# **Automated Creation of Puzzle games with Constraint Programming**

# **Abstract**

# **Contents List**

[Automated Creation of Puzzle games with Constraint Programming 1](#_Toc520459215)

[Abstract 2](#_Toc520459216)

[Contents List 3](#_Toc520459217)

[List of figures 4](#_Toc520459218)

[Acknowledgements 5](#_Toc520459219)

[Chapter 1 Introduction 6](#_Toc520459220)

[Chapter 2 Context 6](#_Toc520459221)

[Chapter 3 Requirements and Analysis 6](#_Toc520459222)

[Chapter 4 Design and Implementation 7](#_Toc520459223)

[Chapter 5 Testing (or Results Evaluation) 8](#_Toc520459224)

[Chapter 6 Conclusions and Project Evaluation 8](#_Toc520459225)

[List of References 8](#_Toc520459226)

# **List of figures**

# **Acknowledgements**

# **Chapter 1 Introduction**

– Right at the start, very clearly outline the problem you are working on, why it is interesting and what the challenges are. Make sure your reader is clear on what your project is about after reading the first few paragraphs. Don’t begin waffling on about how technology is changing the world, how great the internet is, or how you intend to revolutionise computing!

– List your aims and goals. An aim is something you intend to achieve (e.g., learn a new programming language and apply it in solving the problem), while a goal is something specific you expect to deliver (e.g., a working application with a particular set of features).

– Give an overview of how you carried out the project (e.g., an iterative approach).

– A brief overview of the rest of the chapters in the report (a guide to the reader of the overall structure of the report).

This chapter is relatively short (2-4 pages) and must leave the reader very clear on what the project is about and what your goals are.

There is a history of generating puzzles using A.I., from Chess problems to Sudoku levels. Sudoku is a classic puzzle which is also named as “the Rubik’s cube of the 21st century” and it became popular in Japan in 1986. Then Sudoku achieved international popularity in 2005. 【3】Sudoku, Chess and other puzzles can be described by constraint problems and then programmer can establish corresponding models based on these constraints. Additionally, models could give an objective measure of the difficulty of a puzzle instance with grades such as from the easiest to the hardest. Then programmers apply this measuring method to problem instances for the public.【1】

Besides, recently there has been progress in generating puzzles, and also measuring the difficulty of these puzzles, for a human player. In 2017, the AI player, Alpha Go, had beaten Lee Sedol and this remarkable competition also represent a result that it is nearly impossible to win an AI player for puzzles. Alpha Go had greatly promoted the solving usability of AI player to solve puzzles. In this report, I would introduce an effective way to solve puzzles, constraint programming. People establish constraints, describing puzzles, as corresponding models to solve puzzles. There are various kinds of puzzles and I would implement one classic puzzle named “match three” in this project. This project aims to solve “match three” puzzles and measure the difficulty of “match three” puzzles.

The main aim of this project is to automatically generate puzzles and create engines to solve puzzles. This project could involve formulating “match three” rules, implementing simulations of “match three” puzzles, and performing experiments which measure how well the difficulty measure lines up with real users. Through implementing engines of solving “match three” puzzles, puzzles would be divided into three difficult levels ---- easy, medium and hard. Then, real users try to solve the same puzzles and record their time spending in solving puzzles and also divide puzzles into three difficult levels. Compare the results performed by computers and humans and evaluate the engine ability on classifying puzzles difficult levels.

“Match-Three” is a typical puzzle video game where the player manipulates tiles in order to make them disappear according to a matching criterion. The matching criterion is that there are at least three tiles of the same type adjoin each other [1]. A field has a N×M matrix of spaces, where in N and M are integers greater than three, and wherein each space of the matrix includes one of a plurality of different items. Then an object is allowed to exchange to the nearest object once three are three or more identical items results in the items . Finally these items would be removed from the field.

