UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Title of the master thesis

Subtitle (optional)

Dissertation presented by
Zélie MULDERS

for obtaining the Master's degree in
Computer Science and Engineering

Supervisor(s)
Kim MENS, Philippe BARET , Adrien DOCKX

Reader(s)
Hélène VERHAEGE,

Academic year 2017-2018

Contents

1 Introduction **2**

2 Problem analysis in market gardening **3**
2.1 Some vocabulary about market gardening 3
2.2 Daily life scenario 4
2.3 Profitability of market gardens 4
2.4 Researches in gardening 5
2.5 Existing tools 5
2.6 Conclusion 6

3 Problem analysis in software engineering **7**
3.1 Clients and their needs 7
3.1.1 UCL 7
3.1.2 Market gardeners 7
3.2 Vocabulary in software engineering 8
3.3 Which platform? 9
3.4 Which methodology 9
3.5 Which languages? 10
3.5.1 Programming language 10
3.5.2 Web framework 10
3.5.3 Query language 11
3.5.4 Written language 11
3.6 Maintenance 11

4 Solution **12**

5 Architecture **13**

6 Implementation **14**

7 Validation **15**

8 Conclusion and future work **16**

Chapter 1

Introduction

Chapter 2

Problem analysis in market gardening

In this chapter, we will talk about market gardening. It contains all pieces of information needed to understand the thesis, even without any knowledge of this field. We will also analyse this field and define different problems encountered during the management of a market gardening farm.

2.1 Some vocabulary about market gardening

First, it can be useful to define some common terms used in gardening.

Market gardening A market gardener is someone who produces fruit and vegetables on a relatively small area. The difference between a farmer and a market gardener is principally in the type of final product. Where a farmer will produce more cereals, a market gardener is specialized in fruits and vegetables. We will use the term *Truck farmers* for gardens that are cultivated with heavy machinery.

Bed A bed in a market garden is a surface of production, usually a line. It is used to divide the field in smaller cultivated areas. A picture of beds is shown on figure 2.1. Market gardener usually choose the width of their beds according to the width of the tools they are going to use.

Figure 2.1: Beds in a market garden

Crop rotation In order to preserve the soil from draining and to eliminate some diseases specific to some plant species, some market gardener rotate their cropping. If they plant one type of vegetable on a bed, the year after they will plant another type of vegetable on this bed. They will not plant two years in a row the same vegetable on the same bed.

2.2 Daily life scenario

Seasons Market gardeners live by the rhythm of the seasons: they have a peak of work during the Spring and the Summer. Harvests continue during Autumn but during Winter they usually have less things to do in the garden.

Planning Most gardeners plan their cropping during winter[4], when they have more time to think about what they want to grow this year. Planning the coming year has several advantages :

- Know in advance what amounts of seeds and fertilizers they will have to order
- Take the time to decide what to grow and in what quantities
- Look back at the previous year to see which vegetables were the most profitable and adjust cropping according to this experience.
- Gain time during the rush season by having clearly in mind what has to be done
- Organise the year to spread the work as most as possible (everything can not be seeded the same week)

While really useful, this planning part is not always done by market gardeners.

Adaptations Once the work season has started, this planning has to be adapted to the reality on the ground. The weather is the major factor of changes in the planning. Indeed, some seeding requires several days of dry weather followed by one day of rain for example. In the case of difficult weather (late frost, large humidity,...), whatever was the initial plan, the gardener will have to adapt his schedule to the weather. Others factors that disrupt the work set-up can be diseases in the crops, short staffing or hardware issues. One example scenario could be: We are the first week of July, the season is in full swing. Tim is a market gardener and had planned to plant endives this week. The weather conditions are perfect, so he could stick to his plan. Unfortunately, his tomatoes have mildew[1] and if he wants to save his tomatoes' crops, he has to treat them immediately.

