GENETIC IMPROVEMENTS TO PROCEDURAL GENERATION IN GAMES

Examensarbete Systemarkitekturutbildningen

Johan Forsblom
Jesper Johansson

HT 2018:KSAI02


Syftet med examensarbete på systemarkitekturutbildningen är att studenten skall visa förmåga att delta i forsknings- eller utvecklingsarbetet och därigenom bidra till kunskapsutvecklingen inom ämnet och avrapportera detta på ett vetenskapligt sätt. Således måste de projekt som utförs ha tillräcklig vetenskaplig och/eller innovativ höjd för att generera ny och generellt intressant kunskap.

Abstract

One of the biggest industries today is the gaming industry. A multitude of games are sold each year, competing for the players’ attention and wallets. One of the common important techniques used today to produce game content is procedural content generation, where the computer generates small or larger parts of a game which often affects the gameplay experience.

The purpose of this study is to design and implement a framework which can be used to evaluate and improve the procedural content generation in games, so that the gameplay experience for players in procedurally generated games can be increased.

The research method used was design science, and the theories upon which the framework is built with are flow, procedural content generation and the genetic algorithm. The framework first was designed, and then implemented as an artifact in the form of a roguelike game, so that the framework’s functionality could be evaluated and validated. The game was then set up on a webpage so that anyone could contribute to the research by playing by giving feedback of how well the procedural content generation was performing in the form of a questionnaire within the game. Hence, the results of the study was twofold. The framework itself, and secondly the implementation of the framework in the form of a roguelike game.

Keywords: genetic algorithm, flow, procedural generation, games, gaming, experience
Sammanfattning

En av de största industrierna idag är spelindustrin. En stor mängd av spel säljs varje år, vilka konkurrerar om spelarnas uppmärksamhet och plånböcker. En av de vanligaste teknikerna idag för att skapa innehåll till spel är procedurell generering där datorn genererar små eller stora delar av spelet, vilket ofta har inverkan på spelupplevelsen.

Syftet med denna studie är att designa och implementera ett ramverk, vilket kan användas till att evaluera och förbättra den procedurella genereringen av spelinnehållet i spel, så att spelupplevelsen för spelare i procedurellt genererade spel kan förbättras.

Forskningsmetoden som användes var design science, och de teorier som ramverket bygger på är flow, procedurell generering och den genetiska algoritmen. Ramverket designades först, och implementerades sedan som en artefakt i formen av ett rouge-liknande spel, så att ramverkets funktionalitet kunde evalueras och valideras. Spelet lades upp på en hemsida, så att vem som helst kunde bidra till forskningen genom att spela och ge svar på hur väl den procedurellt genererade innehållet presterade. Härav blev resultaten av studien tvåfaldiga, ramverket självt, och implementationen av det i formen av ett rouge-liknande spel.

Nyckelord: genetiska algoritmen, flow, procedurell generering, spel, upplevelse
Contents

1 Introduction ..................................................................................................................... 1

1.1 Research Contribution and Delimitations ................................................................. 2

2 Related Research .......................................................................................................... 3

2.1 Genetic algorithms .................................................................................................... 3

2.1.1 Selection Methods ............................................................................................... 5

2.1.2 Roulette Wheel Selection .................................................................................... 5

2.1.3 Random Selection ............................................................................................... 6

2.1.4 Ranking Selection .............................................................................................. 6

2.1.5 Tournament Selection ......................................................................................... 7

2.1.6 Mutation ............................................................................................................. 7

2.1.7 Elitism .................................................................................................................. 7

2.2 Procedural Content Generation for Games ............................................................... 7

2.3 Flow .......................................................................................................................... 8

2.3.1 The components of Flow ..................................................................................... 8

2.3.2 Flow Measurement .............................................................................................. 10

2.3.3 Flow in Games .................................................................................................. 13

3 Methodology ................................................................................................................ 14

3.1 Research Approach & Strategy ................................................................................. 14

3.2 Method Research Framework .................................................................................... 14

3.2.1 Explicate problem and Defining Framework Requirements .............................. 15

3.2.2 Design and develop the artifact ......................................................................... 15

3.2.3 Demonstrate the Artifact .................................................................................. 17

3.2.4 Evaluate Artifact .............................................................................................. 17

3.2.5 Ethical Considerations ....................................................................................... 18

4 Developing the Framework .......................................................................................... 20

4.1 Scope ....................................................................................................................... 21

4.2 Parameters .............................................................................................................. 21

4.3 Applying the Genetic Algorithm ............................................................................. 21

4.4 Data Storage ........................................................................................................... 22

4.5 Data collection “demands” ....................................................................................... 22

4.6 Analysis of the GA .................................................................................................. 23

4.6.1 Quantitative Analysis of Generated data ......................................................... 23

4.6.2 Qualitative Analysis of Generated Data .............................................................. 23

4.6.3 Flow Evaluation ................................................................................................ 23

5 Demonstrate and evaluate artifact ............................................................................ 26

5.1 Scope ....................................................................................................................... 26

5.2 Parameters .............................................................................................................. 26

5.3 Data Collection ....................................................................................................... 26

5.4 Applying the Genetic Algorithm ............................................................................. 27

5.4.1 Data Storage ....................................................................................................... 28

5.5 Results ..................................................................................................................... 28

5.5.1 The fitness of the individuals............................................................................. 29

5.5.2 Comparison between generations ..................................................................... 30

6 Discussion & Conclusions ......................................................................................... 34

6.1 Methodological Reflection ...................................................................................... 35

6.1.1 6.1.1 Generalizability ......................................................................................... 36

6.1.2 Validity and Reliability ....................................................................................... 36

6.1.3 Reproducibility .................................................................................................. 36

6.2 Future Work ............................................................................................................. 36

References ...................................................................................................................... 38
1 Introduction

The gaming industry is one of the biggest today. A lot of games are released each year, both by big companies and smaller independent development studios, all fighting for the customers’ attention and wallets. The challenge of making games, is not just that they should look good, the gameplay must also be fun. If a game is not fun in general, no one will play it and the reviews will keep others from buying. When a game is fun, the players come back over and over again to the good experience that the game is providing for them (Fullerton, Swain, & Hoffman, 2008). This is where the importance of discovering why and what it is that makes players keep on playing a game. If the player base of a game is large, the longer the game can live on, and the more money can be collected. One of the best examples of this is World Of Warcraft which has been running since 2004, with the peak of ten million players in 2010, still having 5.5 million players in 2015 (Wikipedia, 2016).

One of the researchers that have spent many years to investigate what it is that makes a person immersed in an activity is Csikszentmihalyi. He has defined a term he calls flow, which represents the mental state of when a person is completely immersed in an activity, with a full sense of enjoyment and focus upon what he is doing, not aware of anything else other than the activity being performed (Csikszentmihalyi, 1990). One interesting fact, is that the descriptions between when people have experienced flow in various activities and when players are immersed in computer and console games are identical (Chen 2007). This means that players enter a flow-like state when playing computer and console games. It is therefore possible to measure the amount of flow the players’ experience when playing a game.

Another important aspect to computer and console games is variety. In order to bring more variety into games, game developers are using procedural content generation (PCG), which is the process of having the computer create game content algorithmically. By letting the computer randomly generate new worlds or levels, the players get a less static gaming experience each time they play the game, thus making the game more appealing to play again (Hendrikx, Meijer, Van Der Velden & Iosup 2013).

One of the main problems when it comes to game design today is to find the right challenge, so that the game becomes immersive and fun to play (Fullerton, Swain, & Hoffman, 2008), in other words, the players must enter a state of flow where they want to keep on playing. Since many games today utilizes procedural content generation techniques, to let the computer generate different kinds of game content, and the PCG in some way is affecting the gameplay experience, it is therefore necessary to evaluate the PCG and see if it can be improved to affect the gaming experience in a positive way.

The chosen approach to find a solution for this problem, is to create a framework which utilizes the so called genetic algorithm, which will tweak the different procedural generating parameters, based on player feedback, and improve the gaming experience for players in procedurally generated games, where the framework is built by applying Csikszentmihályi’s theory of flow, genetic algorithm theory and procedural content generation theory. The genetic algorithm is a machine learning algorithm, part of the evolutionary computation paradigms, which tries to find the mean value that works best for the large population. It is based upon the theory of evolution, where only the fittest survives. By letting the strongest individuals live on into the next generation and breed among themselves, the genetic algorithm will produce a better and better population each generation, until it reaches a point where the optimal individuals are found, due to no significant changes in the input from player feedback.
Once the framework is created, it will be empirically tested by implementing it and developing an artifact which applies specific selected parts of it.

The researchers did not find previous research which covered how one on a general scale can improve and evaluate PCG in games. Hence the following general research question have been formulated:

*How can machine learning be used to improve the gaming experience for players in procedurally generated games?*

The definition of a framework according to the IT standards and organizations glossary is the following: “*A framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful*” (TechTarget 2015). The structure of the framework will be conceptual, which means it will leave room for other tools and practices to be included, however in itself provides enough to complete the process. The framework will serve as the skeleton on which the procedurally generated content can be evaluated.

### 1.1 Research Contribution and Delimitations

The performed research is contributing with a framework that can be used for improving the gaming experience in procedurally generated games by evaluating how the procedurally generated content is affecting the gameplay.

