

Lecture #1

Agenda

- Course Structure and Expectations, Syllabus
- Logistics
 - One weekly class, Wed, 6:00 pm EST
 - Six modules, (1) lecture, (2) assignments & quizzes
 - Term project team meeting, weekly
 - Submissions are due Wednesday morning, bi-weekly
- Module 1 . Introduce a context.
 - Globalization
 - Requirements
 - Engineering Management

Learning Objectives

Upon successful completion of this course, you will be prepared to:

- ✓ • Manage global product development
- ✓ • Solicit, define and scope requirements as part of the product backlog grooming
- ✓ • Play an effective role of a Software Engineering Manager in a context of IEEE CSDP
 - Select an estimation method that is appropriate for a specific phase of a product life cycle. Oversee adoption of a consistent methodology to narrow the Cone of Uncertainty.
 - Support the Scrum delivery framework; become aware of several agile certification paths.
 - Play a role in a peer review, request and provide constructive and concise comments.
 - Assess common security threats and establish corresponding deterrents
 - Evaluate software development tools (approved, allowed, restricted), while following the Magic Quadrant technique. Maintain a logical relationship between tools and processes to optimize their variety throughout an organization.
 - Articulate a strategy for system & unit test leading to continuous integration and delivery.
 - Structure a project asset library aiming at a single-click navigation to a requested artifact.
 - Provide leadership to a process program using SEI CMMI as an improvement model.

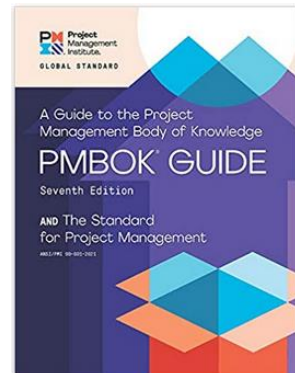
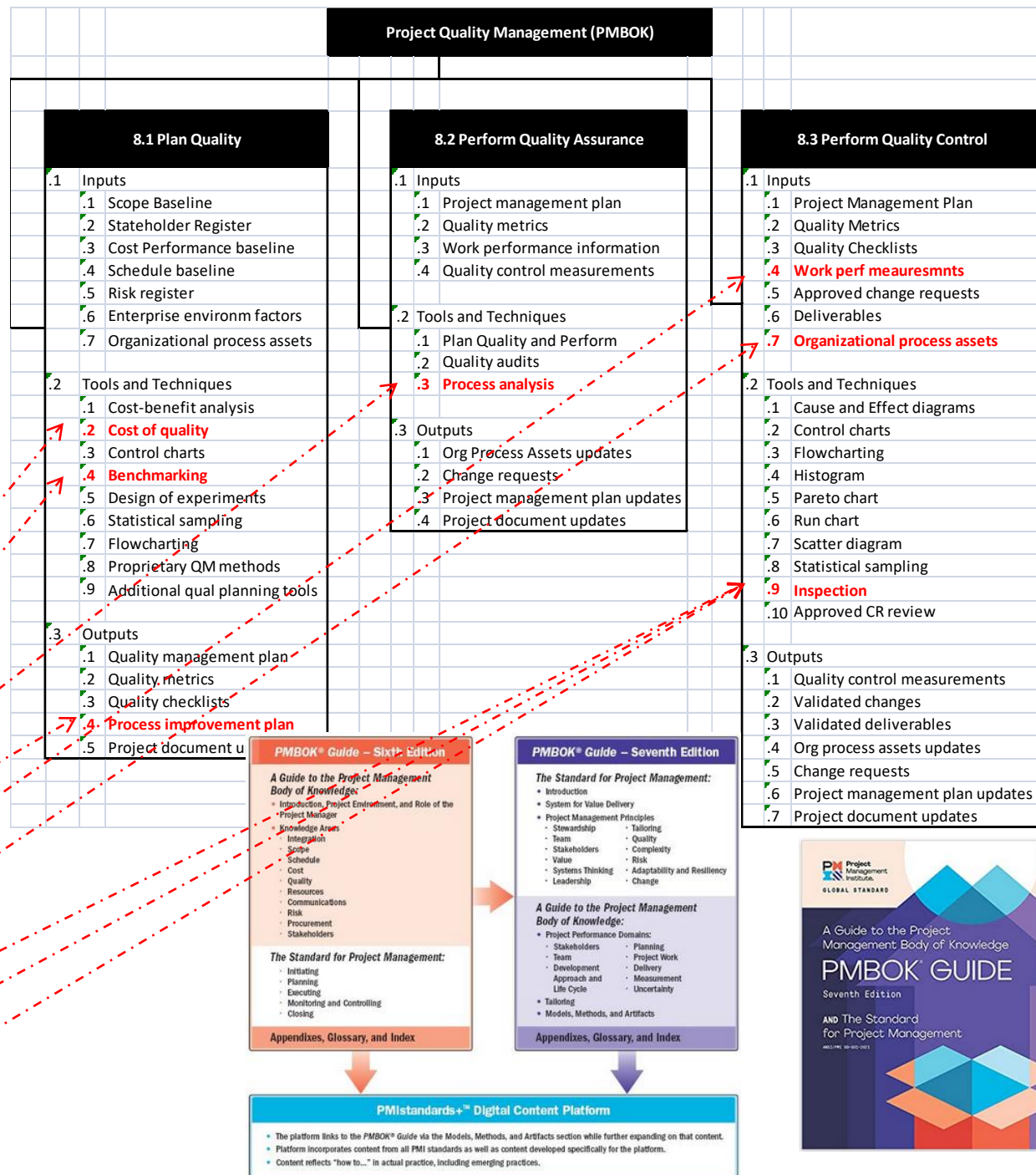
Mapping This course into PMBOK Quality Management

It is important to map the topics of this course into a well-accepted industry standard, PMBOK. Topics here are not improvised each running of the course, but in fact founded on a broad engineering experience.

Reference: "PMBOK Guide", PMI ANSI

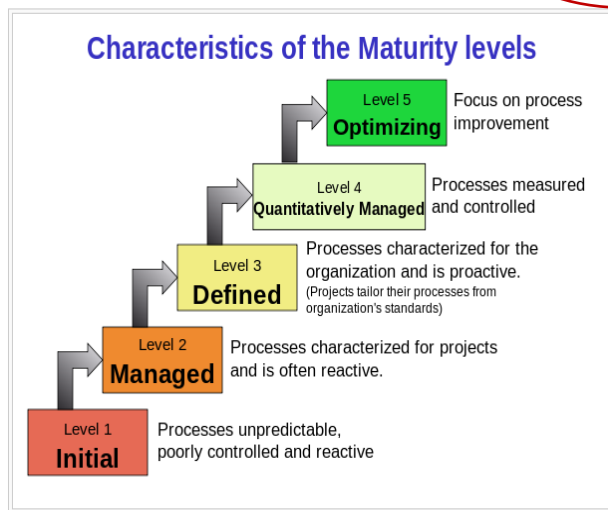
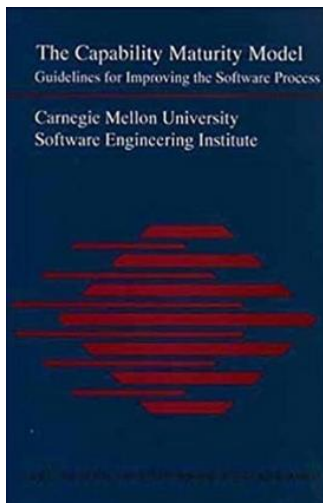
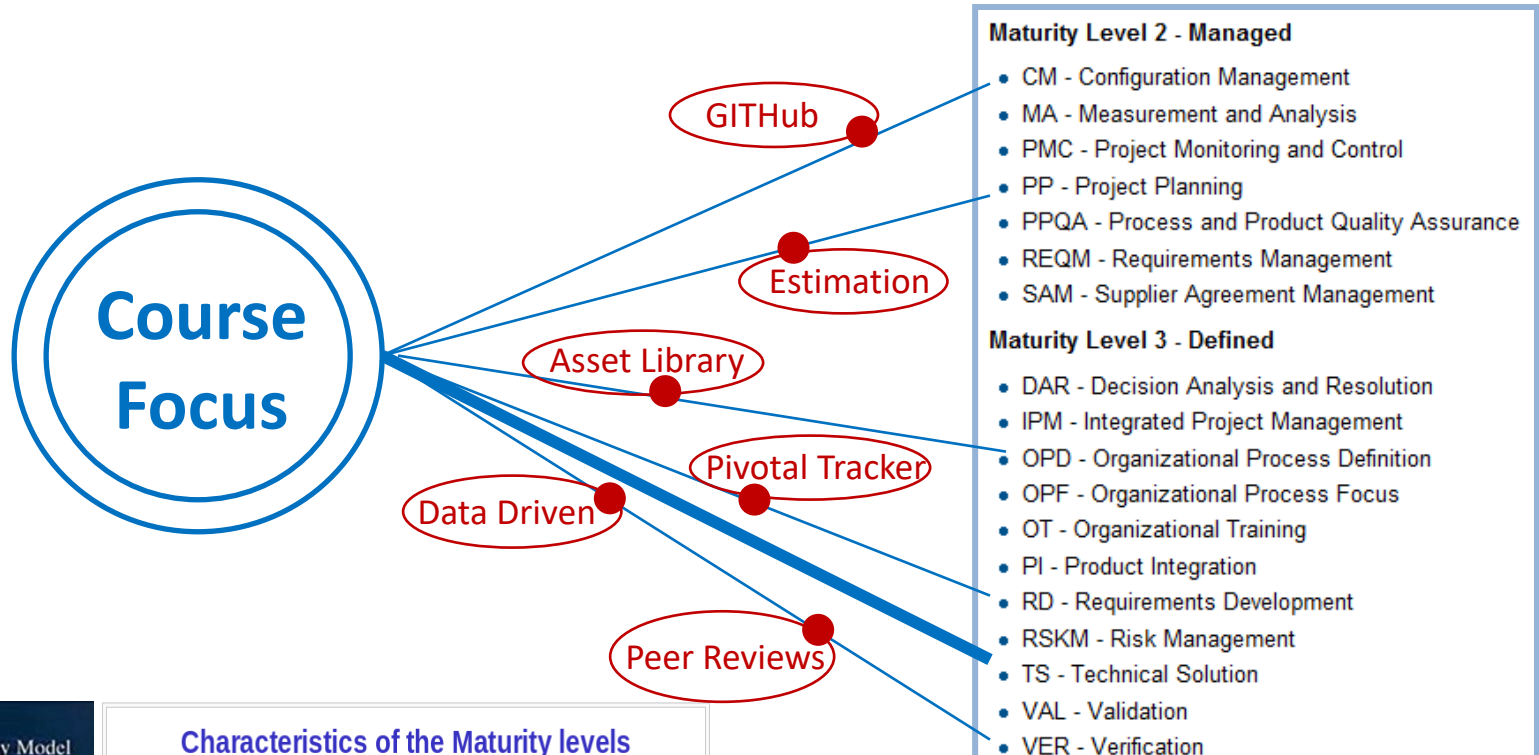


MET CS This Course
Cost of Delay assignment 1 mod 4
Benchmarking assignment 3 mod 1
Analysis of engineering capacity Quiz 2
Agile adoption plan assignment 3 mod 3
Cone of Uncertainty estimation mod 2
Process Assets Library mod 6
Peer Reviews mod 3
System test mod 5
Unit test mod 5



CMMI - Capability Maturity Model Integrated

– Where are you?



Reference: "CMMI For Development", version 1.3; Software Engineering Institute

How does CS 633 fit into overall BU program (MSCIS and MSSD)

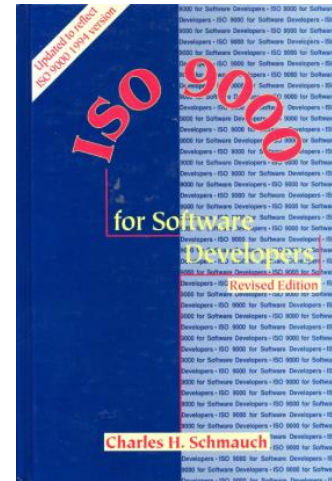
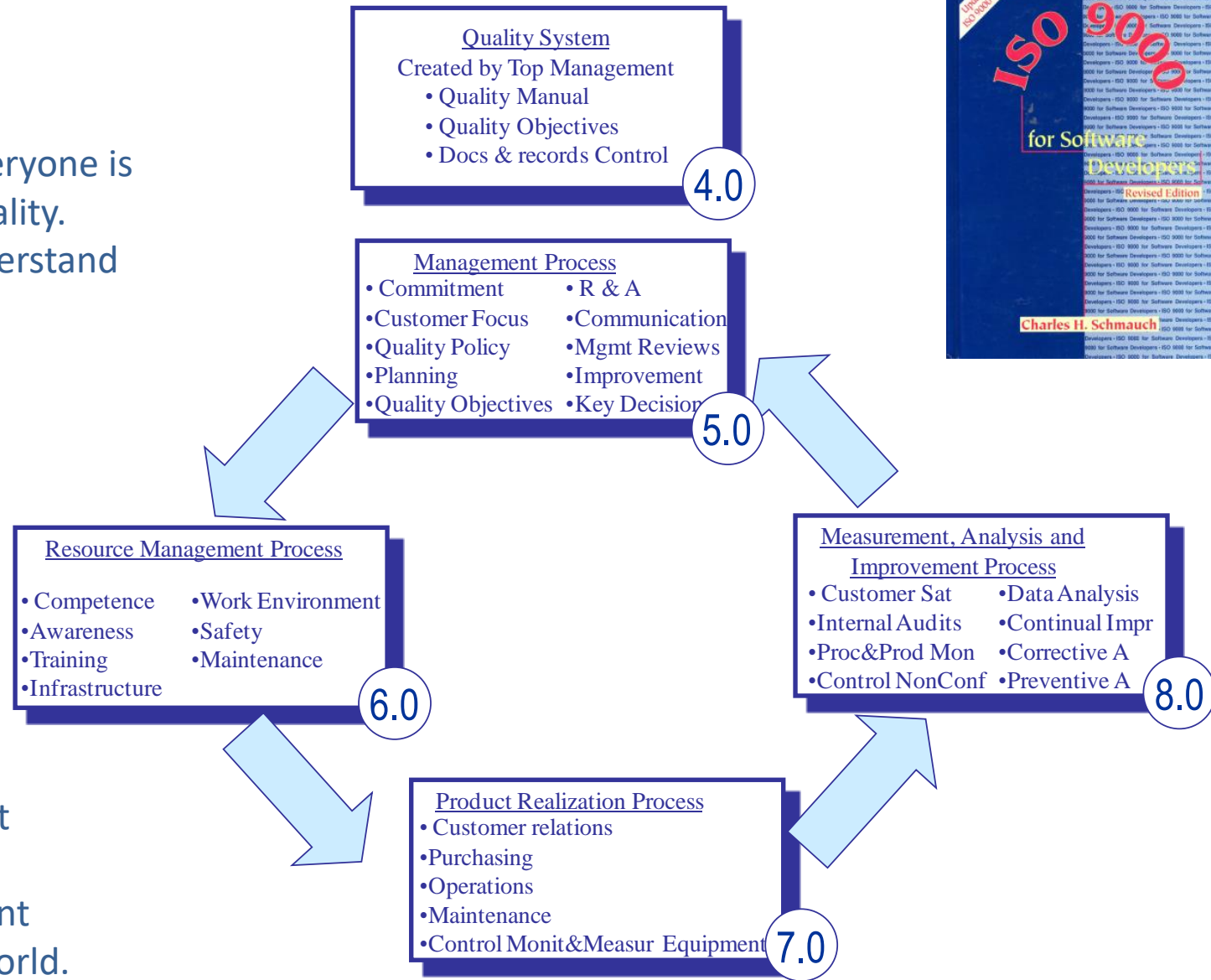
Courses are expected to be in-lock with each other, have common threads and have some overlap. Also, each course should have a distinct angle and objectives.

- **IT Project Management (CS632)**
PMI is not a focus of CS633
CS633 maps into the section of PMI called Quality Management
- **Agile Development (CS634)**
CS633 interprets Agile as a means to deliver software, not as a focus of the whole discussion.
For term project, CS633 uses a simplified Pivotal Tracker instead of Trello
- **Software Engineering (CS673)**
CS 673 is somewhat more technical, since CS 673 expects each student to develop a piece of code, as part of the term project
- **Security Policy and Procedures (CS684)**
CS 633 has an in-depth examination of current events, focusing on SSO as part of overall system design
- **Systems Analysis and Design (CS682)**
CS 633 is a logical continuation of CS 682, as it relies on UML skills for a term project
- **Enterprise Architecture (CS783)**
CS 633 covers an in-depth examination of Process Architecture and Asset Library

ISO 9001:20015 Quality Management Systems

Pictorial below shows the interaction of key ISO processes. It implies that "key" processes are being distinguished from "non-key" ones.

ISO means "equal", suggesting that everyone is responsible for Quality. Everyone shall understand key processes.



ISO 9001 is the most widely used process improvement framework in the world.