Because “match-three” games are very simple games with a very limited number of rules, even the younger children or the elder people can easily understand the rules. Then these “match-three” puzzle games become more and more popular. Bejeweled‎, Candy Crush Jelly Saga and Ruby Blast are the most famous “match three” puzzle games that the number of downloads from app store have more than billions of times [2] [3]. Many popular games are all based on this puzzle and the most famous series is “Candy Crush”. According to the reported by Wall Street Journal in 2013, nearly 15 million people in Western countries were addicted to Candy Crush Saga [4]. These data show that people are enjoy playing “match-three” puzzle games. Thus, in this project, I would also use “Candy Crush” as the typical “match three” puzzles to implement the following experiments which could measure the difficulty levels of puzzles.

根据XX在XXX所写，我们追求AI主要有两个目的。第一个是为了让我们更好地了解intelligent entities,然后同时更好的了解人类自己。毕竟我们认为人是intelligent的。第二个就是build intelligent entities,更多的是 for fun and profit.因为这些intelligent entities， 人类的生活变得更加有效率也更佳丰富多彩。【2】而本文恰恰就是二者的结合，使用constraint programming来解决puzzle games正是为了fun and profit.而如何得到这样的artificial engine，就需要我们了解什么是我们需要的entities.

Russel and Norvig [Russel 95] define AI as the creation of computer programs that emulate four things: 1. thinking humanly 2. thinking rationally 3. acting humanly 4. acting rationally. 而这四个方面，我们之前比较追求的是rationally thinking and acting.因为这样我们能够得到尽可能聪明的AI，比如alpha go. 在thinking humanly上，很多只是为了通过Turing test使得人们无法分辨人和机器的差别。而在 acting humanly上，人们又往往是在机器人上努力，力求达到一个真假难辨的机器人。但是在真正的有趣的游戏里比如象棋，一个人类想要和机器人下象棋。由于现在技术的发展，人类已经很难下赢机器了，因为机器已经把thinking rationally 发挥到了极致。所以在它把所有可能性都排列出来以后，以人类的现有思维方式很难赢。因为大多数人类下象棋并不是为了学术或者训练，而是为了fun， most of them want to be soundly beaten every time. The one of most significant aim of playing game is to obtain a great player experience(PX) [5]. However, it is not easy to evaluate the player experience using any device or other physical experiments.

Elisa and her group members has introduced a great experiment design and analytical method for measuring enjoyment which had been identified as a central component of the player experience [5]. However, PX has various, overlapping concepts so that it is difficult to develop valid measures and a common understanding of game enjoyment. Firstly, they structured the review of 87 quantitative studies into general methodological observations in an experiment. Then according to the “Purpose of the studies”, “Participants”, “Games and Genre”, “Study setting, gameplay duration and game metrics”, and “Measuring point of the critical element of experiment” these five aspects, more in-depth measures taken by the institute. Finally, the determining factors affecting the enjoyment of the game are Game System, Player and Context and Relationship between enjoyment and other PX components such as flow, presence and immersion. Thus immersion is great critical factor of a great player experience. 那么换一句话说，玩家对于一个游戏有着想要不断玩下去的欲望，也就能够证明玩家在这个游戏中得到了很好的游戏体验。

所以在象棋游戏中，需要一个机器人with a suitable challenge, but doesn’t overwhelm the human by always making the best move. 这样的对手需要有一点点的短视，偶尔还会犯一些错，就像人类一样或者比人类的正常水平略低一些，配合人类的水平不断的调整自己。这样的象棋比赛才是大多数人类为了fun 和放松说需要的。但是在非对抗游戏中，比如“candy crush”就并不需要这样的一个机器人。但仍然需要的是一个能被玩家攻破的棋局，那么这就需要给游戏分难度了。所以给游戏分难度是一个可以快速提升游戏者体验的一个方法。这也和现在市场上的大多数游戏的开发十分契合。

# **Chapter 2 Context**

This chapter should cover background information, related work, research done, and tools or software selected for use in the project.

