UNIVERSITY OF ALBERTA
DEPARTMENT OF COMPUTING SCIENCE

INTRODUCTION TO SOFTWARE PRODUCT MANAGEMENT

COURSE NOTES

Table of Contents

Module 1: Software Product Management – The Discipline

Upon completion of this course, you should:

- Be able to identify three goals to achieve better software.
- Be able to define the role of a Software Product Manager.
- Describe several ways a project can be deemed successful.
- Have reflected on the core values in the Agile Manifesto.
- Have reflected on the 12 principles described in the Agile Manifesto.
- Be able to describe how process, requirements, planning, and monitoring are integral components of software product management.
- Be able to explain the structure of courses within this specialization and the expectations of coursework.

Introduction

Welcome to the University of Alberta specialization in Software Product Management on Coursera. This program will present a philosophy for developing great software through effective handling of: client expectations, developer expertise, and timelines.

Realize that this is not a course in “project” management; the focus is on software product management. How does product management differ from project management?

“Project management” is a broad field that can be applied to any development scenario, for example, physical construction of a building. Certainly you will recognize aspects of project management within this specialization. However, building software presents some unique challenges. This specialization will describe practices specifically developed for creating software products.

One key challenge is change. Producing software may be like building a house for a client; you may start with a blueprint, but as the clients see the building take form, they visualize and request changes that make the home more liveable. Similarly, a software product is refined into a state of usability, with ideas and feedback from everyone involved, especially the client and end-users. A software product manager needs a unique set of skills to assess whether the product is meeting client expectations, and to work with a development team to solve the client’s problems.

In this specialization, you’ll learn effective techniques to produce better software products.

Better Software

It takes the sustained work of many people with different skill sets to make great software products. A software product manager needs to understand a number of goals or viewpoints to achieve better software.

One viewpoint aims to provide the right software product for the clients. That is, it meets their needs, solves their problem, and they are happy with it. That is, the software product is validated.

A second viewpoint aims to have the software product done right. The software implementation conforms to a specified design, and in turn, the design satisfies a stated set of requirements. Developers can conduct reviews and tests to ensure the requirements, design, and implementation line up and do not have defects. That is, the software product is verified.

A third viewpoint aims to have the software project itself managed right. The idea is to adopt just enough process and suitable practices to organize the work of everyone involved. These practices ease communication and feedback, so that everyone is clear on the next steps. That is, the project is managed.

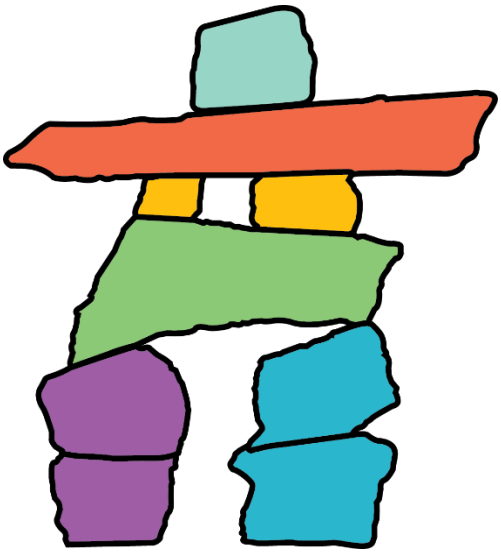
Consequently, to achieve better software, three goals are: the right product, done right, and managed right.

Role of Software Product Manager

A software product manager is in charge of the success of a software product. But the role is much more than managing a project. Great software product managers can speak to both clients and developers in their own terms. The role involves understanding the product from the client's point of view. The role also requires effective communication and motivation of development team members.

Structure of the SPM Specialization

There are six courses for you to take, including this introduction course. The specialization concludes with a capstone project course. These courses are represented as an inuksuk.



Upon completing the **Introduction** course, you'll be ready to move on to the four core courses. These courses provide a deep dive into techniques and practices that make a software product manager a valuable asset to any development team. The Introduction course is the ground on which the inuksuk stands.

