Comp8081
Management Issues in Software Engineering

Donna Turner

Agenda

- Attendance
- Review: Feature Set Control and Project Recovery (McConnell Chapters 14 & 16)
- Teamwork and Team Structure (McConnel Chapters 12 & 13)

$\left.\right\}$ 1 hour

- Assignment 3 $\longrightarrow$ 2 hours
- Next Week: last lecture

Review:
Feature Set Control

McConnell Chapter 14

Aside – Formal Requirements

We looked at two types of requirements:

- System Requirements
- User Based Requirements
  - Use Cases
  - User Stories

System Requirements - Examples

- **Shall** Requirement
  - The system shall process updates from the data source within 6 seconds from initial receipt.
- **Will** Requirement (describing another system)
  - The data source will provide updates every 10 seconds in Json format.
- **Should** Requirement (design requirement)
  - The system should not prevent users from carrying out other activities while it is processing updates from the data source.
- Note:
  - *Will* and *should* requirements are typically to provide context
  - Notes are common, to provide further context

User Based Requirements - Examples

As IT Security, I want Sharepoint to have a whitelist of attachment file types so that we can prevent the upload of potentially malicious files.

- IT Security – Who
- a whitelist of attachment file types – What (i.e., the feature)
- prevent the upload of potentially malicious files – Why

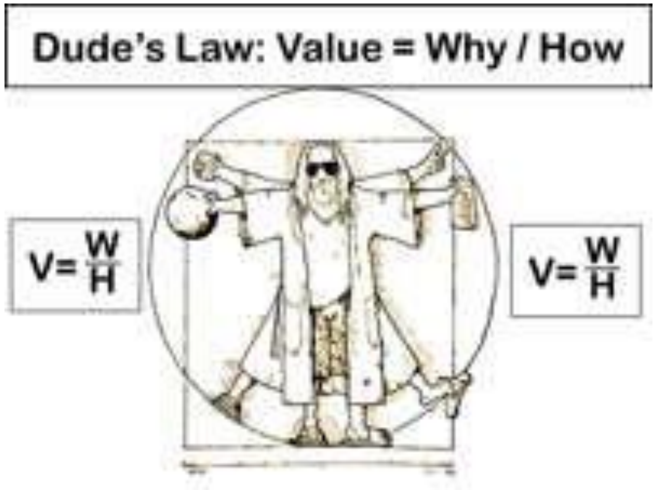
User Based Requirements - Examples

User Stories – Help to provide the 'why' to those implementing and testing the requirement.

Dude's Law: V = W / H,
where V is value,
W is why (intent) and
H is how (mechanics).

If (H)ow increase and (W)hy is constant, then (V)alue is reduced.

If your W is constant (you know what you expect) and you reduce H (less process) then the V increases.



As you drive the (H)ow towards zero, which you could call leaning out your processes, (V)alue increases even if Why is constant.

https://devjam.com/2010/08/05/dudes-law-gordon-pask-shoveler/

SMART Requirements

- **S**pecific
  - A good requirement is specific and not generic. It should not be open to mis-interpretation when read by others.
- **M**easureable
  - This answers whether you will be able to verify the completion of the project. You should avoid signing up for any requirement that cannot be verified as complete.
- **A**ttainable (Achievable, Actionable, Appropriate)
  - This is intended to ensure that the requirement is physically able to be achieved given existing circumstances.
- **R**ealistic
  - Answers whether the requirement is realistic to deliver when considering other constraints of the project and requirements.
- **T**ime Bound (Timely, Traceable)
  - Where appropriate each requirement should be time-bound or specify by *when* or *how fast* a requirement needs to be completed or executed.