– Provide necessary context and background information to describe how your project relates to what is already known or available.

– A description of the research carried out to learn out about the nature of the problem(s) being investigated and potential solutions. The form of the research will vary widely depending on the kind of project. For example, it might involve searching through research publications and online resources, or might involve an exploration of design possibilities for a user interface or program structure.

– The sources of information you are drawing on (papers, books, websites, etc.) should all be cited or referenced clearly. In addition, state how each source relates to your work and avoid the temptation to pad out the chapter by including sources that you didn’t make use of during the project.

– If relevant, a survey of similar solutions, programs or applications to yours, and how yours is differentiated. – Introduce the software, programming languages, library code, frameworks and other tools that you have used. Discuss the available choices and make clear which you made use of and why.

You should not include well known things (e.g., HTML or Java) or try to give tutorials on how to use a tool or code library (use references to books and websites for that information). Everything you include should be directly relevant to your work and the relationship made clear. This chapter is likely to be fairly substantial, perhaps 8-10 pages.

首先介绍一下match three。

Match Three games are a type of casual puzzle games. The major task consists in forming lines/chains/groups of 3 or more same identical tiles. The traditional game board is square-patterned and filled with various tiles which could shift, select or rotate. Eugene Alemzhin created the first Match Three game by swapping adjacent balls named “Shariki”in 1994 which was released for DOS. If there is no more possible matches in the board, then the game is over with current score shown in the screen. Then this kind of influential game led to the popularity of “Match Three” puzzle games. Nintendo published “Panel de Pon” in 1995 and “Tetris Attack” in 1996. The former one is the original version released and the following one is the first game in Puzzle League series. In 2000, “Pokémon Puzzle League” developed by Nintendo which features the same gameplay as in “Panel de Pon”. However, different to the predecessors, it was developed with a 3D mode instead of the traditional 2D mode. With the development of mobile phones, people realized that they would like to play more mobile games, a convenient and relaxing kind of gameplay.

In 2001, “ Bejeweled”, the first most famous movile “Match Three” game released by PopCap Games. There are more than 150 million times of Bejeweled downloaded from App stores. The objective of this game is to swap one gem with an adjacent gem to form a horizontal or vertical chain of three or more gems. If there are more than three gems connected, it provides bonus points. Once these gems are connected, they would be disappear and fill new gems in. That would lead to chain reaction, called cascades, are triggered, where chains are formed by the falling gems. Cascades bring more bonus points as well. Additionally, there are two main goals for players, including complete a certain score in a limited time or limited steps. Since the great popularity of Bejeweled, more and more “Match Three” puzzle games appear in App stores, such as Candy Crush Saga, Bubble Witch Saga, Jelly Splash. Those games have long since caught on all over the playing planet.

The reasons why a huge number of people are addition to “Match Three” puzzle games as listed.

