How Will a Mixed-Initiative Level Editor that Predicts User Requirements Affect the Levels Created?

Tristan Barlow-Griffin - 1607804

***Abstract*—Mixed-initiative (MI) systems are becoming more common both in and outside of the games industry. These MI systems are taking an ever-increasing number of roles. These roles range from generating entire levels, to fostering creativity in game designers. With the increasing cost of game development, having a tool that can help to quickly and effectively prototype levels will reduce the costs of development significantly. This paper explores some of the effects that introducing an MI component may have to a level editor. To identify what effects to look for, this paper uses existing research into predictive texting and MI level editors. From this research, five hypotheses are proposed. To test the hypotheses, this paper proposes an experiment where the same group of users create different levels with and without the MI component present. The results demonstrate that the MI component caused an increase to the time the participants spent on a level, an increase in the number of objects in the level and an increase to the number of clicks made to create the level. The results also suggest that the MI component influences participants to explore alternative level designs.**

I. Introduction

THIS research project looks into how a prototyping tool with a Mixed-Initiative (MI) component that predicts user requirements will affect the process and design of a level. A prototype is the initial design of an object [1], the prototyping phase is used to quickly test certain aspects of a products' design so the designer can identify and solve any problems before full production commences [2]. Fullerton *et all* [3, p. 150] state there are two kinds of prototyping in games: Physical and Software prototypes. Since the publication of Fullerton's*et al* [3] book in 2004, the accessibility of tools that help designers prototype levels has increased. Fullerton *et al* [3, p. 164] also describes a level editor as a good way to prototype levels. The free to download game engine called *Unreal Engine 4* (UE4) has a level editor built into it. Within this editor, the designers can create basic geometry scaling them to fit their needs. In addition, designers can add custom meshes and programmable objects to turn their levels into games.

The prototyping phase is meant to test the design of a product, the less time and resources required to produce an artefact that can demonstrate the proposed design, the better. It could be argued the less time a designer puts into a particular design the less attached to the design they become. When collaborating in a group, differing opinions can cause different constraints to be set on the design of a product. While a given design may satisfy the original designers set constraints, the prototype may have to be discarded as it did not meet the other requirements set by the team. Identifying and discarding concepts early in development can save a lot of time and energy [4, p.489]. Arguably, this will reduce the negative impacts to interpersonal relations that idea dismissal may have.

In this paper, we propose a level editor that is meant to only prototype a levels' design. This will allow for less focus on the polish of the level created as the aim of the prototype is just to test if it fits the users' requirements. This paper will build upon a standard level editor by adding an MI component. The component implements features requested from Alvarez *et al* [5] study on their MI level editor. The implementation of these features is meant to increase the overall ease of designing a level by predicting what the user may require. In addition, this paper looks at the Sentient Sketchpad [6] an existing MI level editor, comparing their design to MI tool design theory and general user interface (UI) design theory. The hypotheses in this paper are only concerned with how the MI component interacts with the participants and what kind of impact the interactions will have when trying to prototype a level. Looking at existing studies on the most effective prediction methods [7]–[9] this paper identifies and evaluates the methods proposed in the context of an MI level editor.

II. Related Work

The lack of documentation about prediction methods being used in game design required this paper to go beyond the scope of just game design examples. The methods of prediction used in this study must be effective for the hypotheses to be tested accurately. To this end, Section V looks at currently used prediction methods comparing the performance results of each method. In Section III, MI tools are grouped into two broad categories using Liapis *et al* [10] definition. The definition of MI used in this paper will also be defined in Section III. Defining these groups makes it easier to distinguish between the most common types of MI tools. This section will also explore the different ways MI tools are being leveraged both within and outside of the games industry. In Section IV there is an evaluation of the existing MI level editors. Section IV also contains a discussion and comparison of these MI editors in the context of more general MI and UI design theory.

III. Mixed-initiative

The term mixed-initiative was first introduced in *1970* by Jaime R [11]. It describes a process whereby a computer

and a human designer work together to achieve a goal. Other definitions of MI build on the idea of human-computer co-creativity. This paper will use the first definition of MI presented, as trying to define creativity is a very complex matter in itself.

There are two broad categories that MI tools can be grouped into, Interactive evolution and Computer-aided design [10].

- *Interactive evolution*(IE) is where the designer has the idea, and the computer helps them realise it. The computers' role is to evaluate the humans' design, presenting alternative solutions if any constraints are broken.
- *Computer-aided design* (CAD) is where the computer generates the content, but does not evaluate the quality of the produced work. In CAD, the human designer will evaluate the work and use these evaluations to move towards a more desirable product space.

The first documented MI tool created helped students to learn the English language [11]. The uses of MI tools have greatly expanded since 1970 and have been described as a backbone tool for designers [5]. The application of MI systems in more complex environments have given mixed results [12]. Barnes *et al* [12] found that in most cases, systems that divide decision making between a human and an intelligent agent were generally more effective than if decisions were just dependent on the one. This can be seen with predictive texting. Only choosing the words suggested to you by your phone can result in unexpected and hilarious results [13]. Barnes *et al* [12] found that human designers were better at making abstract decisions and inferring the significance of an object or event. Kantosalo *et al* [14] focus their MI tool on user-centred design, this means their AI agent had less responsibility than the human user. Kantosalo *et al* [14] propose future work where the agent and the designer have an equal role in the system, they do not make any conjecture on the anticipated results.

Procedural content generation(PCG) and CAD are closely related fields, the difference between CAD and PCG is in the evaluation period. If a designer were to use a PCG algorithm to generate a level and then look through the produced results, evaluating and picking maps this would be considered CAD [10]. For it not to be considered CAD, the PCG algorithm would have to perform the evaluation itself. For example, this might include checking if the map is completable or if it is a certain size. The field of PCG has advanced significantly in recent years [15]. The uses of PCG are ever increasing as publishers seek to lower costs of production [16], [17].

One may choose to use PCG in games as it will increase the quantity and variation of levels produced ensuring replayability [18]. PCG algorithms can also be shipped with the games they are made for, this allows for an inexhaustible source of new maps [19]. Doing so will extend the games life-span giving the players an amount of content that would otherwise be impossible. Although there is no guarantee that this content will be interesting or unique. However, using CAD tools to generate maps there will always require a human element to evaluate the maps. With this input the human designer may discard maps that are similar to existing maps, thus providing quality assurance not present in a PCG algorithm.

Prototyping a level should be a fast iterative process [20], the method used to prototype should allow for instant feedback on the design. This feedback needs to have an easy channel back to the designer so that amendments can be made easily. Map generation algorithms, even in under ideal circumstances will only provide the designer with a set of parameters to change [21]. Small variations in the given parameters can produce large changes in the maps design [22]. This will make it difficult to make small changes when given feedback. As a result, this reduces the effectiveness of a CAD approach when trying to prototype levels.

Within an IE environment, the core of the creative process relies on the human designer. As the main creative driver, the human has the most input. Ideally, the computer will add value by providing supplementary support, for example, alerting the designer when constraints have been broken. It can be argued as the constraints are determined more by the human, the size of the possible output space will be larger. Yanakakis *et al* [23] claim that CAD examples like PCG limit the designers' intentions as they follow their own algorithms. Doran and Parberry [21] inadvertently corroborate Yanakaki*et al* [23] view, as they say, a PCG algorithm should ideally, have a set of designer-centric parameters as their only form of control. This limited control over the creation of levels may, in some situations, be enough. The user requirements of a map change most often during the prototyping phase. The level of control provided through CAD might not be enough to handle the necessary changes.

IV. Mixed-Initiative Level Editors

A user interface should be intuitive to use and not require any additional helping systems [24]. Liapis *et al* [6] aim to achieve this by creating a design tool that allows users to create levels using a low-resolution graphical interface. Figure 1 shows the interface used to design levels for a strategy game. The designer can place tiles on the map which will colour in the given tile with the placed tiles colour. During a tile placement, the tool will test for the playability of the map, checking to see if placing the current tile will break any of the games' constraints. The Sentient sketchbook also provides alternative viewing modes, examples of the viewing modes can be seen in Figure 2. There is no mention in the study how frequently these tools were used, Galtiz [25, p. 752] warns against too many graphics on the screen.