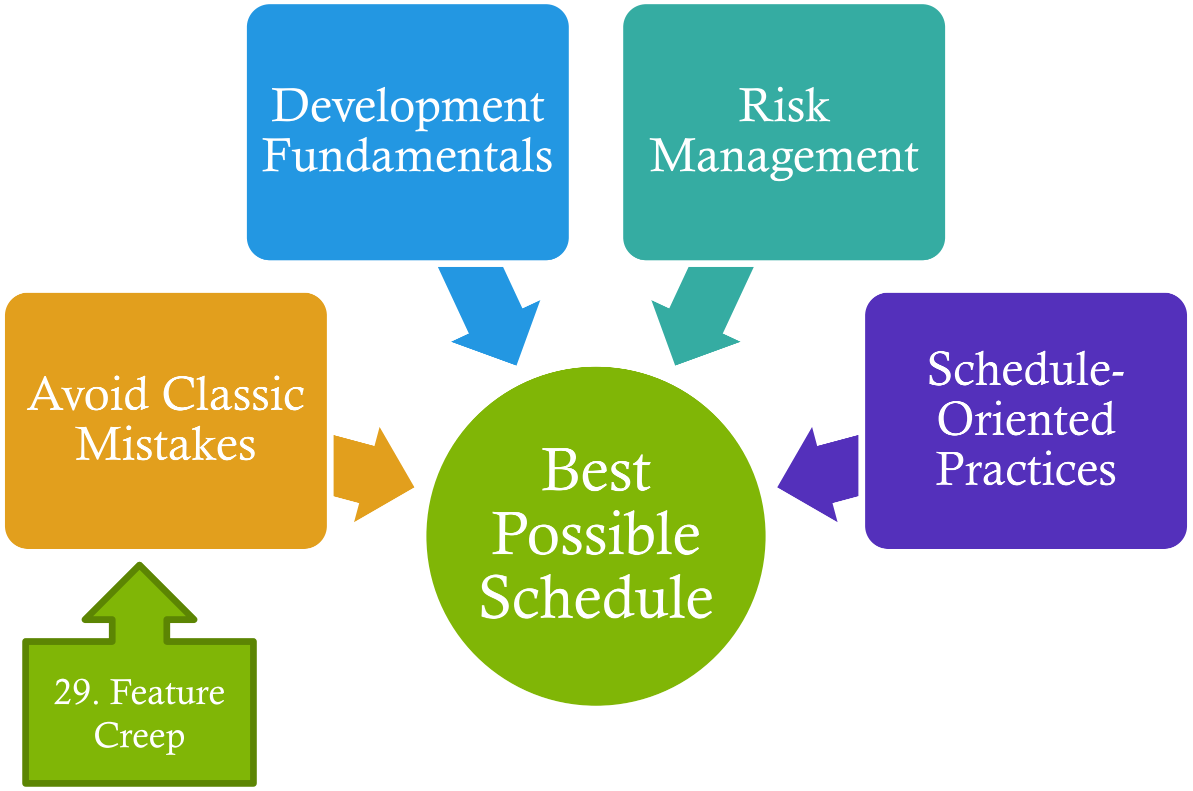
Requirements Management

Scope Creep

- We've looked at this topic a number of times already
- It's a consistent, important theme in what we do