Each of the core courses is expected to be four week's worth of work, with the course material organized into weekly modules. Each module has a quiz, and each course ends with a final exam, which are graded towards your overall specialization grade.

Each course builds upon the knowledge gained in the last.

The foundational “legs” of our specialization structure are the first two courses: 1) **Software Process and Agile Practices** and 2) **Client Needs and Software Requirements**. Once launched, these courses can be taken on demand in either order. Completing these two courses is essential for the subsequent two courses that form the “body” and “arms” of our specialization structure: 3) **Agile Planning for Software Products** and 4) **Reviews and Metrics for Software Improvement**.

The culmination of the specialization is applying knowledge from the four core courses to the **Capstone** project. The capstone is a six-week course in which you will evaluate and make decisions with a simulated team of software developers in a quest to create a product for a challenging client.

In order to participate in the capstone, you will need to have taken all of the prerequisite courses.

After you’ve completed all of the prerequisite courses and the capstone, you will have earned your Coursera Software Product Management certificate.

DID YOU KNOW?

The University of Alberta is proud of its Canadian and Arctic heritage. A pertinent figure is an inuksuk, which is a man-made landmark resembling a human used by the Inuit peoples of the Arctic to mark paths.

The traditional meaning of the Inuksuk says “you are on the right path.” This is an excellent metaphor for the journey you are about to embark on with this specialization.

Coursework Expectations

Within each course, a module represents one week’s worth of work and will have a workload of 2–4 hours. Just because the workload is small does not mean that there is not a lot to learn in this course.

The coursework estimate is based on the amount of video lesson time, as well as all required readings. We recommend doing the readings so that you are able to track with the material presented in the videos. Additional resources are provided in the course notes.

The more you put into the courses, and participate in the discussion forums, the more you will get out of them.

Goal of SPM Specialization

In preparing this specialization, we believe that if you apply yourself to learning the concepts and techniques presented, you will build confidence based on your knowledge of software product management.

Module 2: Specialization Preview

Defining Project Success

Developers spend countless hours and sleepless nights thinking about the best way to design software. Their efforts directly impact the parameters for measuring project success:

- On-time delivery
- Completed within budget
- Delivered with all features complete

Project success could also be measured by:

- The number of post-release bugs
- The support needed after a release
- The software product's customer rating
- The revenue generated
- The client's satisfaction

A software product manager will help to identify attributes of product and project success and assign metrics to ensure that development efforts are driving towards successfully delivering a quality product.

Agile

"Agile" is set of software development principles created for effective and adaptive software development. These principles take shape as practices that have become commonly adopted in the software development industry.

Four core values define the Agile philosophy for software development.

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

The Agile philosophy values the items on the left more than the items on the right. This is not to suggest that you ignore *processes and tools*, *comprehensive documentation*, *contract negotiation*, or *following a plan*. It is merely that the items on the left are the more important things to focus on.

Let's reflect on the adjective definition of **agile**, from the Merriam-Webster dictionary:

> ***adjective:*** having a quick resourceful and adaptable character;
> e.g., an **agile** mind

So, as a software development approach, Agile seeks to be quick, resourceful, and adaptable.

Twelve Principles of Agile

The four core values are illuminated by 12 principles of Agile software development.

Agile software development brings focus to ensuring a steady, manageable, and efficient development pace. It strives to avoid situations where team members are overwhelmed with project tasks. Happy and productive developers do consistently better work.

Agile practices also encourage frequent testing and demonstrations. This helps to reduce bugs and keeps the client involved in the development process, so the end deliverable is a product the client can use.

Agile practices encourage communication with the client. Managing expectations is important. Keeping your client involved throughout the entire project allows them to see the product as it develops and give feedback. This ensures that the client is getting exactly what they asked for. Or better yet, something better that they didn't even imagine was possible.

Manifesto for Agile Software Development

The four core values, and 12 principles are available online at: www.agilemanifesto.org. Visit the website to see the founders and to see a list of signatories—a large number of people that agree with the manifesto and have signed to show their support for Agile software development.