Baldwin *et al* [26] have also implemented an MI dungeon designer called the evolutionary dungeon designer or (EDD) for short. EDD is closer to a CAD tool than the IE tool Sentient Sketchbook [6]. While the approaches may be different see Figure 3, both MI tools allow for large customisation of the levels generated see Figure 3. Baldwin *et al* [26] take a different approach to Liapis *et al* [6]. Baldwin *et al* [26] core concept is to identify design patterns within the level design. These design patterns consist of multiple tiles that constitute common patterns found in games. Alvarez *et al* [5] builds upon the EDD suggested in [26] by adding an

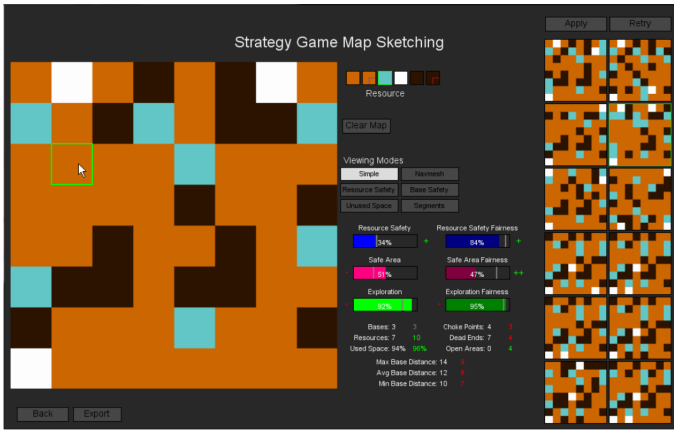

Fig. 1. Liapis *et al* Sentient Sketchbook during a design session [6].

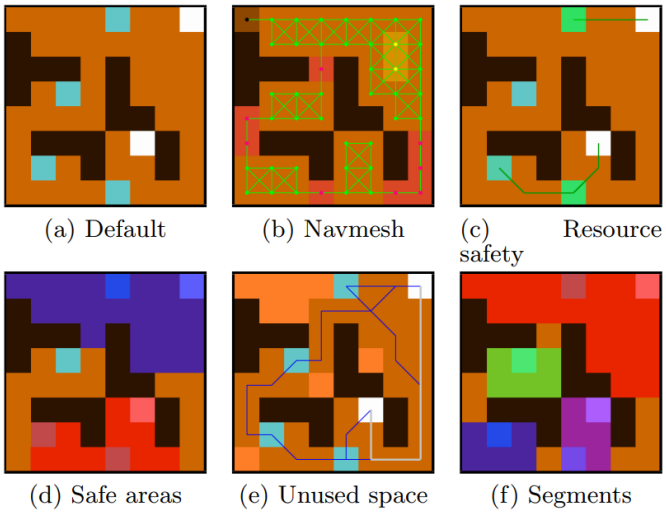

Fig. 2. Liapis *et al* Sentient Sketchbook Different viewing modes [6].

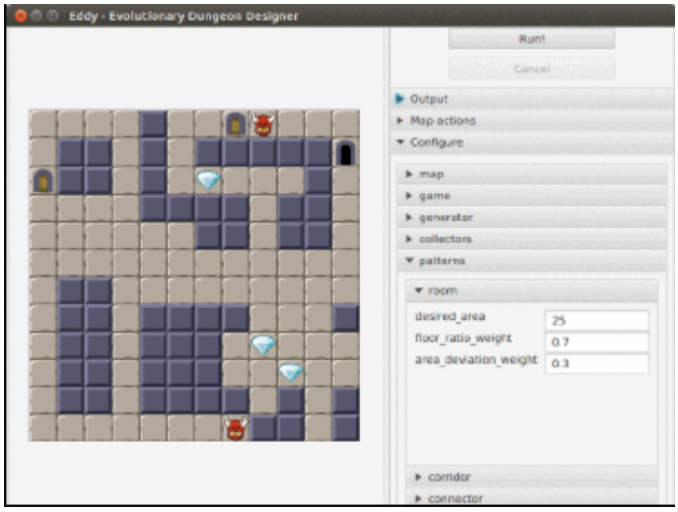

Fig. 3. Baldwin *et al* Evolutionary Dungeon Designer user interface [26].

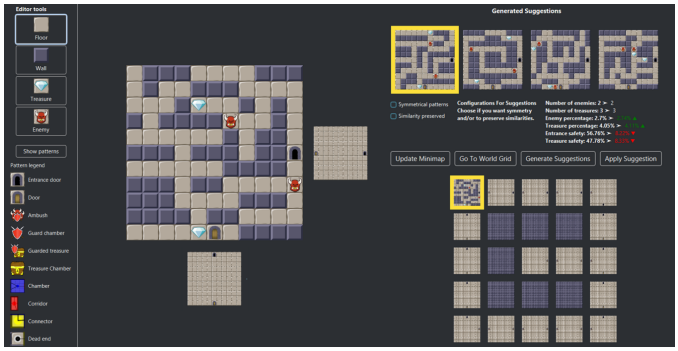

Fig. 4. Alvarez *et al* Evolutionary Dungeon Designer with modifications user interface [5].

IE element to it. It could be argued that the new version of the EDD has an improved interface with a reduction in excess drop-down menus, see Figure 4. Beyond the aesthetic differences, Alvarez *et al* [5] dungeon designer integrates key aspects of the Sentient Sketchpad [6]. The second edition of the EDD allows the user to design their own levels, it will then offer suggestions based upon the map the user created, this can be seen in the top right corner of figure 4. The results from Alvarez *et al* [5] experiment focused on whether their tool fostered creativity in the participants using it. Included with the results is a table of requested features made by the participants of the study. One key element highlighted is that the EDD should do a "bit more automated assistance when doing manual designs, which can reduce clicking around the program" [5, Table 2]. Another significant feature request was for the dungeon designer to take into account the pattern of the entire dungeon, using the map of the dungeon to generate new rooms.

Horvitz [27] proposes 12 critical factors to take into consideration when making an MI user interface. While Horvitz [27] focuses on an MI assistant for Microsoft Outlook (emailing software) it can be argued that some of these factors are relevant for a level designer. The first factor that is listed is that an MI tool needs to add significant value through the automation of services. An example of a service automated by emailing assistant is the sorting of a users' emails into different categories. Within the context of [6] they satisfy [27] first critical factor by allowing the computer to automate some of the map design services like checking for broken game constraints. Liapis *et al* [6] also allows their algorithm to take on a creative role albeit based on an original human designed map. Within this project, the focus will not be on the creative aspect as the definition of creativity is hard to for a computer to understand [28]. Alvarez *et al* [5] found their MI tool is better at providing controllability than expressivity, when the user imposes their vision, as it is hard for a computer to capture the designers' vision. It can be argued that an MI tool could not consistently add value if it cannot capture the designers' vision. Smith *et al* [20] believe that human designers strengths lay in creativity and their ability to evaluate good content and which the computer lacks.

Another factor raised by Horvitz [27] is that a tool must consider minimizing the costs of poor guesses about the users' goals. Even with an extensive history of the users' requirements, novel use of the tool might be required. It is important for a system to recognise when something novel

is happening and for it not to attempt predictions. Some authors find value in these missed guesses and even seek to find novelty search spaces [6]. Other authors [5], [23], [29] claim these kinds of mistakes can foster creativity and alternate suggestions that do not aim to predict the user can be beneficial. On the right-hand side of Figure 1 the results of the guessing algorithm are shown. Clicking retry will remove the current maps and create new ones for the designer to evaluate.

None of the above examples [5], [6], [26] satisfy Barnes *et al* [12] statement that an MI systems UI must provide transparency to the reasoning behind the agents' actions. In all cases given above [5], [6], [26] , when generating new suggestions the reasoning behind each suggestion was not given to the designer. The designer is presented with the statistics of the current map generated (density, number of resources etc.) but it is not clear from the interface how these statistics are being used. Lee and See [30] agrees with Barnes *et al* [12] that MI systems should be transparent, they go so far as to say it is required for a human to trust the automation process [30].

V. Prediction Methods

Predictive texting increases the average message length users send to each other [31] as well as the speed the words are written [32]. The same theory may apply to game design. If a users' game design patterns are learnt, an AI system may be able to assist in the design process. In this section, this paper will look into different methods of predicting human requirements, this paper will also discuss the pros and cons of each technique.

Chipalkatty *et al* [33] have tested alternate methods for predicting human input so as to abstract the low-level movements of the robots the humans were controlling. They built on the idea that humans are good for high-level abstract tasks, but an AI agent was much better at performing low-level repetitive control tasks. In addition, they found that when trying to predict the next human input, trying to identify patterns in a history of events was not successful. Instead, using just the last event yielded much better results, hence their title "less is more". Instead of using current human inputs, [34] used the history of the users' social media page to predict the users' interests. Perhaps if the authors of [33] had looked less at the input history of the human and instead focused on grouping inputs together to create larger actions. Similar, to how modern day phones often predict entire sentences rather than just single words.

