Agile Talks

Welcome

2013-03-20

sergiotaborda@zbra.com.br

Agile Talks

- Inception (Sérgio)
  - Origin of Agile
  - Principles of Agile
- Planning (Sérgio)
  - Planning Concepts
  - Practices Workshop (next week)
- Engineering (Fábio)
  - Agile Engineering
  - Agile Engineering Workshop

Agile Talks

Inception

2013-03-20

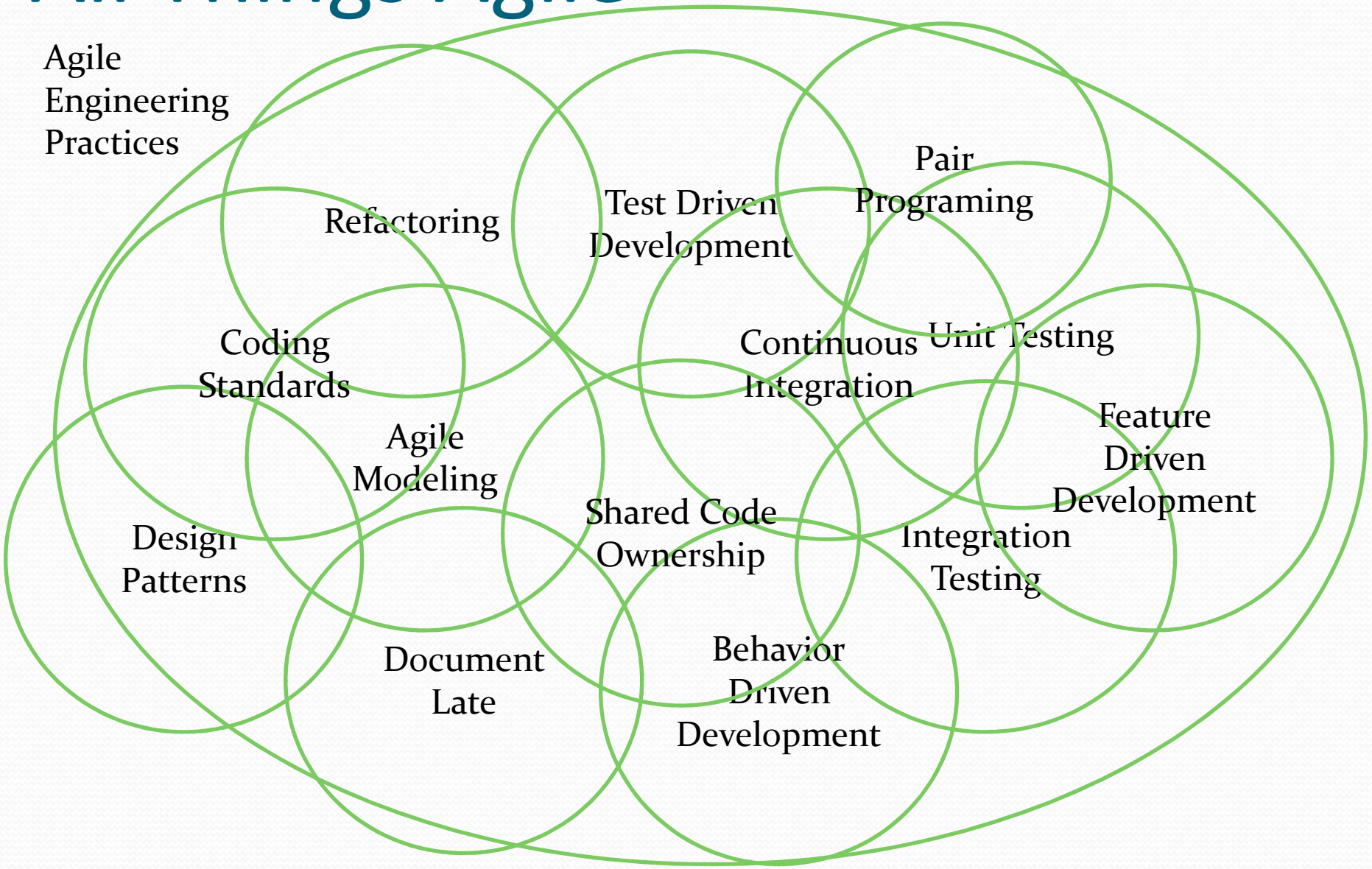
sergiotaborda@zbra.com.br

Contents

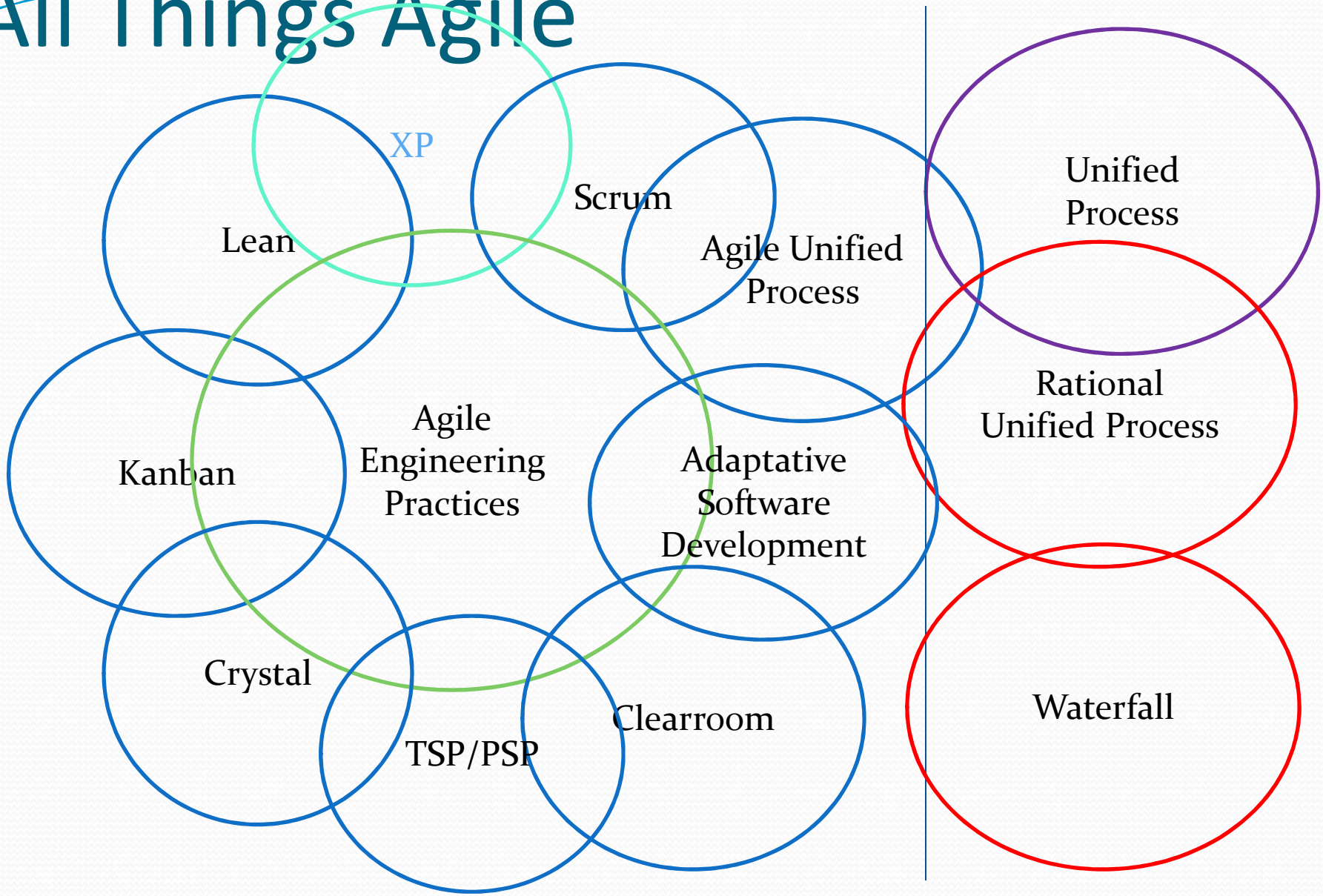
Part II

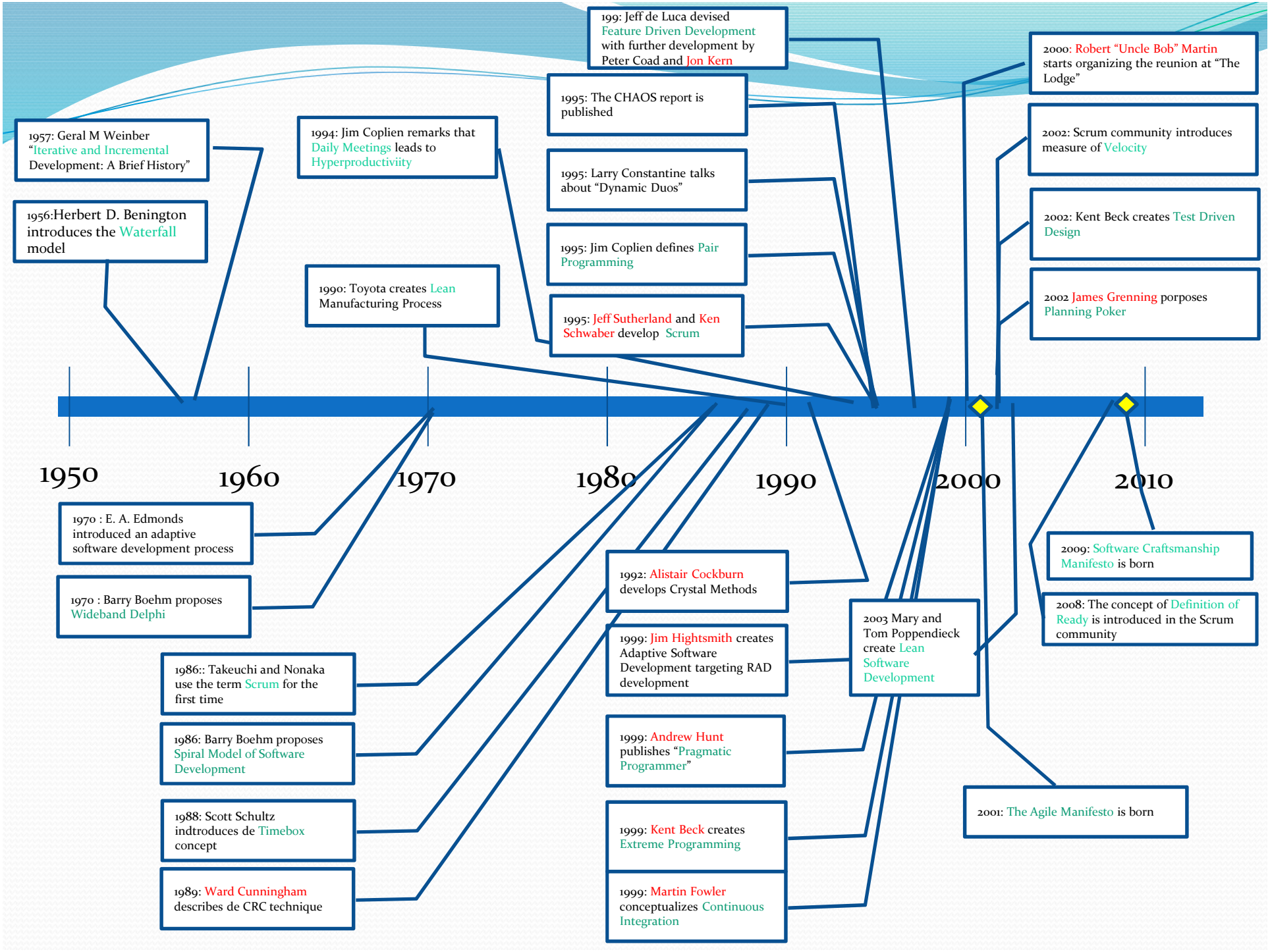
Agile Origins

All Things Agile



All Things Agile




1957: Geral M Weinber "Iterative and Incremental Development: A Brief History"
1956:Herbert D. Benington introduces the Waterfall model
1994: Jim Coplien remarks that Daily Meetings leads to Hyperproductiviity
1990: Toyota creates Lean Manufacturing Process
199: Jeff de Luca devised Feature Driven Development with further development by Peter Coad and Jon Kern
1995: The CHAOS report is published