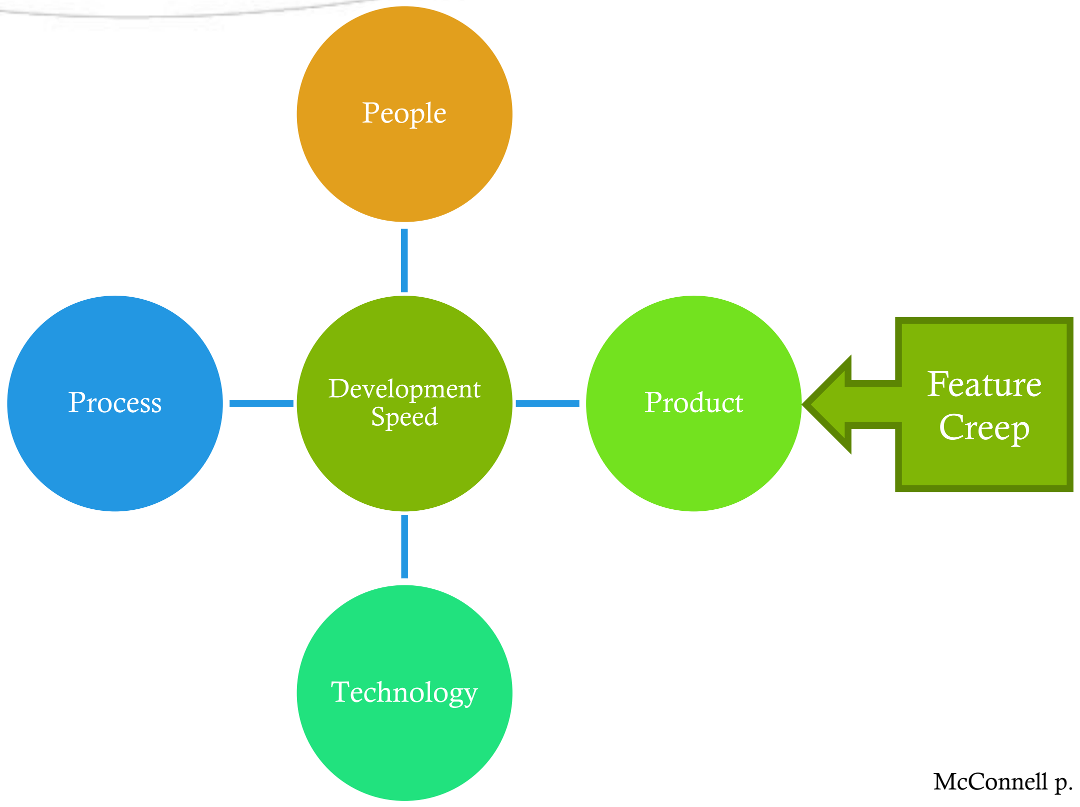
I HIRED A CREEP TO HELP DETERMINE OUR PRODUCT'S FEATURES.
www.dilbert.com scottadams@aol.com
YOU NEED MORE FEATURES.
GOOD WORK.
© 2001 United Feature Syndicate, Inc.
WHEN CAN YOU HAVE THAT DONE?
GAAA!!

Four Pillars of Rapid Development



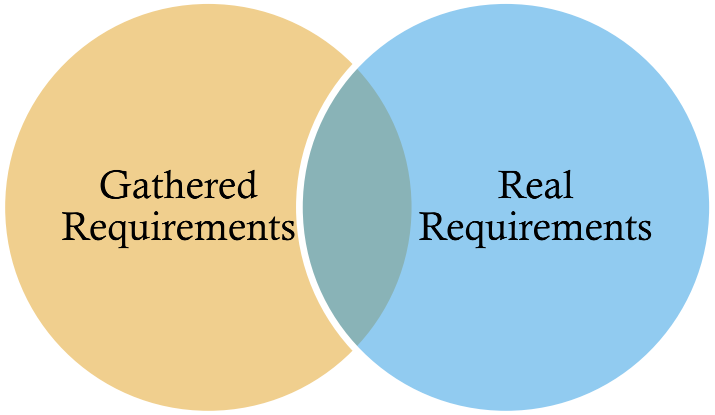
McConnell p. 9

Four Dimensions of Development Speed

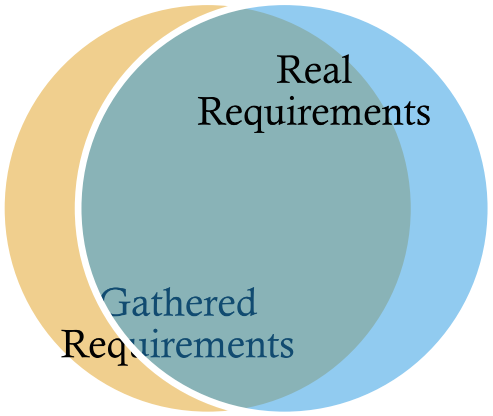


McConnell p. 11

Customer Oriented Practices



"Typical" requirements gathering processes



Customer-oriented requirements gathering processes

p. 240

Lifecycle Model Considerations

Different projects have different development needs – even if they all need to be developed as soon as possible

Requirements Considerations

- How well do my customer and I understand the requirements at the beginning of the project?