During testing of the implementation of the framework, not all parts will be tested. The excluded part of the framework is “flow evaluation”. The researchers have chosen to collect data from one source instead of two in order to validate the framework, meaning that quantitative questionnaire data from players will be the input used for validation. How the framework was implemented and what selected parts that were used is described in further detail in chapter 5.

The expected outcomes of this study is a framework which can contribute to improvement with gaming experience in procedurally generated games, and with procedurally generated content. Also, a proof of concept where the main parts of the framework is implemented and tested will be runnable.
2 Related Research
There are many papers and articles about PCG and GA, but not much about the two of them in combination, where the GA is evaluating and evolving the PGC parameters to improve the gaming experience. Therefore, the related research here is only referring to PCG and GA, since there is little or next to no information about the subject to be found.

In this section, the theory for the methods used in the research is presented. The three main theories identified by the researchers that would make the foundation for the framework which was designed and developed to answer the research question, were the genetic algorithm, procedural content generation and flow theory.

Evolutionary computation is a common term within machine learning. There are different paradigms within the evolutionary computation family today: genetic algorithms (GA, which is also the focus of this paper), evolutionary strategies, genetic programming and evolutionary programming, to name the most common ones. The difference between them lie in the way they represent their schemes, how their selection method works, and what reproduction operators they use (Sivanandam & Deepa 2008).

The evolutionary computation algorithms take their inspiration from Charles Darwin’s evolution theory. The principle “Survival of the Fittest” is applied, where those in the population who can adapt to the environment are the ones that can live on. The individuals which gather the needed resources and have a high propagation are counted as “fit”, and will live on. Those in the population which cannot fulfill the basic requirements for survival, will have less descendents and eventually become extinct (ibid).

2.1 Genetic algorithms
The first scientist to write Genetic Algorithms through adapting and applying the evolutionary idea to optimization problems, was John Holland (1975). He wrote the book “Adaption in natural artificial systems”, in which he described this idea. His theory has since then been improved, refined and has become a powerful tool in the hands of researchers today for solving optimization and search problems.

Of the four evolutionary computation paradigms, the genetic algorithm is the most popular one. It is usually represented as a fixed-length bit string (see figure 2.1), which is equivalent to chromosomes in systems of biology, where each position in the string represents an individual’s feature.

![Fixed length bit string representation of the genetic algorithm](image)

The most common operator to use in the genetic algorithm is the cross-over operator. By taking two fit individuals (in this case bit-strings) and cross them with one another at a cross-over point, two new “offspring” will be produced (see figure 2.2). There are also other methods used such as inversion, where a part of the string is reversed, and bit-flipping mutation (see figure 2.3) where flipping one bit in the string will produce a new offspring (Sivanandam & Deepa 2008).
The first GAs only used one cross point when crossing individuals, thus not making it possible to give a new offspring the features from both the head and the tail of the string. If the best features lie in both the head and the tail, none of the new offspring will contain the best features. Hence the problem is solved by introducing two cross-over points (see figure 2.4). The outcome will here be two children, where the head and tail of the string is preserved, and the part in-between is crossed.

According to Goldberg (1989), reproduction is the process of selecting the fittest individuals from the population. This is referred to as the fitness function, which calculates how fit an individual is to solve a problem. The fitness function is used to maximize a value (such as goodness, profit or utility) with the outcome, that strings with higher values have a greater probability to be part of breeding offspring to the next generation. With the use of reproduction, cross-over and mutation, many practical problems can be solved with good results.

There have been observations that stochastic errors can occur in genetic algorithms, which can
lead to either a *genetic drift*, where the selection of fit individuals are favored, or *premature convergence* where the offspring no longer can outperform their parents, which leads to less diversity and a non-global optimum in the search (Kumar & Jyotishree 2012).

2.1.1 Selection Methods
Selection is when two parents are chosen from the population to breed (cross-over), and to produce better offspring. Within GA there are different selection methods one can use. Roulette Wheel Selection, Tournament Selection, Random Selection and Rank Selection, are the most common ones. When selecting individuals for crossing, the term *selection pressure* is a common term. The higher the pressure, the higher the chance that fit individuals are chosen, the lower the pressure, the lower chance that fit individuals are selected for crossing (Sivanandam & Deepa 2008).

2.1.2 Roulette Wheel Selection
As the name suggests, the concept of the selection is taken from the roulette wheel. The roulette wheel selection is one of the first selection methods used in GA, described by Goldberg (1989) in his book “Genetic Algorithms in Search, Optimization and Machine Learning”. Each individual (string or chromosome) in the population has a portion of the roulette wheel, where the size assigned to it is proportionate to its fitness. The table 2.1 shows a sample problem with four strings, with their fitness and total percentage. When applied to the roulette wheel, the wheel becomes weighted, where the more fit strings have more space assigned to them on the wheel, as shown in figure 2.5.

<table>
<thead>
<tr>
<th>String No.</th>
<th>String</th>
<th>Fitness</th>
<th>% of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>010101</td>
<td>94</td>
<td>9.7</td>
</tr>
<tr>
<td>2</td>
<td>111001</td>
<td>165</td>
<td>16.9</td>
</tr>
<tr>
<td>3</td>
<td>010110</td>
<td>256</td>
<td>27.2</td>
</tr>
<tr>
<td>4</td>
<td>110011</td>
<td>450</td>
<td>46.2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>965</strong></td>
<td><strong>100.0</strong></td>
<td></td>
</tr>
</tbody>
</table>

To create new offspring, the roulette wheel is spun for as many reproductions that are needed, where a minimum of two parents are required to create offspring. The strings with a higher fitness have a higher chance to reproduce (see table 2.1), and will therefore have more offspring in the following generation.
When a string has been selected by spinning the wheel, it is placed in a mating pool, where it will reproduce with other selected strings. After reproduction has been made, the strings in the mating pool are paired for mating, whereafter they go through cross-over (see figure 2.2), by selecting a random cross-over point in the string, where the starting point for the cross-over cannot be the first character, since in that case the strings would only be swapped, thus not creating a new offspring. After cross-over, there is a small probability that mutation takes place, to ensure genetic diversity (Goldberg 1989).

If the fitness value of the strings differ too much, the roulette wheel will have problems selecting new strings for mating. For example if a string occupies 90% of the wheel, then it will be hard to select other individuals to the mating pool (Sivanandam & Deepa 2008).

2.1.3 Random Selection
Random selection takes a random parent from the population and puts it in the mating pool for breeding. The result from using random selection is unpredictable, and causes more disruption on average than roulette wheel selection (Sivanandam & Deepa 2008).

2.1.4 Ranking Selection
Ranking selection is used to solve the roulette wheel selection problem when the fitness strings begin to differ too much. In Figure 2.6, the last string takes up about 80% of the wheel, making it hard for other strings to be selected for the mating pool, thus causing a genetic drift. Ranking selection works by sorting the population according to fitness value. Each individual is then appointed a selection probability according to its rank, thus setting up the roulette wheel to work with rank instead of fitness value. The result is that each individual’s chance for selection is more evenly spread out on the roulette wheel as shown in figure 2.7. Ranking selection makes sure that convergence does not increase too fast, so that stagnation and premature convergence is prevented. However, this can have the side effect of slower convergence rate, thus finding the optimal solution will take longer time (Kumar & Jyotishree 2012).
2.1.5 Tournament Selection
In tournament selection, two individuals are randomly chosen from the population for a fight. The individual with stronger fitness, counts as the winner and is moved to the mating pool. Reselection for a tournament is allowed, therefore the individual that loses a fight, is placed back into the population again. Tournament selection is repeated until the quota for the mating pool is filled, and the strongest individuals can then reproduce and create new offspring (Koza 1992).

2.1.6 Mutation
Mutation is what makes it possible for the genetic algorithm to not be trapped within a local minimum. It modifies the population by randomly changing its current genetic structure so that diversity within the population is present at all times. Mutation can be performed by either bit-flipping as shown in figure 2.3, or by swapping two random chromosomes within the individual (Sivanandam & Deepa 2008).

2.1.7 Elitism
Elitism is the process of letting the strongest, or a few of the strongest individuals live on to the next generation. This brings a significant improvement to the performance of GA. If the best individuals are not selected to be part of the next generation, they can be lost due to mutation and cross-over, hence diminishing the GAs performance (Sivanandam & Deepa 2008).

2.2 Procedural Content Generation for Games
Procedural content generation (PCG) for games is the process of having the computer create game content algorithmically. It can also be used to determine what content which is interesting and present that to the player. PCG is not easy. Generating content for a game requires both computational power and the ability to determine both the cultural and the technical values of the PCG content (Hendrikx et al. 2013).

During the last twenty-five years, the amount of people required to create a game has increased tremendously. In the 1990’s, it took only a team of five to six developers about a year to create and get a professional game published, compared to the large triple-A titles today, which takes years to build, and hundreds of people who create and build the content required for the massive game-worlds which are present inside these games (ibid.,). To cut down on the production time needed for these games, many PCG techniques have been discovered to generate different kinds
of game content. It is today possible to generate cities with buildings and a road networks, elements such as water, fire, gases, textures for models, forests, and dungeons to name a few.

2.3 Flow

The beginning of the flow concept, stretches back to the 1970’s when Csikszentmihályi became fascinated how different artists got completely lost in their work. When observing painters, he discovered that they got so immersed in their work that they ignored hunger, thirst, and even their sleep. The curiosity to understand this phenomenon, became the starting point for his research (Nakamura & Csikszentmihályi 2001).

