Software Engineering for Business Professionals

Duration:	1 day
Type:	beginner

Description

This course enables experienced managers to participate in the process of software development. It is designed for business professionals who are about to commission, sponsor, fund, oversee or provide requirements for a bespoke software product. The course does not prepare delegates to act as software project managers, but will allow them to promote their interests appropriately at all stages of the software lifecycle.

The course is divided into two sections. In the first delegates learn the key activities of a software team and how it operates, both ideally and in actuality. In the second delegates explore the interaction between the business and the software team and how they can influence and support the development process.

Prerequisites

Delegates should have three or more years experience working in a management role outside the software industry. Prior involvement in software development is not required, but delegates should have basic IT literacy.

List of Modules

Questions that Need Answers

- I'm commissioning software - what have I let myself in for?
- What does a software development team do all day?
- How do get programmers to do what I want?
- Why doesnt my software work the way it should?
- How can I get the most value from my investment?

Different Types of Software Project

- Producing software for external customers
- Producing software for internal customers
- Automating internal business processes
- Automation in a manufacturing environment
- The challenge of enterprise integration
- Highly specialised types of software

Understanding the Software Developers Job

- Programming languages and compilers
- Versioning and source code control
- Libraries, frameworks and power tools
- Build management and releases
- Installers and documentation

One (Idealized) Week for an Agile Developer

- Taking on stories and clarifying requirements
- Agreeing on acceptance tests with the client
- Creating a clean development environment
- Prototyping solutions for technical risks
- Writing code, refactoring and unit testing
- Integrating the work and releasing to QA
- Fixing bugs and showcasing to the client

Software Development as a Process

- Traditional waterfall-style projects
- Incremental and evolutionary prototyping
- Iterative, lean and agile processes
- An agile process in depth - Scrum

An Interlude: Ways to Kill Working Software

- Losing ownership of the source code
- Losing the source code from version control
- Creeping obsolescence of platform and hardware
- Allowing the 'truck count' to drop to single digits

Influencing the Requirements Process

- Customers rarely know what they want
- Most of what is developed will not be used
- Requirements must be gathered from the right people
- Requirements must be continually refined through feedback
- Defining requirements must include acceptance tests
- Stakeholders must have an input into prioritization
- Changing requirements always has consequences

Metrics for Ensuring Progress

- Working features are the best guide to progress
- The dangers of insisting on incremental deliveries
- Identify and insist on solutions to key challenges
- Measuring the velocity of progress via story points
- Distinguishing between functional and non-functional tests
- Distinguishing between internal and external quality
- Metrics that can lead to false confidence

Managing Problems and Setbacks

- Balancing scope, quality, time and cost
- Encouraging communication between stakeholders
- Adjusting the iteration plan when velocity falls
- Diagnosing failures in requirements gathering
- Detecting deficiencies in the testing process
- Watching out for showstopper problems
- Examples of typical last minute issues

Summary: Key Points to Remember

- Software is hard because it is intangible prior to delivery
- Most problems are psychological rather than technical
- Developers know coding not your business model
- The requirements process drives everything else
- Requirements should be gathered repeatedly based on feedback
- An iterative development process usually works best
- Greater stakeholder involvement inceases success
- Poorly chosen metrics and inflexible demands lead to failure
- A feature is only complete when it has passed acceptance tests
- Working software is always losing value unless maintained