Introduction

Life Hacks (Helpful Hints)

Focus on learning . You should assume that you will be successful with this course and hence, grades will be reflective of that. Focus on those best practices that you are able to actually apply. In other words, in this course you should do seven (7) things ... learn, learn, learn, learn

- Prior to a lecture - scan through blackboard and get familiar with new material
- Slides are uploaded before each lecture. Some students make notes during sessions.
- Prior to a live session – copy and review assignments from blackboard; prepare questions.
- Be active during live sessions; the primary learning method for some students is to engage in a discussion, bringing about personal experience, which could be quite stimulating.
- Quiz is reviewed during a live session and then opened to students; make first attempt while review is still fresh in your mind; multiple attempts allow the knowledge to sink in

This is an overview course. It delves into multiple topics. Buckle up. You cannot afford spending all your time on a single topic. Time box your assignments. This is not about a PhD thesis. This is about becoming aware of an important concept and then moving to the next concept. Assignments, quizzes and term project deliverables - are thoroughly discussed in the class.

This course is not about a specific language or a framework. Discussions - are optional, aimed to stimulate your thinking.

Term project brings it all together. All predefined deliverables of a term project are introduced during lectures. Do not wait, start coding at day one.

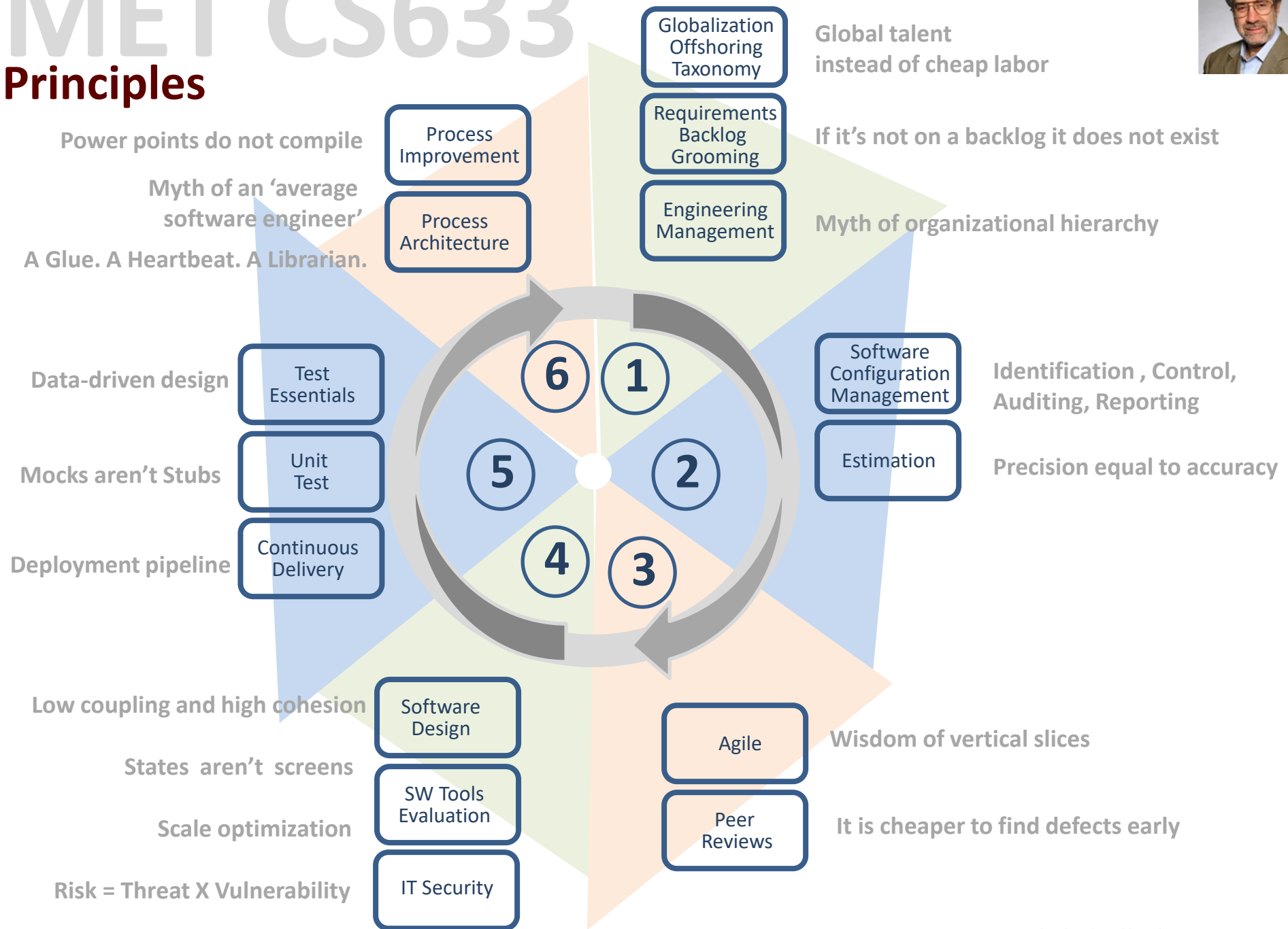
It is all about how to do software better, about software engineering and program management. **You cannot function as a professional software engineer or a project manager if you are unaware of these key concepts covered in the class.**

Students' Testimonials

1. I recently completed my master's degree and have very fond memories of my final class before graduation. As a Principal Software Developer at Liberty Mutual – class covers a plethora of topics relevant to my everyday work, providing me with the best-practice tools I need to succeed. Most specifically, the term project, which very accurately simulates a real-life product life cycle, was exceptionally rewarding. My team and I were able to build a cloud-based, production-ready application; all while fine-tuning and reinforcing the learning that coincided in the classroom sessions. I most highly recommend this class. Zac Fermanis. Software Engineer, Liberty Mutual.
2. The most impactful technique, to me, is what we have learned about Process Improvement. Not only is it important for software development, it also applies to many areas of the enterprise. In the government, we produce lots of standard documentation -- unfortunately, there is not always a standard way to create these artifacts. Therefore, more often than not, projects "reinvent the wheel" -- they create satisfactory artifacts, but the journey is not a straight path, it's long and arduous. There is no feedback, no "retrospective", so subsequent projects are destined to have the same fate. I will work to create a PAL in my division. I may not be able to affect processes at the enterprise level, but I can positively affect processes at the working level. Jason Morales. Project Manager Pomerleau.
3. Specifically, I hope to adopt aspects of continuous delivery into my projects. As a former writer, I am interested in the process of dividing branches into smaller pieces, always having the goal for a particular branch in mind. More than anything, though, what I have learned in this class is that software engineering is a realm of cultish devotion to trends and that rising above to understand the substance behind these trends, and to gauge for oneself whether they are worthwhile or the strange fleeting whims of people who are bored and want to make more work for each other, is crucial to anchoring oneself against the waves of hype. Michael Nizzary. Software Developer.
4. Tool selection is a best practice that I learned in this course and will be able to apply it to my job as a DevOps engineer. Since I work for a software company, we are constantly announcing a myriad of tools that we can leverage. However, it is important to make wise choices on which tools to use by listing the criteria that they should be judged on. Too often we simply get overwhelmed and randomly pick a tool because it is cool or popular, but not take into consideration its effectiveness. One of the techniques that I found to be especially useful for my job is predictive analysis. The ability to glance quickly at a chart based on historical data and deciding based on the direction of the curve is a good skill. At my job we deal too much with exceedingly small minute details and often forget to "look at the bigger picture" that may convey some critically important message for our business. Alicia Gallagher DevOps Engineer.
5. One of the best practices which I plan to adopt in my own work environment is refactoring. Refactoring of test code is just as important as the refactoring of the software's design as testing must be kept coherent. Tests which are difficult to understand may undermine the quality assurance process as we may not be testing everything that we think we are. For this reason, we should also consider using black box testing to discover issues that our team may have missed. Black box testers who are not familiar with the code may have different ideas about how it should work and be able to find issues accordingly. The combination of our usual QA process with these practices can greatly improve the efficiency of our team. Lupe Delgado Quality Assurance Engineer Morgan Stanley
6. Dear Professor Elentukh, I write this note to you to tell you that you have imparted, not only knowledge, but wisdom to me during this course. You have a very unique teaching approach. And when I say "unique", I mean that in the most positive sense. I have never been through a course where the professor actually gives out the answers to the quiz before the quiz. At first, it seemed counter-intuitive. However, if, as you always say, the goal is to learn, it makes sense. After all, how many times do students take tests and not ever know whether they got the right answer? That just leaves a hole in people's minds. So, going over the quiz each week and challenging us to think about it actually achieved its purpose: to actually retain the knowledge and learn. There are many things that took time to sink in. But, as you always say, "take some time to let it sink in" and, invariably, it did. There are many things I will take with me from this course but if I had to pick one it is this: If, during a peer review, someone starts attacking an author, that attack is always about something else. That really resonated with me and, I have to say, it changed my thinking. This is not always true for peer reviews; it is true for many things in life. Someone cuts you off in traffic, it is about something else. Someone gets angry in a meeting; it really is about something else. I can't be sure that I "got it" but that is what it meant to me. I told my wife about this concept and it was a real light bulb moment. It is so obvious but, at the same time, not obvious. You taught a very interesting course full of phrases and ideas that were brief but packed with meaning and with a lot of depth. So, I thank you. I thank you for not making us memorize a bunch of facts that we would forget shortly after the exam but, instead, teaching us to ponder, think about, and retain important ideas. Ian Edwards Program Manager WorkSafeBC
7. Hello Prof. Elentukh, My favorite parts of the class are your bi-weekly lectures from where I personally learn the most. This is the first time in my educational experience when answers of the quiz are open and discussed in the class. This approach teaches so much and so well. I read the paper "*Improving teaching and Learning Effectiveness of Computer Science Courses - Case Study*" authored by prof. Kanabar and you, to learn more about the approach and overall about the class. It is very interesting. Thank you, Professor! Best greetings from Sofia, Denitsa Koynova
8. One of the best practices from this course that I plan to take away is using functional tests more. In my company we tend to write unit tests first and then write the functional tests - as we are shipping the code to QA. David Dell. Software Engineer Maxar Technologies .
9. The most impactful best practice I learned in this course is how to conduct peer reviews. The current peer review process at my company begins with the developer adding every single person on my scrum team to the review, then waiting a week, and if there are two approvals on the code, merging it into the main branch. Now I know that this process is incomplete and ineffective. What we should do going forward is add less people, so nobody is intimidated but there is still the possibility of enough relevant feedback being collected. We also need to try scheduling Face to Face meetings to talk about the piece of code under review. The code base that we work on has existed for 30 years, and everyone is an "expert" about a difference piece. There often isn't cross-collaboration, and so people don't understand enough about the author's work area to detect defects. The usual process is one senior developer will look over it, possibly give a few comments that the author will then fix, and then once everyone sees the senior developer's approval, they blindly follow suit without even looking at the code. This must stop, and I plan to start scheduling Face to Face meetings where I can help my coworkers really understand what I was trying to do and motivate them to look at it themselves instead of waiting for approval from someone else. Piper Lincoln. General Dynamics.

MET CS633

Principles



MET CS633

Responsibilities



MET CS633

Do Not



Do not develop a process for its own sake, for a 'gram

Do not reduce architecture to an option

Do not intertwine test data with steps

Do not change code unless get a failing test

Do not toggle features if you can split them

Do not color wireframes

Do not introduce a tool that does not support the common process

Do not focus on threats before addressing vulnerabilities

Process Improvement

Process Architecture

Test Essentials

Unit Test

Continuous Delivery

Software Design

SW Tools Evaluation

IT Security

Globalization
Offshoring
Taxonomy

Requirements
Backlog
Grooming

Engineering
Management

Do not offshore if unable to monetize

Do not rely on grapevine requirements

Do not evaluate performance of individuals based on process metrics

Do not make a version - a part of file's name

Do not accumulate changes before implementing them

Do not average estimates

Software
Configuration
Management

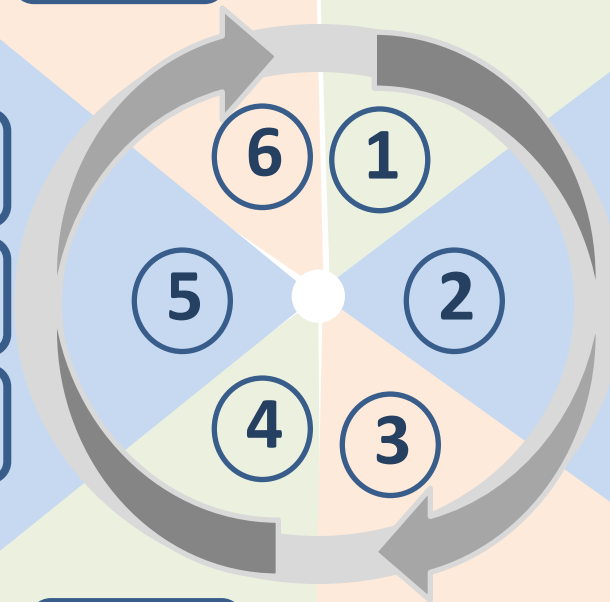
Estimation

Do not extend a sprint. Ship increments.

Do not defend your work product
Do not take a personal credit

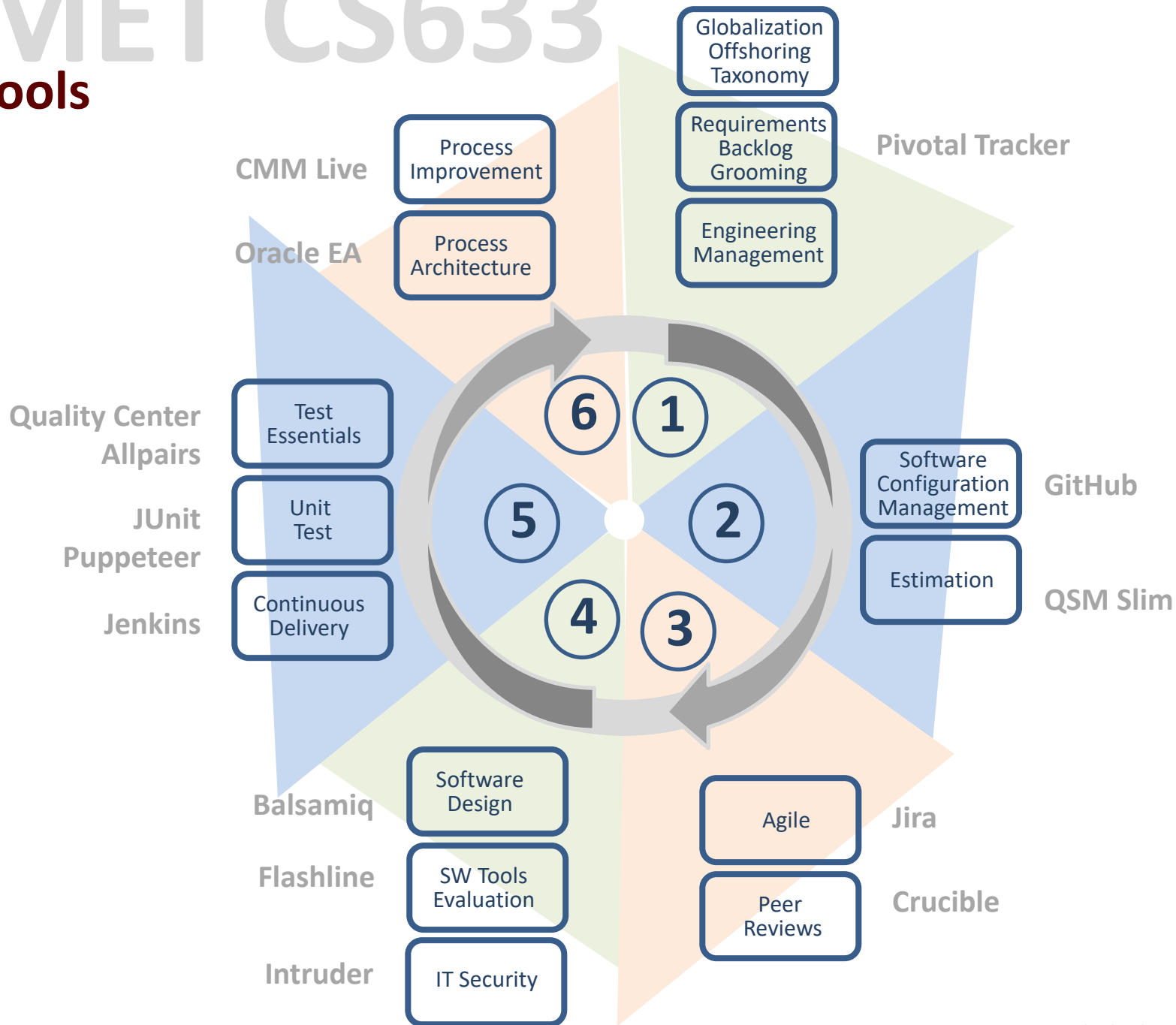
Agile

Peer
Reviews



MET CS633

Tools



Terminology and Taxonomy

Terminology and Taxonomy

The purpose of this section is to introduce initial taxonomy and key concepts of Software Quality Management.

