Agile Software Development

Lecture 3: Pleasing Your Customer

Mahmoud El-Gayyar

elgayyar@ci.suez.edu.eg

Slides are a modified version of the slides by Prof. Kenneth M. Anderson

Presentation Topics

1) *Software Process Models (at least 3 other than Agile)*
2) *Scrum: Agile Framework*
3) *Extreme Programming (XP)*
4) *Example of Software Disasters*
5) *Git and Subversion: Distributed Configuration Management*
6) *Software Architectures*
7) *Cloud Computing*
8) *Web Services*
9) *Content Management Platform: WordPress*
10) *Software Teams Structures*

Exercise

A	C	E	G	I	K	M	O
10	20	30	40	50	60	70	80

Task Switching

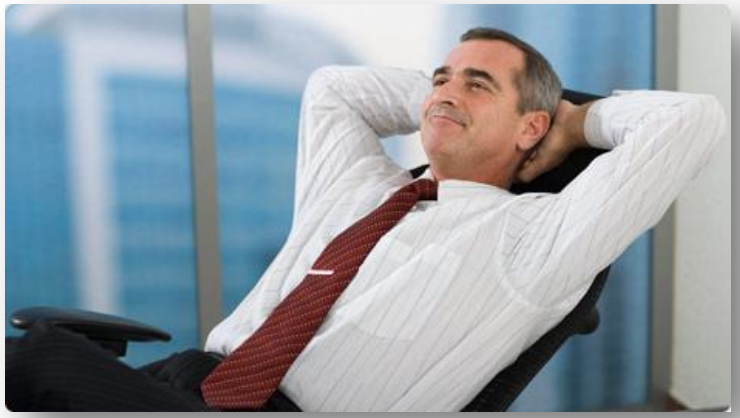
Number of Tasks	%Time on each task	%Cost of Switching
1	100	0
2	40	20
3	20	40
4	10	60
5	5	75
More than 5	Random	Even more

Outline (Pleasing your Customer)

- *Ultimate goal of software development*
  - Make your customer happy
- *Importance of Iteration*
- *Initial steps in planning a development activity*

Pleasing Your Customer

- *If your customer is unhappy, you're unhappy*
- *The ultimate goal of software development is*
  - solving your customer's problem by delivering
    - *what is needed*
    - *on time*
    - *on budget*
  - This is NOT easy



- *Important: Need to have a clear idea of who your customer is!!*

Who is the Customer?

- *The person or persons playing the role of the customer can vary across development contexts*
- *In this chapter, we have a situation in which we are being hired by Tom, a small business owner, to create a website*
  - Tom is clearly the customer
  - he is providing requirements and paying for the work
- *But when the website is deployed, who becomes the customer?*
  - Tom's customers

Customer == User

- *HCI research shows that systems live or die by **how happy** the "end users" are with the system*
- *Tom's customers in this case are the end users*
  - However, in this initial sequence of development, we only have Tom and we have to go by what he says
  - In the future, Tom will be hearing from his customers about the utility and usability of our website and he will convey that feedback to us
- *What's the difference between **utility and usability**?*

Other Customers

- *You (!)*
  - Often for only small scale software
- *CTOs*
  - Acquiring enterprise level systems for an organization
  - Who are the end users in this situation?
- *New Application Development (e.g. mobile app)*
  - For version one: development team
  - How can you avoid this? Who are the end users?

Example: Tom's Trails - I

Conference in 3 months

How Long?
How Much?

Example: Tom's Trails - II

- *Big Bang Approach to Software Development*
  - The developer that Tom hires
    - *takes his initial ideas*
    - *disappears for a month and codes like crazy*
    - *displays final product to Tom*
  - What happens?

Example: Tom's Trails - III

- *The developer delivered software that did not meet Tom's expectations*
  - And if the customer is unhappy…
- *Reasons*
  - Did not fully understand Tom's requirements
  - Tom's view of what he wanted can shift over time
  - Did not check in with Tom during development
  - So any unclear requirements were resolved by developer making arbitrary decisions

Example: Tom's Trails - IV

- *What do you think if the Tom said: "The customer should be able to search for trails"*
  - A customer should see a world map, then he can select a location and search for trails nearby.
  - The user should scroll through a list of tourist spots and find trails that led to and from those spots
  - The user should enter a zip code and a level of difficulty to find all trails that match the difficulty and near the zip code.