1995: Larry Constantine talks about "Dynamic Duos"
1995: Jim Coplien defines Pair Programming
1995: Jeff Sutherland and Ken Schwaber develop Scrum
2000: Robert "Uncle Bob" Martin starts organizing the reunion at "The Lodge"
2002: Scrum community introduces measure of Velocity
2002: Kent Beck creates Test Driven Design
2002 James Grenning porposes Planning Poker
1950
1960
1970
1980
1990
2000
2010
1970 : E. A. Edmonds introduced an adaptive software development process
1970 : Barry Boehm proposes Wideband Delphi
1986:: Takeuchi and Nonaka use the term Scrum for the first time
1986: Barry Boehm proposes Spiral Model of Software Development
1988: Scott Schultz indtroduces de Timebox concept
1989: Ward Cunningham describes de CRC technique
1992: Alistair Cockburn develops Crystal Methods
1999: Jim Hightsmith creates Adaptive Software Development targeting RAD development
1999: Andrew Hunt publishes "Pragmatic Programmer"
1999: Kent Beck creates Extreme Programming
1999: Martin Fowler conceptualizes Continuous Integration
2003 Mary and Tom Poppendieck create Lean Software Development
2001: The Agile Manifesto is born
2009: Software Craftsmanship Manifesto is born
2008: The concept of Definition of Ready is introduced in the Scrum community

Agile is not a Manifest

Manifest

Individuals and interactions over Processes and tools

Working software over Comprehensive documentation

Customer collaboration over Contract negotiation

Responding to change over Following a plan

Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Insights

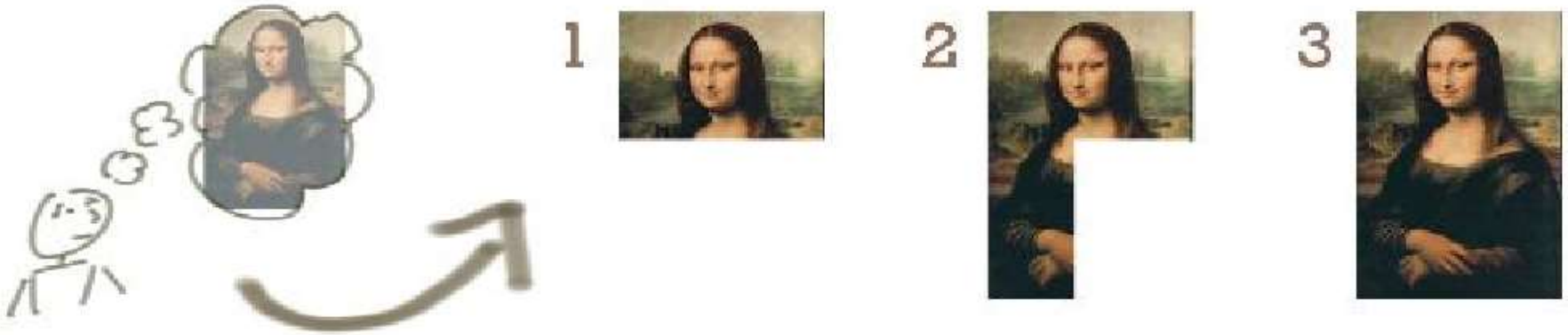
- A waterfall process is ill-suited for common software development
  - Requires a cyclic process
  - Requires an iterative process
- Software is not like building a bridge
  - Does not rely on predictive rules
  - Requires an adaptive process
- People are not drones
  - Building Software is an intellectual endeavor
  - Face-to-Face Communication works best

Iterative vs Incremental

Incremental

Iterative

Incremental



- Each step is not deliverable as the value in the whole
- If money runs out , nothing is usable
- The customer forces termination by any means necessary

Iterative

Each step is deliverable with different value

- Each step is deliverable with different, increasing, value
- If money runs out , a simpler version will run
- The customer has return over investiment at any point

Software is a Product

Project

- Is Limited in Time
  - Money runs out
  - Interest runs out
  - Oportunity phases out
- Short Term Objectives
  - Objectives do not change
- Responsability
  - Project Manager
  - Stearing Comitee
- Delegatable

Product

- Is Endless
  - It makes money
  - Is a brand
  - Its a payed investment
- Long Term Objectives
  - Objectives will change
- Responsability
  - Product Departament
  - Marketing
  - Finance
- Not Delegatable