After completion of this section you should be able to do the following,

- establish a mechanism to keep refreshing and clarifying the project taxonomy
- become aware of natural hindrances one overcomes while establishing clarity of a project environment

Taxonomy

- terminology
- taxonomy

Terminology and Taxonomy (T&T)

It is the responsibility of a Project Manager, at the beginning of a project, to document key terms used throughout the product life cycle. Everyone on a project team shall become clear about the basic concepts and terminology. As Jeff Sutherland puts it in the Scrum Guide "When a Product Backlog item is described as “Done”, everyone must understand what “Done” means”.

We attempt to follow the same at the beginning of our class.

For us to communicate effectively we need to establish and keep refining with each class – the set of consistent terms. We shall understand each other through the usage and manipulation of these terms. What is the difference between terminology and taxonomy ? Terminology defines individual terms. Taxonomy goes much deeper, it explores the connections and dependencies between various terms, it defines their scope and how they change over time. We shall always come back to the definition of terms and re-clarify them. Needless to say that every module of this course starts with definition of taxonomy relevant to the module at hand.

Low Coupling and High Cohesion Quality Center
 Continuous Delivery Process Improvement Engineering Management
 Backlog Grooming MVC Faults & Failures
 Process Architecture

Defect Density & Examination Rate Black Swan
 Cone of Uncertainty 5/20 Rule Regression UML
 Estimation SW Tools Motivation CMM ISO
 Software Configuration Management Scrum Git

Requirements Unit Test Cost of Delay
 Story Points
 Peer Reviews
 Mocks & Stubs
 Test Essentials

Asset Library Pivotal VersionOne Rally Agile Manifesto



Three Trends (TT)

There are three trends that dominate the discussion of taxonomy.

- First attribute of taxonomy is its unavoidable ambiguity. Take the term "epic" as an example. It's a bona-fide English word. One might think of a Leo Tolstoy with his multi-volume novels. In software engineering, the term is used as a high-level requirement that encompass other low-level stories.

Epic is a so-called “special word” that have a certain meaning within the boundaries of a certain projects. At the end of this session there is an assignment to research several software engineering terms and to overcome their apparent ambiguity.

- Second, terms are like coins. After putting them in a circulation, they rub against each other for a while, and they lose their identity, as their face-value become obsolete. Think about, how does it happen? what is the mechanics of losing value after an extensive circulation? Apparently the term "epic" could have many meanings that rub against each other, so their colloquial meaning become an average of each individual meaning.

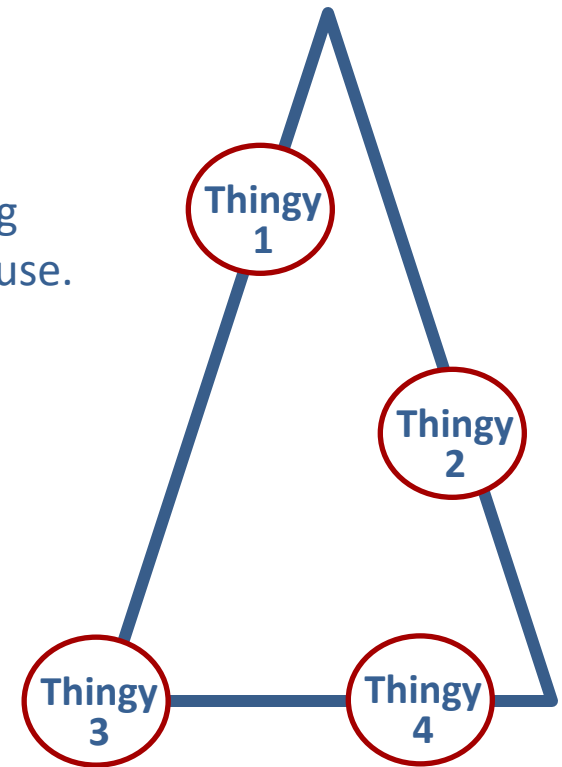
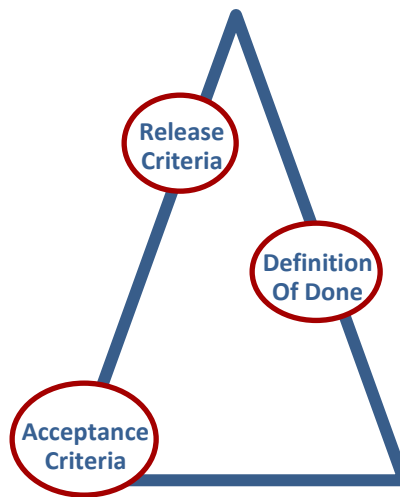
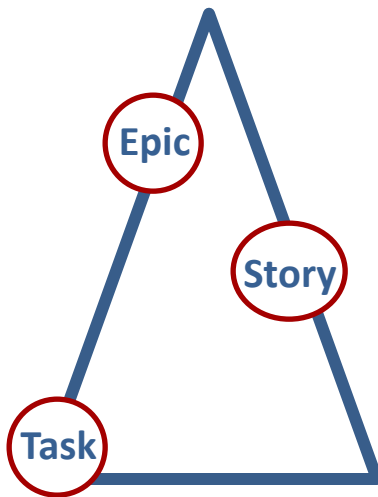


- Third attribute of many terms is their metaphorical dimension. When Andy Grove titled his book "Only The Paranoid Survives", he did not imply a literal meaning of a clinical condition requiring a medical attention. So it is up to a reader to interpret and to apply this apparent metaphor. When Michael Fagan writes in his paper ... moderator's task is to invite the phantom inspector... he did not mean to chase after ghosts, but he meant to assure the team collaborate synergistically. So the task in front of us is to overcome the shallow literal interpretation and to reach the real author's intent.

Tree of Thingies (TT)

It is the responsibility of a program manager at the beginning of a project to establish a consistent terminology for team's use.

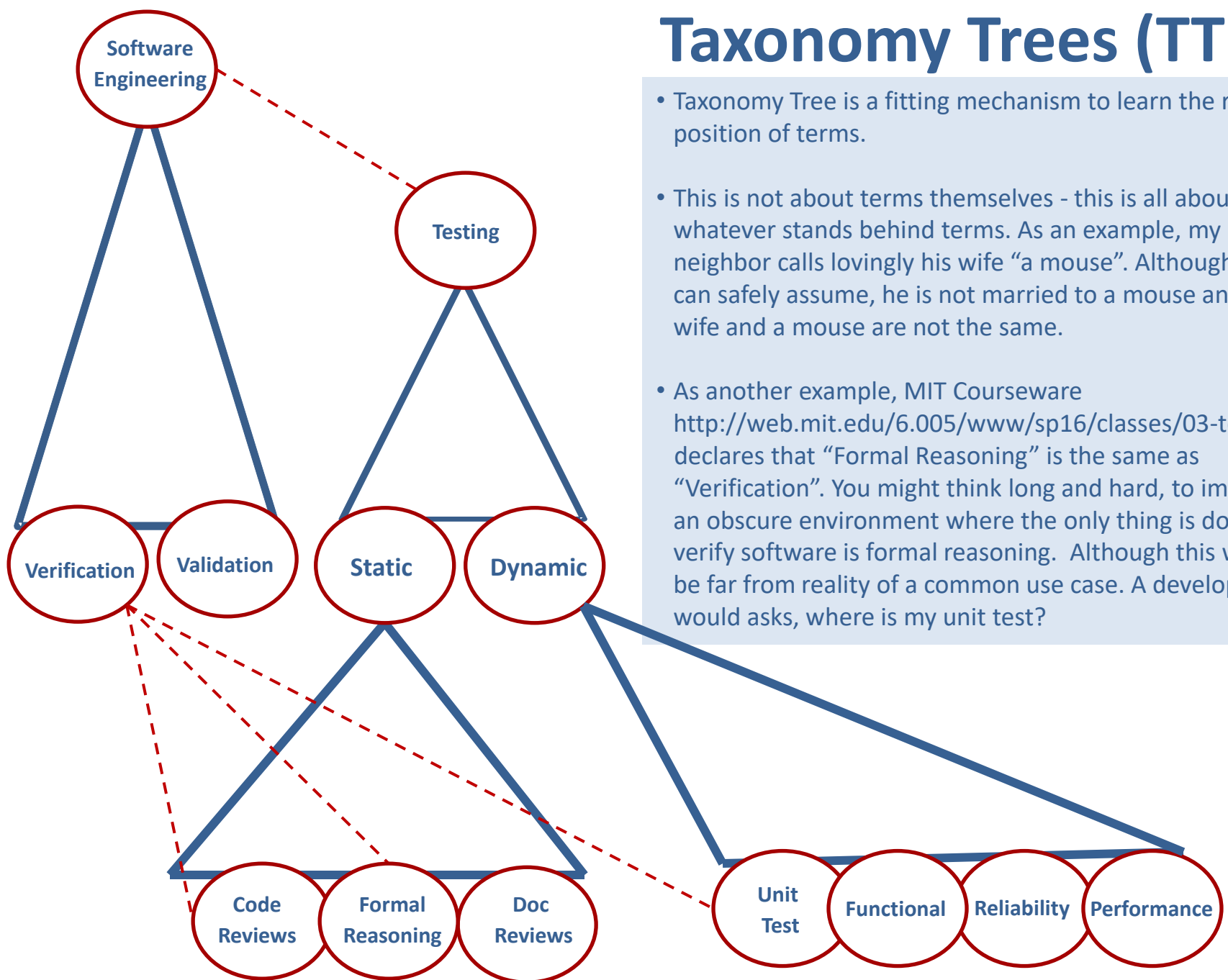
As a first step, before documenting a detailed terminology, it is prudent drafting a hierarchy, a tree with several levels and empty slots. So the second step is to start hanging actual terms on a tree.



How many TT levels do you have?
Hopefully not too many.

Do you have terminology conflicts?
Need to resolve them.

Taxonomy Trees (TT)



- Taxonomy Tree is a fitting mechanism to learn the relative position of terms.
- This is not about terms themselves - this is all about whatever stands behind terms. As an example, my neighbor calls lovingly his wife “a mouse”. Although, we can safely assume, he is not married to a mouse and his wife and a mouse are not the same.
- As another example, MIT Courseware <http://web.mit.edu/6.005/www/sp16/classes/03-testing/> declares that “Formal Reasoning” is the same as “Verification”. You might think long and hard, to imagine an obscure environment where the only thing is done to verify software is formal reasoning. Although this would be far from reality of a common use case. A developer would ask, where is my unit test?

The "A"-word

There are certain words around us that are over-used, over-loaded and have started losing their original meaning. Such an over-exposure is also driven by the key-words approach to the search of relevant content.

The “A” Word is a perfect example of such an evolution. Conventional wisdom suggests that you cannot publish a book or a paper these days without sticking this word inside the title. It became synonym with the “GP” – Good Practice. For example, writing down test cases or reviewing a project objective is a good thing to do - hence it must be an “A” Word.

So, unfortunately, many folks had to stop using this term, since it does not mean anything any longer; and for all practical purposes it is dead. Hope you have guessed already which Word I am talking about.

Word Of the Day : Lingua Franca

We have to quickly develop a common language so to understand each other. It is more effective to come up with such language before a project starts. To develop a common language for a project that has been completed is too late.

1620s, from Italian, literally "Frankish tongue." A stripped-down Italian peppered with Spanish, French, Greek, Arabic, and Turkish words, it began as a form of communication in the Levant. The name probably is from the Arabic custom, dating back to the Crusades, of calling all Europeans *Franks* (see **Frank**). Sometimes in 17c. English sources also known as *Bastard Spanish*.

A *lingua franca* ([/ˌlɪŋgwə ˈfræŋkə/](#)),^[1] also known as a **bridge language**, **common language**, **trade language** or **vehicular language**, is a language or dialect systematically (as opposed to occasionally, or casually) used to make communication possible between people who do not share a native language or dialect, particularly when it is a third language that is distinct from both native languages

Sources of Terms

There are several standard ways to research and clarify the terminology in front of us, Wikipedia, Margaret Rouse...

Welcome to Wikipedia,
the free encyclopedia that anyone can edit.
6,151,668 articles in English

..... engineering is the
study and application of
engineering to the design,
development, and
maintenance of
software

Margaret Rouse



Margaret Rouse

WhatIs.com

TechTarget

BIO

FOLLOW

integration.

A note from the author:

I speak fluent Geek, Biz-speak, Cloud, SAPanese, VAR, BI/BA, Storage, Security, Agile, Networking, SEO and marketing. I spend my days translating these highly specialized languages into plain English.

I try hard to make my definitions simple and straightforward so that anyone can understand them. It's not always easy and some days I do better than others. If you have a suggestion for how to improve a definition, please [contact me](#) and let me know! I'll be happy to list your name as a contributor.

.....Continuous software
development is a blanket
term that covers several
aspects of
an iterative

Off Shoring

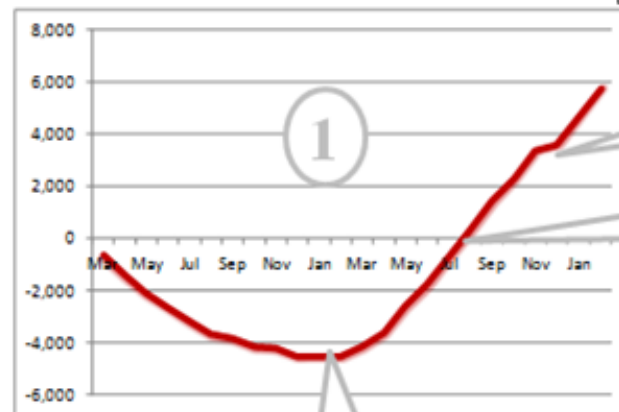
Distributed Software Development

From the case cited by Erran Carmel "Offshoring IT"

As we are considering moving the bugfixing from Boston to Singapore. By analyzing the shape of Cost Curve and playing what-if scenarios, we are able to optimize and justify offshoring.

US Fully Burdened Rate	150
20% cost of interruption	30
Cost to fix a defect in US	7
Cost to fix a defect in Singapore	4

	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb
Defects # fixed in Singapore	0	2	20	30	40	50	60	70	80	5	90	150
Incremental Bug Fixing Benefit	0	6	60	90	120	150	180	210	240	15	270	450
Cumulative Bug Fixing Benefit	0	6	66	156	276	426	606	816	1,056	1,071	1,341	1,791
One-time training event	-250											
Knowledge transfer	-100	-90	-80	-70	-60	-50	-40	-30	-20			
Cross-functional meetings	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20
20% cost of interruption	0	0	0	5	10	20	30	40	50	0	60	70
Training for new releases				-200				-200				-200
Translation services				-20				-20				-20
Common req process and tool	-100	-100	-100	-100	-400	-400	-100	-100	-100	-100	-100	-100
Common SCM process and tool	-150	-500	-500	-150	-150	-150	-150	-150	-150	-150	-150	-150
Common peer reviews training	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25
Common life cycle curriculum	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25	-25
Incremental expense	-670	-750	-750	-605	-670	-650	-330	-530	-290	-320	-260	-470
cumulative expense	-670	-1,424	-2,174	-2,785	-3,455	-4,105	-4,435	-4,965	-5,255	-5,575	-5,835	-6,305
Total	-670	-1,424	-2,174	-2,629	-3,179	-3,679	-3,619	-4,149	-4,199	-4,504	-4,494	-4,514



National Holiday

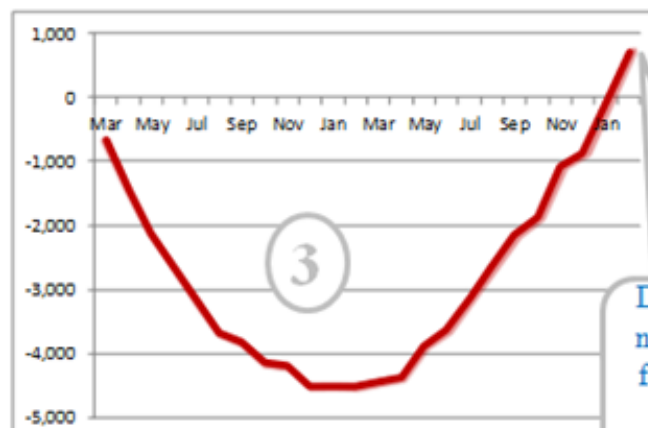
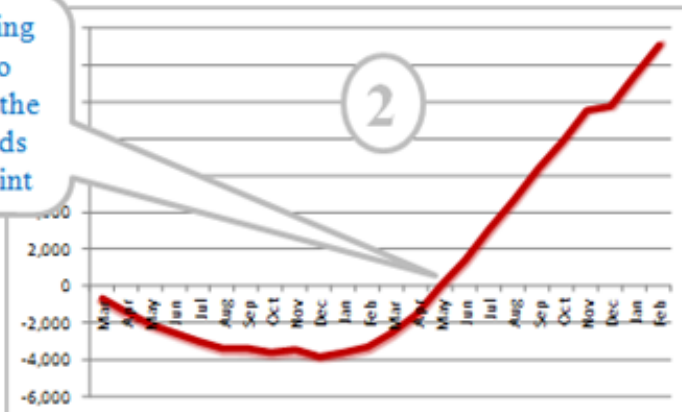
18 months break-even point

Significant investment upfront

Cost of interruptions is added back

Mandatory common process, training and tool

Increasing bugfixing cost from \$4K to \$5K – diminishes the benefit and extends the break-even point



Dropping the total number of defects from 200 to 100 – diminishes the benefit

Following Red Curve (P&L)

The actual [spreadsheet](#) depicted here is attached to Class Discussions as a separate file. You are encouraged to play what-if scenarios by inserting various parameters into the spreadsheet and observing the red curve is being re-shaped.