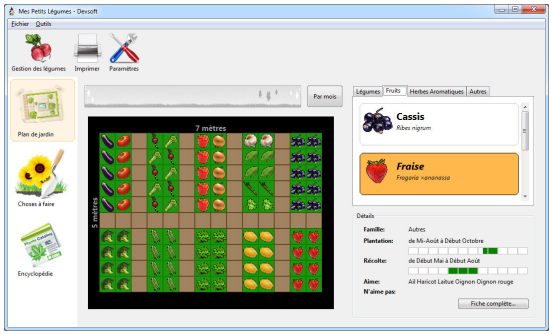
This example shows that some events have priority over the initial plan and confirm the idea that initial planning is meant to evolve.

It is essential for a market garden to be able to adapt its plans to a specific situation and to keep a clear head as we go along the season. Planning is already not and easy task, but adapting to changes is even harder.

2.3 Profitability of market gardens

Workforce Market gardeners don't count their hours as regular workers. They work all day in order to reach their objectives of the day. Most of them have no idea of how long each culture takes. It also means that they have no idea how profitable their cultures are. Moreover, they often need external workers to help them during the peak season. These external workers represent 50% of productions costs [9]. Consequently, organizing the work to reduce the need of external workers can have a considerable impact on the garden's profitability.

[1] an epidemic fungus `https://en.wikipedia.org/wiki/Mildew`



(a) Visual representation of a garden planning

(b) Intercropping advices

Figure 2.2: Screenshots of the software *Mes petits légumes*

Vegetables profitability Some vegetables are more profitable than others. For example, Jean-Martin Fortier [9] in his book gives data about the profitability of the vegetables he's growing. However, most farmer don't do this analysis on their production and have therefore, no idea of which cropping is the most profitable. Even in the table of Jean-Martin Fortier, we have no idea of the work time needed for each culture. And yet we have seen before that workforce represent a significant cost. Moreover, from one area to another it is reasonable to think that some crops will be more profitable than others. Depending on the clients' preferences or the soil type, some vegetables will be easier to sell or to crop. Gardeners are mostly not analysts and don't have the right tools to give them an idea of how profitable their business is and how they could be more efficient.

Others profitability factors

- Retail strategy: different retails strategies will give different revenues, the more intermediaries there are between the producer and the client, the less the producer will gain.
- Pricing strategy: of course, the selling price of vegetables will affect the profitability of this vegetables. Depending on which retail strategy is chosen, prices will be more or less flexible.

Antoinette Dumont has dedicated her doctoral thesis on the subject of market gardens' profitability.[8]

2.4 Researches in gardening

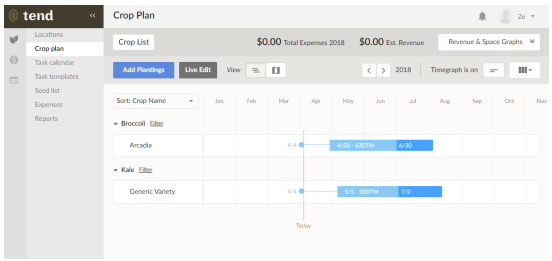
Complete this section

2.5 Existing tools

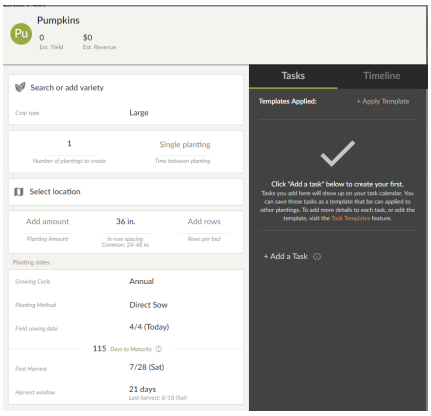
From our researches, there is not a lot of software to help farmers of all kind in their daily life. Furthermore, most of them are not open source.

First, we found software like *Mes petits légumes*[3] intended for non-professional market gardeners, with a great library of data about lots of vegetables. The software can be bought once for 19 € or one can use the free incomplete version. Two screenshots of this application are shown on image 2.2

Then, we have softwares like *LEA*[2] more focused on the management of the business and intended for big farms. It can generate bill from tractor work. It helps managing stocks and uses of fertilizers. Once again, the software is not open source and a subscription is required to use it.



