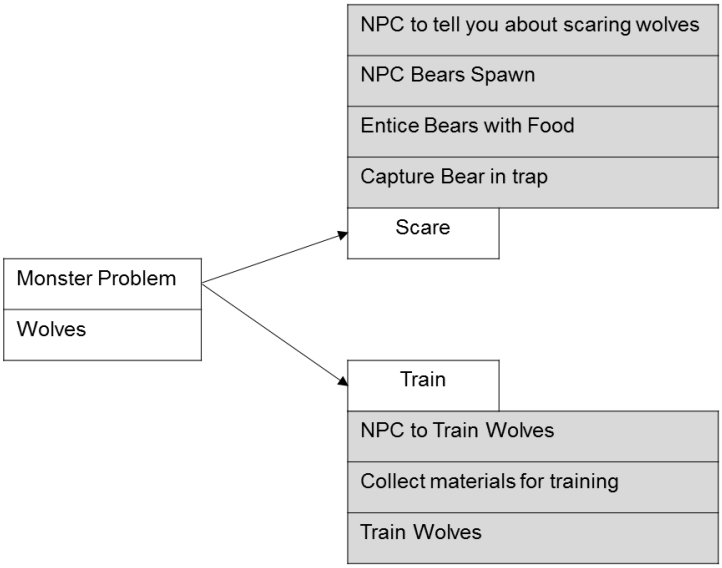


*Figure 12. Library Template Specifics* – An example with wolves showing two template choices and the slots that need to be filled.

Figure 12 shows that for the option scare the slots that need to be filled are an NPC who knows about what wolves are scared of (possibly having the trait 'Wolf Trainer'), that an NPC Bear needs to be spawned somewhere within the area, that there needs to be a specific item that bears like, and you need a trap to capture the bear.

## 3.1 Quest Generation

The quest itself will be generated using all the information passed to it by the character, world state, and story to fill in slots in a library file using a contextually linked generator. This is very similar to the tool described in Section 2.4.2. All objects within the world will be tagged with certain traits so that the word association, in conjunction with the contextual linking, can work efficiently. An example of the contextually linked information generating part of the *dynamic quest* can be seen in the figure below.

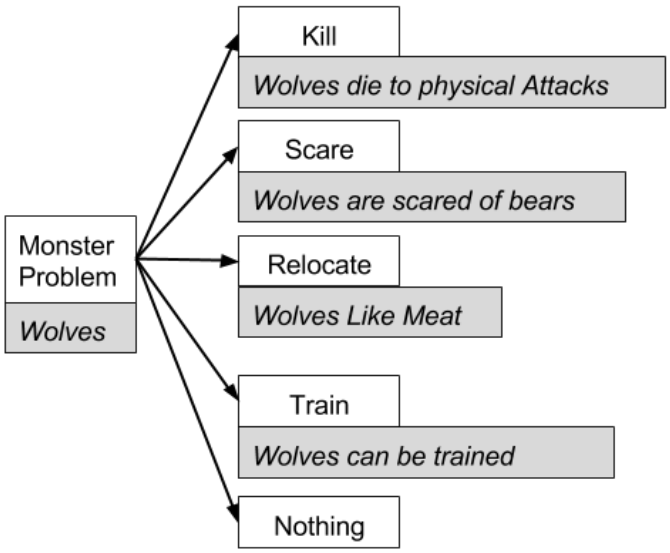


*Figure 13. Contextual Dynamic Quest Generation* – In the master problem wolves, the contextual word association tool shows that wolves die to physical attacks, are scared of bears, they like to eat meat, and they can be trained.

## 3.2 Benefits

Having four different components feed into one to generate the quest means you can very easily customise what, and how much, information is taken from each component. This allows the framework to be easily customisable, so if you want to focus more on social aspects you can modify the framework to draw heavily from this component and less on the world state and vice versa.

The library-template system allows new content to easily be created and slotted in at any time, as it will work with all other current information. As you are only providing a template and simple instructions, the quest generator itself just has to fill in the blanks with the information it has, and suddenly you have a range of brand new quest types. This approach would be useful to online games, for example, massively multiplayer online role playing games, because uploading a template to internet to get a game with a new set of quests is less expensive than uploading a whole new version of the game.

By using a multistage library system, with many different ways to complete quests and many different outcomes, we can provide a much greater diversity in quests. This also enables us to provide small stories within the quests so they link together.

Due to the monitoring of the world state many small events are taken into account and can provide anything from a reason for a quest to small changes in dialogue. This will help players get more engrossed in the game, especially if the dialogue is tailored specifically for their character and doesn't feel disjointed.

The Ethos Framework aims to improve upon the aforementioned frameworks by creating a complicated system that monitors many different elements of the game and brings them together to help create unique quests for the player every time.

# 4. Discussion

## 4.1 A Comparison

### 4.1.1 TRUE STORY Architecture

The Ethos Framework improves on the TRUE STORY Architecture by extending the 'memory' system of the character component to include traits and relationship levels, as well as a record of past interactions with the player's character, for each individual NPC.