- Suppose your offshoring team is going through an extended national holiday. At a first sight, it is difficult to connect such an event with anything similar across the globe. Bostonians are unfamiliar with a month-long, all-out celebrations that are common at other places, e.g. at Rio de Janeiro carnival. Bostonians are unfamiliar with protracted hunger strikes of politicians at Indian provinces. In order to limit the scope of our investigation, the only question we need to ask, is how such an outside event is relevant to the "red curve". Apparently, during a carnival, folks are unable to fix defects, so the corresponding zeroes have been inserted into the row titled "Fixed defects in Singapore".
- Suppose the offshoring team has asked for a raise, so the new cost to fix a defect will be \$5K. It is enough to insert the "5" instead of "4" into the cell titled "Cost to fix a defect in Singapore" to observe the movement of red curve.
- Suppose you received numerous requests from engineers to deploy a new source control tool. You have to insert a new line among the "Expenses" with the cost of the tool, along with its training. The result will be that the red curve will shift to the right.

In summary, first, we created the red curve metaphor, along with its underlying spreadsheet. We limited the scope of our analysis and made it more manageable. We moved from reality of hunger strikes and extended carnivals into a new reality of EXCEL cells.

Second, we started manipulating parameters of the metaphor of our offshoring venture. By inserting alternative numbers into the spreadsheet we are able to connect events that looked unrelated at first sight. We are able to measure how does the break-even point is affected by the hunger strike in India and a salary hike in Singapore. We are able to arrive at certain decisions based on changes at our Red-Curve-Metaphor.

Here is another important thought about off shoring. The nature of software development suggests not to chase after cheap labor. What we are looking for are talented individuals who are capable of making a difference. It is irrelevant where these talents came from. Modern tools bridge the distance, making all of us, in effect, sitting in the same room. Since we could find high achievers in any place in any country, it is prudent for a global enterprise to spread offices around the world.

Requirements

Requirements Engineering

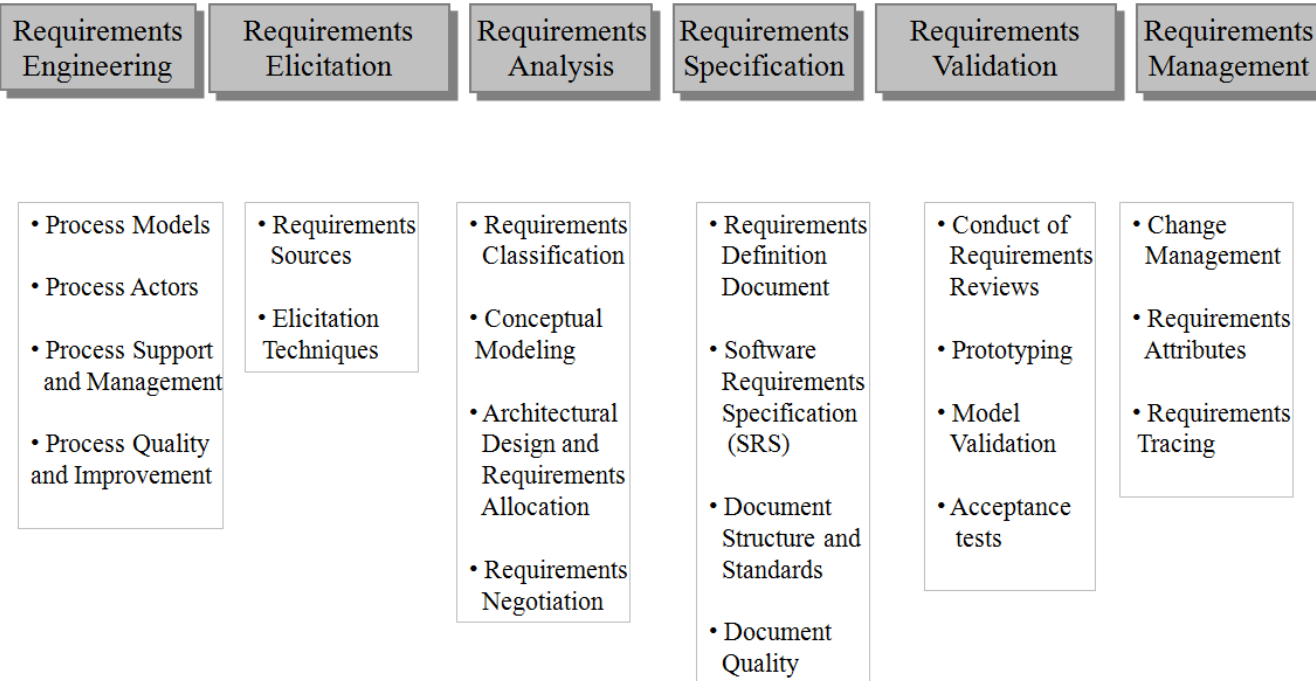
The module covers responsibilities of sw practitioners involved in managing requirements. After completion of this section you should be able to do the following,

- Position Requirements Management within a broader context
- Classify given defect in a requirement statement
- Document Roles (users, personas)
- Use Canonical Form to develop requirements
- Map process workflow into documented requirements

Taxonomy

Requirements Engineering

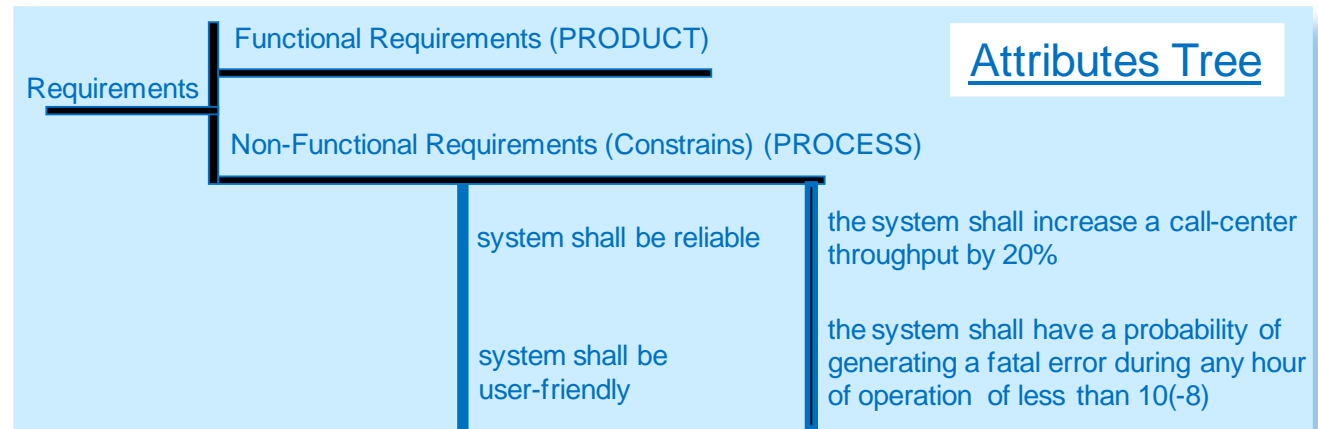
- Elicitation
- Analysis
- Specification
- Validation
- Management



Each sub-process is not static. It continues throughout the whole product development. Context-free Interviewing is a perfect examples of Requirements Elicitation.

Attributes of a Requirement

Following pictorial shows the decomposition of attributes of a requirement. It is useful to create such a tree when analyzing the completeness and quality of all requirements for a specific product.



Attributes of a solid requirement are comprised below.

- Unambiguous
- Non-overlapping
- Complete
- Testable
- Clearly stated
- Consistent
- Correct
- Modifiable
- Prioritized
- Traceable & Traced
- Uniquely identified
- Design-Free

These attributes serve as a buckets, when classifying defects in requirements. My experience shows that first bucket "unambiguous" is always full. As this is the most common defect type.

Forbidden Terms

Karl Wiegers [1] has a long list of terms that should be avoided within the language of a solid requirements. Simply scanning through the list provides a clear direction away from ambiguity toward a more detailed refinement.

It sounds paradoxical but the reason to remember these terms so you will not be using them.

- Acceptable
- Adequate
- Efficient
- Fast
- Flexible
- State-of-the-art
- User-friendly
- Several
- Graceful

Definition of Personas

We should have a clear understanding who is the user of the system we are building. It is prudent to start the requirements process (as well as, to start the whole project) with documenting the so-called personas (users, actors, roles). Assuming each persona has a different set of needs. Each need creates a different angle of the same product. We are aiming to assemble a comprehensive list of product requirements. R.A.S.C.I. is a common method to define external system roles. There are obvious advantages in providing a consistent definition of personas across several product lines.

Example RASCI for a "defect database"		ROLES				
Task	Submitter	Tester	Developer	Guest	Manager	
Enter Defects	R	R	R	R	R	
Assign User Priority	R	I	I	R	I	
Assign Dev Severity	I	C	R	I	A	
Resolve Duplicates	I	R	C	I	I	
Reproduce Defects	C	R	C	I	I	
Provide Fixes	I	I	R	I	I	
Retest Fixes	I	R	S	I	I	
Close records	I	R	C	I	I	
Distribute Reports	I	I	I	I	R	
Analyze Common Trends	I	S	S	I	R	
R: Responsible	Owns task execution and outcome					
A: Accountable	To whom "R" is accountable: Respons					
S: Supporting	Support the task completion					
C: Consulted	Has expertize or capability to complete the task					
I: Informed	Must be notified of the results, but need not be consulted					

Note that definition of personas is a controlled document. Changes to such document shall result in a profound product ramifications. Decisions we make in the beginning of a project - stay with us throughout the whole endeavor. Definition of personas is one of those decisions. For example, at the end of a project, the number of test cases, as well as the time you need to execute them, is bound to quadruple with each additional persona.

Cost of Defining Requirements

- We should be careful with time spent on this initial task
- Wiegers' Cosmic truth # 10, "you will never get a perfect requirements"
- Need to draw a line where to stop doing requirements and move on
- Some people say that, "a story is just a topic for a discussion"

done

backlog

White Space

Anti-Persona
a someone who
will NEVER use
the product

- agree
- understand
- estimate

WISH LIST / ICE BOX

- Too small to talk about and to spend time on definition
- Too big and needs to be split
- Will be done anyway so why bother to define it
- No need to review, since was just done in previous release
- A fantasy, a pie in a sky

I always pay attention to common mistakes that students make. Particularly at the first attempt of writing requirements and defining personas. Definition of personas is the first logical part of writing requirements. Question I have, why would you put yourself as part of the well-defined and controlled personas? It really does not make sense. These are two different groups, one group are those who use and pay for the product and another group are those who build the product. Hence, when you are asked to 'define personas' you need to find and elaborate on members of the first group only – those who will use the product

Warm and Fuzzy

David Platt who was named in 2002 by Microsoft a Software Legend, offers an elaborate template for defining personas. www.joyofux.com . For many years David is teaching a class at Harvard Extension. He has collected a wealth of experiences with the subject. He suggests spending at least a day on a definition of one persona. Such an estimate is apparently exceeding expectations of this class. David's passion and entertaining details are always appreciated by his students. His template has the following parts,

- Picture, photograph
- Personal information
- Character Cue
- What people say about this persona
- Personal Essay written (purportedly) by the user herself

Lots of details offer an insight into the human psyche of the user and create an emotional connection with a software developer.

RASCI is the tool suggested in this class to define personas. It focuses on traceability between product artifacts. Tasks stated in RASCI are much higher level than activities in stories or in a project plan. The only purpose is to outline a persona and then move to the next step of a project. Tasks in RASCI do not intend to create an exhaustive list of project tasks.

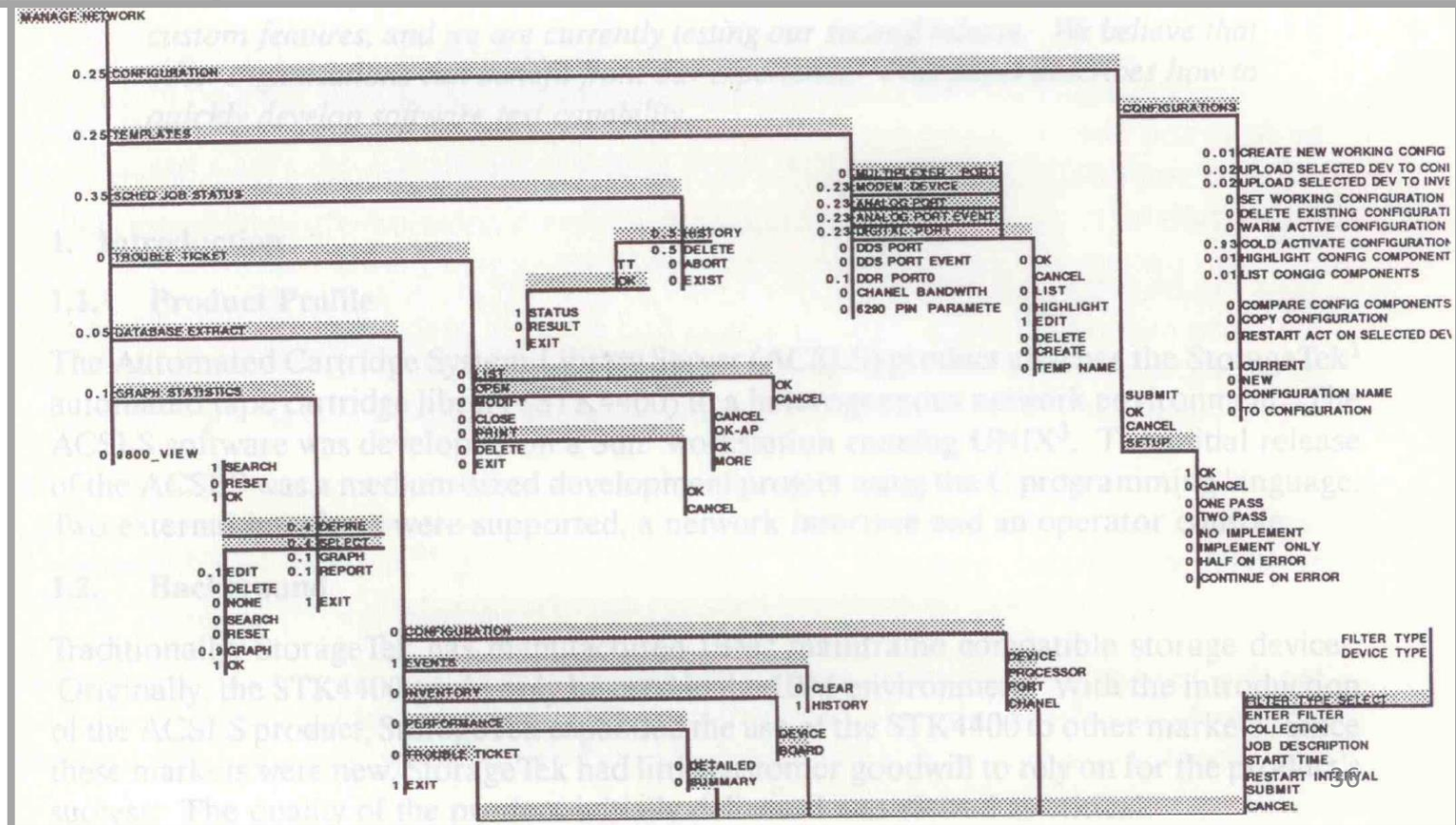
Hope the difference between David's template and RASCI is clear. In our projects, we are not fixated with emotional connection, warm and fuzzy. As we are simply making sure that project artifacts are traceable to each other.

Operational Profiles

NM COMMAND	
0.45 VIEWS	
0.45 MANAGE RESOURCES	
0.03 MANAGE NETWORK	
0.05 MANAGE EVENT	
0.02 UTILITIES	

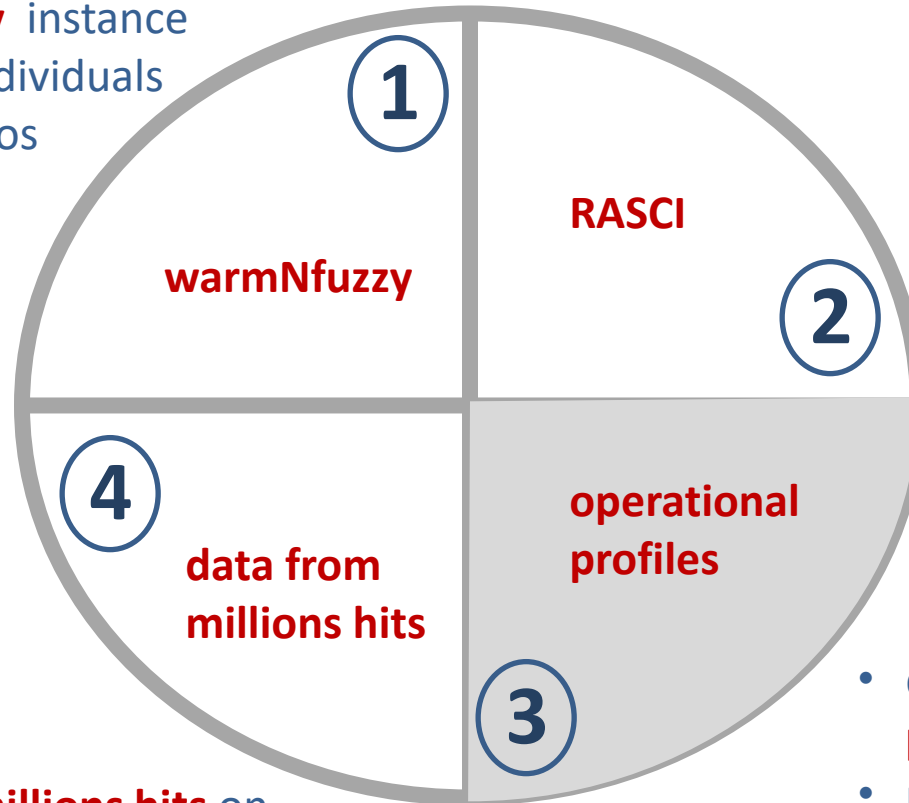
Maintaining operational profiles is an effective method of dealing with customer requirements. Pictorials below show operational profiles that I used to collect for Motorola Network Management System. There are two basic methods to collect operational profiles. First, interview NMS operators, and second, utilize a key capturing software.