Csikszentmihályi (1996) describes how they during the studies of flow found that every flow activity had one thing in common: The sense of discovery and giving the feeling that the person was transported to a new reality. The person’s performance was lifted which led to a higher state of consciousness where the self grew. According to Csikszentmihályi, this growth is the key to flow activities. In figure 2.8, the arrowed line represents the state of which the person is in, and the grey area is the zone where flow is present. As long as the person’s skill is proportionate towards the challenge, he can experience flow. However, as soon as the challenge becomes too hard, due to the person lacking in skills, the feeling of frustration starts to sets in. On the other hand, if the person’s skills are high, but the challenge is too low, the person will experience boredom. The more a person practices, the more skilled and experienced he becomes. Therefore, the challenge has to be increased in order to keep the person from losing the state of flow.

2.3.1 The components of Flow

During the years of researching flow, Csikszentmihályi discovered eight main components which make up the state of flow (Csikszentmihályi 1996):

1. A challenging activity that requires skill
   Flow often occurs when an activity is “goal-directed”, thus bound by a set of rules which
requires an amount of skill to perform. Those who lack the skill to perform the activity at hand, will most likely find it frustrating, while a person who has the skills, but unaware of the outcome will find the activity challenging in a positive way.

2. **Merging of action and awareness**
   When a task arises which requires all of a person’s skills to be performed, that person will be completely immersed by the activity. No energy is left to comprehend any other information than which what the task at hand requires. All focus and attention is completely turned towards the activity being performed.

3. **Clear goals and immediate feedback**
   Csikszentmihalyi (1996) describes that clear goals with immediate feedback is what makes it possible to achieve complete involvement in a flow experience. Just as a tennis player knows the rules of the game, and after performing the move to get the ball to the other side of the tennis-court, get immediate feedback to whether he performed good or bad. The same rule apply to other activities as well whether it is playing music, rock-climbing or playing golf. When an activity has clear goals, the person performing the activity knows what needs to be done in order to reach the goals of the activity, and the immediate feedback tells them how well they are progressing. He further emphasize that, unless a person has a set goal, measures and recognizes the feedback leading towards the goal, the person will not be able to enjoy the activity.

4. **Concentration on the task at hand**
   One aspect of the flow dimension which frequently repeats itself, is the effect of forgetting the things in life which is unpleasant. Due to enjoyable activities requiring a complete focus and attention to be performed, it leaves no room for the mind to process any other information. Hence this becomes an important side-effect of this feature of flow. A mountain climber expressed it in the following way (quote from Csikszentmihalyi, 1996):

   "When you’re climbing, you’re not aware of other problematic life situations. It becomes a world unto its own, significant only to itself. It’s a concentration thing. Once you’re into the situation, it’s incredibly real, and you’re very much in charge of it. It becomes your total world."

5. **The paradox of control**
   Today there are people that practice dangerous sports which can lead to the loss of their lives, but still they continue to practice the sport. The reason for this is that people enjoy having a sense of control in difficult circumstances. However, it is not possible to experience the sense of control, unless there is an unpredictable outcome, meaning that the person performing the activity really is not in complete control.

6. **Loss of self-consciousness**
   In flow, one of the things that also happens, is that the focus upon oneself disappears. When performing a flow activity, the person experiences a union with the environment he is in, such as the mountain, the sea or the forest. The activities of flow is experienced in having clear goals, a challenge which is proportional to our skill and solid rules. When performing them there is very little opportunity for the person to feel threatened. However, during normal life, the self is faced with threatening situations all the time.
As Csikszentmihalyi (2013, p. 63) exemplifies: “For instance, if walking down the street I notice some people turning back and looking at me with grins on their faces, the normal thing to do is immediately to start worrying: “Is there something wrong? Do I look funny? Is it the way I walk, or is my face smudged?” Hundreds of times every day we are reminded of the vulnerability of our self. And every time this happens psychic energy is lost trying to restore order to consciousness.”

7. Transformation of time
When a person is experiencing an optimal experience, the sense of time is lost. Hours seem to pass in minutes, and any normal time measurement such as the clock or the shifting of night and day is irrelevant. Generally, people report that when performing a flow activity, time seems to pass much more quickly. However, there are also examples of when the reverse happens. For instance, ballet dancers reported that when performing a difficult move which only takes about a second, can feel like taking minutes to complete. The generalisation Csikszentmihalyi makes of this, is that the sense of time during a flow experience have little or no relation to how time passes by the clock. There are of course exceptions to this generalisation, when time is the essential challenge of the flow activity.

8. Experience becomes an end in itself
When all of these parts of flow comes together, people begin to enjoy their experience with the activity, it becomes autotelic. Autotelic is a Greek word built from two other Greek words: Auto - meaning self and Telos - meaning goal. Autotelic refers to an activity performed without expecting any future benefit from it. Simply doing the activity is reward in itself. For instance; If one teaches children for the purpose of turning them into good citizens, is not an autotelic experience. However, if one finds enjoyment from interacting with them through teaching, it is an autotelic experience.

2.3.2 Flow Measurement
According to Giovanni (2012) researchers agree that the original definition of flow by Csikszentmihályi is correct, but when it comes to measuring flow, they disagree upon how this best can be performed.

To provide an example of how a flow questionnaire can be performed when using our framework, the so called flow questionnaire, created by Csikszentmihalyi & Csikszentmihalyi (1988) is described in more detail below.

Flow Questionnaire
The flow questionnaire (FQ) was the first method for measuring flow. Based upon interviews from many different occupations, Csikszentmihalyi & Csikszentmihalyi (1988) selected the best and most clear descriptions from the interviews and condensed them into the first measurement method for flow.
Table 2.2 – Twelve dimensions related to flow experience
(adapted from Csikszentmihalyi & Csikszentmihalyi (1988))

<table>
<thead>
<tr>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>I get involved</td>
</tr>
<tr>
<td>I get anxious</td>
</tr>
<tr>
<td>I clearly know what I am suppose to do</td>
</tr>
<tr>
<td>I get direct clues as to how well I am doing</td>
</tr>
<tr>
<td>I can feel I handle the demands of the situation</td>
</tr>
<tr>
<td>I feel self-conscious</td>
</tr>
<tr>
<td>I get bored</td>
</tr>
<tr>
<td>I have to make an effort to keep my mind on what is happening</td>
</tr>
<tr>
<td>I would do it even if I didn’t have to.</td>
</tr>
<tr>
<td>I get distracted</td>
</tr>
<tr>
<td>Time passes (slowly...fast)</td>
</tr>
<tr>
<td>I enjoy the experience, and/or the use of my skills</td>
</tr>
</tbody>
</table>

As shown in figure 2.9, a FQ starts with describing three experiences of flow from different activities. It then asks if the person has experienced something similar. If yes, the person is qualified to describe situations where they have experienced a similar state

1. Please read the following experiences:
   “You are so involved in what you are doing, [that] you aren’t thinking of yourself as separate from the immediate activity….You don’t see yourself as separate from what you are doing.”
   “...the concentration is like breathing-you never think of it. The roof could fall in and, if it missed you, you would be unaware of it.”
   “Her reading is the one thing that she’s really into, and we read together. She reads to me, and I read to her, and that’s a time when I sort of lose touch with the rest of the world, I’m totally absorbed in what I’m doing.”

2. Have you ever experienced something similar (Yes/No)?

3. If yes, in which activities was it that you had similar experiences?

4. How frequent are you involved in these activities?

5. Please write the one activity of those you wrote down which best correspond to the quoted experiences from section 1.

6. On the following page there is a list of items one can feel when performing an activity. For each item, please rate which feeling you have when performing the one activity you choose in section 5.

7. Now, please do the rating (6.) again towards your job and then towards another normal daily activity such as family life, watching television or spending time with a friend.

Figure 2.9 – Flow questionnaire (Adapted from Csikszenmihalyi & Csikszenmihalyi 1998, and Giovanni 2012)
After the participant chose the activity which corresponded the strongest to the quoted experiences in the beginning, he will then rate the one activity he or she chose towards the twelve dimensions described in table 2.2. In theory, the person experiencing flow will report being immersed, not being bored, not having to make an effort to be able to concentrate, enjoying the experience and so forth. After rating the activity, the user is asked to repeat the process and rate their job, and another normal daily activity such family life or watching television. After the questionnaire is completed, the participants are asked to describe how they enter the state of flow, how they maintain it, and what ends it (Csikszentmihalyi & Csikszentmihalyi 1988).

Strengths and weaknesses of the flow Questionnaire
FQ has according to Giovanni (2012) four main strengths:

1. Estimating the prevalence of flow

2. FQ does not assume that everyone experiences flow generally or in a specific context, thus making it a more valid measuring method for flow.

3. Participants are asked to freely write down the activities where they have experienced flow, which makes it possible to use FQ to estimate how flow is prevalent in different contexts.

4. Due to asking the flow-experiencing participants to rate different aspects of their subjective experience, and how they perceived the skill and challenge in the performed activity, FQ allows for testing if flow occurs when skills and challenges are in a relative balance towards one another, and if the subjective experience is more predominant than in the states of boredom and anxiety.

When looking at FQ, there are three weaknesses to address. The first one is: Does the flow quotes describe a single description of a flow state? The original flow quotes were divided into different sections of the FQ, where one was designed to measure shallow flow and the other deep flow. By mixing the two of them, it creates an uncertainty whether the participant answered yes/no to deep or shallow flow (ibid).