Software is a Product

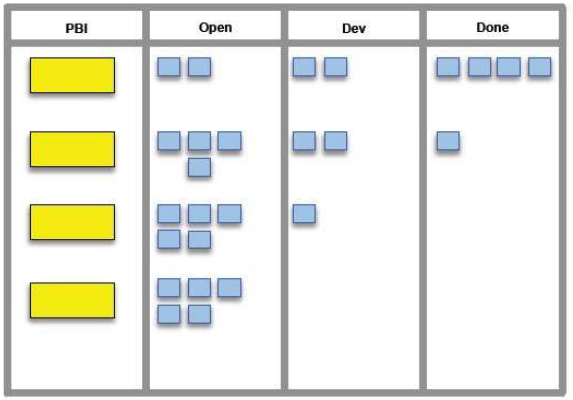
- Evolution needs to be controlled by Product Development
  - In the pivot figure of the Product Owner
  - Product Owner has responsibility for the features and the results (ROI)
- Product Evolution is based on observation of users
  - Measures to guide decision and not gut feeling
  - Features not used by users are not features
  - You are not the user

Traditional View

- The Project is the Software and vice versa
- The Project will end => The Software will end
  - Money runs out
  - Time to Market run out
- The Software has components => The project can be split in “phases”
  - Deliver in “working” parts
- We sell a Project/Software and steer a Software/Project
- Projects need documentation => Software needs documentation
- The customer is always right. Do it.

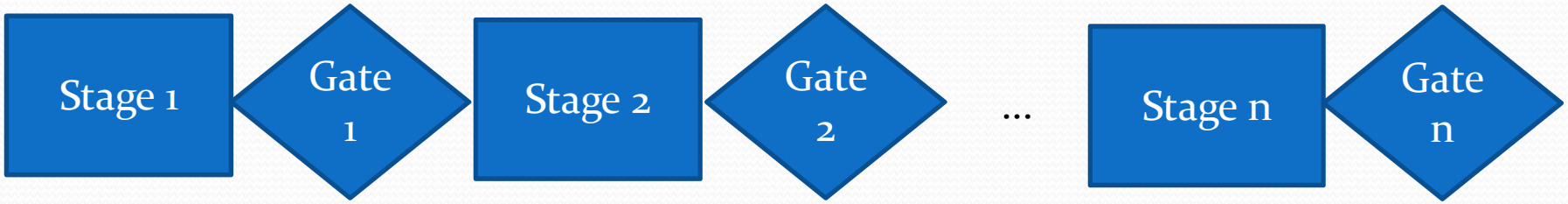
Agile View

- The Software is a Product
- The Product will never end
  - Because it makes money
- The Project is an endeavor
  - To implement a part of the software
  - The endeavor has an end, and can fail, the product doesn't
- We sell a Project and steer a Project
  - The client steers a Product and ROI
- The project needs documentation, not documents
- The software needs the documentation the client needs
- The customer is not always right, but always has a point. Listen.