Architectural Considerations

- How well do I understand the system architecture?
- Am I likely to need to make major architectural changes midway through the project?

Reliability Considerations

- How much reliability do I need?

Future Version Considerations

- How much do I need to plan ahead and design ahead during this project for future versions?

McConnell, p. 154

Feature Set Control in Project Stages

Early Project Control	Mid-Project Control	Late-Project Control
• Define a feature-set consistent with schedule/budget constraints	• Controlling creeping requirements	• Trimming feature to meet schedule and budget
1 • Minimal spec	• Change control	• Feature cuts
2 • Requirements scrubbing		
3 • Versioned development		

Review: Project Recovery

McConnell Chapter 16

Project Recovery

What does McConnell Say?

- General Recovery Options
  - Cut scope
  - Increase productivity
  - Slip the schedule
  - A combo of the first three

- First Steps
  - Assess your situation
  - Think win-win
  - Prepare yourself to fix the project
  - Ask the team for input
  - Be realistic
- Dimensions of Development Speed
- Then, Timing

Recovery: Dimensions of Dev. Speed

People

- Restore group morale
- Clean up major personnel and leadership issues
- Add people carefully
- Focus people's time
- Allow differences
- Ensure pace

Process (1)

- Identify/fix classic mistakes
- Fix the clearly broken parts of your development process
- Create detailed mini-milestones
- Link schedule to milestone completion
- Track progress meticulously

Process (2)

- Record reasons for missed milestones
- Recalibrate after a short time
- Only commit to a meaningful schedule
- Manage risks painstakingly

Product

- Stabilize requirements
- Trim the feature set
- Assess your political position
- Take out the garbage
- Reduce the number of defects – and keep them reduced

Did we agree on ommiting 'Technology'?

McConnell, p. 376-384

Teamwork

McConnell Chapter 12

Teamwork

Some questions to start

- We have discussed Motivation and …
- Under "Employee Engagement" we said, "having positive working relationships" is a factor
- Can we link Motivation to Teamwork a little more directly?
- What enables high performance on a team? Top one or two factors?

Creating a High Performance Team

- Shared, elevating vision or goal
- Sense of team identity
- Results-driven structure
- Competent team members
- Commitment to the team
- Mutual trust
- Interdependence
- Effective communication
- Sense of autonomy
- Sense of empowerment
- Small team size
- High level of enjoyment

McConnell, p.278-279

Details of Some Factors

Results-driven Structure

- Clear roles
- Accountability
- Effective communication
- Individual performance monitoring
- Feedback
- Fact-based decisions