* **人口统计学**：与传统视频游戏相比，休闲游戏更倾向于女性和35岁以上的受众。（IGDA 2005，第11页）
* **发行**：休闲游戏主要由用户下载，一般下载大小不超过10MB。
* **硬件**：按照惯例，休闲游戏的目标是低端机器和旧机器。在撰写本文时，新的休闲游戏仍然支持Windows 98。
* **经济模式**：休闲游戏主要可以通过先试后机模型下载，玩家通常可以在60分钟内玩完整游戏，之后玩家必须付费才能继续玩游戏。
* **允许短暂的比赛**：大多数休闲游戏可以在很短的时间内进行; 开始游戏需要很短的时间，并且通常很容易打断游戏会话。这并不意味着玩家实际上总是玩短暂的会话：在Trymedia网站的一项调查中，66％的玩家报告他们的典型游戏时间超过一小时（Macrovision 2006）。关键是休闲游戏*允许*短暂的游戏，因此玩家更容易投入游戏。
* **自动保存**：即使玩家关闭游戏窗口，大多数休闲游戏也会自动保存，因此玩家可以在以后轻松放下并恢复游戏。自动保存可能会让玩家更容易在许多情况下玩游戏，在这种情况下，无法进行更传统的游戏 - 例如在工作场所。
* **鼠标控制**：休闲游戏几乎完全由鼠标控制。尽管存在很少的硬数据，但轶事证据表明，休闲游戏玩家发现使用键盘控制游戏非常困难。
* **非常简单的规则**：Steve Meretzky说应该可以用三句话来陈述休闲游戏的规则。（Barwood＆Falstein 2006，＃107）
* **适度创新**：学习休闲游戏必须非常容易。这往往意味着休闲游戏几乎是现有游戏的克隆，具有新的图形，或者创新发生在小的增量步骤中。
* **多个级别的成功：**大多数休闲游戏通常会奖励玩家以更具挑战性的方式完成子任务。在匹配的平铺游戏中，通常有奖励用于制作组合（同时几个匹配）以及用于匹配比所需更多的区块。
* **很多积极的反馈**：休闲游戏往往旨在为玩家提供早期成功的经验。（Barwood＆Falstein 2006，＃107）
* **很少的负面反馈：**与其他游戏类型相比，休闲游戏通常非常*容易*，并避免因错误而惩罚玩家。

因为此次项目是用“Candy Crush Saga”，所以也要简单介绍一下。“Candy Crush Saga”是由King在2012年研发的三消游戏，这是由网页游戏“Candy Crush”发展而来的移动端游戏，有IOS， Android， Windows Phone 和Windows10等不同游戏版本适应各种不同平台。“Candy Crush Saga”是以糖果为主题，基本规则是将三至五颗一样的糖果排成一线消除糖果得到分数。 游戏需要把不同的糖果组在一起并引爆，会产生不同效果威力。当串消糖果产生连击的期间，玩家无法移动其他的糖果。游戏采取逐一解锁的方式进行，每一关样式不同，随机掉落糖果。糖果也有很多种，首先基本的board是由普通糖果组成的。而普通的糖果也有六种，分别是红色糖果 – 果冻，豆橙色糖果 – 润喉糖， 黄色糖果 – 柠檬糖， 绿色糖果 – 口香糖， 蓝色糖果 – 棒棒糖 和 紫色糖果 – 果汁糖。并且还有一些特殊糖果，条纹糖，包装糖和彩糖炸弹。条纹糖有两种，横纹和竖纹。就像下图所示，当横向移动糖后，四个糖果消失，然后可以生成统一颜色的横向条纹糖，如果是纵向移动，那么生成的是纵向条纹糖。横纹的条纹糖可以消灭糖果所在位置那一整行，竖纹的条纹糖可以消灭那一整列。第二种特殊糖果是包装糖，它可以由五个L型或T型的同色糖果组合消除而生成。包装糖被消除后，可以产生两次3\*3的爆炸，消除附近所有糖果及障碍物。最后一种是最大威力的特殊糖，彩糖炸弹，五个同色糖果在同一列或者同一排消除才可以生成一个彩糖炸弹。只要彩糖炸弹与相邻的任一普通糖果交换位置，即可消灭场上所有该颜色的糖果。

# https://upload.wikimedia.org/wikipedia/commons/thumb/f/f4/Striped_candy_formation.svg/260px-Striped_candy_formation.svg.pnghttps://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wrapped_candy_formation.svg/320px-Wrapped_candy_formation.svg.pnghttps://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/Color_bomb_formation.svg/260px-Color_bomb_formation.svg.png

|  |  |  |
| --- | --- | --- |
| **糖果A** | **糖果B** | **效果** |
| **条纹糖** | **条纹糖** | 消灭一整行、一整列的糖果 |
| **条纹糖** | **包装糖** | 变成巨大糖果，并消灭3整行，3整列的糖果。 |
| **条纹糖** | **彩糖炸弹** | 如果彩糖炸弹和条纹糖交换位置，那么该颜色的所有糖都随机变成横向或者纵向的条纹糖，然后都会爆炸消除。 |
| **包装糖** | **包装糖** | 5\*5的大爆炸两次 |
| **包装糖** | **彩糖炸弹** | 如果彩糖炸弹和包装糖交换位置，那么该颜色的所有糖都变成该颜色的包装糖，然后爆炸消除。 |
| **彩糖炸弹** | **彩糖炸弹** | 可以清除整个画面的糖果 |