A. Markov Chains

Markov Chains implements a theory similar to the methods previously discussed by Bhatia and Hasija [34]. A Markov chain is a special kind of process that works under the assumption that the state at time *t+1* depends only on the current state. Another explanation is the state at time *t+1* is exclusively dependent on the state at time *t*. This means State *t+1* is not dependent on the history of the states leading up to it [35]. This technique has been proven useful for predicting anomalies in systems [35]–[37] where the states are heavily dependent on the latter state happening. However, it is hard to see how during a creative process, where the next state is dependent on the vision of the designer, the state at *t+1* can be predicted from just knowing the state at *t*. The introduction of a Markov model makes Markov Chains method more viable as a prediction technique. A Markov model can be used to describe the probabilistic relationship between the previous states in a Markov Chain [38]. Higher order Markov chains relax the rule of the next state being only dependent on the current state by allowing the network structure to look *n* number of states back from the current state [39]. Snodgrass and Ontanon [40] used Markov models as a way to model level data. Using these Markov models Snodgrass *et al* [40] applied different sampling techniques, they found using their higher order Markov chains to be more effective than using standard Markov chains. This conflicts with Chipalkatty *et al* [33] findings of "less is more", Snodgrass and Ontanon [40] found that using a history of states to predict the next state was more effective to achieve their goals.

B. Artifical Neural Network

An artificial neural network (ANN) consists of many nodes that are divided into separate layers. Each node receives inputs from other nodes, the value of these inputs depends on the weight of the connection between the nodes [41]. ANN are used in prediction methods focusing on outputting numerical values [41], [42]. Shepperd and Kadoda [7] found that case-based reasoning(CBR) outperformed an ANN. However, Shepperd and Kadoda [7] highlighted the dependence of the respective techniques on the nature of the training sets used. Figure 5 shows stepwise regression procedure (SWR) proving to be more accurate at prediction than both CBR and ANN on small datasets. A time series approach to neural networks has been found to increase their accuracy when dealing with world state predictions [43]. The number of data points in the training set that Hazarika and Lowe [43] used was much larger than in [7] at 500 points for training and 250 for validation. Looking at examples from [5], [6], [26] the maximum level dimensions are 12 tiles wide by 12 tiles tall. This means that the maximum number of data points provided by one map would be 124, which is far less than Hazarika and Lowe [43] use to train their ANN.

C. Case-based Reasoning and Stepwise Regression procedure

Case-based reasoning (CBR) is a prediction technique that uses a history of completed products to predict solutions to problems in the current product. This is done by comparing features in the current problem description to problems already solved in existing projects. Usually, the problem description that is most similar to the current problem is used to estimate the solution [8], [44]. The core of a case-based reasoner is solving new problems by using or adapting solutions from old problems [45, p. 1]. Watson [46] describes CBR as a methodology, not a technique and provides several different techniques to apply CBR. Mendes and Mosley [8] have compared some of the techniques found in [46] as well as additional techniques. They found that using a CBR technique

	Small training set (20)				Large training set (100)			
Dataset	SWR	RI	CBR	ANN	SWR	RI	CBR	ANN
Normal	9.90	23.00	20.03	15.25	9.77	13.41	17.90	38.96
	10.32	20.70	22.14	17.63	9.31	15.67	17.80	9.09
Normal + outliers	36.57	166.68	205.04	162.39	51.33	96.16	37.62	28.12
	63.95	256.05	57.64	160.30	40.19	88.11	34.39	32.57
Normal + multicolli nearity	11.11	27.95	26.03	36.71	17.87	27.17	24.65	46.63
	20.65	28.23	34.07	38.36	12.04	19.96	21.33	14.16
Normal + outliers + multicolli nearity	285.73	139.51	26.14	232.65	172.14	17.70	13.71	62.08
	140.59	149.36	22.50	261.63	148.22	18.52	14.55	52.57

Fig. 5. Shepperd *et al* Analysis of Accuracy (MMRE) for Continuous Model (Y1) [7].

that used adaptation rules performed significantly better than techniques that did not use adaption rules. In addition, they found the CBR techniques that used weighted Euclidean distance also gave the best predictions. Much like Shepperd and Kadoda [7], Mendes and Mosely [8] also found that SWR gave the best prediction accuracy "for all measures of prediction accuracy" [8, p. 11]. The dataset used in [8] consisted of 34 data points which could be considered a small data set when comparing the number of data points used in [43]. Looking at Figure 5 there is a strong case for both SWR and CBR as prediction techniques with small data sets. Wen *et al* [9] also state CBR to be accurate with small data sets, validating the results found by Shepperd and Kadoda [7]. If CBR is described as being accurate with a small data set [9] and SWR techniques consistently outperform it with both small and large sets [7], [8]it can be inferred that SWR is also a good technique to use with small data sets. SWR adds to the prediction model the variables with the highest partial correlation to the response variable at each stage [47]. With the aim to have a set of variables (predictors) in the model to maximise F, F describes the association of all of the predictors to the response variable [47]. For a variable to be added to the model it must increase F by a constant specified amount a commonly described as the(Alpha-To-Enter) value. Similarly for a value to be removed from the model it is measured by its reduction in F and is also compared against a constant described as the (Alpha-To-Leave).

VI. Methodology

This paper introduces a tile-based level, editor which is can be both an MI system and a regular system. The AI aspect of the level editor can be switch on or off and thus will be referred to separately from the level editor. For the remainder of this paper, the AI aspect of the editor will be defined as the MI component. The research question proposed in this project is: How Will a Mixed-Initiative Level Editor that Predicts User Requirements Affect the Levels Created? To that end, a comparison is made between the levels created when the MI component is active against when it is inactive. This experiment was designed to see if a predictive MI component fulfils the design request raised in Alverezs' *et al* [5] literature. The experiment proposed was used to test if the predictive texting findings of Ling [31] and Dunlop and Crossan [32] are applicable to level design.

A. Hypotheses

When creating a level editor the aim is to increase the ease at which levels could normally be created. For the scope of this paper, the focus is on the MI component within the context of the editor, not the editor itself. Table I shows the proposed hypotheses for this paper, it also shows from which source of data each hypothesis depends. Ling [31] found that predictive texting increased the speed at which messages are written. Also, Dunlop and Crossan [32] found that predictive texting also increased the size of the messages written. Hypothesis 2 and 1 explore if these findings are also applicable to level design with the MI component replacing the predictive texting element. Hypothesis 3 investigates whether a predictive component will satisfy the feature proposed by Alverezs' *et al* [5]. Hypothesis 4 builds on the work of Alvarez' *et al* [5] and evaluates if an MI level editor can make designers consider alternative level design approaches. Finally, hypothesis 5 examines if designers with more experience are more likely to be negatively impacted by the MI component than designers with less experience, this is based on conjecture.

B. Participants

All of the participants for this experiment were required to have some level of design experience. Sampling participants from game design environments ensured that all of the participants had some degree of level design experience. Some of the participants were sampled from a Reddit post on the r/GameDev sub-Reddit but the majority of participants came from within the Falmouth Games Academy. A power analysis for this experiment resulted in a sample size of 23. The power included a two-tailed *t*-test with an effect size of *0.8*, an alpha error probability of *0.05* and a *power(1- beta error probability)* of *0.95*. In this experiment, all of the participants were in one group and were required to design levels using the level editor provided. By the end of the experiment, all of the participants designed the same number of levels. Everyone in this study had to consent to the experiment before they had access to the interface and all participants were given the option to quit at any time. The information sheet only gave basic information about the project so as to not invite acquiescence bias [48]. None of the data collected in this experiment could be used to identify any of the users.