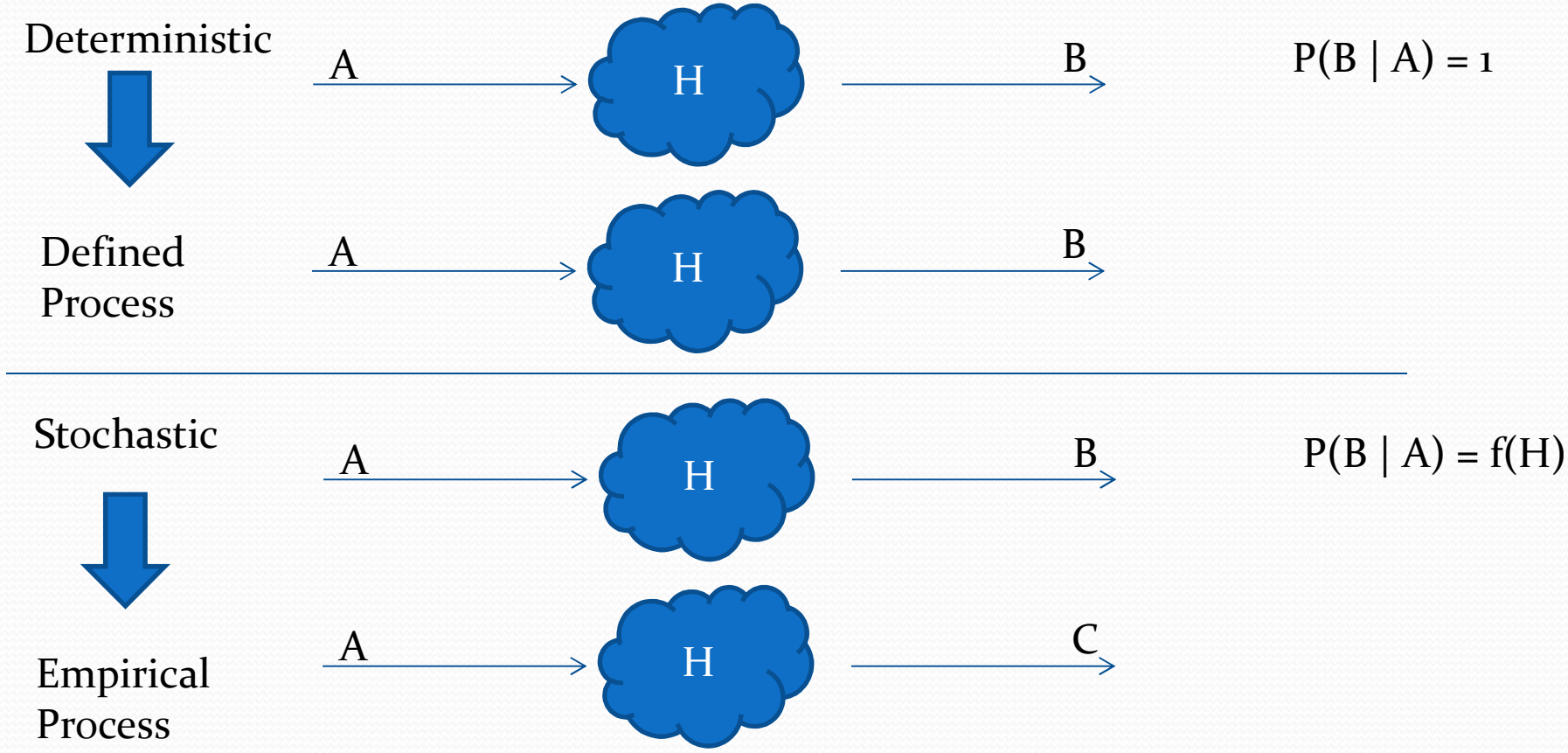
	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