Competent Team Members

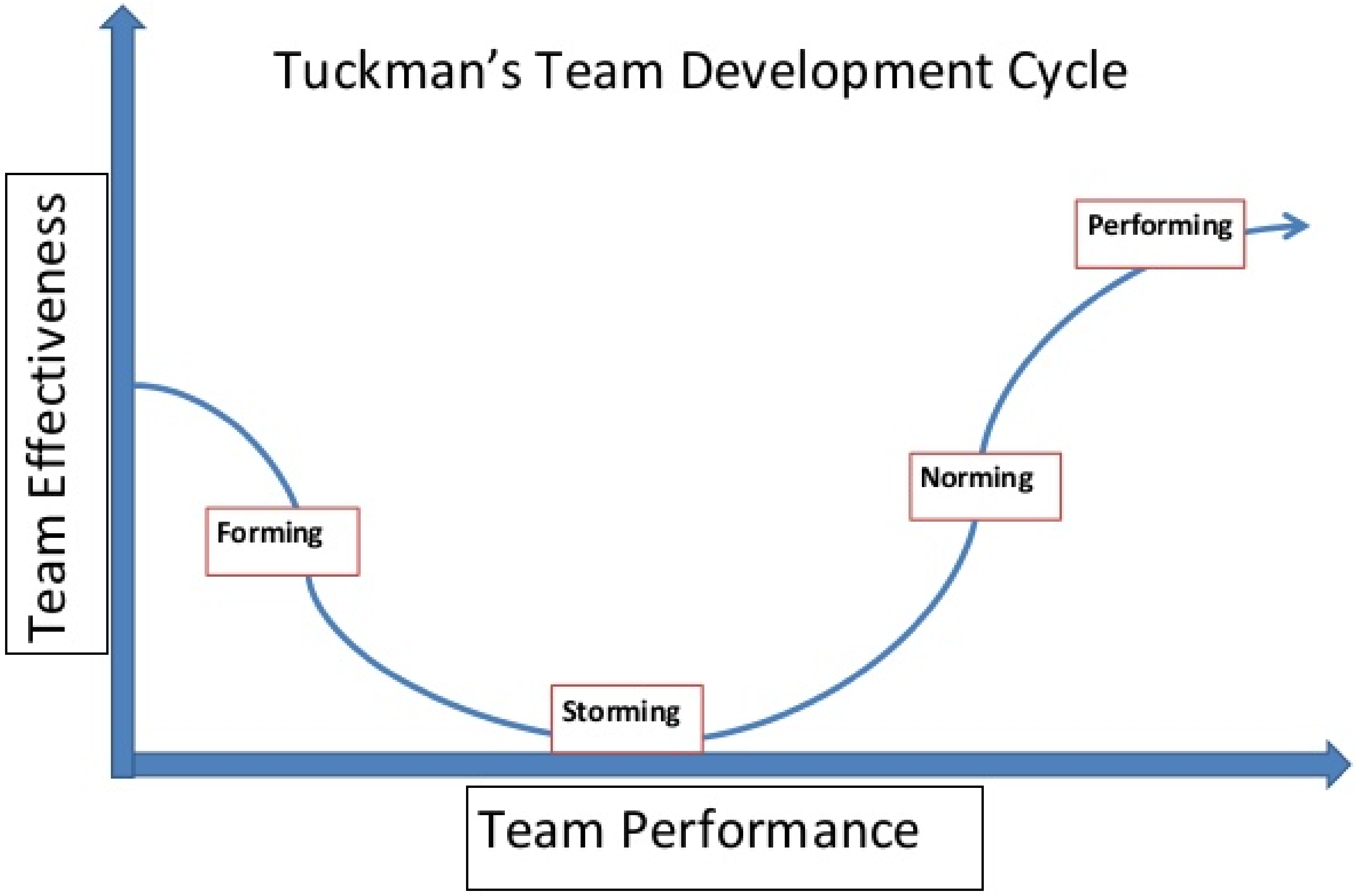
- Selection processes
- Specific technical skills
- Strong desire to contribute
- Specific collaboration skills
- Mix of roles (types)

Management Tips

- Establish a vision
- Create change
- Manage the team as a team
- Delegate tasks to unleash energy and talents
- Leave details of "how" to the team

When you look at this, which part(s) is/are owned by the individual?
Which part(s) is/are owned by the manager?
Which part(s) is/are influenced by the team?

Stages of Team Development



Conclusions

- What conclusions can we draw?
  - For individuals?
  - For team leads and managers?
- How will you incorporate the “peopleware” topics we have now reviewed for into your next team-based projects?

Team Structure

McConnell Chapter 13

Questions to start

- What kinds of team structures have you experienced?
- How much time do you think one “direct report” takes?
- What do you think can happen when companies grow, if they don’t have a team structure plan in place?

Video: Spotify Squads

- https://www.youtube.com/watch?v=4GK1NDTWbkY
- 13 minutes

Next week

- Course Review
- Final Review

Assignment 3

- Open book/internet
- Try to avoid eavesdropping on other groups, if possible
- Must submit to D2L by the end of class

Comp8081

end of Week 13

Donna Turner