Operational profiles are used to drive system test (exercising heavily those functions that are used more frequently) and determine priority of issues that are more close to the heart of customers.



Four Ways To Define Personas

- create **warmNfuzzy** instance
- describe specific individuals with their live photos
- include children
- Preferences
- frustrations
- life styles
- whatever other people say about them



- select a minimum number of reps
- segment a complete population of users
- define archetypes
- document **RASCI** with a high-level responsibilities
- each archetype has many warmNfuzzy instances

- document **operational profiles**
- maintain a tree of frequencies of usage of certain product features
- apply these frequencies for testing, ranking defects, etc
- map these trees into user roles

- collect **data from millions hits** on various sites and pages – based on cookies
- compile habits and preferences, then match them with projected demographics and geographical locations

Canonical Form

Canonical form is an established language of defining requirements. It's simple template leading one to clearly state three parts of a requirement (a) role, (b) action, and (c) value. All stories in Pivotal Tracker are written in canonical form.

Attributes of a well-written story

I ndependent	to an extent possible, does not depend on other stories
N egotiable	a story is a conversation starter, not the end result nothing about "how"
V aluable to the user	something a user can use not a "piece of something" a user can use
E stimatable	no research required, well-understood
S mall	
T estable	



An acceptance criteria

S pecific
M easurable
A ttainable
R elevant
T ime-sensitive

Cards from "Braintrust", see above, are commonly used to document product requirements (stories). Personas could be both "humans" and "non-humans". In many cases, a key requirement arrives from interaction with another "software system" in addition to interaction with a certain "individual".

What's Wrong With These Requirements?

Finding and reviewing defects in someone else's requirements is a great method to avoid such defects in the future. Pictorial below has a series of common defects.

	As a	I want to	so that to
	Role	Action	Value
1	Submitter	enter defects	receive fixes within a reasonable time
1.1	External Customer	assure field defects are entered	customer issues are addressed expediently
1.2	Field Support staff	enter field defects	resolve issues at customer site
2	Developer	receive defects relevant to my component	fix defects in a timely manner
2.1	Developer	mark defects as Fixed and Closed	follow the bug life cycle
3	Tester	receive the list of fixed defects	retest fixes; announce release availability
4	Tester	mark defects as Closed or Re-open	assure Development is aware of Reopen defects
5	Manager	run a report with defect trends	identify common issues act upon trends
6	Executive Sponsor	create two radio buttons that glow in a dark	mark defects that I originate as Super Extra Hot
7	Development Lead	assign defects to a specific developers	utilize expetize of developers effectively
8	Defect System Admin	find defects in prohibited states	maintain integrity of defect tracking system
8.1	DBA	archive old defects	focus on priority issues only
8.2	Systems Manager	monitor current limits of defect tracking	assure the software is able to support process
9	Proj Mngmnt System	synch naming conventions	maintain integrity of project reporting
10	Test Case System	synch relevant test cases	execute regression suites quickly
11	Guest (Observer)	learn about defect tracking system	actively participate later
12	Auditor	assess compliance against industry standard	communicate to customers and sponsors
13	Defect Tracking System	* data entry response within 0.5 seconds * runing report within 3 seconds	saving valuable user time
14	Sales Rep	run random reports	include bug reports into sales pitch

Ambiguous.
Need to be expounded,
attachments / screenshots

Ambiguous.
Need specific SLAs

Outside of scope
Customers are
read-only

No use case
Developers
do not close

Not design-free.
Non negotiable.

Redundant.
Merge into one

Ambiguous
Define "old"

Not testable.
Define standard

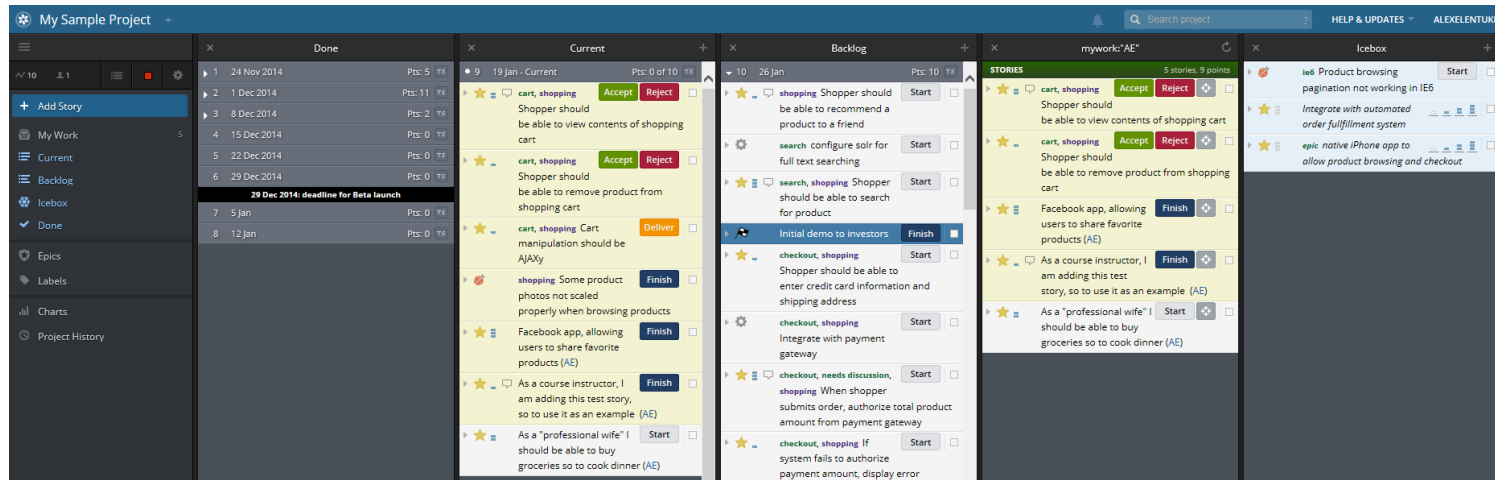
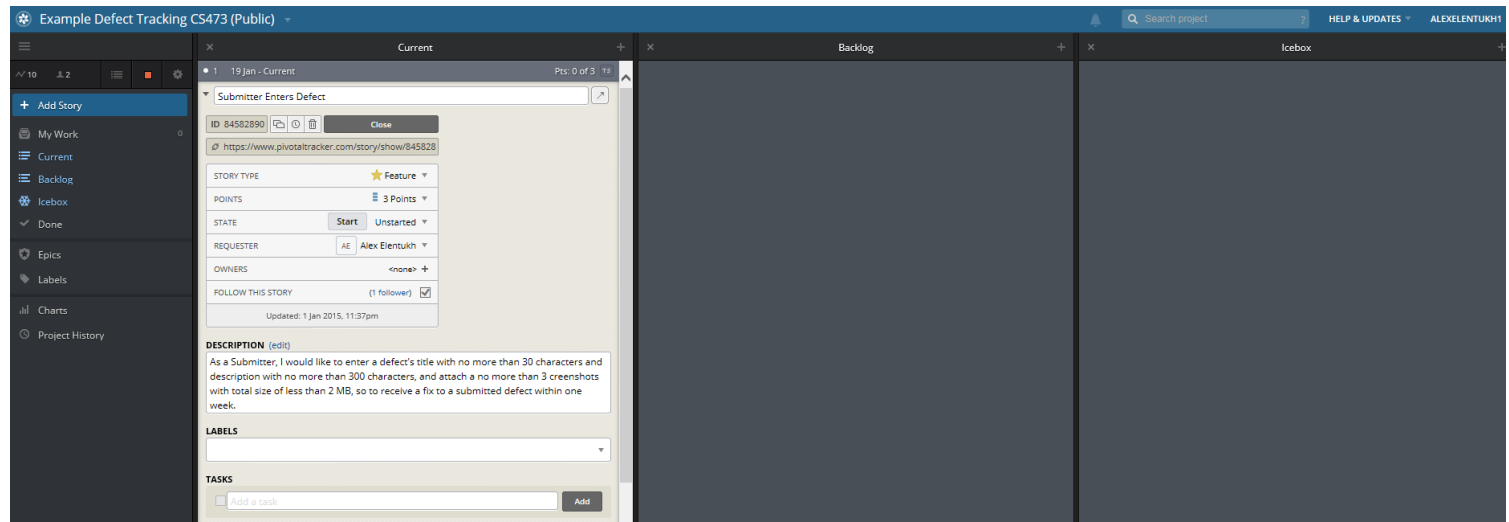
Good One. Non-functional
are usually forgotten.

Has little value.
Undefined persona

Pivotal Tracker

<https://www.pivotaltracker.com/help/gettingstarted>

Individual requirements propagate from right to left, from "Icebox" into "Backlog", then into "Current" and eventually into "Done." Note that transition into "Done" is automatic. In a traditional environment, one might be accustomed to a separate Project Plan, Gantt Chart, Tasks beneath Activities, etc. Pivotal Tracker combines these pieces into a fairly simple dashboard.



Two Types of Requirements

In a context of this class, both terms, 'requirements' and 'stories' are used interchangeably. Obviously, 'requirements' is a more general term applied to whatever life cycle a project follows. Which is opposite to 'stories' which is an Agile term.

It worth distinguishing two types of stories. First type has external personas. These are users of the final product, customers, who will purchase it. Second type, has internal personas, these are developers who design, build and test it.

Suppose a product development life cycle consists of a dozen iterations. It is most likely that at the initial iteration, only stories of the second type are used. For example, the following story belongs to the first iteration. *As a software engineer, I need to set up a branching and merging for our source control, so to be able to collaborate with my team.* This story is expected to be accepted and moved to DONE area by the end of the first iteration. Here is the example of a story from the last, a so-called, hardening iteration. *As a user, I need to verify all product features are part of the release and the final acceptance test has passed.*

The first type of requirements belongs to posterity. These are the features that will be retained indefinitely and will be exercised by millions of users. The second type of requirements constitute a temporary path, that is only a means to achieve the posterity.

- Question. Should we focus on the first type or the second type? You do need both. There is no other way but to go through the temporary in order to achieve the permanent.
- Question: If I accept a story, which I have not really done, is it a problem ? Yes, it can grow into a problem with the following sequence of events. Pivotal will automatically move this story into DONE column. You will be unable to alter its status. During final presentation, you will be unable to demo its functionality. Will need to do some explaining, as to why traceability is lost between requirements and actual product.
- With reference to SAFe (Scaled Agile Framework), there is a single development pipeline that is fed by all sources. The 'architectural runway' deals with a flow of requirements that are invisible to end users.

Lessons Learned from Using Pivotal Tracker

The first lesson you learn from using this tool is nothing to do with the tool itself, it is all about discipline. The tool forces you into a certain workflow. I saw some folks call it a 'Nazi' tool, as it does not allow moving accepted stories into DONE. The tool does it for you. The underlying reminder is that 'time flows one way only', it is irreversible. If you miss an opportunity to accept a story within an iteration, you will not be able to recover, never. By the end of a project, during final presentation, the tool will display all iterations and some iterations will be empty.

Another lesson is derived from the selection of stories. Question is which stories to document? You do not want documenting miniscule stories that will be done anyway. You want focusing on stories that involve convergence and monitoring.

Pivotal Tracker is in a direct competition with other means of collaboration. Make sure Pivotal wins the competition and you do not use Gantt charts and EXCEL to track actions. Make sure you do not maintain an elaborate project objectives and extensive minutes from team meetings. Pivotal should suffice. As you realize, the last thing on my mind is to sell you some specific tool. What I am struggling with is redundancy. If you use Pivotal, then Gantt chart should fade away into being redundant.

If you are a Project Manager of Business Analyst, you are expected to be well-versed in many other abating techniques.

- Story Mapping. This is done during iteration planning and project initiation. Key stories are positioned into iterations. New work is being discovered and accounted for.
- Epics. The hierarchy of stories is important to characterize pieces of work.
- Story Points and Velocity Tracking are used in any mature organization. These concepts should never be a project's focus. Their goal is to improve consistency and effectiveness of product delivery.
- Pivotal allows to automatically create a traceability matrix. Which could be used as a starting point for a test planning, while understand that the long road to an adequate coverage is still ahead.

Measuring the size of a project

My document has more pages than your document.

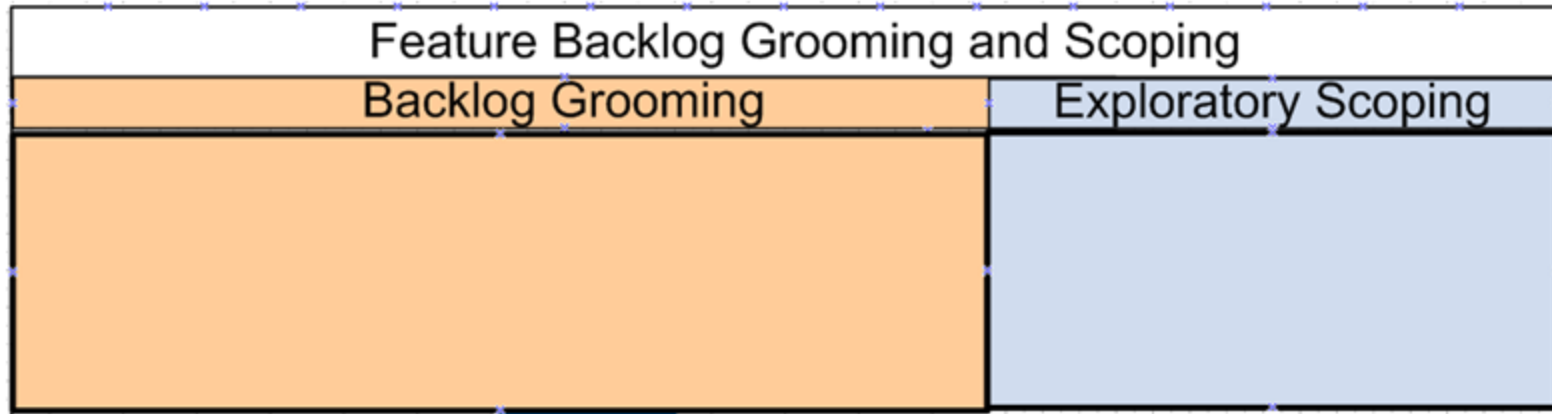
My document has more words than your document.

I have more shall(s) and will(s).

I have more requirements that are standardized and are used as one of the measures.

Feature Grooming and Scoping

Examine the key software process called Feature Backlog Grooming and Scoping, based on an example provided below. The swim flow diagram depicts both Grooming and Scoping in different colors. Grooming deals with feature definition and scoping deals with initial estimation.