### 4.1.2 Context Aware Petri Net

The Ethos Framework improves on the dictionary aspect of the Context Aware Petri Net, the base used to supply elements to quests. The original was very basic, simply allowing for a few different elements to be slotted in to create a unique quest. The Ethos Framework improves this by making the quest multi-staged and dynamic, with multiple solutions and outcomes, turning a single quest into a small quest line or series of smaller quests to tell a small story. This enables more storytelling, more involvement from the player, and much more variation in quests. Apart from the dictionary function, the Ethos Framework does not include any other aspects of the Context Aware Petri Net. The Petri Net is focused more on analytics rather than interactions in the game and essentially restricts the player to only a subset of quests, and this goes against the aim of the current paper.

### 4.1.3 Hierarchical Generation

The Ethos Framework adopted the idea of the world monitor from the hierarchical generation implementation, as well as the combining of the quest library. The Ethos Framework expanded on the library by providing a much more in depth library system with multi-stage quests that form quest lines, where you play out a small story that is connected to the larger overarching story of the game. The Ethos Framework also took the idea of the world monitor, combining it with three other modules to create a new quest, rather than just combining it with the library. This way the components all contribute towards the generation of a quest, increasing diversity of the quest exponentially, and providing a much more unique experience than what could have been experienced with hierarchical generation.

### 4.1.4 The Grail Framework

The Grail Framework is an excellent implementation of dynamic quest generation and a fantastic step forward in terms of achieving this goal. The Ethos Framework set out to expand many of the ideas presented within the Grail Framework, specifically the idea of multiple layers. The Grail Framework receives information from four major components to determine how a quest should be generated; player, story, characters, and world. The Ethos Framework simplifies this information and provides a much more in-depth world monitor and a vastly different library system in which the multi-stage quest, outcomes and solutions provide a far more diverse experience.

## 4.2 Conclusion

The Ethos Framework presents a viable interpretation of a way to effectively create *contextually linked dynamic quests* especially in role playing games (RPGs), and, upon review against the examined previous implementations, solves many of their problems to create a new framework absent of their deficits. This framework provides an exciting new opportunity for future games to adopt and provide players with truly *dynamic quests* and games that really do provide a unique experience every time you play. Finally, to prove the Ethos Framework is viable, an implementation and testing of the framework needs to be performed. This is a separate project of its own and has been left for future research. For implementation purposes of the *contextually linked dynamic quests*, one straightforward way is to implement it using a set of Boolean requirements, as once the requirements are met, the corresponding quest would be unlocked. For better implementation, we will recommend to use fuzzy logic in order to give virtually infinite versatility to quests.

## 5. References

Lee, Y.S., & Cho, S. B. (2011*)* Context-Aware Petri Net for Dynamic Procedural Content Generation in Role-Playing Game, *IEEE Computational Intelligence Magazine, 6,* 16-25.

Lima, E.S., Feijo, B. & Furtado, A.L. (2014). Hierarchical generation of dynamic and nondeterministic quests in games, In Proceedings of the 11th Conference on Advances in Computer Entertainment Technology (ACE '14). ACM, New York, NY, USA, 24 , 1-10. doi:10.1145/2663806.2663833

Pita, J., Magerko, B. & Brodie, S. (2007) TRUE STORY: Contextually Linked Quests in Persistent Systems*, Proc. Conf. Future Play*, 145-151

Sullivan, A. (2012) The Grail Framework: Making Stories Playable on Three Levels in CRPGs. *PhD thesis*, University of California Santa Cruz.

Salen, K., & Zimmerman, E. (2004) Rules of Play: Game Design Fundamentals. The MIT Press, Cambridge.

## 6. List Of Figures



*All figures are my own and have been created as a supplement to the text to explain the ideas presented.*

# About the Authors

## Jacob Dodunski

Jacob Dodunski is a current student of Media Design School in New Zealand studying game programming. He has postgraduate degrees in neuroscience, computer science and has a passion for games.

**Contact:** Jakedodunski@gmail.com

## Dr. Imran Khaliq

Dr. Imran Khaliq has a PhD in Computer Science from the University of Auckland. Dr. Khaliq's research focuses on Algorithmic Games played on finite graphs. He is passionate about investigating, teaching and playing games. Currently, he is a Lecturer at Media Design School, Auckland.

**Contact:** imran.khaliq@mediadesignchool.com

**The Journal of Creative Technologies (JCT)**
https://ctechjournal.aut.ac.nz

ISSN: 2230-2115

Colab, Auckland University of Technology, New Zealand


*The Journal of Creative Technologies (JCT)* is an online, open access, peer-reviewed journal for the publication of research and innovation about new technologies, creative practices, and critical theories. The journal aims to explore applied, methodological and theoretical perspectives on emergent technological platforms and strategies through thematically focused issues.
*JCT* is a research communication platform published by Colab at the Faculty of Design and Creative Technologies, Auckland University of Technology.