Not Guesswork

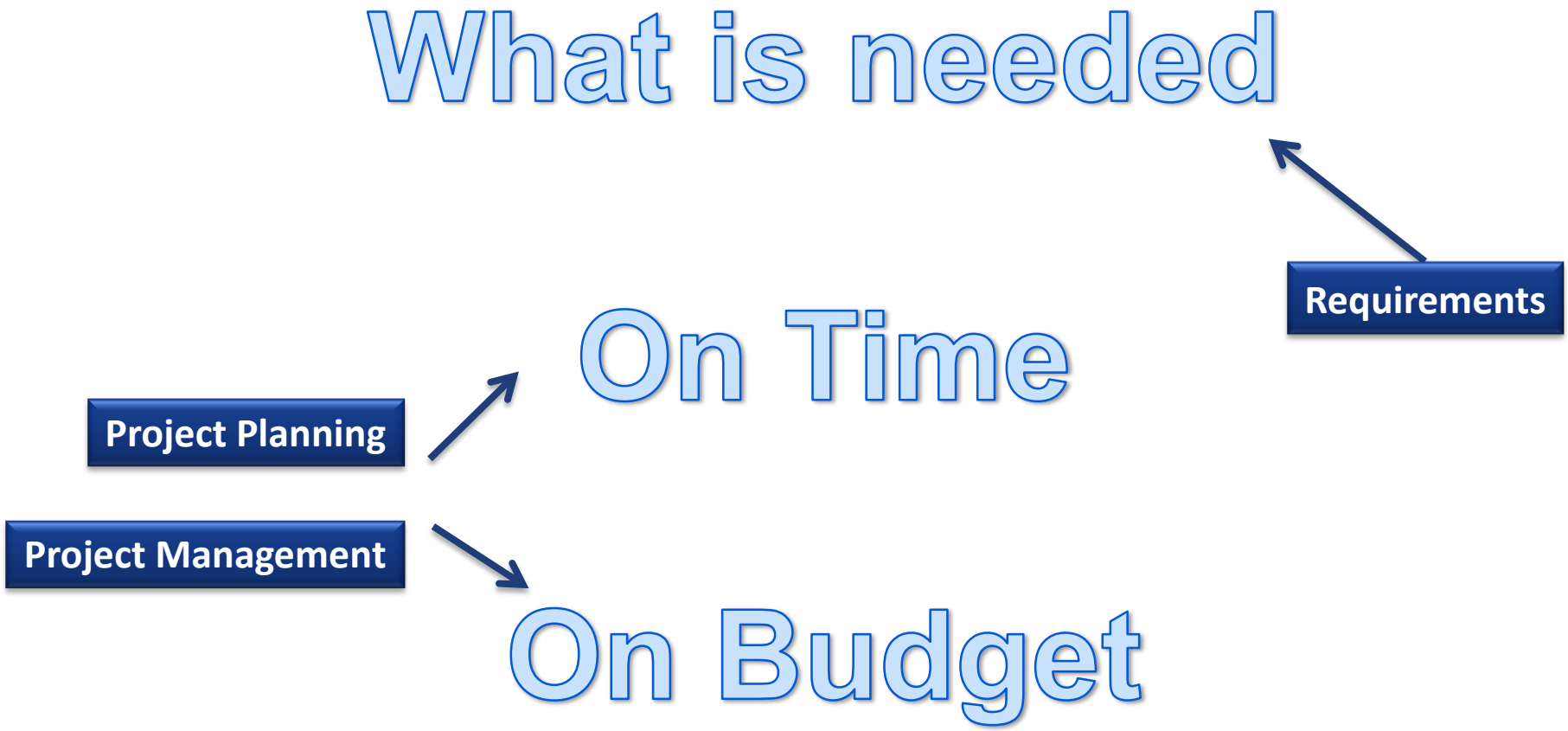
Software

is NOT

Guesswork

You need to keep the customer in the loop to make sure you're on the right path
even when you're SURE you know what the customer wants… why?