(a) Planning view



(b) Adding a new plantation

Figure 2.3: Screenshots of the software *Mes petits légumes*

Finally, we have found a software that seems to have a purpose and a target audience similar to this project. *Tend*[7] is a software developed in the USA by a Startup. It has lots of features, including a databases of vegetables, a task calendar and an expenses section. The main feature (constitute a crop plan by adding plantings) is shown on figure 2.3.

These software show that farmers are in need of tools to help them in their planning and management. The poorness of software really adapted to their needs show that this field has been forgotten by technology.

2.6 Conclusion

From this analysis of the field of market gardening,

+ say that this is where this thesis comes in

Chapter 3

Problem analysis in software engineering

For each thing you explain also say why you explain it, what role will it play later in this thesis

3.1 Clients and their needs

3.1.1 UCL

As said before, market gardening is getting more and more interest from searchers these last years. The UCL is thus currently leading researches in this field. The university has recently bought an ancient farm and plan to do some research in market gardening on the fields around.[1] The point of these researches is to gather data about the viability, efficiency and profitability of different gardening principles. There are many theories about gardening on small surfaces, but not that much researches on this topics and the efficiency of most of them have not been proven. Hence, UCL would like a web application that would help them gather data from this forthcoming gardening project but also from gardens around the country. From our meetings with UCL's searchers we have defined some requirements:

- The application should have a searcher backend to allow easy access to gardens' data
- Users should be able to choose which pieces of informations they agree to share with the university
- Every task should have a *note* field so growers can give more details to everything they do.
- Every task should have a *duration* field in order to collect data on the time needed for each cropping.

3.1.2 Market gardeners

Before starting the project, we met several vegetable grower in order to have their opinion on the application and their advice. As the application is intended to help them, it was essential to meet them. From these meetings, we defined what were their requirements:

- The application should help them in their planning
- In order to help their planning, the application should remind them about cultural operations

[1] `http://fermedelauzelle.be`

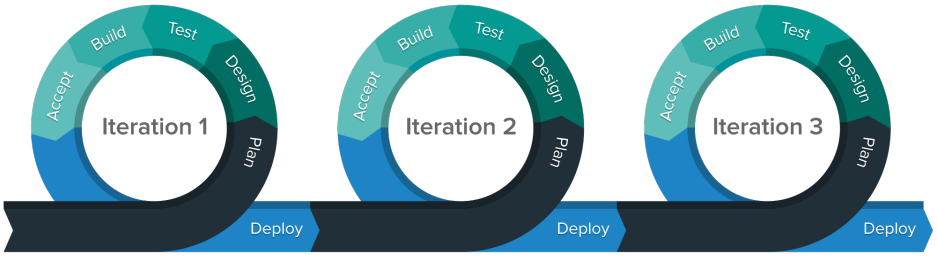


Figure 3.1: Iterations using Agile methodology

- The application should give them information about the previous years and their previous harvests
- The application should have an economical side: one should be able to see which cropping is the most profitable.

Meetings

3.2 Vocabulary in software engineering

In this section, we define some concepts specific to software engineering. These concepts will be used in the following sections. This section can be skipped by experts of the subject.

Agile Methodology The Agile methodology is a set of techniques and principles for conducting a development project. The main principle is to iterate over short periods (called sprints) divided in subphases in order to build the final product in an incremental way. A sprint lasts between one and two weeks and is divided as follows:

- At the start of each iteration, we plan with the client what we are going to do this iteration.
- Next step is to think about how to build a good design to achieve the objective
- Then, we develop the features
- After, we test these features. If we switch these last two steps, we apply what is called Test-Driven Development. With this methodology, a team write tests before developing the corresponding features, ensuring that the tests will cover all the cases
- At the end of each sprint, we meet the client again to validate the changes and new features, collect his feedback about the project's progress and to define together the future work.