The second weakness in the FQ is that no intensity of the flow experience was measured in the different activities performed. Even though there were scales measuring the experience intensity when performing an activity of flow, disregardless to if the respondent experiences flow when engaged in the activity due to people only experiencing flow a percentage of the times they are involved in the activity. The measurements of intensity, does not specifically measure the intensity of the flow in the performed activity (ibid).

Finally, because the participants only were asked to give their average skills and challenges in the flow activity they experienced the most flow in, there is no direct assessment of how the challenges, skills and the ratio between these two, influence the state of flow. The problem with an average rating, is that it is also affected by the frequency of how often flow is experienced in the respondent’s best flow activity (ibid).

Assessment of FQ
According to Giovanni (2012), the FQ method for measuring flow is a good tool to study the prevalence of flow. However, it is limited when it comes to inspecting how a subjective
experience is affected by skills and challenges. A FQ can also not measure the intensity of the flow experience in general or specific cases.

**Experience Sampling Method**
The experience sampling method (ESM), is a technique that collects data when individuals are performing their daily activities. The data collected contains what the participants was doing, and their experience of it. The ESM works by beeping the subjects at random times during the day, urging them to fill in an experience sampling form (ESF). In the form, the participants fill in what they are doing, where they are, and select how they feel from a list of thirteen different adjectives, how they perceive the challenge and skill of the activity, whether they are in control, and why they were doing the activity. ESF in combination with ESM has been used by different researchers in many different countries, thus providing the possibility for comparing cross-cultural results (Giovanni 2012).

**The Componential Model**
According to Giovanni (2012), the two most frequently used method for measuring flow today is the Flow State Scale-2 (FSS-2) and Dispositional Flow Scale-2 (DFS-2). FSS-2 measures intensity of flow as a state, while DFS-2 measures the intensity of flow as a domain-specific or general trait. The difference between the two questionnaires is small, where the difference lies in the instructions of how to think while answering the questionnaire. The flow state questionnaire gives the instructions to answer the questions thinking about the flow activity they just performed, while the dispositional questionnaire asks the participants to answer according to their general average flow experience of flow activities.

**2.3.3 Flow in Games**
Descriptions of when people have experienced flow in different activities, are identical to the experience players have when being immersed in games (Chen 2007). Flow elements such as losing the track of time, having clear goals and being completely immersed in the activity is present. After more than three decades of competition between game companies, today’s games include and uses the eight components of Flow. The feedback is immediate and the player is offered clear goals which he can overcome through mastering specific gameplay skills. When evaluating the quality of the flow experience in video games or other similar experiences, the duration of the Flow, is the most important factor which determines if a player has entered the zone of flow (ibid).

In the same way as with flow (described earlier in section 2.4), Chen (2007) emphasizes that the activity must have a balance between the challenge and the players skill. If the presented challenge is above the player’s skill, the activity can become overwhelming, thus generating anxiety. On the other hand, if the challenge does not engage the player, he will soon lose interest and quit the game.

Chen (2007) further emphasize that when designing a game, the center of focus must be upon how players can maintain flow through its duration. The larger the potential audience, the harder it becomes to design a challenge that fits the audience. This is due to the fact that no one experiences the same thing in the same way. When it comes to computer and video games, players will have different skills and expectations to what the challenge should be. Some games provides a static and narrow experience which keeps most players within flow, but is not fun for those with a novice or hardcore skill. When designing towards a large audience, the experience of players will differ, therefore the design must offer a challenge which fit the different players’ level of skill.
3 Methodology

This section presents the process of how the researchers developed the framework which would provide an answer to the problem statement and the research question. To distinct between the framework created by the researchers and the method framework by Johannesson & Perjons in the text, the term “methodological framework” will refer to Johannesson & Persjons.

3.1 Research Approach & Strategy

In order to find an answer to the research question, design science research was used. According to Hevner et al. (2004), design science takes the approach of creating an artifact to solve the problem at hand. Due to the research question being closely related to design science, the decision was made to use design science as research method. Design science research is technology-oriented and focuses upon creating things that serve human purposes (March & Smith 1995). The performed research designed an artifact in the shape of a framework, which also was implemented. Its purpose is to answer the research question which focus upon machine learning and procedurally generated games, and how they in combination can be used to improve the gaming experience for players, thus serving a human purpose. To develop an artifact fits very well with the research question, since it includes both machine learning and PCG, which are techniques used in both program and games development.

Since the researchers could not find any previous research about how to improve and evaluate PCG in games, a good foundation had to be laid out. This foundation for the framework was laid out by collecting and analysing existing theories which would serve as qualitative data input for the framework. When designing our framework, building blocks (see table 3.1) was found through analyzing data from the chosen theories. These blocks were identified as objects that could be used to extend the framework to the point where it could function in a way so that the problem and research question could be answered.

There are many articles written separately about PCG and GA, but not many which combine the two of them to evaluate and evolve the gaming experience. Since there is little or next to none written about this, we are adding something new with our research.

3.2 Method Research Framework

The design research methodological framework (see figure 4.1) used to describe the design process for developing our framework for improving procedural generation in games, has been adopted from Johannesson & Perjons (2014) book “An introduction to Design Science”. How the researchers applied the steps in methodological framework are described in chapter 3 (step 1 and 2), chapter 4 (step 3) and chapter 5 (step 4 to 5).
3.2.1 Explicate problem and Defining Framework Requirements

The first step in the design process, following the methodological framework shown in figure 4.1, is to explicate the problem. This step involves formulating the initial problem, justifying its importance and to investigate its underlying causes. After the problem has been explicated, the next step is to define requirements that identifies and outlines an artefact which can be a solution to the explicated problem (Johannesson & Perjons, 2014).

First the initial problem and its explication was laid out (see chapter 1). The second step was to define the requirements of the framework. The framework had to provide a solution for the problem and the research question, where machine learning was a central part. In order to do this, literature studies were performed for the different theories which had to be incorporated into the framework to serve as the knowledge base and foundation. During previous courses at the university, the researchers’ interest for machine learning had increased, and was what motivated the construction of the framework in the first place. The machine learning algorithm chosen for the framework was the genetic algorithm. Further what gave influence to the researchers framework was Csikszentmihalyi’s studies of flow (Csikszentmihalyi & Csikszentmihalyi 1988; Csikszentmihalyi 1990, 1996, 2013; Nakamura & Csikszentmihályi 2001) for measuring the gaming experience, and procedural content generation (Hendrikx et al. 2013), which could provide parameters that could be evolved by the GA (Sivanandam & Deepa, 2008; Kumar & Jyotishree, 2012; Koza, 1992; Holland, 1975; Goldberg, 1989). These three theories was not only referred or related to, but also applied as parts into the framework, which is described in further detail in chapter 4.

3.2.2 Design and develop the artifact

The next step (see figure 3.1), is to design and develop the artefact. When developing the artifact, input is taken from the previous step, where the outline of the artifact and its
requirements are set. The output, is an artifact which meets the outline and the requirements. In this activity, knowledge from research literature and other written sources make up the base.

Designing and developing an artifact has, according to Johannesson & Perjons (2014), four different sub-activities: (1) Imagine and Brainstorm, (2) Assess and Select, (3) Sketch and Build and (4) Justify and Reflect. Imagine and Brainstorm, is where the researchers come up with new ideas, or elaborate on existing ones which can be incorporated in the design of the artifact. In the second sub activity, Assess and Select, the ideas that were produced are assessed in order for the designers to choose one or several of them to serve as the base where the design can be further developed. The third sub-activity Sketch and Build is where the construction of the artifact takes place. The fourth and final sub-activity Justify and Reflect, the design decisions are argued and reflected upon by the researchers. Practically, these four sub-activities are performed iteratively and in parallel.

The process of designing and developing the framework, actually started by the researchers already in the beginning of their participation in a computer science programme at the university of Borås. Many ideas and existing theories were pondered and brainstormed upon during two years finally concluding in using the theories of: the genetic algorithm, flow, and procedurally generated content as the foundation for the design of their framework. After the theoretical foundation had been assessed and selected, the researchers acquired deeper understanding of how the theories worked, and how they could be used together to design the framework. The approach for sketching out the framework and other building blocks that would have to be added to it, was acquired by analyzing the three theories and their relations to one another, which then made it apparent what parts that was needed. After this process, the identified building blocks (described in further details in chapter 5) which were used to build and assemble the framework, is described in table 3.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Design</td>
<td>In the game design building block, the decision of what parts of the game that should utilize PCG is taken. It is the parameters for the PCG which the framework is working on to improve.</td>
</tr>
<tr>
<td>A Game</td>
<td>The game functions as the platform where the PCG is tested.</td>
</tr>
<tr>
<td>Players</td>
<td>Players are the testers of the game who provides the quantitative and qualitative feedback needed to evaluate how well the PCG is performing. The quantitative feedback is received in the form of a questionnaire from the players.</td>
</tr>
<tr>
<td>Flow Evaluation</td>
<td>Flow evaluation interviews or questionnaires with players about the gaming experience is performed with regular intervals to see how the current state of the PCG is affecting flow.</td>
</tr>
<tr>
<td>Data storage</td>
<td>The data storage is where the data for the genetic algorithm and the feedback from players is stored and updated.</td>
</tr>
<tr>
<td>The Genetic Algorithm</td>
<td>The genetic algorithm is the evolutionary engine that takes the players feedback and uses it to improve the parameters of the PCG.</td>
</tr>
<tr>
<td>Parameter Manager</td>
<td>The parameter manager has the purpose of passing PCG parameters from the data storage into the game.</td>
</tr>
</tbody>
</table>
3.2.3 Demonstrate the Artifact
The purpose of this step is to prove the feasibility of the artifact by demonstrating how it can be used in one case. Here it is mostly descriptive knowledge that is presented of how and why the artifact works (Johannesson & Perjons 2014).