C. Experiment Design

The experiment required the participants to create five different levels each with different editor settings. Before the session starts the users were prompted with a multiple choice question. This question asked the participants' to self-assess their own game design experience, from "not a lot" to "a lot" of design experience. After this, the participants were presented with the main level editor interface. They were then prompted with any of the requirements this design session

TABLE I
HYPOTHESES TABLE

	Hypothesis	Null Hypothesis	Data Source
1	The MI component will affect the speed at which the participants create levels.	The MI component will not affect the speed at which the participants create levels.	Design Session Statistics
2	The MI component will affect the size of the levels the participants create.	The MI component will not affect the size of the levels the participants create.	Design Session Statistics
3	The MI component will affect the number of clicks the designer makes.	The MI component will not affect the number of clicks the designer makes.	Design Session Statistics
4	The MI component will influence participants to explore other level designs.	The MI component will not influence participants to explore other level designs.	Questionnaire
5	The participants with more experience will be Negatively impacted more often by the MI component.	There will be no correlation between experience and the negative impact of the MI component.	Questionnaire

had. The different editor settings the participants used can be seen in Table II. The editor settings were given to each user in a random order, this was done to reduce the impact of the practice effect on any data gathered. Before each level starts on-screen prompts informs the user of the rules of the current level, for example: ”During this level, there will be a component that predicts your requirements.”. All of the levels were silently timed so as not to influence participants to work faster than they would usually. At any point during the experiment, the participants could withdraw from the study, doing so caused any data gathered to be deleted. The exploration setting was used to try and influence the designers to consider alternative design decisions. During this setting, the MI tool will suggest prop placements that the user has not considered.

After completion of each level, the participant was asked four multiple choice questions. Each question has five possible answers that follow a Likert scale, the responses example responses proposed by Brown [49]. After the questions have been answered any data gathered was stored along with the current configuration. Once the participant finishes, the data gathered during that session is sent to a remote server using an HTTPS Post message. The questions are:

- How frequently did the level editor make you consider an alternative level design?
- How likely are you to use this level editor when prototyping the design of a level?
- How difficult did you find it to make the level you wanted?
- How pleased are you with the final design of the level?

An alternate method could have had a questionnaire at the end of the experiment once all levels have been completed. However, as each set of responses correlates to a different editor configuration, doing it after each level made data management easier. Leaving the questions until the end in one questionnaire may invite levelling and sharpening bias, this is where details are lost and others are heightened over time [50].

D. Level Editor Design

In this experiment, there were two kinds of game objects that can be placed in the level rooms and props. Rooms occupy tiles on the map, whereas the props can be placed within any

TABLE II
EDITOR SETTINGS

Editor Settings	Predictive Placements Enables	Unrestricted size	Exploration
Settings 1	X	X	
Settings 2		X	
Settings 3	X		
Settings 4			
Settings 5	X		X

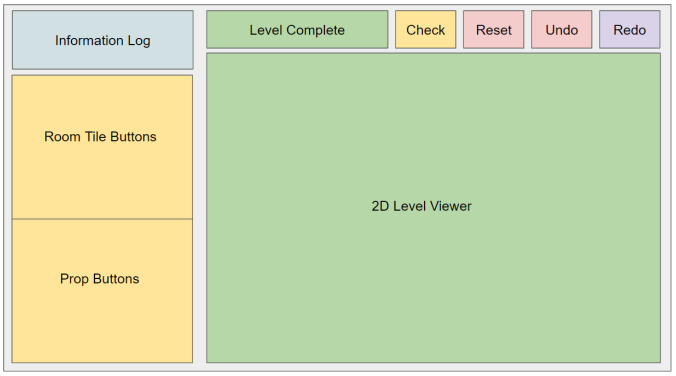

Fig. 6. The first iteration of the proposed level editor

of the rooms. Figure 6 shows the first prototype design for the level editor. Barnes *et al* [12] suggests that any MI component should provide the reasoning behind the agents' actions, to do this the design contains an information log. The information log informs the participants why the level editor has made a particular decision. The level view is a tile-based view of the current map, it allows users to place tiles and props according to their respective rules. Figure 6 also shows the layout of the level editor, on the left is where the buttons to switch between what props/rooms are placed when the participant next clicks. There is a checking algorithm that makes sure there are a start and an end tile, it also checks these tiles are connected by a valid path. The level complete button executes this checking algorithm before finishing to make sure a valid level is being submitted. When the checking algorithm returns a false, the information log informs the user why it has been rejected.

E. Prediction Technique

When deciding the prediction technique for this experiment the main limiting factor to consider is the number of data points available. In the MI level editors discussed in Section IV, the number of tiles presents within a level did not exceed 144 [5], [6], [26]. By treating each tile as one data point this gives a maximum number of 144 data points to use when training the MI component. It has been proven that CBR techniques are effective, even with a small data set [7], [9]. However, SWR has been proven to outperform CBR on multiple occasions [7], [47]. The application of SWR does not seem possible in the multivariable system presented in this level editor, instead, CBR was used to provide the probabilities for a type of prop to be in a given tile. These probabilities were then roulette sampled to produce the props that will actually be placed in the tile.

VII. Software Development

The front end of the level editor was written in HTML which provided the web nodes, CSS which provided styling for the web nodes and JavaScript with Canvas as the rendering API. The front-end logic and AI components were also written in JavaScript. All of the front end development was written using the WebStorm IDE, with Google Chrome to run and debug the application. Writing the application in the form of a website made the experiment easy to share which reduced the friction gathering results can create. It also reduced the chances of investigator/observer bias [51] to occur. The questionnaires that were asked after the creation of each level were embedded in the application to provide a complete and contained experiment. Git was used as the version control software, with the repository for the Git project hosted on Digital Ocean server. On the server side, a Git post-commit hook was set up to automatically deploy any changes in the master branch to the website. The server code that handles the processing of the data sent to the server is written in python3 using the HTTP server module. Python3 was used because it was easy to set up and the run time performance of the application was not a priority. The Python3 module called TinyDB was used for the databasing API. TinyDB does not scale well but as this was a small experiment, the convenient way TinyDB stores data outweighs the performance losses.

A. System Development Life cycle

The development of the application used for this experiment was undertaken by only one person. The level designer can be used to create levels for the game called ”What is that Meat?”, however, there are no dependencies between the two systems. This meant the application of a life cycle method such as Agile Development which focuses a lot on team dynamics and communication [52] would be less applicable to this kind of solo project. For this project, the incremental Model was used. The incremental model is made of phases, where at the end of each phase the product is given to the users. The feedback from the users is used in the creation of the next product, or increment [53]. An incremental model approach is appropriate to use when the full requirements of a system are not known as it gives development flexibility [54].

The first phase (increment) of the application had all of the features and layout as shown in Figure 6. The application was then given to a pilot group of three users. All of the sessions in this small group were supervised so that quality could be assured first hand. In the first increment of the application, the AI would only place props when it already had a history of what props to place in that room. The placing of the props would only happen when a new room was placed. Through observation, it was noted that users tended towards placing all of the rooms for a level first and then filling the rooms with props. This meant the prop placement part of the MI tool was not being used.

The observations and feedback from the first phase were used to fuel the second phase of development. The lack of engagement with the prop placer was solved by a system that showed the user the props that the MI wants to place in the existing rooms. When the users are placing props in a room, low opacity props are placed in other rooms to indicate what props the MI tool wants to place. Another button was added to the toolbar which places all of the MI suggested props. Figure 7 is a screenshot of the final application, along the top bar the Prop Fill button can be seen. The Prop Refresh button can be used to change the seeds of the random number generator used to generate the props. These features are not available in the levels where the MI tool is disabled. Upon completion of this phase, the application was given to two experts in the fields of AI and game development for heuristic evaluation. Cockton and Woolrych [55] believe that heuristic evaluations do not encourage a comprehensive view of an interface. However, a heuristic evaluation performed early on the development process is good [56] and it is cheap in terms of resources [56]. From the evaluation, three main problems were identified.

- The mixed-initiative tool did not take into account the location relative to the room the props were placed.
- The Make Paths should connect all rooms to a path that reaches the start and end tile.
- There is no context for making the levels, or what the props mean.

The usability issues identified were once more fed back into the development cycle. The CBR algorithm was re-designed to take into account the relative position of the props in the room. The Make Paths button was extended to make paths from all of the rooms to a path that connected to the start and end rooms. To give the users context and insight into their design decisions a small turn-based game was created alongside the application. Adding a way for the users to playtest the games they make is crucial to making a great game [57]. The game that was made is a simplified 2D version of the game ”What is that Meat?”. The participants can play test their level at any time provided the minimum requirements for the level have been met. The application was given to a pilot study of four users and was observed to check for usability issues. This increment of the application did not contain any problems and was considered ready to start the experiment.

