# ISYS2102 Software Engineering 2

## Specification

## The “DevFortress”

Your group project for this course is to write a computer game. Your game is based on the real-world challenges of developing software.

### Starting the Game

The game starts with the player taking the role of a development team manager. At the start of the game, the player can hire seven (7) developers.

1) Players can accept a project even if the developers don’t have the appropriate skills.

The player can choose the skills each developer starts with. These skills can be programming languages, APIs, technical documentation skills, design skills, artistic skills etc. A suggested list of the minimum skills is provided in ***skills.xls*** spreadsheet. ***Students may add to the skills list***.

The player has a starting budget of $1000 (dev-dollars). Each skill point costs a number of dev-dollars to train, the cost for each skill is listed in ***skills.xls***, ***Skills cost progression*** worksheet.

The starting budget is also used to buy hardware for the team. Each developer hired needs a PC.  ***Students determine how much a PC costs.***

2) Balancing beer consumption is a key to success.

Finally any budget left over should be spent on pizza, coffee, red bull and beer. Developers **will** **not** work without coffee, red bull and pizza.

Beer makes developers happy. However the work they do is worse when they have beer.

### Playing the game – basic premise

Each month, the player will be presented with a choice of projects. ***Students determine how many random projects are presented (suggested minimum of 3 / month).***

Each project lasts for many dev-weeks and requires different skills. It is possible for a player not to have any team members with matching skills – for example the three projects could all be VHDL projects and the player only has Java developers.

The player can choose to accept a project. Accepting a project allows the player to earn more $ to pay for the developer salaries, new hardware and hiring more staff to be able to accept bigger projects.

After accepting a project, the player has to monitor the developers and keep them happy. Developers become unhappy when they are working on things they don’t like, when they haven’t had a beer, or when requirements change. Other events can also make developers unhappy (see the events list in the appendix).

Developers are defined by their strongest technical skill. For example Kim has Java (1), C (4) and SQL (2), Kim would be classed as a C developer to determine if she is happy working on a project. If two skills are equal, then the skill that is alphabetically first is chosen (i.e. C before Java).

For example Kim is a C developer but she is assigned to work on a VB project – this makes her unhappy.

At the end of each turn (month), each unhappy developer may leave. ***Students determine the chance that this occurs (either a flat %age, or a formulae based on number of developers who are unhappy and length of project, salary etc).***

### Ending the game

The game ends when the player runs out of budget, or when all the developers are dead or have left.

### Projects

Each project is divided up into ***function points***. The player can see how many function points a project is worth before accepting the project. After accepting a project, the player must assign developers to the project.

3) Each project has a main skill requirement.

Each project has a main skill requirement which is used to determine how effective a developer is. For example a project could be considered a VB project or an Oracle project.

Project characteristics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Time** | **Max function points** | **Total functional areas** | **Number of unknown** |
| 1 | 1-4 months | 100/month | 1-4 | 1 |
| 2 | 1-8 months | 100/month | 1-8 | 2 |
| 3 | 6-12 months | 200/month | 6-12 | 3 |
| 4 | 12-24 months | 200/month | 12-24 | 4 |
| 6 | 1-24 months | 400/month | 5-30 (random) | All unknown |

The level of the project should reflect the level of income the player can earn from it – Level 1 projects provide little income, Level 5 projects can be worth $millions. ***Students determine the $ value of the projects.***

When the player is deciding if to choose the project, not all of the function points will be shown – this adds a realistic element of risk to each project.

Developers are assigned to one of the functional areas in the project. If a developer produces more function points in the turn (month), than are needed to complete the work in that area, the additional function points are wasted.

### An Example Project

4) A turn in the game is a month, but events occur four times in a turn.

Build a social website (Level 2 project):

* DB, 20 function points
* Analysis, 50 function points
* Scalability, 100 function points
* Coding, ? function points
* Documentation, ? function points

Developer Tran is assigned to the DB functional area. Tran produces forty (40) function points of work in the turn. Twenty (20) points are used to complete the DB requirements, but the additional twenty (20) points are wasted.

### Events

Game time is broken into ***weeks*** and ***months***. Each week, random events occur to each developer, e.g. Take sick leave, resign, demand a pay rise, demand to be moved to a different project, get killed (by another developer). These events are triggered randomly by the game engine. A list of suggested random events is provided in the appendix.

### Skills

Each developer has skills in different technologies, tools etc. At the end of a project a developer might improve their skills in a particular area. This represents a developer becoming more experienced over time. For each month that a developer is assigned to a project there is a 5% (\* project level) chance that the developer improves their skill 1 level for the main project skill.