In order to demonstrate our framework in one case, a prototype of a roguelike game was implemented where the approach taken was to implement and utilise the main parts of the framework consisting of all the framework parts. However as stated in the delimitations, actual “flow evaluation” and “data analysis” were left out. The framework was then applied to the roguelike game, which was built with the help of Unity game engine. The game was then uploaded to a website where anyone in the world could play and submit feedback about the current state of the game in the form of a questionnaire (see section 5 for in depth details how the framework was applied to the game).

3.2.4 Evaluate Artifact
The next step in the methodological framework by Johannesson and Perjons (2014) is to evaluate the artifact. It is here one determines the artifacts ability to solve the explicated problem, how well it solves it, and why. When evaluating an artifact, there can also be different sub-goals that might be of interest to investigate. Firstly, what are the functional and nonfunctional requirements of the artifact. Secondly, investigate the formalised knowledge about the designed artefact, and its utility. This can for instance be the main theories upon which it is built and its implementation principles. The goal here is to enhance, disprove or confirm the design theory. Thirdly, compare and study the artifact towards other similar artifacts which are aimed to solve the same or similar problem. Fourthly, investigate if there are any harmful or unintended effects when using the artifact. Finally, formative evaluation and summative evaluation are considered (ibid). Formative evaluation is where opportunities for further improvement of the design can be identified. The artifact is designed and evaluated iteratively so that improvements and corrections can be made. Summative evaluation assess an artifact when it has been completely designed and developed. Once the assessment has been made, it is used as the final assessment of the artifacts usefulness, and does not feed back into the design process as with formative evaluation. Both these methods can be used together to find out the artifacts utility by investigating the artifacts utility by comparing it to other artifacts and then in the end do a summative evaluation.

Johannesson and Perjons (2014) also points out that there are different evaluation strategies which can be used when evaluating an artefact. One approach is to use ex ante evaluation, and ex post evaluation to classify and characterise these strategies. With ex ante evaluation the artefact is evaluated without being used or fully developed. In this case, different experts in the area can be interviewed based on a specification and early prototype of the artifact. Ex ante evaluations can be carried out without large resources or access to big organisations, and can also be carried out swiftly. This makes them excellent for formative evaluations when assessing a prototype or an initial design so that feedback can be acquired for further refinement. There is however a downside when using ex ante evaluation, since there is probability of false positives in the results, thus making the artifact appear better than what it is. Ex ante evaluation only looks at a partially developed artifact, and can therefore not give results that are reliable for summative evaluations.
With ex post evaluations, the artifact has to be employed. This requires that the artifact is developed, thus making them less likely to produce false positives, since a completed artifact is being evaluated, and not a prototype. When using this approach, more resources are usually required and takes more time, where access to organisations and people who can do the evaluations is needed, thus making it suitable for summative evaluations (ibid).

Johannesson and Perjons (2014) emphasize that when performing the evaluation, it is possible to use artificial evaluations or naturalistic evaluations. Artificial evaluation takes places in an artificial settings, such as a laboratory, where the artifact is assessed. Naturalistic evaluation takes place in the real world, in the environment where the artifact was intended to be used and is assessed there. When using naturalistic evaluation, real people uses a real system to solve a real problems. Due to the research results coming from the real world, it is possible to transfer them to similar settings or to generalise them. Artificial evaluations have the same advantage as ex ante evaluations. They are inexpensive and quick to perform and can find out if the artifact is efficient or not by controlling variables within the artificial environment. But just as the case with ex ante evaluations, there is also the risk of producing false positives.

The implemented framework by the researchers was evaluated both by using formative evaluation and ex post evaluation. First, formative evaluation was used to develop their framework through iterations where improvements and corrections was made. From the beginning, GA was the main focus, due to being the machine learning algorithm centerpiece of this research. After iterating over different approaches that could serve to answer the research question, PCG and flow were added and the researchers found that these three could serve as the foundation for the framework, and would suffice to be able to help answer the research question. Secondly, ex post evaluation was applied by developing a roguelike game which utilized the framework with naturalistic evaluation, where the game was deployed to a website to see how the framework worked when applied in reality.

The roguelike game which utilized the framework showed that the framework had the ability to solve the explicated problem. As feedback was collected from participants, one could see how the PCG was affecting the gameplay experience. The participants replied through a questionnaire what parts of the PCG that was positive and the parts that were negative.

3.2.5 Ethical Considerations
Whenever studies involves people, it is important that they get to choose whether or not to participate in the research. The participants must not be put under psychological or physical harm and that the information gathered from and about them must be handled confidentially, and in some cases with anonymity (Robson 2011; Johannesson & Perjons 2014).

Since the framework is targeted towards games but also can be used by other software utilizing PCG, the ethics regarding the framework has very little impact on people and the real world. However, the framework has two parts where one has to have ethics in mind, (1) the questionnaire when collecting data for the GA, and (2) the flow evaluation. Both of these require interaction with people where they have to reply to questions and take part in the research. Both people and the information gathered must be handled according to research ethics so that the personal integrity of the participants is kept.
The data collected for the researched performed in this paper was collected by the questionnaire (see figure 5.2), in the roguelike artifact. There was no questions that asked for any personal data collection such as the players’ real name, age or gender, thus letting all participants to be anonymous. The only data collected was how the player experienced different parts of the gameplay, which in its turn is used by the genetic algorithm to breed better generations of individuals, hence, the collected data does not hold any sensitive information about any person, and can be handled without taking extra security measures, such as encryption or safe storing.
4 Developing the Framework

The purpose of this framework is to let the genetic algorithm based on user feedback improve a procedurally generated game, or components of a game which uses PCG. The design of the framework was inspired and built upon three theories. The (1) GA (Sivanandam & Deepa, 2008; Kumar & Jyotishree, 2012; Koza, 1992; Holland, 1975; Goldberg, 1989), (2) flow theory (Csikszentmihalyi & Csikszentmihalyi 1988; Csikszentmihalyi 1990, 1996, 2013; Nakamura & Csikszentmihályi 2001) and (3) PCG (Hendrikx et al. 2013).

The framework is composed of several different components which work together to achieve this. As seen in figure 4.1, there are different parts used in the framework: Game design, where the foundation for a PCG game is laid.

![Figure 4.1 - Framework Illustration](image)

A game which uses PCG (the dotted box at the bottom right), players who provide feedback for flow evaluation and questionnaires, a data-storage, the genetic algorithm, a parameter manager to pass data between the game, a data storage, and data analysis.

The main cycle of the framework starts by picking an individual of the current GA generation from the data storage. The individual consists of the procedural parameters which are passed into the PCG of the game. The game then generates the procedural content based upon these parameters, which in its turn affects the gaming experience. The next step in the cycle is a player who plays the game and evaluates how well the procedural parameters (the GA individual) performed. The evaluation is then placed in the data storage in the form of a questionnaire, and a flow interview can be performed to measure the player’s gaming experience.
4.1 Scope
Scope for this framework means on what level one will use the GA to optimise the PCG parameters. There are three different levels of scope for the GA optimisation: global scope, group scope and individual scope. In the global scope, anyone that plays the game will affect the GA in the optimisation process. In the group scope, one selects what type of group the GA optimisation should work with, this can for instance be age spans, gender or how familiar one is with playing computer games. In the individual scope, each player will only affect their own GA optimisation.

When looking on how different scopes can affect the GA process, the global scope would probably be the one that moves the quickest to make new generations, but would also be the one that takes the longest time for finding an optimal value, due to receiving data from many different people with vastly different opinions. However in the end, the general scope will provide the “mainstream” parameter optimisation. The group scope would have about the same effect as the global one, however instead of creating an average on all the people that played, one will get a more focused optimisation for a specific targeted group of people. The group scope will probably go through less generations compared to the global one, when finding the optimal parameters. Given that the targeted group have similar opinions about the game, it could lead to a good result for what the game should be like for them. In the individual scope, parameters will be optimised specifically by an individual, according to his opinions. However this process is also the slowest in reaching the best optimized parameters for the scope, since only one person is affecting the optimization.

4.2 Parameters
Selecting what parameters the GA will work on is an important part which comes in two steps. The first step is to determine what parameters from the game that the GA will work on. This is straightforward since the game have various types of PCG. It is here one choose what parts to optimize on and how the parameters should be represented that affects these parts. The second step is how to represent them inside an individual in the GA. Here the choice is made if there should be a one to one parameter connection, or if one parameter inside the GA should affect several parameters inside the game.

4.3 Applying the Genetic Algorithm
Once the design decisions have been made, and there is a data storage in place, it is time to determine how the genetic algorithm should work with evolving the parameters used for the procedural generation.

When the method of storage is place, the genetic algorithm itself has to be implemented. The following six steps must be in place to determine how the GA should function:

1. Setup Individuals
   The first thing to consider is what the first generation of individuals, (the first Adam and Eve) should look like. How many initial individuals should the first generation consist of? Should they be randomly generated, or created by estimation of what one think might be good starting values? Keep in mind that the more individuals that one use, the more feedback is needed to evaluate a generation.
2. **Cross-Over**  
Secondly, decide how many cross-over points to use. Should one or two cross-points be used, or a combination between the two?