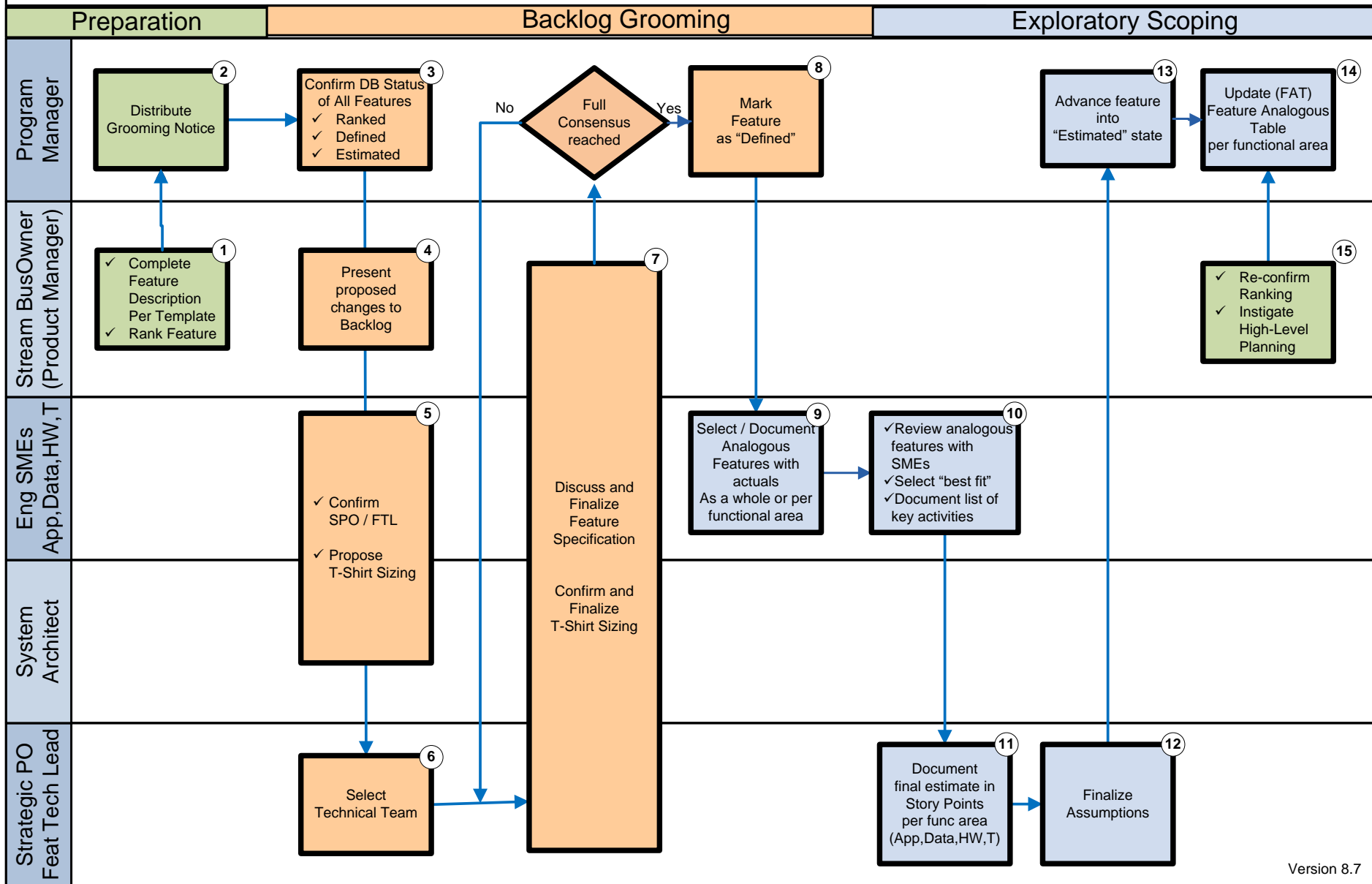
Additional context

These swim lanes are taken from a real organization, to clarify the context.

- organization consists of approximately 2,000 people
- involved 30 architects; 30 product managers; 21 scrum teams
- after re-organization from waterfall into agile, 600 test engineers were divided; 200 testers have joined scrum teams and 400 testers remained supporting the system test
- revenue generated by this product line exceeds \$1B per quarter
- there is uninterrupted flow of features proceeding through this process; at any point of time there are 500 active features in the database

Apparently, the scope of these swim lanes does cover requirements and does not include development, testing, release, etc.

Feature Backlog Grooming and Scoping



Description of key activities

You have an exercise covering these swim lanes. Benchmarking is important in developing an "organizational compassion", a skill to relate to other teams/environments.

1	Complete Feature description per template	This implies that there does exist a controlled template. A template might be very simple, <ul style="list-style-type: none">- Functional- Non-functional- Acceptance Criteria with some basic regression
	Rank Features	<ul style="list-style-type: none">- Relative priority based on a business customer need- Independent from a particular release or feature size- Implies a <u>single</u> list of features marked with their importance from a user perspective
2	Distribute Grooming Notice	This regular distribution has a wide audience. Process changes and analysis of metrics are announced as part of the same distribution.
3	Confirm Database Status	All features are advanced through a consistent set of states
4	Select SPO / FTL / Technical Team	<ul style="list-style-type: none">- In some cases, folks who do initial scoping (architects) are not the folks who implement the feature (dozens of software engineers)- There is a 3 hours per week limit (time box) for an engineer, to be spent on this process
7	PM facilitates a regular meeting	This is the heartbeat of the whole process. At this meeting all issues are resolved
10	Following estimation techniques are used	<ul style="list-style-type: none">- T-Shirt Sizing- Analogy- Decomposition- Expert Judgment- Fibonacci Numbers
12	Finalize assumptions	Assumptions are reconciled among SMEs before converging on a coherent estimate

To support the notion of universality of requirements, let me bring about an example from a different area. When you order food in a restaurant, in effect, you provide your "requirements" to a chef. A waiter has a notebook that is a bona-fide centralized repository of requirements. And then there is a size estimation, when you match your appetite against the order (five course meal or a cup of coffee).

Human Factors

What Product Owner feels and should overcome

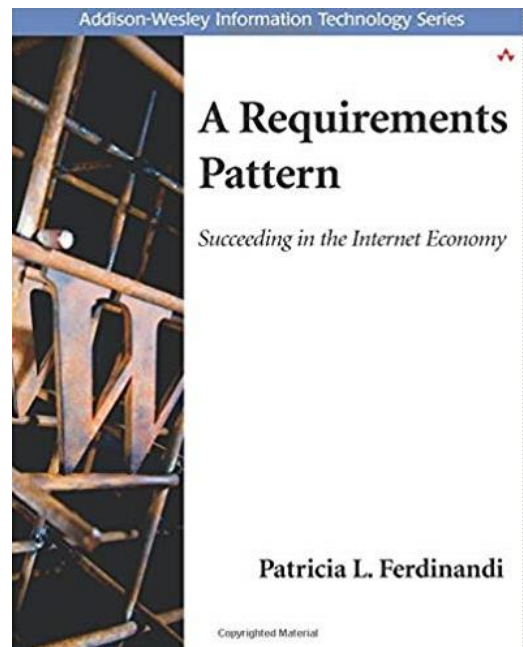
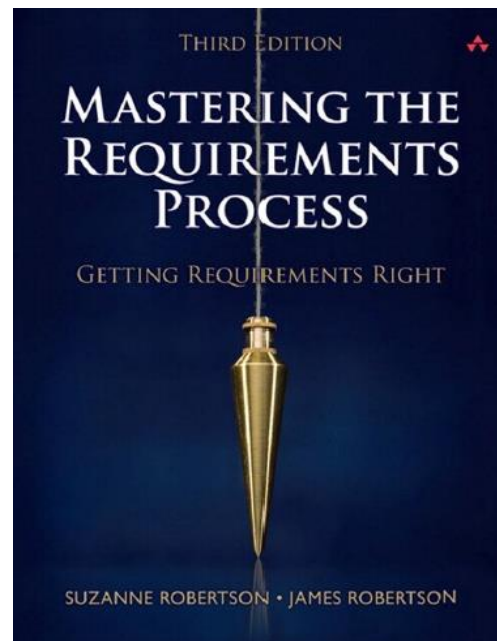
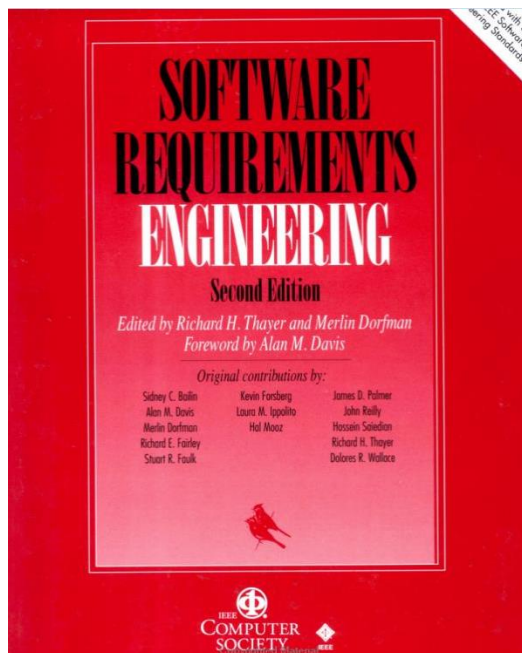
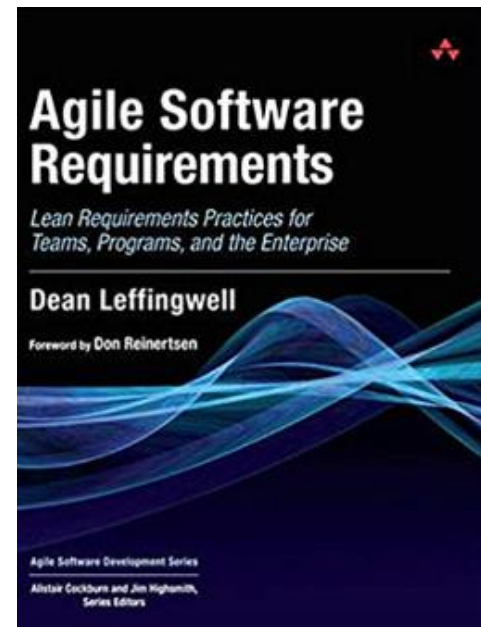
- A feature must be linked to a specific release, a certain team, timeframe , implementation
- A feature is simple and could be implemented quickly.

What product Owner should focus on

- Product Owner should describe a feature as standing on its own feet and as independent as possible from other features, considering the difficulty of breaking such dependencies.
- A feature should be described without any reference to a timeframe or a projected release

Requirements Management process is like any other process needs a very definite attention and a discipline. It is not enough to distribute a requirements template with predefined headings. It is necessary to have a real conversation between all parties involved. To simplify the exchange, Technical Lead has to ask questions and Product Manager should respond to these questions. As a result of this conversation Product Manager actually edits the feature description and assigns a new version.

Everyone should expect to learn something new from Requirements Management process. Tech Lead will learn about the feature so to pave the way to next implementation steps. PM is bound to learn from responding to various questions and from reducing to paper these responses.



Engineering Management

Engineering Management

The purpose of this section is to delve into selected aspects of engineering management. Software engineers often ask - what do managers do? In a context of our course, it is important to shed some light on presumed responsibilities of a software leader.

- perform a function of a manager / generalist
- align company culture with project goals
- establish consistent development process
- leverage organizational cause and effect relationships to optimize product delivery
- motivate software engineers
- avoid excessive multitasking
- maintain software development policies

Taxonomy

Universality of software management

Process consistency

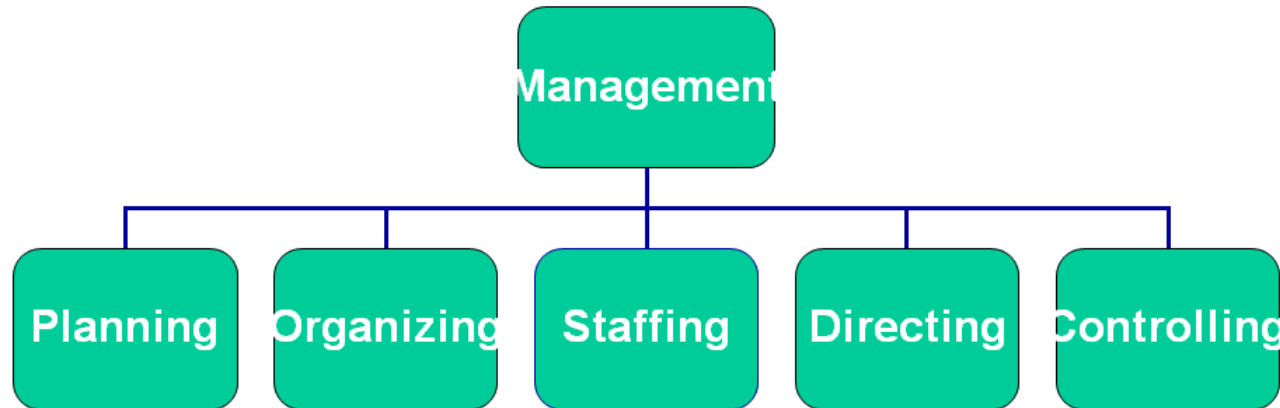
Organizational culture

Cost of a defect

Maslow Motivation Model (MMM)

Universality and Specifics of Software Management

A software manager is bound to perform the same tasks as any other non-software manager. The universality of management with its five aspects - is reflected on the diagram below.



A software manager is supposed to master some definite expertise in these five aspects.

- Planning - Predetermining a course of actions for accomplishing organizational objectives
- Organizing - Arranging the relationships among work units for accomplishment of objectives, and granting of responsibility and authority to obtain those objectives.
- Staffing - Selecting and training people for positions in the organization
- Directing - Creating an atmosphere that will assist and motivate people to achieve desirable result
- Establishing, measuring, and evaluating performance of activities toward planned objectives

Specifics of Software Management

- 1 Perception of clients of a lack of appreciation for inherent complexity and profound ramifications of requirements changes
- 2 Software process itself generates a need for changes in client requirements
- 3 Software is often built as an iterative process
- 4 Software engineering maintains a difficult balance between creativity and discipline
- 5 Software engineering is largely lacking an underlying theory
- 6 Software is intangible and is difficult to test
- 7 Degree of software novelty and complexity is extremely high
- 8 Software Engineering is faced with extremely high degree of change

Organizational Culture

Any introductory module needs to cover a concept of a company culture.

What is a company culture? It is a set of assumptions and unwritten rules that guide individuals in their day-to-day activities. Those, who have been with the company for a while, have already become a part of the culture. However, those who just arrived need to answer many questions.

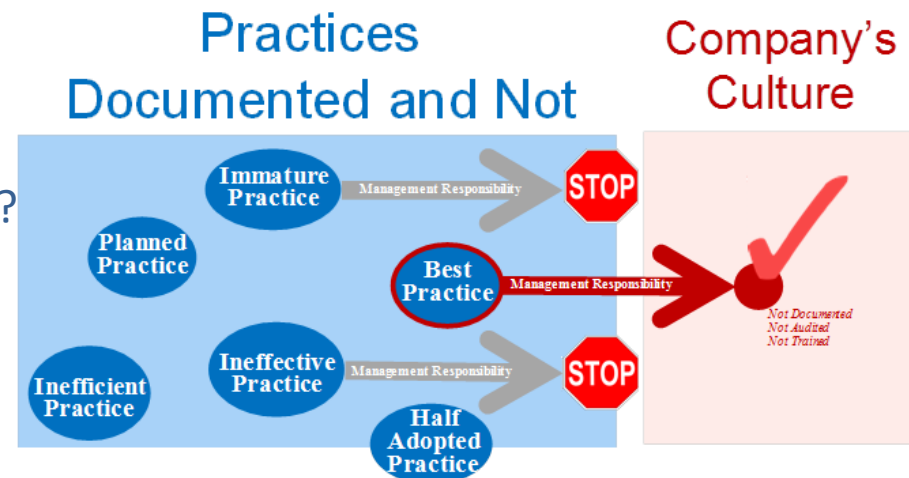
- Do people sit all together in the cafeteria or separately?
- When should men wear ties and women suits?
- What are the consequences of not delivering product on-time?
- Will my boss be mad at me if customers find a bug in my code?

Aligning company culture with the company mission is an important management task.

Whatsl defines [six models](#) for a company's culture.

The blue-suit strictly hierarchical culture of IBM is very different from the free-wheeling, grab market first culture of Facebook. Both of these very successful companies enjoy and prosper because of their strong culture.

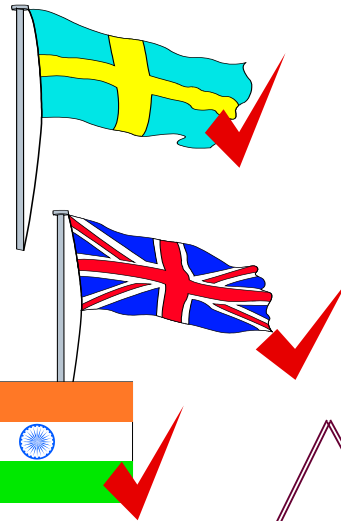
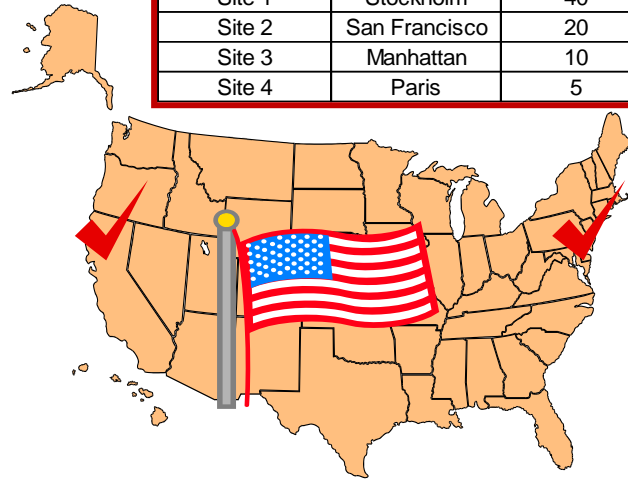
A question in front of a manager is ... How to transition a documented Best Practice into a company culture, push it into fabric, make it stick? How to align the Culture with project goals? It is a responsibility of a manager to become Aware of various company's cultures, compare and perform the "mental benchmarking", so to be able to act accordingly.