A visual representation of this iterative approach is shown on figure 3.1

A good summary of the Agile methodology is the agile manifesto[6], that states the main principles of this methodology.

Agile Manifesto [6]

Individuals and interactions over processes and tools.
Working software over comprehensive documentation.
Customer collaboration over contract negotiation.
Responding to change over following a plan.
That is, while there is value in the items on the right, we value the items on the left more.

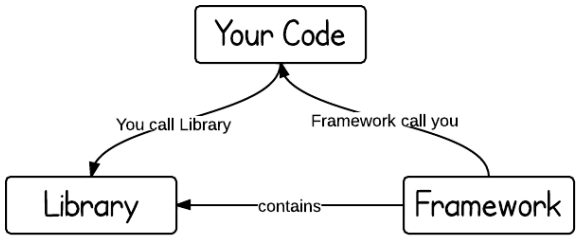


Figure 3.2: Framework versus library

The main advantages of this methodology are to have flexibility and regular feedback on the product delivered.

Software framework Give clear definition of a framework

A framework is different from a library in the sense that when using a library, we call the methods of the library while when using a framework, our code is called by the framework. The difference is shown on figure 3.2. Frameworks help to build reusable and maintainable applications

give some examples of frameworks + why you talk about it, integrate Django with it?

Object Relational Mapping here

Continuous Integration Continuous integration (often referred as CI) is a software engineering practice to help development teams working on a set repository. When using CI, we automate a set of checks at each push request on a repository and developers should merge to this repository regularly. With continuous integration, we will detect bugs faster and more easily. We can also add tests to our continuous integration tool, these tests will then be run at each push request and we will receive a notification if they fail. Thanks to that, we can see exactly which commit implied a failure in the tests. If we do small regular commits, it will be really easy to find where the failure comes from.

3.3 Which platform?

First question to ask is which platform are we going to choose. We have two main clients; searchers at the UCL and gardeners. While searchers work mostly on their computer, gardeners spend most of their time on the ground and don't always have regular access to a computer. We could have think about a mobile application, but then we have to deal with different operating systems (Android and IOS mainly). For these reasons, we have opted for a web application. A web application is available on every device having access to internet, whatever the operating system is. With this solution we assure satisfaction of both clients.

3.4 Which methodology

When this project began, we had no clear specifications about the final result to produce. The requirements have evolve during over weeks. Because of this incremental definition of requirements and specification, it was natural to follow and Agile methodology (see 3.2). As we were only one developer working on the project, we did not apply all principles of this methodology. We kept the iterative approach by doing regular meetings with the clients. We had meetings with the UCL client once a week and with a gardener once a month. For each meeting, new features were

presented to the client in order to have his approval (or not). These meetings were also the good moment to define priority for the coming weeks.

3.5 Which languages?

In this section we will analyse the different choices we had concerning all type languages.

3.5.1 Programming language

To guide our selection of the perfect programming language we had several guidelines:

- As we want the application to be reusable and maintainable by future students or searchers from the UCL, we had to choose a language easy to learn and to understand, with a fast learning curve. From next year, first year student at the EPL faculty will be taught Python as first programming language. In the agronomy faculty, it is common that searchers use Python
- As we are targeting a web application, we needed a language with web frameworks available.
- Because of the nature of our application and the requirements defined at the beginning of the project, we also wanted an object oriented language.

From these guidelines it was natural to choose *Python* as programming language.

Python Python is a programming language used in many fields. Its main advantage is great readability. Its first release was in 1991 and its creator is Guido van Rossum. Python is an interpreted language which means there is no compilation stage before running the program. The interpreted executes it directly. This feature makes it fast and easy to use even for very small projects, but for bigger projects Python is known as being slow because there is no compilation optimisation possible.[5] However, Python has a big active community and thus great support and documentation which makes it a reliable language. Python is currently at the fourth place of the TIOBE index [2] which suggests that Python is a good choice of programming language.