3. **Mutation**  
Thirdly, decide what, when and how mutation should take place. What parameters that should be affected by it? How much should the mutation alter a parameter? What is the probability that a mutation takes place? These are the main questions to keep in mind while setting up the parameter mutation.

4. **Selection Method**  
The fourth step is to determine how the selection of individuals for breeding the next generation should take place. Choose one of the common methods for selection which seems to fit best for the task at hand.

5. **Fitness Function**  
The fifth step is to decide how the fitness of an individual is calculated. Usually, the better fitness an individual have, the better it is. Tie the fitness function to the feedback received from the players, since it is their evaluation which is the foundation of how well a particular individual of procedural parameters is performing.

6. **Elitism**  
Finally, determine how many of the best individuals of the current population that should be part of the next generation and copy them over. Also determine if, or how many of the better fit individuals that automatically should be part of mating in the current generation.

Once these parts of the genetic algorithm have been selected, the genetic algorithm can be implemented.

4.4 **Data Storage**  
Depending on what scope to use, the way to store the data can be different. If it is a global or a group scope, a database of some sort would be needed so that data can be saved to it from various locations. It is also important to have the history of all the data stored, so it later on can go through analysis. If the scope only is for a single person, there is no need for the data to be analyzed and can therefore be stored locally, and the data will be accessible locally.

4.5 **Data collection “demands”**  
Data collection can be done in different ways, such as automated or manual. Either the game has to collect the data, or the user has to fill in a questionnaire. Both requires specific timing of when to collect data so that the data collected will be as accurate as possible.

Automated gathering can be a bit easier to do in the sense that it won’t have to interrupt the user as they are using the program. On the other hand it is harder, given that one would have to guess how good or bad the user’s experience with the program is from changing values inside the code.

Manual data gathering, will interrupt the user as they use the program in one way or another, therefore one need to select a time to prompt the questionnaire when it affects the user
experience the least. It is easier to capture the user’s experience with the program through a set of questions, then by using automated gathering.

When designing the questionnaire, each question should affect one or more of the different parameters used in the procedural generation.

4.6 Analysis of the GA

When all points about the program is done the analytic part of the framework should be considered, reflecting how the data can be analyzed both in a quantitative and qualitative way.

4.6.1 Quantitative Analysis of Generated data

To do the quantitative analysis one look towards the automatically collected data or the data collected through questionnaires. The data will be stored inside the different generations that have been generated over time by the GA. Each of these generations consists of several individuals, holding the parameter data used by the PCG.

There are different analyses that can be made: (1) individual level, (2) generational level, and (3) all generations together. On the individual level, one see the data of a single set of parameters used to create the PGC. Each of the individuals will most likely more or less differ, which allows for analyzing and comparing them to one another to determine if different users experienced the program differently or not, and if an individual was better or worse than the other. On the generational level, generations of individuals are analyzed and compared to one another, by doing so, one can here gain the insight of what individuals’ that were good or bad in a specific generation. Finally, when looking at all generations together, the approach is to look at them from the beginning to the end, investigating if the individuals and their entries get higher and higher fitness values the further along the generations they come, thus validating that the GA have optimized the PCG parameters.

4.6.2 Qualitative Analysis of Generated Data

The qualitative analysis can just as the quantitative work be based on the gathered data from users through the questionnaire, however this will have to be written data in the form of comments that the user made. Other ways to do this would be through interviews or observations of the players, or a combination of the two.

4.6.3 Flow Evaluation

When performing a larger game project, one might want to do a more thorough evaluation to frequently measure the current state of flow the players experience when playing the game. In combination with the framework, this can be a powerful tool. By using either a qualitative or quantitative approach where flow evaluation of the current state of the game is performed at a regular basis, a history of the players’ gaming experiences is created. This experience history is an important indicator, which shows when the game was giving the best gaming experiences for the players.

If the experience history is showing a negative trend, it is a good time to analyse why the trend is negative, and what it was that made the experience to deteriorate. If most of the game is procedurally generated, it is possible to go back to a previous version of stored parameters in the data storage which performed better and reset the process from there.

To do the flow evaluation one would need to consider the eight main components of flow that Csikszentmihalyi (1996) brings up, so that a questionnaire or an interview can be constructed
which will capture the player’s flow experience. An important thing to have in mind, is that all
the components of flow are contributing to the flow experience, so no one single component on
its own can determine the how good the flow experience is. The components can also overlap
in some areas and be more or less interdependent on one another. Below is some suggestions
to what type of questions one can ask to each respective flow component while measuring the
player’s flow experience of a game:

The first component of flow talks about challenges that requires skill to complete. This could
be measured by asking the user about his experience with the game, if it required any skills or
if it felt like everything was left to chance. These sorts of questions are asked to see if, and how,
the players’ experience of skill and difficulty changes between the modifications made to the
game.

The second component of flow is merging of action and awareness. Here the question to ask is
how immersed the player was while playing the current version of the game. This will indicate
how well the current state of the game is at capturing the players’ attention, and whether or not
the game is at a point where it can keep them occupied for longer periods of time.

After immersion, the third component is to determine how the player experienced whether or
not the game had clear goals, and if there was immediate feedback. The questions to ask here
includes if the player knew what he needed to do, why he needed to do it and whether there
were any immediate feedback. If it’s a larger game the questions could be set up as categories
of different parts of the game, for instance the quest system, combat, dialogues or the shop
system of the game. The goal here is to make all important parts of the game easy to understand
and operate and to provide immediate feedback for the actions performed. The clear goal does
not have to be a big general one, it could be smaller things which adds to the experience as a
whole. Take the shop for instance, the clear goal is that the player should be able to buy and
sell items with ease. There should not be any doubts to whether or not an item bought or sold
or if the money was withdrawn. The immediate feedback has to verify for the user what the
result of the action performed were.

The following fourth component is concentration on the task at hand. Questions to ask here are
questions about the amount of concentration required by the player, if he forgot everything else
in life while performing the activity, if his mind was fully occupied with the task at hand. This
will show how well the game is at bringing the player into its world. If the player easily is
distracted by other things, the current version of the game is lacking in this area.

The paradox of control is the fifth component of the eight components of flow. What to discover
here is whether or not the user experienced if he was in control, and have him try describe what
it was that made him feel that way. This component is closely related to the challenge
component. What to discover here is if the player experiences control over the challenge the
game provides. If the challenge is too though, the user won’t feel in control.

The sixth component of flow is loss of self-consciousness. What one want to measure here is
the player’s awareness about himself. Find out if the player felt less aware about himself and
his problems and if the focus was moved from oneself to the task at hand.

The penultimate component of flow is transformation of time. Here the users experience about
how time went by while playing the game is investigated. Ask questions such as if the sense of
time was lost, if hours seemed to pass like minutes, if the player noticed the shifting between night and day. Generally when people are experiencing flow, time passes more quickly.

Lastly, experience becomes an end in itself, is the final component of flow where people enjoy their experience with the activity they perform. Here, the obvious question to ask is whether or not the player enjoyed the experience. Also, the experience of enjoyment for the player might have changed since the last version of the game, if so is the case and the chosen method is qualitative, then ask a follow up question to why the player think that is the case.

In section 2.4.3, Csikszentmihalyi’s flow questionnaire measurement is described in further detail. It also contains an example of an FQ which can be used to determine whether or not a participant experienced flow when performing an activity. The FQ can be modified to fit better towards one’s needs.
5 Demonstrate and evaluate artifact

The artifact the researchers created using the designed framework (see Chapter 4) can be described as a game as well as a server. The genre for the chosen game was a roguelike one, in figure 5.1 we show a screenshot of the implemented game. Roguelike games uses procedural generation for most of its game components such as levels, enemies and items, which made it a good fit for implementing and applying the designed framework to.

![Figure 5.1- Screenshot of the Roguelike game that was implemented](image)

5.1 Scope

The target scope selected for the roguelike game was a global one (see section 4.2), where all the players could send in their feedback about the current generation of the game. The feedback was presented in the form of a questionnaire in the game. The collected feedback was then saved in the server, and used by the genetic algorithm to produce new generations of individuals.

5.2 Parameters

Most aspects of the game was made to use some form of PCG. The level generation parameters controlled aspects such as room size, number of dead ends, the amount of chests on a level, the number of enemies per floor, their damage, health, scaling, loot drop and the amount of experience the player would get from killing them. Also the player’s start health, damage and scaling when leveling up were decided with parameters. Each of these parameters were stored as chromosomes in each individual in the GA.

5.3 Data Collection

For the data collection a questionnaire was put together, where each question was represented by using sliders. There were two different types of sliders to represent the different questions asked: First was a scale that went from a low to high, which could represent bad to good or the
reversed. The second one had the pattern of low-high-low, which could represent questions that have two “bad” sides to it, for example level size, which could be too small or too large. The Questionnaire (see figure 5.2), was displayed at the end of each round for the player to answer and submit. Also, if the player wanted it was possible to enter the questionnaire from within the game at any time, thus ending the current round to give immediate feedback. For instance if the current set of parameters created a level which gave an obvious unbalanced gameplay, there would be no need to continue playing, and feedback could be given directly without having to finish the current round.

![Questionnaire](image)

*Figure 5.2 - The Questionnaire used to collected feedback from the players*

### 5.4 Applying the Genetic Algorithm

When setting up the genetic algorithm, the following six steps as described in the framework (see section 4.4) was used to determine how the algorithm should function.