Consistent Process (horizontally and vertically)

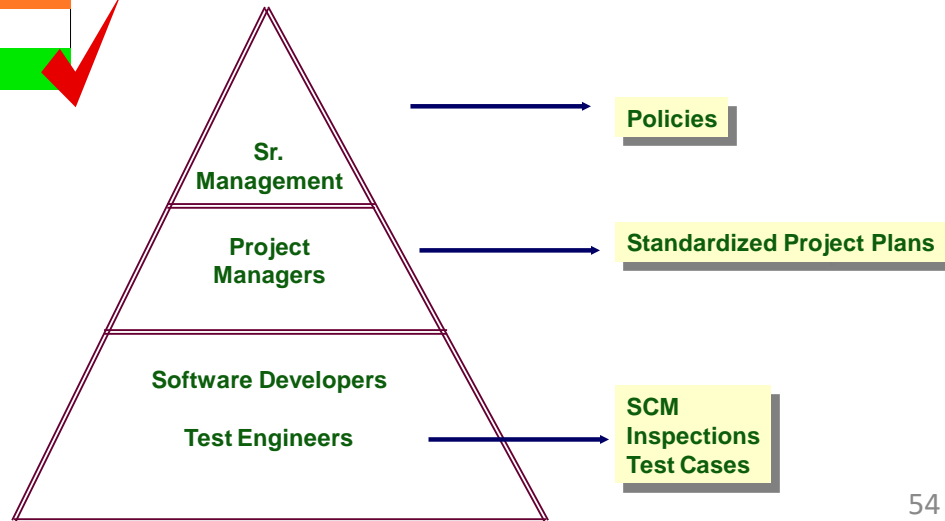
Consistency is the fundamental attribute of a mature software process. Pictorial below focuses on a distributed aspect of software development. Note that two processes do not need to be identical in order to be consistent. Apparently, it is impossible to develop and deliver a common product across multiple continents, if the overall process is inconsistent.

Site	Location	# Employees	Product Specialty
Headquarters	Brookline	85
Site 1	Stockholm	40
Site 2	San Francisco	20
Site 3	Manhattan	10
Site 4	Paris	5



In a context of a geographically distributed team, an example of a severe inconsistency would be ... product manager documents ambiguous requirements in Brookline, resulting in testers in Paris unable to verify certain aspects of product features.

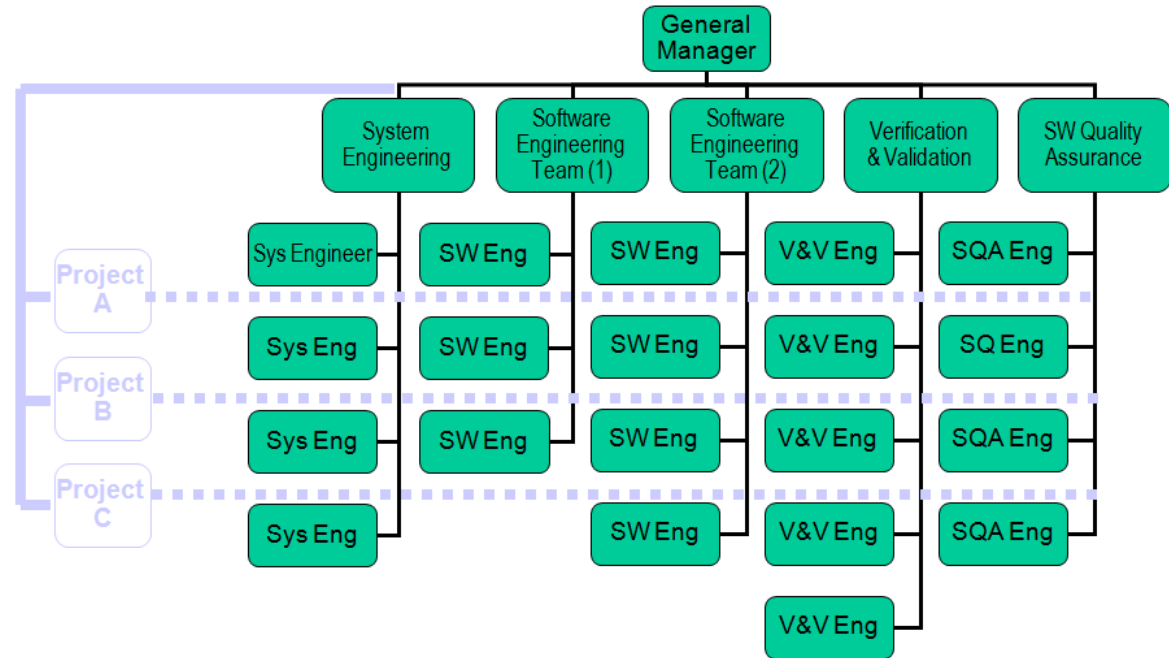
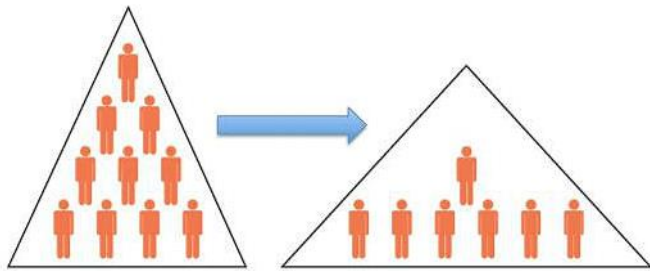
Consistency across organizational levels is equally important. It would be unproductive to have a gap between a top level direction provided by Senior Management and a bottom level product implemented by engineers.



A Myth of Organizational Hierarchy

When you are sitting in front a blank sheet of paper trying to capture a key thought that is both elusive and fundamental, the idea of you reporting to someone else - seems so irrelevant. What difference does it make how many organizational levels are above you or below you? How could it possibly help with solving the task at hand? In fact, it does not.

Having said this, it is still important to consider various org charts. Since when people talk about an "organization" they usually mean the "org chart". Here is the chart of a so-called matrix organization.



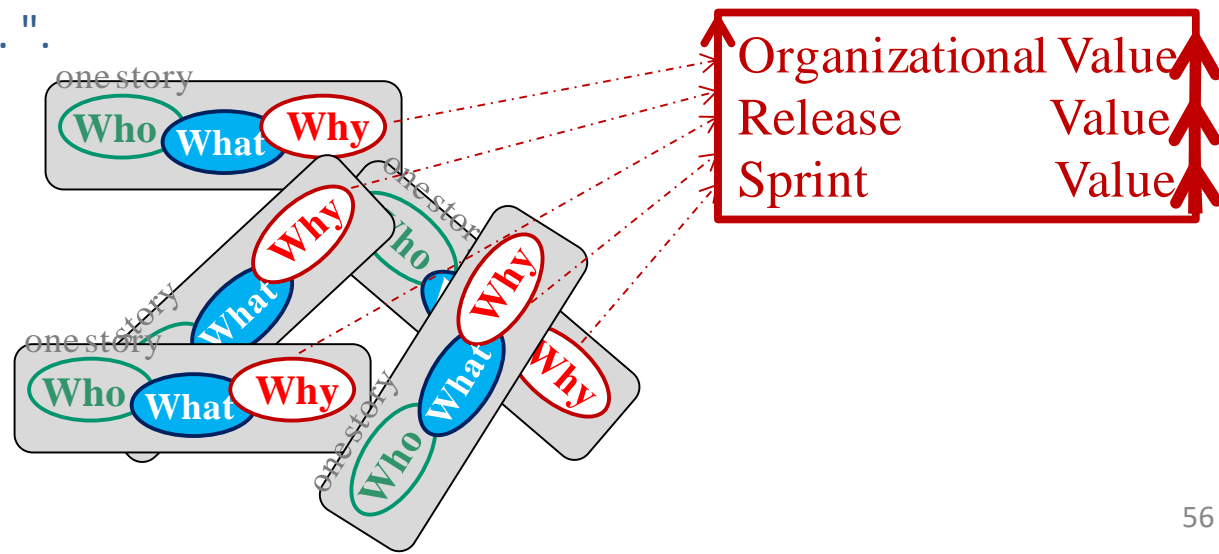
Here is pictorial from Craig Larman <http://less.works/> elaborating on apparent advantages of a flat organization. It is a natural tendency to flatten the hierarchy, to open up barriers and shorten communication paths. Although, realities of a software company teach us differently. Managers do own a certain function that must be accomplished one way or another. Reading about companies that grew quickly, e.g. Akamai, Google, we learn that, at some initial point, a senior manager decided to abandon a hierarchy and make a large group of people reporting directly to him. Such decision could go only so far. Eventually a certain structure had to be put in place.

Staying Close To Reality and Value

It is not coincidental that the name of the biggest yearly conference in Bay area with participation of CEOs of largest companies - is "The Code"

While navigating around a big organization, with everything that is going on, let me confirm that "The Code" is the only reality you and I could possibly share. Understanding what is real what is not real - is a key skill of a professional software manager.

- With all my appreciation for peer reviews, I should confirm that the idea of sitting around a table and examining some mental models looks like a fantasy in comparison with real test defects revealed while exercising the actual system.
- Agile Principle # 7 states that "... working software is the primary measure of a project progress ... ". It is not the number functional specs written, not the number of management reviews, but the software itself is the measure.

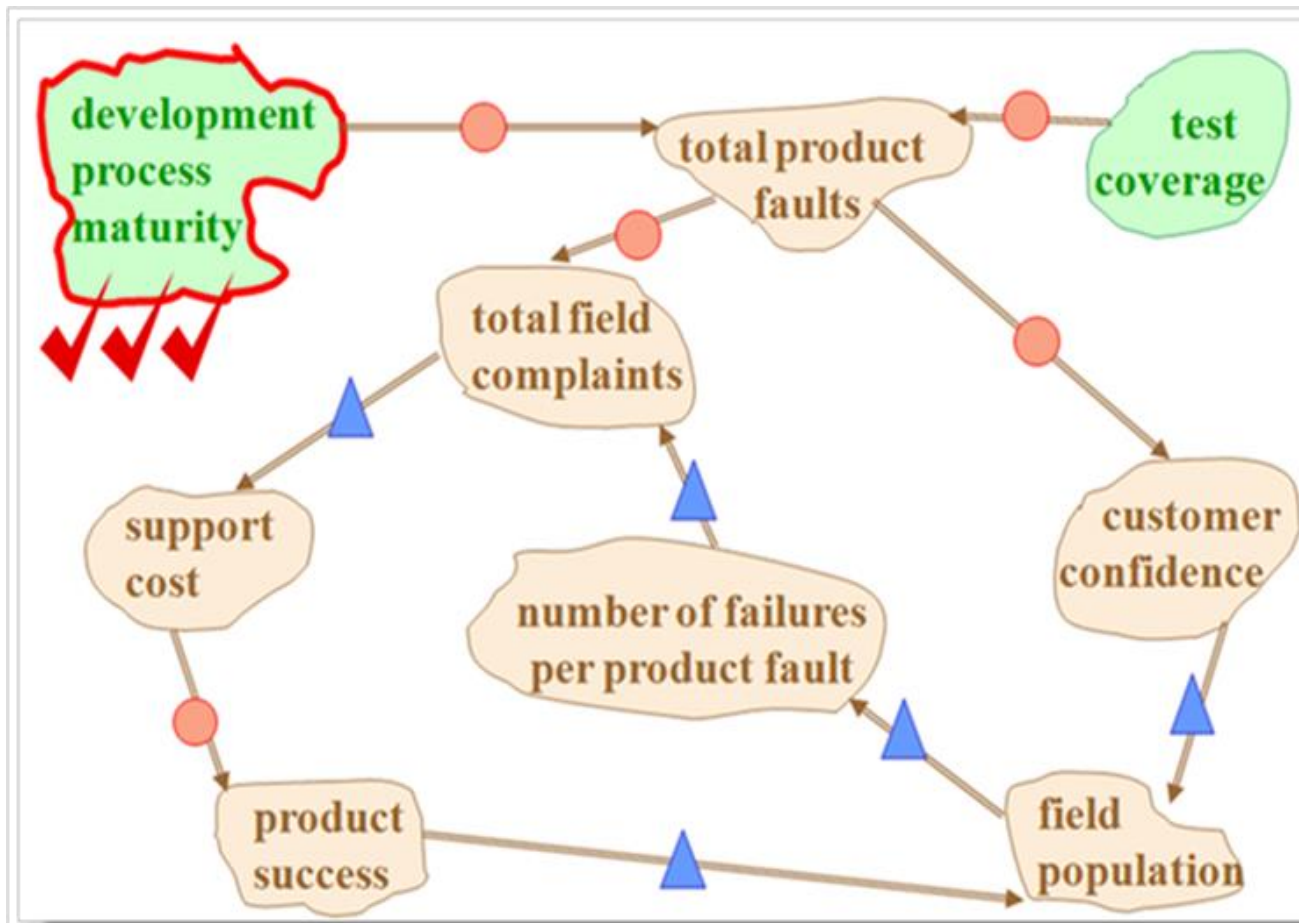
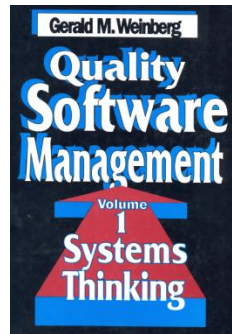


Organizational Cause & Effect Relationships

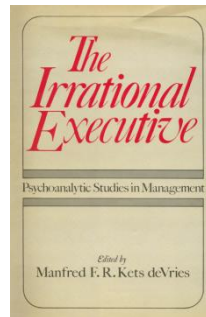
Examine notation introduced by Gerry Weinberg in his book "Systems Thinking".

- A blue triangle designates a positive correlation.
- Red circle denotes a negative correlation.

We should traverse the diagram to make sure the relationships between all areas are clear. I observed in many small and medium-size companies, that both green areas are managed by the same person/role, for example, by a Quality Assurance Director.



All of us should find a way to acquire the prudent, forward-looking thinking, as it is not a readily available trait of software managers.



The book by Manfred Kets de Vries "The Irrational Executive", goes into a great detail of exploring some unusual behaviors within federal government. The study shows how the work environment interacts with forces of human character in a ways that stimulate decisions that are difficult to justify logically. Many pages of this book are dedicated to examples of behaviors when folks are striving for personal advancement at the expense of agencies they are hired to lead. Decisions they make defy logic and in a way represent a great lesson for us to draw upon.

Should note the difference between correlation and causation. Although, for the high-level view presented at the pictorial, both notions are aligned.

Motivational Models

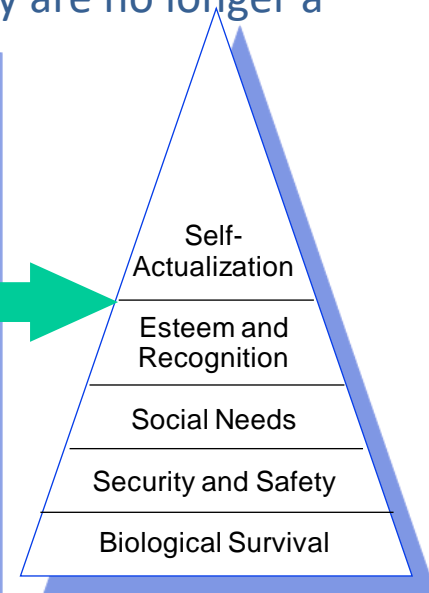
A software manager motivates team members. It is important to distinguish a good motivational strategies that support the long-term organizational objectives. On a contrary, applying a poor strategy could de-motivate engineers or create a conflict that is bound to destroy a team.

Maslow hierarchy is worth pondering.

While managing an off shoring team in India, one day I heard that folks are unable to come to the office. This is because software engineers were afraid for their lives, due to an unrest in their remote province. Looking at the Maslow hierarchy, I see "safety" belongs to the second level. Although, assuring "safety" for engineers in Boston would hardly be an appropriate motivator. Since those needs that have been satisfied already are no longer a motivator.

Facilitating a learning environment, where folks are able to unleash their true potentials - is the motivational strategy belonging to the top level of Maslow hierarchy.

Motivation Model	Elaboration
Frederick Taylor	Workers will respond to an incentive wage
Elton Mayo	Interpersonal (group) values override individual values
Kurt Lewin	Group forces can overcome the interests of an individual
Douglas McGregor	Managers must understand human nature in order to motivate them
Maslow	Human needs can be characterized in a hierarchy. Satisfied needs are not motivators.
Fredrick Herzberg	A decrease in environmental factors is dissatisfying; an increase in environmental factors is NOT satisfying. A decrease in job content factors is NOT dissatisfying; an increase in job content factors is satisfying
Chris Argyris	The greater the disparity between company needs and individual needs, the greater the employee's dissatisfaction
Rensis Likert	Participant management is essential to personal motivation
Arch Patton	Executives are motivated by the challenge of work, status, the urge to achieve leadership, the lash of competition, fear, and money
Theory Z	A combination of American and Japanese management style. People need goals and objectives; otherwise they can easily impede their own progress and the progress of their company.
Total Quality Management (TQM)	A strategy for continually improving performance at each level and area of responsibility



Many motivational models suggest "taking money off the table" so to speak. Here is the [link](#) to an insightful video with Dan Pink saying that ... “The best use of money as a motivator is to pay people enough to take the issue of money- off the table.” Your assignment illustrates this notion further.