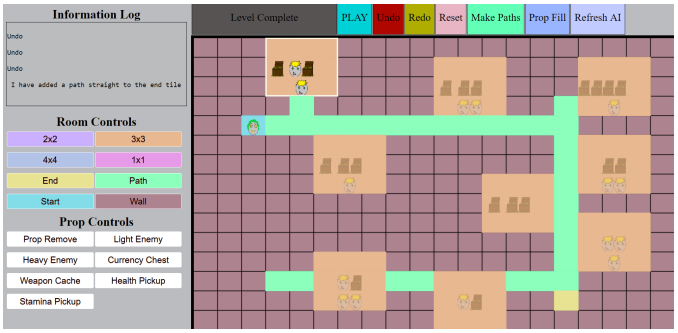

Fig. 7. The final view of the interface during a Mixed-Initiative level

B. Software Testing

Unit testing was introduced early on in the life cycle of the application. Jest is a testing library for JavaScript applications run in the Node.js runtime environment. Jest was used in this experiment because it contained all of the features needed to create unit tests for this application. In addition, Jest has very good documentation and there are many good examples on GitHub. To ensure a consistent styling throughout the application JSHint was used to perform checks on the applications source code. The tests for this application were integrated into the main project through the use of the library called pre-commit. Pre-commit is a Node.js library which provides an easy way to integrate unit testing into Git using git hooks. Before any changes could be committed, all of the unit tests would need to pass and there must be no styling mistakes. If a test were to fail or a styling mistake identified the commit would not occur.

VIII. Results and Analysis

The data from the experiment were processed in Python and was retrieved from the server using an HTTPS Post request. Three Python libraries were used to analyse the data. Scipy was used to provide the statistical functions, Numpy for more statistic functions and data handling and Matplotlib to generate the graphs. Python was used so that the graphs could also be generated when new data comes in. Each following subsection focuses on a different hypothesis and the results gathered to test that hypothesis. Some subsections contain additional results to provide greater context for the discussion.

A. The MI component will affect the speed at which the participants create levels

This hypothesis focused on the time it took for the participants to finish creating each level. The results showed a large distribution of times when all of the maps were combined into one list. Figure 8 shows the distribution of the maps with and without the MI component. Figure 8 shows there is a cluster of times around the 100-300 second mark but both MI and no MI contain some maps that took much longer. The large distribution in time can be attributed to the differing work speeds of the individual participants. To reduce this the times were compared on a per user bases. A ratio of the time it took with the MI tool divided by the time without the MI was calculated for each user and added to a list. A *t*-test produced a *p*-value of 0.001254 and a *T*-Value of 3.4381. The *p*-value for this test is very low, which means for this experiment the null hypothesis can be rejected. The mean ratio of time was 1.636 which means that on average it took each user 1.636 times longer to create maps with the MI component. The Cohen's *d* effect size was 0.9923. A Cohen's d effect size of greater than 0.8 would suggest a large effect size [58].

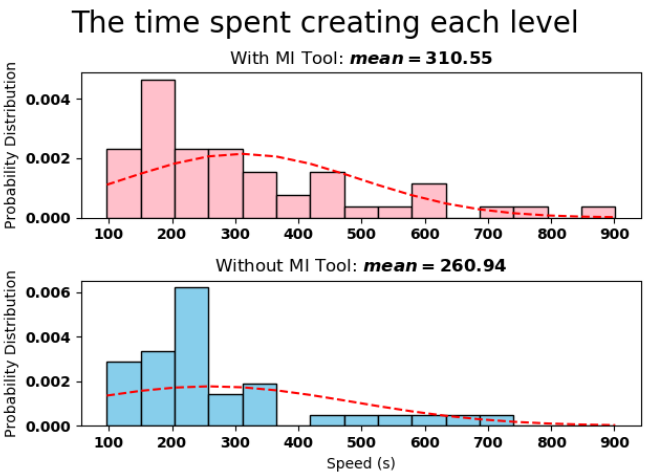

Fig. 8. Histogram of the time taken to create levels with and without the MI component

B. The MI component will affect the size of the levels the participants create

For this hypothesis, the two dependent variables used were the number of props and the number of tiles present in the level. There is a large distribution with the number of props and tiles. Figure 9 shows a box plot for the number of objects that were at each level. From Figure 9 there can be seen a general skew towards a larger number of objects with MI tool, than without. A ratio of the number of each object with divided by without the MI component was calculated. This removed the uncertainty created by comparing different participants design styles. Table III shows the values calculated from a *t*-test on the list of ratios generated from the data. It would appear that the *p*-value for both the ratios of props and tiles are significant as they are less than 0.05. Both the tiles and the props have roughly the same Effect sizes at 0.7, which suggests a large effect size [58]. It is interesting to note the large difference in means between the props and tiles. For every prop in a no MI level, there were two in an MI level. With tiles, there were 1.353 tiles in an MI level to everyone in a no MI level. Table IV shows the percentage of MI placed objects that were removed by the participant, it shows that the participants removed about 50% of all tiles placed by the MI and about 30% of props.

The increase in objects could have been caused by the increase in time spent creating the level described in Section VIII-A. However, a *t*-test performed on the number ratio of props against the ratio of time it took to complete the

level returned a p-value of 0.4107 which is too high to show significance. A similar t-test performed on the ratio of tiles against the ratio of time produced a p-value of 0.2789 which once again is too high to prove a meaningful correlation.

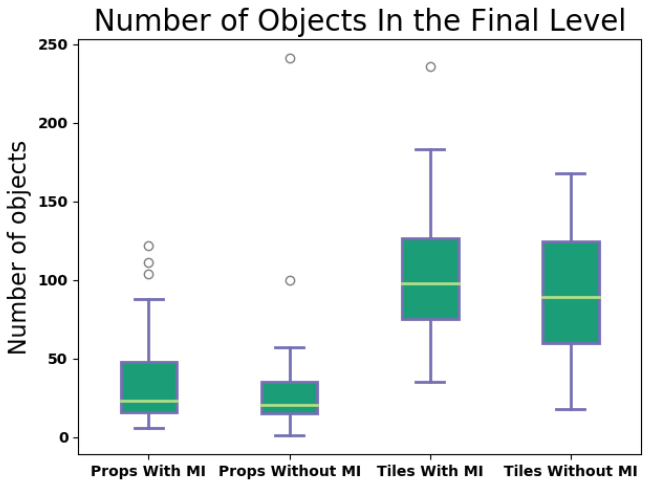

Fig. 9. Box plot of the number of objects in the final level

TABLE III
NUMBER OF OBJECTS VALUES TABLE

Object Type	*P*-Value	*T*-Value	Cohen's *d* Effect Size	Mean Ratio MI/No MI
Tiles	0.01457	2.539	0.7329	1.353
Props	0.01570	2.509	0.7242	2.002

TABLE IV
PERCENTAGE OF OBJECTS PLACED BY THE MI THAT WERE REMOVED

Tiles	Props
47.48%	28.86%

C. The MI component will affect the number of the clicks the designer makes

To test this hypothesis the number of clicks the user made in each level was recorded. The clicks with and without the MI component were calculated as a ratio on a per-user basis. A t-test on the ratios produced a p-value of 0.008758 and a T-Value of 2.738. The Cohen's d effect size was 0.7904 which would suggest a large effect size [58]. The mean click ratio was 1.622, which means that on average for every click done on a no MI level, 1.6 clicks were done on an MI level. Figure 10 shows several bins with a high number of clicks for MI enabled levels.

The average increase in clicks may be related to the size of the level. Section VIII-B describes that on average the levels with the MI component have twice as many props and 1.353 as many tiles than levels without the component. A t-test performed on the ratio of clicks against the number of props returned a p-value of 0.4133 and against tiles a p-value of 0.3181. Both of the p-values are too high to show a significant relationship between the number of clicks and the size of the level.