1. **Setup individuals**
   The approach chosen was to set up ten different initial individuals with different values which were estimated to be reasonable starting values for generating dungeons. These individuals were also tested and approved before accepted to be part of the initial population after the algorithm was fully implemented.

2. **Cross-Over**
   The number of cross-over points to use for the GA was debated, and in the end landed on that only one cross-over point were to be used. The reasoning was that development time would be shorter to use a single cross-over point and that there was no way of knowing before hand, if the evolution of generations would be better if a multiple cross-over implementation were to be used.
3. **Mutation**
Mutation for the individuals were chosen to be fairly low. Each individual had a 5% chance to randomly have one of its values slightly mutated each time a new generation was generated. The reason to keep the mutation fairly low, was to not introduce too much changes to the level parameters each generation, so that the players evaluating the current generation only would experience subtle changes between generations instead of drastic changes.

4. **Selection Method**
For the selection of individuals that would breed the next generation, tournament selection was chosen as the method to use. The individuals had to fight among themselves for the right to be part of the mating pool to produce offspring.

5. **Fitness function**
For the fitness function, the different chromosomes of the individual were given a scale between one, to seven, where the higher the number, the better the fitness. To calculate the fitness value for an individual all the chromosomes were summed together. Each chromosome in the individual was then tied to the different questions in the questionnaire which the players submitted, thus recording the chromosomes fitness based upon the players’ feedback.

6. **Elitism**
To ensure that the GA would not have to redo a lot of work, the most fit individual in the current generation was copied into the mating pool directly to be part of breeding the next generation. Also, instead of completely randomly selecting individuals for the tournament selection, the current three best individuals were chosen to fight randomly against other competitors, thus increasing the odds to bring the better individuals to the mating pool.

During the development a lot of thorough testing was done to ensure that the genetic algorithm was working properly. Many simulations of generating new offspring based on feedback was carried out, until it was performing satisfactory.

5.4.1 **Data Storage**
Since a global scope was selected for the game, a storage was needed for the data collected from players, as well as some place for keeping the different generations from the GA. It was important that all who played the game would be able to submit feedback from wherever they played, therefor a MYSQL database was setup to handle the storage of data.

5.5 **Results**
One of the purposes with the implemented artifact is to validate if the framework upon which it is built is working. In Figure 4.1, all the different components of the framework is presented, showing what must be included in the artifact. The implemented artifact is including all of the parts, except from flow as described in the delimitations, would not be part of the experiment.
When all parts of the framework was implemented in the artifact, the result was a functioning artifact that was delivering the expected output. The results from the implemented artifact is presented in the sections that follow.

5.5.1 The fitness of the individuals

In the conducted experiment, an individual included the parameters shown in the questionnaire (see figure 5.2), such the game level, player stats, enemy stats and object parameters. These parameters, also known as chromosomes in GA, together creates the size, difficulty and prerequisite for the player to succeed, thus determining how the player will experience the game.

After the first round of questionnaire feedback from the players, the results (see figure 5.3) for the individuals showed which ones that had more potential to proceed and become part of creating the next generation of individuals, through cross-over as described in section 2.1

![Figure 3.3 - Result of each individual’s chance in the first generation to be part of breeding the next generation](image)

- 29 -
5.5.2 Comparison between generations

The parameters of the second generation was generated based upon the feedback of the players. After the genetic algorithm had performed the tournament selection with the individuals in first generation, the remaining strongest individuals then crossed over with one another producing the next generation.

Table 5.1 and 5.2 shows the average difference between each chromosome in generation one and two. In order to examine the value changes between generation one and two, the graph in figure 5.4 was generated by calculating the total weighted average of all questionnaires for all individuals in generation one, using the results from figure 5.3 and figure 5.5.

As described in section 5.3, the questionnaire values differed on how to interpret what is good or bad on the scale. There are three scales used: (1) small to large, (2) easy to hard, where the middle of the scale is its optimum, and (3) Bad to excellent, where the higher the value the better.

In order to correctly interpret if a value in the bar graphs is good or bad, see table 5.1 for the scales, and table 5.2 for the mapping of the values.

<table>
<thead>
<tr>
<th>Value</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small to large</td>
<td>Too small</td>
<td>Small</td>
<td>A bit small</td>
<td>Good</td>
<td>A bit large</td>
<td>Large</td>
<td>Too large</td>
</tr>
<tr>
<td>Bad to excellent</td>
<td>Very bad</td>
<td>Bad</td>
<td>Somewhat good</td>
<td>Good</td>
<td>Very good</td>
<td>Awesome</td>
<td>Excellent</td>
</tr>
<tr>
<td>Easy to hard</td>
<td>Really easy</td>
<td>Like a walk in the park</td>
<td>Easy</td>
<td>Good</td>
<td>Hard</td>
<td>Nightmare</td>
<td>Hell</td>
</tr>
</tbody>
</table>

Table 5.2 – Bar graph chromosome mapping

<table>
<thead>
<tr>
<th>Scale</th>
<th>Bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to hard</td>
<td>Easy path</td>
</tr>
<tr>
<td>Bad to excellent</td>
<td>Game difficulty</td>
</tr>
<tr>
<td></td>
<td>Item useful</td>
</tr>
<tr>
<td></td>
<td>Experience gain</td>
</tr>
<tr>
<td>Small to large</td>
<td>Level size</td>
</tr>
<tr>
<td></td>
<td>Enemy attack damage</td>
</tr>
<tr>
<td></td>
<td>Enemy health points</td>
</tr>
<tr>
<td></td>
<td>Player attack damage</td>
</tr>
<tr>
<td></td>
<td>Player start health</td>
</tr>
<tr>
<td></td>
<td>Number of enemies</td>
</tr>
<tr>
<td></td>
<td>Number of chests</td>
</tr>
<tr>
<td></td>
<td>Item drop rate</td>
</tr>
</tbody>
</table>

When comparing the total average of all the questionnaires in generation one (see figure 5.4) to the actual chromosome data for the individuals in generation two, which were generated based upon the player feedback (see generation 2 in figure 5.1 and 5.2), the new individuals
chromosome data corresponds very well to the player feedback. The majority of values had a change towards the right direction. For instance, on average the players said that the enemy attack damage and health points was a little bit too high, resulting in both values slightly lowered. Also when looking at the player chromosome changes, the player attack damage and start health was both too low, resulting in slightly increased values in the new generation.

Based upon this data generated from the GA, the developed artifact built upon the researchers’ framework is working as intended, adjusting the parameters for the individuals in the expected manner, thus producing individuals tweaked upon the players’ feedback, hence improving the artifact towards providing a better gaming experience for the players.

Table 5.2 - Average chromosome values of generation one and two

<table>
<thead>
<tr>
<th></th>
<th>Average easy path</th>
<th>Average level size</th>
<th>Average number of enemies</th>
<th>Average game difficulty</th>
<th>Average enemy damage</th>
<th>Average enemy health points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation 1</td>
<td>28.2</td>
<td>14.3</td>
<td>22.4</td>
<td>0.298</td>
<td>5.1</td>
<td>11.5</td>
</tr>
<tr>
<td>Generation 2</td>
<td>29</td>
<td>17.2</td>
<td>19.9</td>
<td>0.335</td>
<td>4.8</td>
<td>10.4</td>
</tr>
</tbody>
</table>

Table 5.3 – Average chromosome values of generation one and two continued

<table>
<thead>
<tr>
<th></th>
<th>Average player attack damage</th>
<th>Average item useful</th>
<th>Average number of chests</th>
<th>Average experience gain</th>
<th>Average item drop rate</th>
<th>Average player start health points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation 1</td>
<td>3.9</td>
<td>0.285</td>
<td>4.9</td>
<td>0.308</td>
<td>0.286</td>
<td>48.7</td>
</tr>
<tr>
<td>Generation 2</td>
<td>4</td>
<td>0.299</td>
<td>4.8</td>
<td>0.266</td>
<td>0.414</td>
<td>49.1</td>
</tr>
</tbody>
</table>
Figure 5.4 - Total weighted average of all questionnaires for the individuals in generation one
Figure 5.5 – Average fitness for each individual in generation one
6 Discussion & Conclusions

To provide an answer to the research question: *How can machine learning be used to improve the gaming experience for players in procedurally generated games?*, we designed a framework which utilizes machine learning that tweaks PCG parameters of a game based upon player feedback, to improve the gaming experience for players.

When designing the Framework a creative process was used. The main parts of the framework took its inspiration from PCG (Hendrix et al. 2013), GA (Sivanandam & Deepa, 2008; Kumar & Jyotishree, 2012; Koza, 1992; Holland, 1975; Goldberg, 1989) and flow (Csikszentmihalyi & Csikszentmihalyi 1988; Csikszentmihalyi 1990, 1996, 2013; Nakamura & Csikszentmihályi 2001). These three theories were applied directly into the framework, and was not only related to or referred to, but actually used. When looking at the framework, we think that the main cycle is solid, where the genetic algorithm is evolving the PCG parameters based on the players’ feedback of the game.