Motivational strategy responds to a "Why" question. This is the second order question, considering the "What" being the first order. As many folks first learn what they are supposed to accomplish (e.g. obtain a university degree) and then become fully aware of why they are engaged in a degree program.

Consider several individual motivations to obtaining a degree. It is important we map them into the MMM Hierarchy.

- Fear of failing the course and losing paid tuition
- Fear of being unable to get a job and support basic needs
- Fear of getting a poor grade, being rebuked by a supervisor and ridiculed by peers
- Desire to get a new job through a better resume
- Desire to learn new skills and broaden knowledge
- Desire to apply new techniques at current work
- Desire to stay current, as a personal interest, without a specific plan to apply new skills

Should note the personal aspect of motivation..

In agile, the greatest motivation is peer pressure. A poor performer, who is unable to share the load, will be surely dismissed by a scrum team.

References

Blackboard site has an extensive list of references.

Karl Wieger ... on requirements ... multiple videos

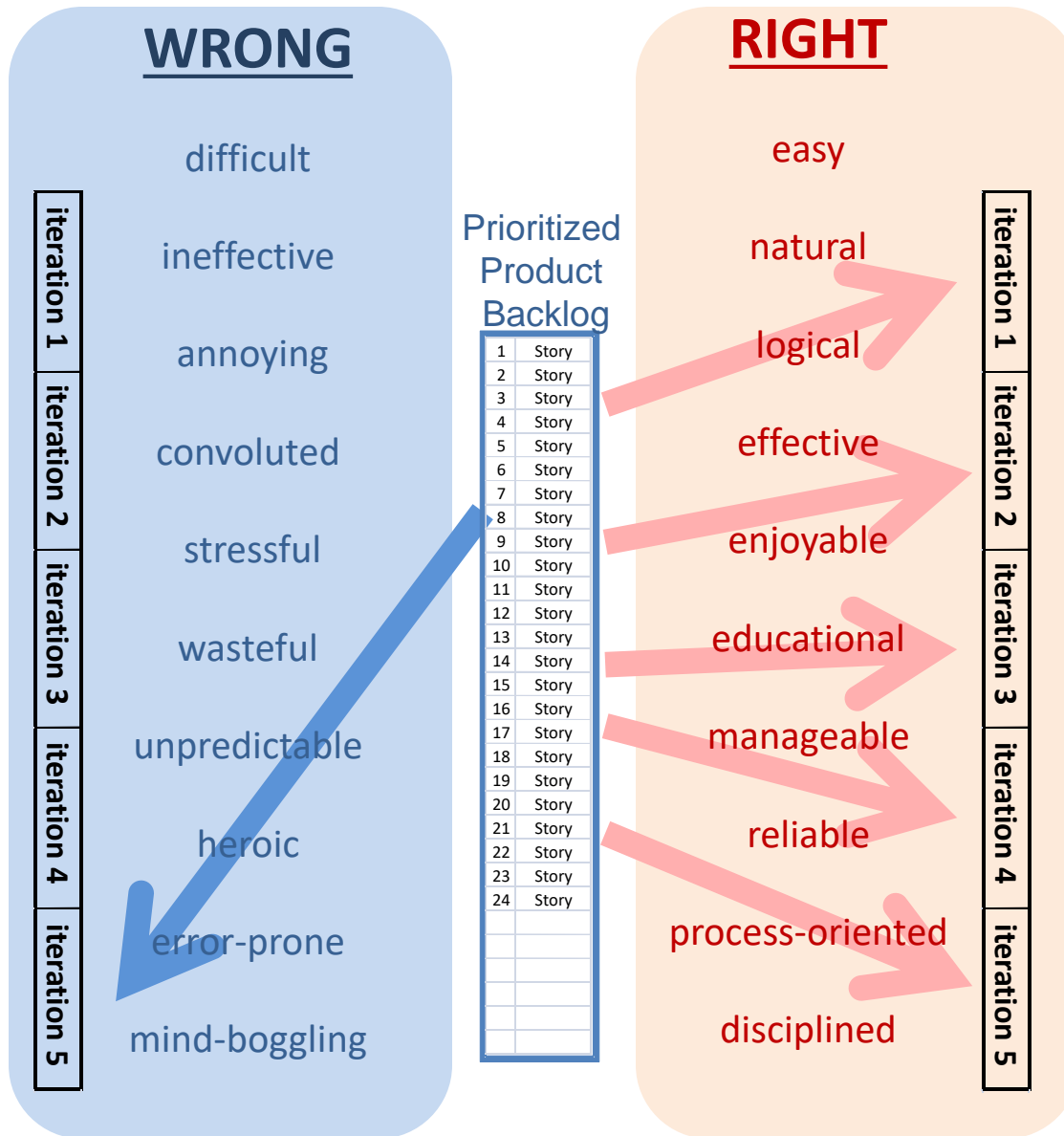
<https://www.youtube.com/playlist?list=PLhbHR06rWeInwall4GolgoingCZFQem22>

<http://www.processimpact.com/elearning.shtml>

CEOs of major companies speak out

<https://www.delltechnologies.com/en-us/broadcast-on-demand.htm?PID=newdawn-hm-postevent#scrollTop=off>

Positioning Stories Into Iterations



Scheduling all stories to be accepted in the last iteration is not only against all agile principles. It is, in fact, ineffective. Since the idea of a quick feedback is lost.

First we learn how to write one requirement. Then we learn how to write many requirements for the whole project, so to accept these requirements within certain iterations.

Actors for a story are not always end-users, but could be internal to a project.

To plan improvement, means to slowly shift from 'left picture' to 'right picture'. What is the logical sequence of tasks to facilitate such a shift? Many folks write that, spending the first week of a project on 'warm and fuzzy holding hands' (instead of moving right into the thick of development) is a wrong thing to do.

As a member of Development Team

I write stories in canonical form to fit each story into a single iteration,
so to follow an effective Scrum process

- ① Story: User Launches Beverage Tracker Application and is brought to login page

Task 1: create wireframe

Task 2: build database

Task3: code front end of this site

- ✓ A vertical slice covers only a part of UI
- ✓ Story does not fit into a single iteration
- ✓ Unable to take velocity credit

- ② As a UX Engineer,
I need to draft a wireframe,
so to enable the coding of UI

- ✓ A Task is turned into a Story to fit into a single iteration

- ③ As a HTML developer,
I need to have a wire frame approved ,
so I will be able to code UI,
while following the wireframe

- ✓ Persona is external to Action.
- ✓ HTML Developer is a customer of UX Engineer







- ④ As a User, I need a wire fame to be drafted,
so to be able to navigate to log in page

- ✓ End User receives Value

Examples of Common Stories

As members of Development Team	we need to converge on iteration cadence, team roles, dev tools	to execute most efficiently
As a members of Development Team	we need to select a topic for a project	so to satisfy objectives of End Users
As a Build Master	I need to set up an environment in GIT - branches	so to be able merging commits and resolve conflicts
As a UI designer	I need to review wireframes within the team and incorporate changes	so to maintain consistency of product
As a Tester	I need to document several basic test cases	so to learn the standard format of a test case
As a Tester	I need to reduce the number of combination of parameters using ALLPAIRS	so to have enough execution time while preserving adequate coverage
As members of development team	we need to submit draft report weekly	so to receive feedback from prof / TA and incorporate comments
As an administrator	I need to edit/delete users	so to maintain the site in good order
As an authenticated user	I need to set and chage my password according with security policy	so to gain access to site
As a user	I want to see a list of puzzles ranked by average time to complete	so that I can select a puzzle by difficulty
As a user	I would like to share my results on popular media streams, Facebook/Twitter	so to widen collaboration

Term Project

			T E A M D E L I V E R A B L E S				
			RE	CM	DESIGN	UX	TEST
			5-Oct	19-Oct	2-Nov	16-Nov	7-Dec
	Team Lead	Component	Personas(RASCI)	CI List	Use Cases	Wireframes	Test cases
			Requirements	Estimation Record	State Transitions		Data-driven table
			PivotalTracker	Tools Connectivity	Components Interraction		
	Project 1 - (XXX)						
Student 1							
Student 2							
Student 3							
Student 4							
Student 5							
Student 6							
	Project 2 - (YYY)						
Student 7							
Student 8							
Student 9							
Student 10							
Student 11							
Student 12							

Roles of a Project Team

Agenda for the initial meeting of a project team,

- Discuss who will play which role, as per the table above, and who will be responsible for which deliverables
- It is best to select one person who will upload all submissions
- Document a project policy, rules of engagement, which is your first submission, see several examples on blackboard.
- Decide on a set of tools that you will be using, e.g. Slack, Git, Sketch, etc.
- Review proposals for a scope of a project

Here are several recommended topics for a project. These are extensions from previous successful projects that add value to the course content.

- 1) Interactive Learning Platform - extension of previous project
<http://association-learning.us-east-1.elasticbeanstalk.com/>
- 2) Application to reduce the number of test combinations using AllPairs algorithm of Module 5
- 3) Collection of tools for a software development project - extension of assignment M4A2
- 4) ML application for SQA - extension of assignment M6A3

Project. Interactive Learning Platform. Add features.

<http://association-learning.us-east-1.elasticbeanstalk.com/>

Detailed description in the paper 'Interactive Learning Platform'
Developers Manual is available.

	Terminology	Requirements	SCM	Estimation	Agile	Peer Reviews	Design	Tools	Testing	Process Architecture	Improvement
Terminology		1	2	3	4	5	6	7	8	9	10
Requirements			11	12	13	14	15	16	17	18	19
SCM				20	21	22	23	24	25	26	27
Estimation					28	29	30	31	32	33	34
Agile						35	36	37	38	39	40
Peer Reviews							41	42	43	44	45
Design								46	47	48	49
Tools									50	51	52
Testing										53	54
Process Architecture											55
Improvement											—

Project. Application to reduce the number of test combinations using AllPairs algorithm of Module 5

Detailed description in notes of Module # 5

	Transition from STATE1 into STATE 2	Guest	Registered User	Golf Course Admin	Web Site Admin	Payment Processor
1	from NOTHING to REGISTERED	YES (1)	NO	YES	YES	NO
2	from REGISTERED to PROFILE UPDATED	NO	YES (2)	YES	YES	NO
3	from REGISTERED to TEE TIME ADDED	NO	YES (3)	YES	YES	NO
4	from REGISTERED to GROUP GOLFERS CREATED	NO	NO	YES (5)	YES	NO
5	from GROUP GOLFERS CREATED to GROUP GOLFERS UPDATED	NO	NO	YES	YES (6)	NO
6	from REGISTERED to PAYMENT SUBMITTED	NO	YES (4)	YES	YES	YES (7)
	Total combinations	30				
	Prohibited combinatins (red)	13				
	Dropped combinations (yellow)	10				
	Selected for execution combinations (white color)	7				

Project. Collection of tools for a software development project - extension of assignment M4A2

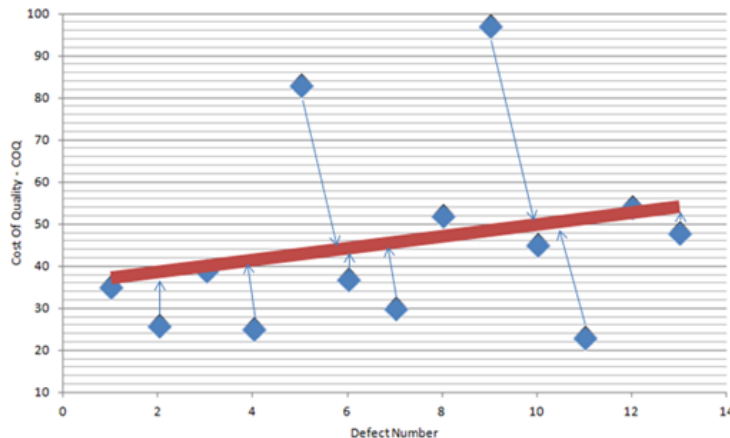
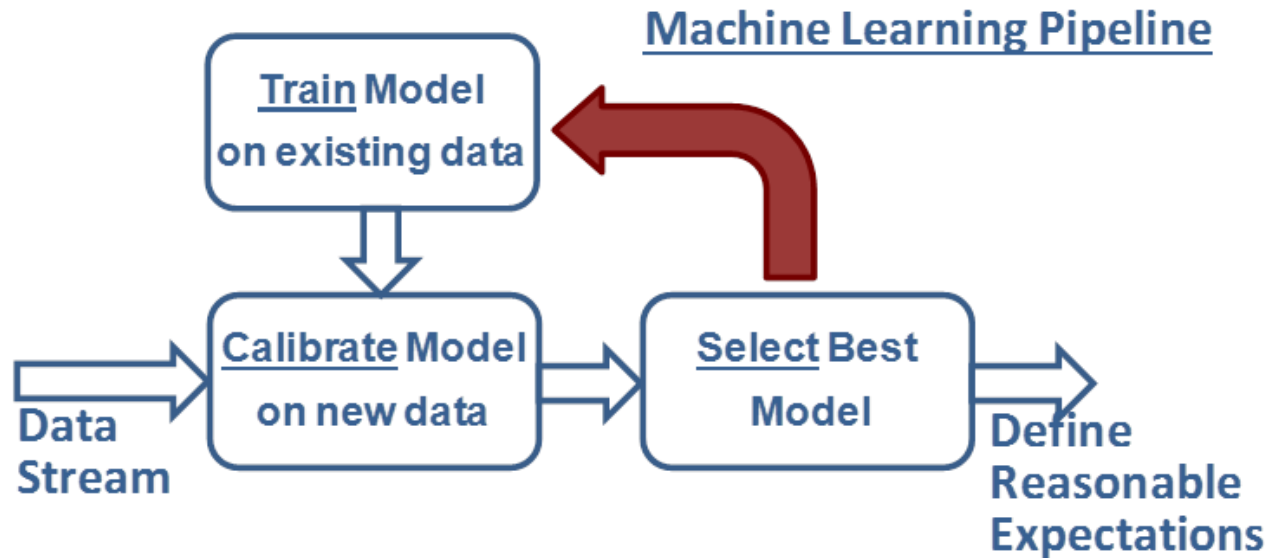
Detailed description in Assignment # 2 of Module # 4

<http://ec2-52-15-156-129.us-east-2.compute.amazonaws.com/>

TOOLS	CATEGORIES										
	Doc Control	Code Control	Peer Reviews	Defect Tracking	System Test Mngmnt	Requirements	Collaboration	Unit Test	StaticAnalysis		
VersionOne										✓	
AccuRev										✓	
Mockito										✓	
Documentum										✓	
Cruible										✓	
Bugzilla										✓	
HP Quality Center										✓	
Rally										✓	
Coverty										✓	
GIT										✓	
Collaborator Smart Bear										✓	
Collabnet Teamforge										✓	
Zephyr										✓	
WebEX										✓	
McCabe										✓	
Jira										✓	
Gmock										✓	
Adobe Connect										✓	

Project. ML application for SQA - extension of assignment M6A3

Detailed description in Assignment # 3 of Module # 6



Project. Repository of completed projects

<https://ec2-3-95-13-178.compute-1.amazonaws.com/>

ARCHIVE			ALL PROJECTS SEARCH UPLOAD PROJECT ADMIN SIGN OUT		
First Project <i>SPRING I 2019</i>	K-Now <i>SPRING I 2016</i>	Project Archive <i>SPRING I 2019</i>			
lorem ipsum dolor sit amet, consectetur adipiscing elit. etiam vitae ultricies odio, id auctor mauris. maecenas vel orci magna. vivamus semper tortor in ornare rutrum. nullam porttitor sem quis nis...	repository of best practices	the goal for this project is to provide a durable, central location where projects can be uploaded and tagged by various identifiers and made available and searchable by the world at large (scope o...			
IS673 Software Engineering Past Project Repository <i>FALL II 2018</i>	Project Portal <i>SPRING I 2019</i>	On Boarding System <i>SPRING I 2018</i>			
to develop a repository for software engineering term projects, where anyone can search through and view term projects developed over the course of previous semesters. the goal for this project is ...	the scope of group 4's project is to improve project portal, an existing single-page web application (spa), which was developed in spring 2018 by a group of met cs students enrolled in software eng...	application for initial training of newly hired			
Code For Kids <i>SPRING I 2019</i>	Student Directory <i>SPRING I 2019</i>	Lean and Mean Calculator <i>SPRING I 2019</i>			
team will create a website for the non-profit "code for kids" to promote coding boot camps for children in communities that are less fortunate. the website will provide information about: the work ...	the student directory is an application devoted for boston university students, professors and tas. the student directory app will allow the users to create account, add their personal and school i...	develop a calculator that accepts a user's input to determine a user's caloric needs, body mass index, and macronutrient breakdown.			

ARCHIVE			ALL PROJECTS SEARCH UPLOAD PROJECT ADMIN SIGN OUT		
TEAM MEMBERS <input type="text" value="Enter names here"/>	PROJECT INFO Project Name* <input type="text"/> Description* <input type="text"/> Year* <input type="text"/> Semester* <input type="text" value="SPRING I"/> Instructor* <input type="text" value="....."/> Facilitator <input type="text" value="facilitator"/>	PROGRAMMING LANGUAGES <input type="text" value="Enter coding languages here"/> FRAMEWORKS <input type="text" value="Enter frameworks here"/>			