For example, Kev is assigned to a level 2, 6 month project where C is the main skill required.

He starts with a C skill level of 2. At the end of the project, Kev has 6 chances to level-up his C skill at 10% (5% \* project level chance/month).

If Kev is very lucky he could level-up from C level 2 to C level 8.

Skill levels determine how many function points a developer can contribute in a week.

Skills come in four (4) different types.

1. ***Technical skills*** allow a developer to work on a project at normal efficiency
2. ***Meta skills*** allow a developer to work more effectively
3. ***Personal skills*** allow a developer to work more effectively with other developers
4. ***Configuration Management*** prevents a developer from wasting effort

### Applying skills to projects (basic examples)

The minimum a developer can produce in a week is 1 function point, regardless of any other factors. For example if you assign three (3) developers to a project you are guaranteed that they will deliver 3 function points / week or a minimum of 12 function points for a month.

***Note:*** Certain special events can reduce this minimum. See appendix for details.

***Joe:***

Skills: Java (1), Databases (1)

Function points completed (when assigned to a Java project)

= >Java skill + (2 \* design skill) + (Java skill \* algorithms)

=> 1 + (2 \*0) + (1 \*0)

=> 1 / week

***James:***

Skills: Java (10), Databases (10), Algorithms (10), Design (10)

Function points completed (when assigned to a Java project)

=> Java skill + (2 \* design skill) + (Java skill \* algorithms)

=> 10 + (2 \*10) + (10 \* 10)

=> 130 / week

From this we can see that James is 130 times more effective on a Java project than Joe. From this we can see that having a high algorithms skill is very valuable.

If instead of having a Java project, we have a C project, we cannot assign Joe or James to the project without accepting a large negative impact. In this situation, to calculate the developer’s productivity, take their lowest technology skill and halve it (minimum 1).

***Note:*** Certain technical skills can be used to instead of the lowest skill (Lisp, Haskell, Forth), this is because developers with these skills can be effective on any project. See the ***skills.xls*** spreadsheet for more examples and example formulae for calculating the number of function points a developer can deliver in a week.

***James:***

Skills: Java (10), Databases (10), Algorithms (10), Design (10)

Function points completed (when assigned to a C project)

=> (Java skill / 2) + (2 \* design skill) + ((Java skill / 2) \* algorithms)

=> (10/2) + (2 \*10) + ((10/2) \* 10)

=> 75 / week

Now we apply the personal skills and the configuration management skills.

***James:***

Skills: Java (10), Databases (10), Algorithms (10), Configuration Management (5), Team player (1)

Function points completed (when assigned to a Java project with 3 team members total)

=> Java skill + (2 \* design skill) + (Java skill \* algorithms) + (Team player skill \* number of team members) / ((10 - Configuration Management Skill+2))

=> 10 + (2 \* 10) + (10 \* 10) + (1\*3) / ((10 - 5)+2)

=> 22.2 (round down to nearest whole integer) => 22

Finally we add in the calculations for coffee, red bull, pizza and beer

Skills: Java (10), Databases (10), Algorithms (10), Configuration Management (5), Team player (1)

Function points completed (when assigned to a Java project with 3 team members total, with coffee, red bull and pizza but [**no beer**])

=> Java skill + (2 \* design skill) + (Java skill \* algorithms) + (Team player skill \* number of team members) / ((10 - Configuration Management Skill+2))

=> 10 + (2 \* 10) + (10 \* 10) + (1\*3) / ((10 - 5) +1)

=> 22.2 (round down to nearest whole integer) => 22

Skills: Java (10), Databases (10), Algorithms (10), Configuration Management (5), Team player (1)

Function points completed (when assigned to a Java project with 3 team members total, with **no coffee**)

=> Java skill + (2 \* design skill) + (Java skill \* algorithms) + (Team player skill \* number of team members) / ((10 - Configuration Management Skill+2)) \* (coffee & redbull & pizza? 1 : 0)

=> 10 + (2 \* 10) + (10 \* 10) + (1\*3) / ((10 - 5) +1) \* 0

=> 0 => minimum is 1 function point => 1

Skills: Java (10), Databases (10), Algorithms (10), Configuration Management (5), Team player (1)

Function points completed (when assigned to a Java project with 3 team members total, with **beer**)

=> Java skill + (2 \* design skill) + (Java skill \* algorithms) + (Team player skill \* number of team members) / ((10 - Configuration Management Skill+2)) \* (coffee & redbull & pizza? 1 : 0) \* (beer? 0.5 : 1)

=> 10 + (2 \* 10) + (10 \* 10) + (1\*3) / ((10 - 5) +1) \* 1 \* 0.5

=> 11

This shows that the player must keep the developers provided with ***coffee, red bull and pizza*** to make the developers productive. If you no longer have any coffee, red bull or pizza, then each developer only produces 1 function point / week.