The core engine in our framework is the genetic algorithm. As described by Sivanandam & Deepa (2008), the GA evolves a population of individuals by breeding the strongest among them with one another. This is performed in our framework based on player feedback. What most players think are the best individuals, are the ones most likely to live on into the next generation, and then breed among themselves to produce an offspring with an improved set of parameters. The GA served as one of the centerpieces in the design of the framework, without it, there would be no evolution for the individuals, no progress forward. When implementing the framework, GA was one of the tougher challenges which required more time and testing then the other parts. There are different weaknesses with GA, which mostly depends on what kind of implementation one uses. For instance, as mentioned section 2.2, if one uses single cross over point, it is not possible to provide a new offspring, carrying the features from both the head and the tail. Further, if one choose roulette wheel as selection method (see section 2.2.2), and the fitness value of the strings differ too much, the roulette wheel will have problems selecting new strings for mating. Then if one decide to use the ranking selection method (see section 2.2.4), which is designed to counter the roulette wheel problem, the side effect of slower convergence rate can take place, hence, finding the optimal solution will take longer time. In the end, it all comes down to what implementation one choose.

The role of PCG in the framework is to determine how game content is created and behaves. As Hendrix et al. (2013) describes, PCG can be used to generate many different things in games, as covered in section 2.3. PCG had a large impact on our design, since it is the procedural parameters that the framework is working with to improve the gaming experience for the players. Implementation-wise PCG was the centerpiece for the roguelike game artifact, used to evaluate the framework, since most things in the game are procedurally generated. It was also one of the main parts in our framework, where the procedural parameters were modified by the GA, and then sent to the game to be evaluated by the players.

Csikszentmihalyi (1996) describes flow as a state where a person is completely immersed in an activity, experiencing full enjoyment in what he is doing, ignoring everything else that happens around him. As mentioned in section 2.4.3, researchers agree that Csikszentmihalyi’s definition of flow is correct, however they are in disagreement on how one best can measure flow. The weakness therefore lies in what measurement method one choose to use. One cannot get a better measurement than what the measurement method provides. The purpose of flow in the framework is to measure the players’ flow experience from playing the game. By doing so once can ensure whether or not the modifications made to the PCG parameters by the GA is affecting
the players’ gaming experience in a positive manner. The impact flow brings to our design is that it serves as a second security layer. Instead of only relying on the GA questionnaire for evolving parameters according to player feedback, we wanted to make it possible to see if the player is kept in a flow state, and not only rely on changes in individual procedural parameters.

When looking for previous studies about how one can improve and evaluate PCG in games, next to nothing was found. Most articles were about PCG in games and how different PCG algorithms work, but none were related to improving and evaluating the PCG to provide a better gaming experience for players. We can therefore not provide a comparison between our research and what previous research says about the subject.

As described in the delimitation in section 1.1, flow was not exercised in this study, however, if one were to measure the players’ state of flow when they play the different sets of procedural parameters, one could in depth determine how their gaming experience was for that particular individual, and see which sets of parameters that performed the best. Thus by choosing what most players thought to be the best procedural parameters, the gaming experience for the general population of players will be improved.

In order for an individual to become optimized, it takes many generations. In our study, we can see that the framework and artifact works. However, in order to obtain better results, more time and collecting of player feedback is needed. Once enough data is collected it is possible to see more correlations between the questionnaires and the individuals for each generation. The more generations collected, the easier it is to see correlations.

When looking back on the performed study, we realized that we should have collected more data about what happened during the GA process, what was happening during the tournament selection. If this data was saved, we could have seen which individuals that was breeding the new generation and how it affected the outcome. However, this was not in our minds at that time, but would be a good tool to prove more insight to the results.

The conclusion of the study became both the designed framework (see chapter 4) and a proof of concept roguelike game that utilized the framework (see chapter 5). The roguelike game showed that our framework was working as intended (see section 5.5), where the procedural parameters for the game were tweaked based on the feedback given from players.

6.1 Methodological Reflection

There are advantages and disadvantages with the chosen method. The benefits of using design science research is that it gives the opportunity to develop new IT artifacts, contributing to the field which the research being performed lies within. Also it is appealing to researchers who likes to do technical development. There are a couple of disadvantages with the method. One is that it can be hard to generalize into other fields, due to design science research often is done towards a specific field where the developed artifact is narrowly tailored to fit the conducted research. Another disadvantage is that it can be risky if the developers lack the competence needed to perform the development, thus resulting in a non-finished artifact that cannot be used for the intended research.
6.1.1 Generalizability

The framework has good but limited generalizability. Procedural generation is not locked to any game genre, so therefore the method is applicable to any game genre which procedurally generates its content. The framework is also not limited to games only, even though it is targeted towards them. It is possible to plug in any other kind of program which uses procedural generation to generate its content to evolve the procedural parameters with the help of GA. However, the framework is only applicable to games and programs which uses PCG. The limitation of the framework thereof is that it only can be applied to PCG.

6.1.2 Validity and Reliability

In the performed study, the framework was designed upon the theories of flow, PCG and GA. These theories were not only referred to or related to, but actually used. To ensure its validity, the framework was then implemented as an artifact in the form of a roguelike game, and then uploaded to a website where anyone could play and contribute with feedback of how well they thought the PCG performed.

The framework is built so that both validity and reliability can be applied when using it. To ensure measurement of validity, players have to evaluate their gaming experience through either a questionnaire or an interview which corresponds to PCG parameters, which later on is used by the genetic algorithm to breed a new generation of individuals. The flow evaluation, performed periodically also ensures that the game is proceeding in the right direction, thus providing validity to the gaming experience as a whole.

In order to increase reliability, there has to be enough quantity of data collected from players to see what most of them agree upon being the best experience. However, the responsibility lies upon the users of the framework to apply good sampling when performing their tests to ensure that their measurements are reliable.

6.1.3 Reproducibility

The framework itself is described and presented in this research. The methodology of how it was designed (see chapter 3), what theories (see chapter 2) it is based and applied upon is well documented. In chapter 4, the development process of the framework and its different parts are described. Hence, it is therefore possible to reproduce the framework.

6.2 Future Work

The research performed contributes with a framework that can be used to improve the gaming experience for players in procedurally generated games. This is done by evaluating the PCG with the GA, based on players’ feedback. As mentioned in section 1.1, the study did not include the data analysis and flow evaluation parts when testing the framework through the roguelike game, since they validate the PCG of the game and the gaming experience, and not the designed framework itself. In this section ideas are presented for what future work that can be done to improve and take the research further.

Different age spans

An interesting aspect would be to use different age-spans when evaluating the PCG parameters to discover what fit best for different generations. It would be very interesting to do a study which involved kids, teenagers and adults, and see what would make the game fun for them.

Different people by interest
Another study that could be done is to take in people by interest. Instead of naturally targeting gamers who already like to play games, it would be interesting to see what people who normally don’t play games think makes the game more or less fun.

**Qualitative research with observation**

In the research performed, the measuring flow was not carried out. If one would conduct qualitative flow interviews where the participants flow experiences was sampled and combine it with observation of how the participants react and respond while playing the game, a more in-depth study of what made the game more or less fun to play can be performed. The important part here is to do the interviews and observations with regular intervals so that the different changes made by the genetic algorithm are sampled. This creates a history that can be examined and analyzed which shows when the GA procedural parameters was the best by comparing them to when the participants had their best experience with the game.

**Applying the framework to a fully featured game**

One of the most interesting studies to try the framework with, would be an already fully featured game which utilizes PCG. Either if the prototype would be fully developed, or if an established game company would be willing to cooperate to tweak their PCG. This makes the research a whole lot larger, but will hypothetically give much better results.
References


Högskolan i Borås är en modern högskola mitt i city. Vi bedriver utbildningar inom ekonomi och informatik, biblioteks- och informationsvetenskap, mode och textil, beteendevetenskap och lärarutbildning, teknik samt vårdvetenskap.

På institutionen Handels- och IT-högskolan (HIT) har vi tagit fasta på studenternas framtida behov. Därför har vi skapat utbildningar där anställningsbarhet är ett nyckelord. Åmnesintegration, helhet och sammanhang är andra viktiga begrepp. På institutionen råder en närhet, såväl mellan studenter och lärare som mellan företag och utbildning.

Våra ekonomiutbildningar ger studenterna möjlighet att lära sig mer om olika företag och förvaltningar och hur styrning och organisering av dessa verksamheter sker. De får även lära sig om samhällsutveckling och om organisationers anpassning till omvärlden. De får möjlighet att förbättra sin förmåga att analysera, utveckla och styra verksamheter, oavsett om de vill ägna sig åt revision, administration eller marknadsföring.

Bland våra IT-utbildningar finns alltid något för dem som vill designa framtidens IT-baserade kommunikationslösningar, som vill analysera behov av och krav på organisationers information för att designa deras innehållssstrukturer, bedriva integrerad IT- och affärsutveckling, utveckla sin förmåga att analysera och designa verksamheter eller inrätta sig mot programmering och utveckling för god IT-användning i företag och organisationer.

Forskningsverksamheten vid institutionen är såväl professions- som design- och utvecklingsinriktad. Den övergripande forskningsprofilen för institutionen är handels- och tjänsteutveckling i vilken kunskaper och kompetenser inom såväl informatik som företagsekonomi utgör viktiga grundstenar. Forskningen är välenkommerad och fokuserar på inriktningarna affärsdesign och Co-design. Forskningen är också professionsorienterad, vilket bland annat tar sig uttryck i att forskningen i många fall bedrivs på aktionsforskningsbaserade grunder med företag och offentliga organisationer på lokal, nationell och internationell arena. Forskningens design och professionsinriktning manifesteras också i InnovationLab, som är institutionens och Högskolans enhet för forskningsstödjande systemutveckling.