虽然有医生认为沉迷玩该游戏容易引发多种健康问题，如老花、颈椎移位、肌肉发炎等。并且专业心理辅导师李馨君指出过度沉迷在玩该游戏容易引发多种心理及生理问题，对青少年产生负面影响。但我们仍然不能否认这款游戏带来的乐趣和满足。“电玩Candy Crush 全球一天玩7亿次”这样的新闻也恰恰证明了这款游戏的魅力所在。

Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games

http://papers.www2017.com.au.s3-website-ap-southeast-2.amazonaws.com/companion/p465.pdf

除了以上对于“candy crush saga”的介绍，我们还需要了解如何去设计各个关卡。这就带来了两个问题。

第一个， 为什么我们需要设计不同难度的关卡，这会对游戏带来什么影响？

第二个，如何认定这些关起的难度，有什么criteria可以作为参考标准？

对于第一个问题，涉及到了PX。而之前在elisa 和她的团队的report上得出的结论是很难有什么具体的工具可以测量出PX的指数，但是沉浸感是一个很好的衡量标准。那么对于沉浸感，可以被转化成投入的时间长短或者玩游戏的次数多少。在一款游戏里花的时间越多，玩的次数越多，沉浸感越强。也就是说，用户在此游戏里得到了很好的PX，所以才会不断地花费时间和精力在这款游戏上。类似的实验是由Su Xue, Meng Wu and their mates 做的一个关于difficulty和engagement的关系的实验。他们convinced themselves of a causal link between the difficulty and engagement， 所以a DDA framework with a global optimization objective of maximizing a player’s engagement throughout the entire game. Then they presented “Candy Crush Saga” and “Bejeweled” with Dynamic difficulty adjustment(DDA) implementations. 然后发现了两个很有价值的关系图。 第一张图里Retained population (red line) at a level is the number of players who have achieved this level as the highest one. There are players churned at each level, thus the retained population decreases as the level increases. The difficulty (blue line) is measured by the average number of trials that are needed to win this level. The more trials it takes, the more difficult this level is. 然后同样分为三个难度级别， easy（<=20）, medium(21-80), hard(>80)。首先是整体难度在升高，而且很多players就会因为难度而停滞不前。第二是在每一个小分段力，比如40-60关，蓝线里面出现了7次小高峰，则表明每3局比赛左右就会出现一局很难的关卡，之后又关卡难度恢复到比较简单或者中等难度。第二幅图是在第46关，the random seed of board initialization。因为初始游戏界面不同，难度级别也就被改变了。Su和她的团队，使用了0-99这样的random seed，然后得到第二张图。可以明显看到同一局的难度系数也在改变，从低到0.15到高到0.75。这个difficulty range可以说是从easy跨越到了hard。而且通过发布游戏后得到的实验，他们得出结论： While existing DDA systems adapt game difficulty in a greedy manner for local benefit, our method maximizes the player engagement throughout the entire game. 所以说一个优化关卡难度设计的游戏会对PX有非常深远的影响。这也就是为什么我们需要设计一个合理的游戏难度给玩家。

为了增加游戏的趣味性，势必要将游戏设计得更加巧妙，比如增加一些游戏道具或者是改变游戏板的样子。 因此我们可以设计成千上万种不同难度的关卡。但同时，我们怎么给这些关卡认定难度又是另外一个问题了。因为我们人工给成千上万种游戏判定难度既低效率又很可能片面或者需要很多人力。那么给这些不同的关卡判定难度的必定是机器本身了。这就涉及到另一件事，机器判定游戏的难度是否和人类所认为的难度是一致的呢？