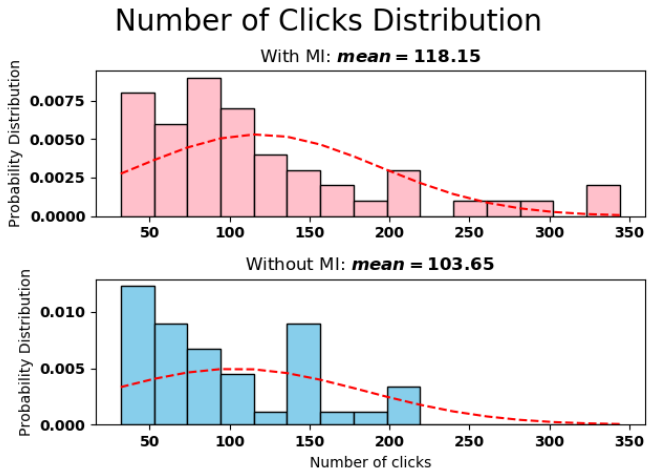

Fig. 10. Histogram of the number of clicks for each level

D. The MI component will influence participants to explore other level designs

The dependent variable used to test this hypothesis came from the questionnaire response to the question "How frequently did the level editor make you consider an alternative level design?". A Mann-Whitney U [59] test was used to determine the significance between the questionnaire response with the MI and the questionnaire response without the MI component. The Mann-Whitney U test was used as the questionnaire data is ordinal and the comparison is made between two independent groups (with MI and without MI).The test produced a p-value of 0.02223 and a u-value of 885.5. Cohen's d effect size is 0.4311, which would suggest a medium effect size [58]. Figure 11 shows a double histogram containing the numerical representation of the questionnaire responses for both the MI levels against the no MI levels. Table V displays how the responses correlated to the numbers shown on the histogram. The mean value of how often users were influenced to explore an alternative level design with the MI component is 1.813 without the MI component the value is 1.292. From the two means we can argue that when the MI component is active, participants found that the level editor challenged them to think about alternative level designs more often. Figure 11 shows that there are some participants that fell between the frequently - very frequently categories with the MI component. It also shows that there are were no participants that thought the level editor frequently to very frequently made them consider an alternative level design.

E. The participants with more experience will be Negatively impacted more often by the MI component

The two dependent variables tested for this hypothesis were both ordinal and from the questionnaire. The users were asked

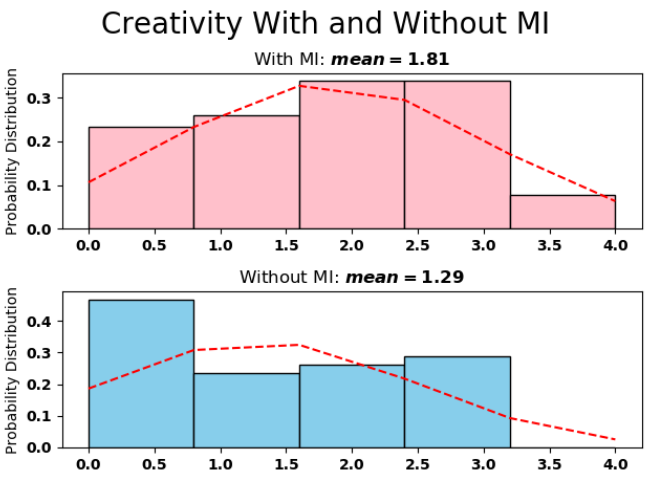

Fig. 11. Histogram of how often users were influenced to explore other level designs.

TABLE V
QUESTIONNAIRE RESPONSE NUMERICAL VALUES

Never	Rarely	Sometimes	Frequently	Very frequently
0	1	2	3	4

at the end of each level: "How difficult did you find it to make the level you wanted?", the responses ranged from very hard to very easy. A Spearman's correlation was used to test the correlation between the users' experience and how difficult they found it to create the levels with the MI component. A Spearman's correlation is recommended to use when dealing with data that is not normally-distributed [60]. The output from a Spearman's correlation on the two variables produced a p-value of 0.4196, which is above 0.05 so the null hypothesis cannot be rejected.

IX. DISCUSSION

The results in this experiment show that an MI component can have a clear impact on the levels created by a designer. It is hard to label the net impact of the tool as either positive or negative. In this section, the potential implications of each hypothesis will be discussed.

A. The MI implementation

The implementation of the MI component plays a large part in the way the participants interacted with it. In this level editor, the MI component could make pivotal design decisions such as the pathways between each room. Table IV shows that the participants often removed the pathways the MI component placed. The layout of the map and how the rooms connect together could be seen as an abstract concept. Barnes *et al* [12] found that human designers were better at making abstract decisions and inferring meaning than an AI agent. It could be argued the results shown in table IV support Barnes *et al* [12] findings as the high percentage of removed MI decisions could mean the MI does not understand the abstract idea for the map.

B. The MI component will affect the speed at which the participants create levels

Section VIII-A shows, at a significance level of 0.1%, that when the MI tool is present the time the users spend on each level increases. These results would suggest that adding an MI component to a level editor would not be effective for the prototyping phase, as prototypes are meant to be quick and easy to produce [2]. One cause for this increase in time could have been caused by the time it took to remove decisions made by the MI component. Section IX-A describes how giving the MI component the ability to make pathways may have caused the MI component to make suggestions that did not align with the ideas of the designer. This misalignment of the MI components vision for the level and the designers could have been the cause for the increase in time.

The results found in section VIII-A indicated that the increase in speed found in predictive texting by Dunlop and Crossan [32], may not apply to level design. However, the way in which users were tested in Dunlop and Crossan [32] differs from how the participants were tested in this paper. In this experiment, participants were not given a number of objects in the scene they had to place, instead, they were allowed to place as many as they wanted. This added some uncertainty as it meant participants could make levels as big as they wanted to.

C. The MI component will affect the size of the levels the participants create

Section VIII-B describes how the mean ratio for both the number of tiles and props is greater than one and both stats are significant to a level of 2%. The results show ratios of above one, which suggests that the MI component did increase the size of the levels, similar to results found by Ling [31] with predictive texting. It could be argued that the increase in the size of the levels is related to the increase in the time it took for participants to create the levels. Section VIII-B includes the results of a t-test looking for a relationship between the size of the MI levels and the time it took to create the levels. The results from the t-test did not show a sign below the 5% level. This would suggest the increase in the size of the level did not come from the extra time spent with the MI levels, but instead from the interaction with the MI. In this experiment using the MI component, on average, doubled the number of smaller objects placed in the level. This would suggest if the map the designer is trying to prototype should have lots of small objects, it may be beneficial to include an MI component to handle the placement of these objects.

D. The MI component will affect the number of the clicks the designer makes

The results described in section VIII-C would suggest the addition of the MI component to a level editor increases the number of clicks the users have to make. It can be argued that adding an MI component to a level editor did not solve the feature requested by one of the participants in the paper written by Alverezs *et al* [5]. One cause for the increase in

the number of clicks could have been the implementation of the MI component. As discussed in section IX-A, the removal of the objects placed by the MI component could have caused an increase in clicks.

E. The MI component will influence participants to explore other level designs

Arguably, the main purpose of prototype levels is to explore potential ideas. These data described in section VIII-D suggests that having an MI component that works alongside a designer, may make them consider an alternative level design. These results also support the findings of Alverezs' *et al* [5]. It can be argued that influencing participants to consider alternatives is a great aid during the design process. Even if the ideas presented by the MI are not accepted by the user, they have still considered search spaces that would otherwise have remained unexplored. This would suggest adding an MI component to a level editor may increase the variety of levels produced by the designers.

The participants also found the level editor influenced them to explore other level designs without the MI component present. This could have been caused by the level editor not allowing them to submit a map that cannot be completed. It can be argued that giving the level editor the ability to tell which maps are completable is a form of intelligence. This would mean, even without the extra tools provided by the MI component, the level editor could still be considered an MI tool.

F. The participants with more experience will be Negatively impacted more often by the MI component

In this experiment, there were no significant results to suggest that the users level design experience affected the way they interacted with the MI component. For all participants in the experiment, it was there first time interacting with the level editor. Regardless of experience all of the users were using a new tool for the first time. This meant the participants had no expectations of what that tool should or could do. This could be why there were no significant results to suggest the users with more experience were negatively impacted by the tool.

X. Future Work

One unanticipated result is how frequently the MI placed tiles were removed. Section IX-A describes reasons to why the MI placed pathways were removed so often. The argument presented in Section IX-A could have affected all of the hypothesis tested in this experiment. Further work should be done to see what effects reducing the MI components ability to make decisions may have.

In section IX-B this paper argues that adding an MI system may increase the time it takes for users to complete levels. The original work by Dunlop and Crossan [32] that motivated this hypothesis took a very different approach to measure the speed the user created levels. Dunlop and Crossan [32] set out a specific goal for the users to type and timed how long it took to achieve that goal. In this paper, to test the speed, the users could finish the level once they had placed a minimum number of objects. They could keep going for as long as they wanted. To make the results more comparable to the results found by Dunlop and Crossan [32] further work should be done that requires the user to place exactly a specified number of objects and time that instead.

The results and discussion presented in section IX-E seem to affirm the findings of Alverezs' *et al* [5]. However, in this experiment all of the results in this area are based on one questionnaire response. Further work should be done, to explore the accuracy of the findings in Alverezs' *et al* [5] and the findings present in this paper. Additional research could include a questionnaire which is more comprehensive as well as a more refined novelty search algorithm.