Topics in Software Product Management

This specialization will cover four core topics in software product management:

- Process
- Requirements
- Planning
- Monitoring

Process

A **process** organizes the work of people into distinct high-level phases or stages to develop a software product. Example phases are:

- Specification
- Design and Implementation
- Verification and Validation

Specification activities discover and define what the software is expected to do. Design and implementation activities structure and construct the software solution. Verification and validation activities test for potential defects and review whether the product meets the client's needs.

Process is necessary to organize our development and make sure that you are completing things in a logical order. They also ensure that steps are not missed or overlooked. Software development is pretty daunting when you don't know where to begin. Processes also provide clarity as to where you should start your project.

Ad Hoc Development occurs in the absence of a process structure and describes a development scenario where work is done reactively to inputs such as developer ideas, or client demands. Ad hoc development can be inefficient with developers potentially wasting time on ill-conceived features.

Requirements

Good development processes help you to develop a product efficiently, but that's only half the story. Every project needs requirements to define the product to be built. How you well define those requirements often dictate whether the project is a success. Therefore, it's important to get requirements right.

In software, requirements are a set of specific descriptions to help capture your client's needs. Requirements can be mapped to features of the software product. When combined, processes and requirements are the backbone of any successful software project.

Clear requirements are important. Ambiguous requirements can create confusion among the development team and with the client. Avoiding confusion is an important part of the software product manager's role. There are techniques to properly elicit and express requirements. By spending the time to refine requirements, you can also detect potential errors in your product before it's even built. By clarifying ideas, development becomes focused and efficient.

Every project comes with a set of requirements, whether you clarify and track them or not. In the Client Needs and Software Requirements course, we will cover proven techniques for eliciting and expressing requirements. This will help you to maximize the effectiveness of your requirements.

Planning

According to Alan Lakein, "Planning is bringing the future into the present so that you can do something about it now." Planning involves organizing tasks and schedules to align software development activities with requirements, in order to make timely progress through the phases of the development process. Creating tasks and schedules requires identifying who should do the work, and estimating how long that work will take. Planning includes:

- Breaking down work into tasks
- Estimating time for tasks
- Assigning work
- Risk management
- Contingency plans

Planning is not just for the beginning of the project. In software development, planning is something that occurs constantly throughout the project. Software is always evolving. Your plans have to be flexible! When you have a process to review your plan, your project becomes more *agile*. Meaning, you can easily adapt to change.

Time Estimation

Poorly written requirements make accurate time estimates difficult. The more specific the requirements are, the clearer it is to estimate how long the work will take to complete!

Assigning Work

The developers and the software product manager determine the time estimates for tasks in software development. The developers know best how long it takes them to develop features, but as a software product manager, you also have a sense for timelines based on your own experience.

Risk Management

If you can identify future risks and do something in the present to mitigate them, then your project will have a higher chance of success.

Risk management entails developing contingency plans to address risk. What are some examples of things that could go wrong?

- What if a developer gets sick?
- What if someone quits?
- What if your client contact changes?
- What if your technology crashes?

Through risk planning or risk management you can anticipate problems that may occur and formulate contingency plans in the event the risks become real.

Monitoring

As a software product manager, you can't establish the process, requirements, and plans, and then step back to watch the development team do their thing. When things change in the project (and they will change), it's important to have an idea of what's going on in the project, so that you can deal with these changes as effectively as possible. Monitoring also helps to make sure you're on track to meet your goals. If you're not, monitoring also helps you to find out why, and what you can do to fix it. In essence, monitoring your progress helps you to stay afloat over the lifetime of the project.

If the client asks for more features, a manager effectively monitoring the project can easily gauge whether their team has available resources, allowing them to adapt to changes within the project for a much higher likelihood of success. This moves you from being a *reactive* software product manager to a *proactive* software product manager.

Transparency is another aspect of monitoring. Transparency means that *everyone* on the team knows the status of the project, not just "management." The development team and the product manager work together to monitor their progress with a positive, goal-driven, and team-based mentality.

Many tools and techniques exist to help you monitor, analyze, and review your progress. We will cover a subset of these tools and techniques in the Reviews and Metrics for Software Improvements course.

Version 1.0.1