我认为可能有三个比较重要的因素，通关所花次数，通关所花时间，

之前的Su和她的团队使用的是win rate。通过一关所使用的次数越高，那么难度越大，同样的，通过一关使用的次数越少，那么难度越低。

Then Christopher Jefferson and his mates had performed an experiment on an automated generation of puzzles named Combination solved by constraints, which indicated that the fun and immersing computer games could be generated by constraints \cite{jefferson2011combination}. They explained how all the levels of Combination were generated, checked for correctness and rated for difficulty completely automatically through the use of constraints. Then they found that running the Constraint Programming a number of times using different variable orderings then averaging the result could provide a more satisfactory player experience. Finally, this application was released in the iTunes and gained a great commercial success and received good reviews. 在这篇文章里他们使用的所花费时间来具体分类各个游戏的难度的。

除此之外，我在另外一篇文章里也看到了另外一种判别难度的方法。他提到是关卡的基本地形对于难度的影响。这涉及到两个核心部分，静态复杂度和动态复杂度。静态复杂度：即基本地形本身构成的复杂度，与游戏过程独立。动态复杂度：即在玩家实际游戏过程中，因消除和下落行为的过程所形成的复杂度特征。而静态复杂度又有一个核心因子： 平均连通数。它是由总连通数/有效格子数量得到的。而且也由此，如果关卡的消除颜色数量相同而言， 他可以将游戏的地形难度分为三个级别：

高：平均连通数>=3.6，这样的关卡玩起来一般比较流畅，Combo频繁。

中：平均连通数在3.2~3.6之间。这样的关卡虽然Combo数量不算太多，但是玩起来还不至于太过吃力。

低：平均连通数<3.2。这样的关卡中，玩家往往要花费一些功夫才能找到盘上为数不多的可操作的组合，因为没有可操作而重置局面也显得相对频繁。

但是也可能出现静态复杂度很高，但是实际游戏操作难度又很低的可能，因为这又牵涉到了动态复杂度。动态复杂度也有一个核心因子：combo问题。当消除一个candy的时候，掉落的candies或者其余candies都会导致chain reaction的发生。而这些combo的出现会大大改变游戏的难度。但是这些在游戏中又是难以预测的。除了地形复杂程度， 他还提到了关卡的目标， 关卡中的特殊元素和关卡的限制条件。关卡的目标可能是某个特定的分数，也可能是candies消灭的数量，也可能是两者相结合的情况。那么目标数量或者分数越高，难度越大。而关卡中的特殊元素比如一些新奇的道具都会增加游戏的难度，比如绳子和巧克力，前者绑住了糖果，后者则是会不断增加占据糖果的位置。而关卡的限制条件比如步数和时间都会影响游戏的难度。但是这四样都只是影响游戏难度，并不能作为直接的判断依据。

综上所述，对于游戏难度的判定最为直接的方法就是过关所花时间，另外通关所花次数和地形复杂程度都可以作为辅助参考。比如在时间差别不大的情况下，可以比较另外几个criteria， 比如关卡的地形，目标或者限制条件.

# Chapter 3 Requirements and Analysis

– Give the detailed problem statement. This elaborates on what you may have included in the introduction chapter, and represents the starting point from which requirements were derived.

– A structured list of requirements.

– Use cases (a use diagram and list of use case titles, with the full use cases appearing in the appendix), or other requirements modelling techniques you used.

– Results of analysing the requirements to extract information. For example, data modelling to find the data to be stored (ER diagram), views/web pages needed and so on.

The level of detail of the requirements and use cases will depend on the nature of your project. If you are doing a Software Engineering based design and implementation project, then they will need to be done thoroughly. If there is a substantial body of requirements and use cases, then a summary should be given in the chapter, with the full set included in an appendix section.

If your project is not Software Engineering oriented, then you still need to describe the requirements you are working to and relevant analysis information. Use cases may not be needed or be relevant.