5) Players can hire more developers to add to the team each month.

Beer makes developers happy (happy developers will not leave at the end of the turn), but it halves productivity.

### Developers

The player can choose to hire more developers to add to their team. The game must randomly create a set of developers that the player can hire.

The developer must have a name, one or more skills and a salary. Developers with more skills and more skill levels require a higher salary to hire. ***Students can determine the salary formula.***

Developers eat pizza and drink coffee and red bull. Each developer uses five (5) coffees / week, five (5) pizzas / week and one (1) red bull / week. ***Students can determine how much each of these costs.***

To generate a random name for a developer, students can use the baby names data from Google’s Python classes (<http://code.google.com/edu/languages/google-python-class/exercises/baby-names.html>).

## The Game Mechanics

### Turn-based

The game is turn based. The game is divided into weeks and months. There are four (4) weeks in each month, and events can occur in each week. Each month represents one (1) turn. Therefore there are twelve (12) turns in one year.

At beginning of each turn, the player can choose to

1. add developers to a project
2. remove developers from a project
3. hire more developers
4. accept a new project
5. cancel a project

Each week, the game engine must

1. produce an event for each developer the player has hired (see events list)
2. calculate the remaining function points that must be completed
3. determine if a project has finished
4. display the current week (within the current month) i.e. week (1 of 4) January or week (3 of 4) July

Each month, the game engine must

1. Pause the game and let the player perform their actions
2. Generate new random developers (in case the player wants to hire new developers)
3. Generate new random projects (in case the player wants to accept a new project)
4. Calculate the amount of money the player has left and decide if the player has lost (no more money)
5. Calculate the total function points delivered by the player
6. Calculate the total weeks played
7. Determine if any of the unhappy developers leave

The game is a management style game and the user interface needs to allow the player to:

1. see all of their developers (display all information, happy/unhappy, name, salary, skills etc)
2. see which developer is assigned to which project
3. assign a developer to a project
4. choose a new project from the selection offered each turn (month)
5. see operating expenses (i.e. developer salaries, coffee, pizza and red bull costs)
6. see when projects will finish (number of function points left)

***Note:***  The game does not need to show the developers working, moving or responding to the player’s actions.

### Losing

The game is lost when the player has $0 or less or when all the developers have left. At the end of each turn the salary of each developer is paid, starting with the most expensive first. When the game is over the game should display a summary screen showing how well (or badly) the player did. The game should store this information as a high score table.

### Income

Each project the player accepts is worth money. On accepting a project, 25% of the value of the project is given to the player. The remaining 75% is given if the project is completed on time. Some projects can have bonuses for completing early. ***Students can determine the level of bonus available.***

## Appendix

### Suggested Random Events

This is a list of random events and their effects. The game should produce random events at a rate of 1/developer/week. If you have three (3) developers working on a project, then there should be twelve (12) random events for each player turn. ***Students can add to these or change the frequency to balance the game.***

|  |  |  |  |
| --- | --- | --- | --- |
| ***Event*** | ***Negative/Positive*** | ***Chance*** | ***Effect*** |
| Developer is sick | Negative | 0.1 | Function point output \* 0.5 |
| Developer kills another developer (can only happen if all developers on all projects are unhappy) | Negative | 0.01 | Maximum output for all developers on project is 1/developer, dead developer is removed from project. |
| Requirements change | Negative | 0.1 | Add 20 function points to 1 area |
| New technology | Positive | 0.05 | Remove 50 function points from project |
| Solution doesn’t scale | Negative | 0.05 | Add 10 function points to 1 area |
| Hacked | Negative | 0.01 | 0 function points produced by developers this week |
| Backup failed | Negative | 0.05 | Add 25 function points to 1 area |
| Holiday | Negative | 0.1 | 1 developer produces only 1 function point this week |
| Features cut | Positive | 0.05 | 1 functional area completely removed from project |
| Bonus | Positive | 0.01 | Additional $ are given at the end of the turn |
| Team building exercise | n/a | 0.05 | Developers only produce 5 function points each, but all developers are happy |
| Redundancies | Negative | 0.05 | A random developer is removed from the team, all other developers are unhappy |
| Idiot’s in marketing… | Negative | 0.1 | Additional requirement has been added at the last minute, additional 10 function points in 1 area, 1 developer unhappy |
| Interns | Positive | 0.1 | An additional 5 function points completed by student interns, 1 developer becomes happy. |
| Nothing | n/a | 0.5 | No effect |