In this experiment, CBR was used in the prediction algorithm. SWR could not easily be adapted to fit with the data structures present in the level editor. If SWR could be modified to work in the context of this level editor, the existing research suggests that it would provide better predictions [7], [47]. More research should be done to measure how the prediction methods use may change the way users interact with the level editor.

XI. Conclusion

Existing literature in the field of MI level design focuses on how an MI component can help with the creative process. The focus of this paper was the effect that adding an MI component may have the level design process. The results in this paper give insight into the kind of effects integrating an MI system into a level editor may have. This paper has argued that integrating an MI component into a level editor will affect both the designer and the levels created by the designer.

This study found four significant effects that adding an MI component may have. These effects are:

- An increase in the size of the levels create.
- An increase in the time it takes to create each level.
- An increase in the number of clicks participants have to make.
- An increase in the exploration of alternative level designs.

When one is planning on integrating an MI system into the level design process, much thought should be put into the MI responsibilities. In this paper, a system was created that could make large scale decisions. These decisions were often not kept by the participants. In the creative endeavour that is level design, perhaps MI systems should not be allowed to make large decisions. As discussed in section IX-A, the implementation of the MI system plays the largest part on the kind of effects it may have. This paper recommends planning carefully the type and scale of the decisions the MI can make. This paper has shown, that users are much more willing to accept the help implementing the finer details, than the larger picture.

References

[1] A. H. Blackwell and E. Manar, "Prototype," [Online]. Available: http://link.galegroup.com/apps/doc/ENKDZQ347975681/SCIC?u=dclib_main&sid=SCIC&xid=0c8f739d, [Accessed 12-Nov-2018].

[2] R. Budde, K. Kautz, K. Kuhlenkamp, and H. Züllighoven, *Prototyping*. Springer, 1992, ch. 5, pp. 33–46.

[3] T. Fullerton, C. Swain, and S. Hoffman, *Game design workshop: Designing, prototyping, & playtesting games*. CMP Books, 2004.

[4] J. Stempfle and P. Badke-Schaub, "Thinking in design teams-an analysis of team communication," *Design studies*, vol. 23, no. 5, pp. 473–496, Sep 2002.

[5] A. Alvarez, S. Dahlskog, J. Font, J. Holmberg, C. Nolasco, and A. Österman, "Fostering creativity in the mixed-initiative evolutionary dungeon designer," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*. Malmö, Sweden: ACM, Aug 2018.

[6] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: Computer-aided game level authoring." in *Proceedings of ACM Conference on Foundations of Digital Games*. ACM, May 2013, pp. 213–220.

[7] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Transactions on Software Engineering*, vol. 27, no. 11, pp. 1014–1022, 2001.

[8] E. Mendes and N. Mosley, "Further investigation into the use of cbr and stepwise regression to predict development effort for web hypermedia applications," in *Proceedings International Symposium on Empirical Software Engineering*. IEEE, 2002, pp. 79–90.

[9] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, 2012.

[10] A. Liapis, G. Smith, and N. Shaker, *Mixed-initiative content creation*. Springer, 2016, ch. 11, pp. 195–214.

[11] J. R. Carbonell, "Mixed-initiative man-computer instructional dialogues." Tech. Rep., Jun 1970.

[12] M. J. Barnes, J. Y. Chen, and F. Jentsch, "Designing for mixed-initiative interactions between human and autonomous systems in complex environments," in *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1386–1390.

[13] S. Ochs, "Just for fun: Show us your best quicktype poetry," [Online]. Available: https://www.macworld.com/article/2692885/just-for-fun-show-us-your-best-quicktype-poetry.html, 2014, [Accessed: 12-Nov-2018].

[14] A. Kantosalo, J. M. Toivanen, P. Xiao, and H. Toivonen, "From isolation to involvement: Adapting machine creativity software to support human-computer co-creation." in *ICCC*, 2014, pp. 1–7.

[15] R. Van der Linden, R. Lopes, and R. Bidarra, "Designing procedurally generated levels," in *Proceedings of the the second workshop on Artificial Intelligence in the Game Design Process*, 2013.

[16] L. Doherty, M. Whitney, J. Shakya, M. Jordanov, P. Lougheed, D. Brokenshire, S. Rao, S. Menon, and V. Kumar, "Mixed-initiative in computer games: Algorithmic content creation in open-ended worlds," in *AAAI Fall Symposium on Mixed-Initiative Problem-Solving Assistants*, 2005, pp. 46–56.

[17] J. M. Font, R. Izquierdo, D. Manrique, and J. Togelius, "Constrained level generation through grammar-based evolutionary algorithms," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 558–573.

[18] D. Karavolos, A. Bouwer, and R. Bidarra, "Mixed-initiative design of game levels: Integrating mission and space into level generation." in *Proceedings of the 10th International Conference on the Foundations of Digital Games*, Pacific Grove, CA, USA, Jun 2015.

[19] L. Johnson, G. N. Yannakakis, and J. Togelius, "Cellular automata for real-time generation of infinite cave levels," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. Monterey, California, USA: ACM, Jun 2010, pp. 1–4.

[20] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.

[21] J. Doran and I. Parberry, "Controlled procedural terrain generation using software agents," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 111–119, Jun.

[22] J. Regier and R. Gresko, "Random asset generation in diablo 3," *Invited Talk, UC Santa Cruz*, 2009.

[23] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity." in *Proceedings of the 9th Conference on the Foundations of Digital Games*, 2014.

[24] R. Oppermann, "User-interface design," in *Handbook on information technologies for education and training*. Springer, 2002, ch. 15, pp. 233–248.

[25] W. O. Galitz, *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, 2007.

[26] A. Baldwin, S. Dahlskog, J. M. Font, and J. Holmberg, "Mixed-initiative procedural generation of dungeons using game design patterns," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. New York, NY, USA: IEEE, Aug 2017, pp. 25–25.

[27] E. Horvitz, "Principles of mixed-initiative user interfaces," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. Pittsburgh, Pennsylvania, USA: ACM, May 1999, pp. 159–166.

[28] A. Jordanous, "Defining creativity: Finding keywords for creativity using corpus linguistics techniques." in *Proceedings of the First International Conference on Computational Creativity (ICCCX)*, Lisbon, Portugal., 2010, pp. 278–287.

[29] A. Liapis, G. N. Yannakakis, C. Alexopoulos, and P. Lopes, "Can computers foster human users' creativity? theory and praxis of mixed-initiative co-creativity," *Digital Culture & Education*, vol. 8, no. 2, pp. 136–153, 2016.

[30] J. D. Lee and K. A. See, "Trust in automation: Designing for appropriate reliance," *Human factors*, vol. 46, no. 1, pp. 50–80, 2004.

[31] R. Ling, "The length of text messages and the use of predictive texting: Who uses it and how much do they have to say," *Association of Internet Researchers, Chicago, IL*, Oct 2005.

[32] M. D. Dunlop and A. Crossan, "Predictive text entry methods for mobile phones," *Personal Technologies*, vol. 4, no. 2, pp. 134–143, Jun 2000.

[33] R. Chipalkatty, G. Droge, and M. B. Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 695–703, May 2013.

[34] V. Bhatia and V. Hasija, "Targeted advertising using behavioural data and social data mining," in *U2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, Jul 2016, pp. 937–942.

[35] N. Ye *et al.*, "A markov chain model of temporal behavior for anomaly detection," in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166. West Point, NY, 2000, p. 169.

[36] W.-H. Ju and Y. Vardi, "A hybrid high-order markov chain model for computer intrusion detection," *Journal of Computational and Graphical Statistics*, vol. 10, no. 2, pp. 277–295, 2001.

[37] R. Gwadera, M. Atallah, and W. Szpankowski, "Markov models for identification of significant episodes," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 404–414.

[38] A. Markov, "Extension of the limit theorems of probability theory to a sum of variables connected in a chain," *Dynamic Probabilities Systems*, vol. 1, 1971.

[39] W.-K. Ching, M. K. Ng, and E. S. Fung, "Higher-order multivariate markov chains and their applications," *Linear Algebra and its Applications*, vol. 428, no. 2-3, pp. 492–507, 2008.

[40] S. Snodgrass and S. Ontanon, "Learning to generate video game maps using markov models," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 4, pp. 410–422, 2017.

[41] Q. Lai, H. Yu, H. Chen, and Y. Sun, "Prediction of total power of agricultural machinery using artifical neural networks," in *2010 International Conference on Computing, Control and Industrial Engineering*, vol. 2. IEEE, 2010, pp. 394–396.