Great Software Delivers



So, how to get there?

- *The developer in the example*
  - did not apply basic project planning/management techniques
  - did not acquire feedback
  - did not clarify requirements with the user

- *The developer made use of a life cycle called*

Code and Fix

actually just "code" and now needs to do a LOT of fixing!

Without Iterations

- *Iteration is important because requirements **CHANGE***

With Iterations

- *With iteration a project can make **corrections** as requirements change so that **what's delivered** matches **what's needed***

An Iteration - I

- *An iteration produces **working software***
  - when you complete an iteration, you have something to show the customer
- *Iterations should be relatively short; where "short" is defined by the estimated length of the project*
  - they can be anywhere from ***one week to six weeks***
  - Our book picks ***20 days***, because that is roughly what you get if you start with a calendar month and subtract out the weekends

An Iteration - II

- *Iterations are "**fractal**"*
  - They decompose nicely into smaller iterations
  - A 20 day iteration can be split into two "internal" 10-day iterations
  - A work day can be split into roughly three 2-hour iterations
    - *Here the customer is yourself: what can I get done in two hours*
    - *did the result match my expectations? should I show this to my co-worker for feedback? etc.*
- *Iterations promote **continuous building and testing***
  - Otherwise, you are not delivering "working software"

An Iteration - III

- *Each iteration is a **mini life cycle***
  - A typical life cycle consists of these major activities
    - *Requirements; Design; Code; Test; Deploy*
- *High-level: ideas come in and working software comes out*
  - Iterations decompose this high-level process such that all of the major steps are performed for each iteration
- *Increase your chances of producing high quality software that solves your customer's problem*

Recommendations for Iterations

- *A published analysis of software development practices in 2001 showed that:*
  - the ***less functional*** the initial delivery the higher the quality of the final delivery!
  - the more frequent the deliveries, the higher the final quality!

- *These results suggest that iterations need to be **short!***

Iterations Help Your Plan

- *Iterations can help you plan ahead to achieve success*
- *Input:*
  - List of features with estimates and priority
    - *Customer supplies list and priority*
    - *Book recommends priority scheme based on multiples of 10*
    - *Developer supplies estimates (we'll see this later)*
- *Output:*
  - List of proposed iterations with features assigned
  - Highest priority features are tackled first
  - Features assigned have estimates less than iteration length

Can you do it?

- *Having an iteration plan lets you keep track of whether the project is feasible*

(Days of Work Left) - (Days left before deadline)

= =

Can you do it?

want it NEGATIVE

15 - 30 == -15 (i.e. fifteen days ahead of schedule)

Iterations Help You Respond to Change

- *And things will change*

- *You need a new plan that shows how you'll respond to the change request*
  - It will most likely create one or two more iterations which can be stressful because, note, the deadline HASN'T changed

- *Later !!!, we'll discuss what to do if the new schedule slips past the deadline*

Working Code

- *Note: Your software isn't complete until it has been released*
- *We measure progress via working software*
  - NOT via number of iterations complete
- *So, if we complete three out of 10 features in two iterations of a four iteration project*
  - We are 30% done NOT 50% done

Time to Try It

- *Iteration Exercise*

Time to Try It

- *Iteration Exercise II*

- *After 37 days you go new Reqs.*

Task	Priority	Estimation
Join Mailing List	10	7 days
Post to Mailing List	10	2 days
Organize a Trekking Group	20	10 days

- *User also change the task "search trails" priority to 10, adjust your iterations and see if you can catch the deadline?!*

Key Points

- *The ultimate goal of software development is pleasing your customer; to do that you need to*
  - deliver (i.e. ship) a system that solves the customer's problem
  - on budget (under budget preferable)
  - on time (ahead of schedule preferable)
  - ***(There is one more thing we should achieve; what?)***
- *Iteration is the key technique to achieving this goal*
  - Provides ability to plan project, respond to change, and deliver what's needed