The length of this chapter depends on the kind of project, but you are typically looking at 5-6 pages.

# Chapter 4 Design and Implementation

– Describe the design of what you have created.

– Start with the application architecture, giving its overall structure and the components that make up that structure.

– Give a description of the design of each of the components that make up the architecture. – Include the database or storage representation.

– Provide implementation details as necessary.

As with other chapters, the structure and contents of this chapter will depend on the nature of your project, so the list above is only a suggestion not a fixed requirement.

Find an ordering and form of words so that the design is clear, focusing on the interesting design decisions. For example, what were the alternatives, why select one particular solution? You have a limited number of pages so be selective about details. Also remember that someone (your examiners!) has to read this so don’t overwhelm them with intricate descriptions of everything that only you can follow – but do make sure the key details of the solution are in place. Use appropriate terminology and demonstrate that you have a good understanding of the Computer Science principles involved.

You can use diagrams and screen shots to help explain the design but don’t overuse them. Diagrams and screen shots should add information, not duplicate what is written in the text, and definitely avoid page after page of diagrams as this will disrupt the flow of your text. Where relevant, UML diagrams can certainly be used but, again, don’t flood the chapter with diagrams. Additional diagrams can always be included in an appendix section.

It may be useful to include sections of code to highlight how a particular algorithm is implemented or how an interesting programming problem was solved. However, avoid lengthy sections of code, as they can disrupt the flow of the text and explanation of your work. Also make sure that your code fragments are readable, easy to follow and properly laid out. It may be better to use pseudo-code rather than actual code, especially when describing an algorithm. If you need to make use of longer sections of code, you can put the code in the appendix and reference it from the text.

An alternative way to organise the content of both this chapter and the preceding one, suitable for some projects, is to have a sequence of chapters or sections for each major iteration of the project. This allows the progression of the project to be shown, with each iteration building on the last, and the opportunity for interesting discussion about the decisions that needed to be made.

This is a core chapter in your report and will usually be quite substantial, 10 pages or more.

# Chapter 5 Testing (or Results Evaluation)

– Describe your testing strategy (e.g., unit, functional, acceptance testing) and how it was carried out. How were test cases selected?

– Examples of specific tests and how they were carried out (e.g., using mock objects to break dependencies). Focus on the interesting cases.

– A summary of the test results and what coverage was achieved. Detailed test reports should appear in the appendix, if they add useful information or you want to demonstrate the kinds of tests and coverage achieved.

If your project requires substantial evaluation of data and results, or other forms of testing that are not code-based, then adapt this chapter to suit.

This chapter will typically be 2-4 pages in length but could be more depending on the depth of testing or evaluation done.

# Chapter 6 Conclusions and Project Evaluation

– A summary of what the project has achieved. Make sure that you address each goal set out in the Introduction chapter, to show that you have achieved what you claimed you would. Don’t leave any loose ends.

– A critical evaluation of the project (e.g., how well were the goals met, is the application fit for purpose, has good design and implementation practice been followed, was the right implementation technology chosen and so on).

– Future work. How could the project be developed if you had another six months. Take care to differentiate between what you have done to satisfy your current project goals, and work that could be done to meet extended goals.

– Wrap-up and final thoughts on your project. This chapter is typically 2-4 pages long but could be longer if the project work requires more extensive evaluation.

# List of References

Give publication details for all the items referred to by references or citations you have made in main text of the report.

Bibliography

This lists all the sources of information that you made use of during the project but are not referenced in the text. The items in the list must be relevant to your project, so don’t just list everything you may have looked at or read. The list of references and bibliography are often combined into one section labelled Bibliography

1. Simonis, H. (2005, October). Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems* (Vol. 12, pp. 13-27).
2. Schwab, B. (2009). *AI game engine programming*. Nelson Education.
3. https://www.theguardian.com/media/2005/may/15/pressandpublishing.usnews