[42] U. Akdag, M. A. Komur, and A. F. Ozguc, "Estimation of heat transfer in oscillating annular flow using artifical neural networks," *Advances in engineering software*, vol. 40, no. 9, pp. 864–870, 2009.

[43] N. Hazarika and D. Lowe, "A neural-network extension of the method of analogues for iterated time series prediction," in *Neural Networks for Signal Processing VIII, 1998. Proceedings of the 1998 IEEE Signal Processing Society Workshop*. IEEE, pp. 458–466.

[44] I. Watson, *Applying case-based reasoning: techniques for enterprise systems*. Morgan Kaufmann Publishers Inc., 1998.

[45] C. K. Riesbeck and R. C. Schank, *Inside case-based reasoning*. Lawrence Eribaum Associates, 1989.

[46] I. Watson, "Case-based reasoning is a methodology not a technology," in *Research and Development in Expert Systems XV*. Springer, 1999, pp. 213–223.

[47] L. D. Schroeder, D. L. Sjoquist, and P. E. Stephan, *Understanding regression analysis: An introductory guide*. Sage Publications, 2016, vol. 57.

[48] D. Watson, "Correcting for acquiescent response bias in the absence of a balanced scale: An application to class consciousness," *Sociological Methods & Research*, vol. 21, no. 1, pp. 52–88, 1992.

[49] S. Brown, "Likert scale examples for surveys," *ANR Program evaluation, Iowa State University, USA*, 2010.
[50] A. Koriat, M. Goldsmith, and A. Pansky, "Toward a psychology of memory accuracy," *Annual review of psychology*, vol. 51, no. 1, pp. 481–537, 2000.
[51] M. R. Phillips, B. D. McAuliff, M. B. Kovera, and B. L. Cutler, "Double-blind photoarray administration as a safeguard against investigator bias." *Journal of Applied Psychology*, vol. 84, no. 6, pp. 940–951, 1999.
[52] K. Beck *et al.*, "Agile manifesto," [Online]. Available: http://agilemanifesto.org, 2001, [Accessed: 2-April-2019].
[53] V. Massey and K. Satao, "Comparing various sdlc models and the new proposed model on the basis of available methodology," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 4, 2012.
[54] P. Isaias and T. Issa, "Information system development life cycle models," in *High Level Models and Methodologies for Information Systems*. Springer, 2015, pp. 21–40.
[55] G. Cockton and A. Woolrych, "Sale must end: should discount methods be cleared off hci's shelves?" *interactions*, vol. 9, no. 5, pp. 13–18, 2002.
[56] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1990, pp. 249–256.
[57] B. Winn and C. Heeter, "Resolving conflicts in educational game design through playtesting," *Innovate: Journal of Online Education*, vol. 3, no. 2, 2006.
[58] J. Cohen, *Statistical power analysis for the behavioral sciences*. Lawrence Earlbaum, 1988, pp. 20–27.
[59] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, vol. 18, no. 1, pp. 50–60, 1947.
[60] N. O'Rourke, L. Hatcher, and E. J. Stepanski, *A step-by-step approach to using SAS for univariate & multivariate statistics*. SAS institute, 2005.

APPENDIX A
ACKNOWLEDGEMENTS

I would like to thank all of the staff at Falmouth University for their support and guidance. I would also like to thank my dissertation supervisor Ed Powley. My thanks go to my classmates for the help provided in the peer reviews sessions.

APPENDIX B
REFLECTIVE REPORT

Each of the following subsections pertains to a different problem encountered during this dissertation. Some of the subsections include a SMART goal that addresses the problem raised in that section.

A. The level editor

From a visual perspective, the quality of the interface leaves a lot to be desired. I was informed by some participants that it was hard to really understand the context of the editor due to poor visuals. I understand the potential for this misunderstanding to have affected the results produced by the interface.

To improve my ability to produce effective visuals for an interface I will watch and engage with the Lynda tutorial titled "UX Design for Developers". I will measure my progress by how far through the tutorial I am and how effectively I can apply the concepts learnt. I will complete this goal within two weeks. From this dissertation project, I have learnt the kind of effects poor UX design can have on a project. I would recommend that anyone who is planning on creating a graphical interface learn more about UX design.

B. Prediction algorithm

From the literature review, the SWR procedure seemed to yield the most accurate predictions. However, I could not fit the SWR procedure to my project. I believe switching to CBR may have caused a significant loss in accuracy. If I had a more advanced understanding of the statistical maths involved in SWR I may have been abled to tailor it to my project.

To advance my statistical maths abilities I will complete ten AQA A-Level statistics past papers. I will measure my progress by how many of the past papers I have completed. If I cannot complete an answer or get it wrong, I will learn the theory behind it. I will give myself five months to complete all of the past papers. Creating the research artefact has shown me that AI and statistical maths are closely related fields. I would recommend that anyone who plans on creating an AI system has a good understanding of statistical maths.

C. Speed test

I think the design of the experiment used to test the speed at which the users created the levels was somewhat flawed. I do believe the results gathered are relevant and do have meaning to the field. However, I do not believe they accurately test the predictive texting findings from which the hypothesis is derived. To accurately test the speed the users created each level, I should have asked the participants to create levels of an exact size and timed how long it took for them to do that. In the future, if I am trying to apply results across research subjects, I will create an experiment that more accurately mirrors the experiment that is proposed in the original research.

To improve my ability to design good experiments that can accurately test a hypothesis I will pick create five hypotheses. Using these hypotheses, I will design experiments that aim to accurately test if any of the hypotheses are correct. To measure my progress I will ask research academics for feedback on the designs I create. I aim to complete this goal before September. Being able to design good experiments is not just useful for academic work. For example, a business may need to find out the effects that changes to their product may have.

D. Play testing

The feature that gave the participants the option to playtest the levels was created approximately halfway through the development of the artefact. It came from the need to give the participants context in regards to the levels they were creating. I debated for some time whether or not to include the time spent playtesting in the total time for the level. After getting feedback from some of the design lecturers I concluded that playtesting a level is an integral part of level design, and certainly should be counted in the total time to make a level. Unfortunately, I believe may have created a lot of noise in the data sets.

E. Tutorial

The tutorial I created contained all of the knowledge required to use all of the features the level editor had. I tracked whether or not the participants did the tutorial or not.

Every participant did complete the tutorial, which consisted of several windows with a GIF and a brief description. Looking back, I believe it would have been more effective to make an interactive tutorial which forced the participants to use every feature at least once. Currently, there were some participants who didn't use some features of the level editor.

To improve my ability at creating tutorials I will start by watching the video titles "How to Make Great Game Tutorials" which can be found on YouTube. This tutorial was created by Game Developers Conference(GDC) which is an international game design organisation. After watching this video I will practice the concepts taught in it by creating three interactive tutorials for games I have previously made.

F. Conclusion

In conclusion, I believe the research project I have undertaken does add knowledge to the field of MI level editors and, to a lesser degree, the field of MI systems. There are some areas where more careful planning and mirroring the existing literature may have produced a better outcome. Despite this, I am proud of the literature I have authored and the complex programming artefact I have created.

APPENDIX C
ADDITIONAL MATERIALS

A. Unit Tests

Figure 12 shows the TortoisGit log after a successful commit. If one of the tests were to fail, the commit is cancelled.

```
'Pre-commit checks...'
PASS Tests/RNG.test.js
  √ RNG Float 1 (2ms)
  √ RNG Float 2 (1ms)
  √ RNG Float 3
  √ RNG Float 4
  √ RNG Int 1
  √ RNG Int 2
  √ RNG Int 3
  √ RNG Int 4
  √ RNG Range 1
  √ RNG Range 2
  √ RNG Range 3
  √ RNG Range 4

PASS Tests/Vector2.test.js
  √ Vector Addition (0,0) + (1,1) (4ms)
  √ Vector Addition (-1,-1) + (0,0)
  √ Vector Addition Rand 1
  √ Vector Addition Rand 2 (1ms)
  √ Vector Subtraction (0,0) - (1,1)
  √ Vector Subtraction (-1,-1) - (-1,-1)
  √ Vector Subtraction Rand 1
  √ Vector Subtraction Rand 2
  √ Vector Scale (0,0) * 10 (1ms)
  √ Vector Scale (-1,-1) - 10
  √ Vector Scale Rand 1
  √ Vector Scale Rand 2
```

Fig. 12.