3.5.2 Web framework

From the previous choice, we already narrowed the range of available web framework. Besides Python as requirement, the principal guidelines for the choice of a web framework were the following:

- It should be easy to use and understand as we have chosen Python partly for its readability
- It should be stable and reliable, so with a good documentation and a community behind reporting bugs and keeping the framework up to date.
- It should have basic web features such as authentication, we do not want to reinvent the wheel and reimplement existing modules.

From these guidelines, we spotted 4 popular Python web frameworks (figure 3.3) Pyramid[3], Bottle[4], Django[5] and Flask[6].

[2] The TIOBE index measure the popularity of programming languages: `https://www.tiobe.com/tiobe-index/`
[3] `https://trypyramid.com/`
[4] `http://bottlepy.org/docs/dev/`
[5] `https://www.djangoproject.com/`
[6] `http://flask.pocoo.org/`

Figure 3.3: Four popular Python Web Framework

Bottle and Flask are more intended for small projects and do not offer lots of support for bigger applications with more needs. For example, they don't have a built-in authentication module. Their key advantages are their small size and their fast installation. We dropped those choices (and other similar lightweight web framework) early on. Django and Pyramid are both designed to create web applications of medium to large size. They are both open source and have a large community. The main difference between Django and Pyramid is while the later is a lightweight framework, the former is a high-level web framework. This imply that Pyramid, compared to Django, is well-known as being a lot more flexible. There is not a framework better than the other, they have different features and we have to choose which ones are the most important for us in our project. We did not choose Pyramid because we did not need the flexibility it offers.

Finally, we opted for Django because of its maturity, its popularity and its wide range of built-in modules. We relied upon its ORM[7], its authentication module, its administration interface, its MVT[8] design pattern and others features that will be discussed in the later chapters about implementation.

3.5.3 Query language

Once again, the choice of Django reduces the choices we have concerning the database we are going to use. First of all, Django does not officially support NoSQL databases. Official documentation is only given for PostgreSQL, MySQL, SQLite and Oracle databases. Our choice was made between PostgreSQL and MySQL because SQLite is intended for small databases and Oracle is not a free alternative. Once again, PostgreSQL and MySQL have both their own advantages. We've choose PostgreSQL because it is the most popular relational database management system (RDMS) with Django.

3.5.4 Written language

Code (including tests) has been written in English for a better readability. But all textual information on the website are in French. This has been decided because our two clients speak French.

3.6 Maintenance

Why you explain this, what CI you use, what CI features and tools you use; why you use these

[7]Object Relational Mapping
[8]Model View Template

Chapter 4

Solution

Chapter 5

Architecture

Chapter 6

Implementation

Chapter 7

Validation

Chapter 8

Conclusion and future work

Bibliography

[1] Continuous integration. https://www.thoughtworks.com/continuous-integration. [Consulted 26-March-2018].

[2] LEA. https://www.lea-agri.com/. [Consulted 10-January-2018].

[3] Mes petits légumes. http://mespetitslegumes.com/. [Consulted 15-August-2017].

[4] Planification des cultures : méthodologie. https://fermesdavenir.org/fermes-davenir/outils/planification-cultures-methodologie.

[5] Why Python is Slow: Looking Under the Hood. https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/.

[6] Manifesto for Agile Software Development. http://agilemanifesto.org/, 2001. [Consulted 22-March-2018].

[7] Tend. https://www.tend.ag/, 2016-2018. [Consulted 26-March-2018].

[8] Antoinette Dumont. *Analyse systémique des conditions de travail et d'emploi dans la production de légumes pour le marché du frais en Région wallonne (Belgique), dans une perspective de transition agroécologique.* PhD thesis, Université Catholique de Louvain, 2017.

[9] Jean-Martin Fortier. *Le jardinier-maraîcher : manuel d'agriculture biologique sur petite surface.* Ecosociete Eds, 2016.

[10] Mireille Navarrete and Marianne Le Bail. Saladplan: a model of the decision-making process in lettuce and endive cropping. *Agronomy for sustainable development*, 27(3):209–221, 2007.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**