The Stage-Gate Process Control

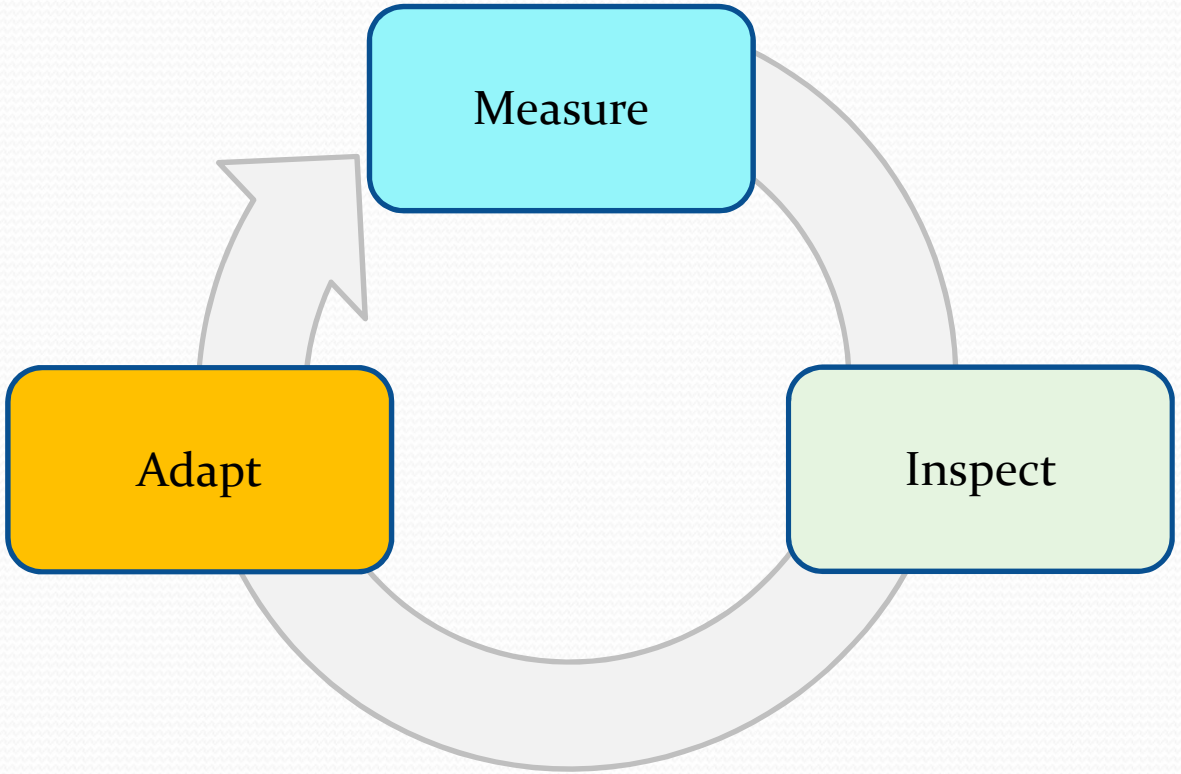


- A sequece of stages protected by gates
  - At a stage actions are performed
  - At a gate artifacts are controlled. If ok, move to next stage
- For Software Stages normally are
  - Inception, Development, Implementation, Test , Deliver
- Stage-Gate Sequence can repeated (Iterative processes)
  - Waterfall is an Iterative Stage-Gate with 1 iteration (do or die)
  - (R)UP is and Iterative Stage-Gate with N iterations

Deterministic vs Stochastic



The Empirical Process Control



- Continuous (Not Stage-Gate)
- In every action , in every individual (Not comand and control)

People Communicating

- Document Late
  - Documenting may be necessary, but it is produced after the software, not before.
  - Software guiding documentation, not documentation guiding software
- Small Teams
  - Minimize the `N(N-1) /2` Channels Problem (0, 1, 3 , 6 , 10, 15)
  - Information radiators
- Maximize Focus
  - Remove distractions
    - Bugs are distractions
    - Messed code is a distraction
    - Wrong requirements are a distraction

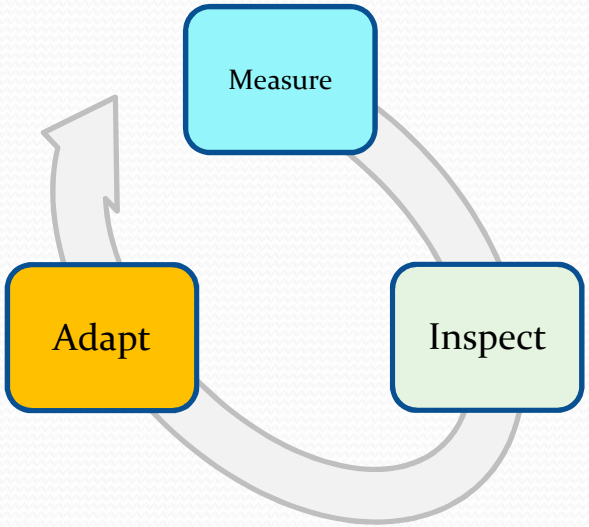
What Agile Is

Interacting People

Searching

Technical and Comercial

Using



Tends To

Empirical Process Control Model

What Agile Is

- People making a product
- Continuously Measuring
  - Times - time to deploy, time to boot, time to create feature, time to maintain, etc..
  - Sizes – size of team, size of scope,
  - Quality Metrics – size of code, size of modules, number of modules, coupling , dependency, etc…
  - Satisfaction – users use feature, users like feature, bugs, etc..
- Continuously Inspecting
  - The Quality (what the numbers mean)
  - The Scope (what to do next)
  - The Product (the objectives are being accomplished)
  - The Software (the quality is there? It still works ?)
  - The Process (is helping or annoying ?)
- Continuously Incrementing Value
  - Deliver Soon
  - Deliver Often
  - Deliver something more each time
- Continuously Adapt
  - Change what is not working
- Repeat

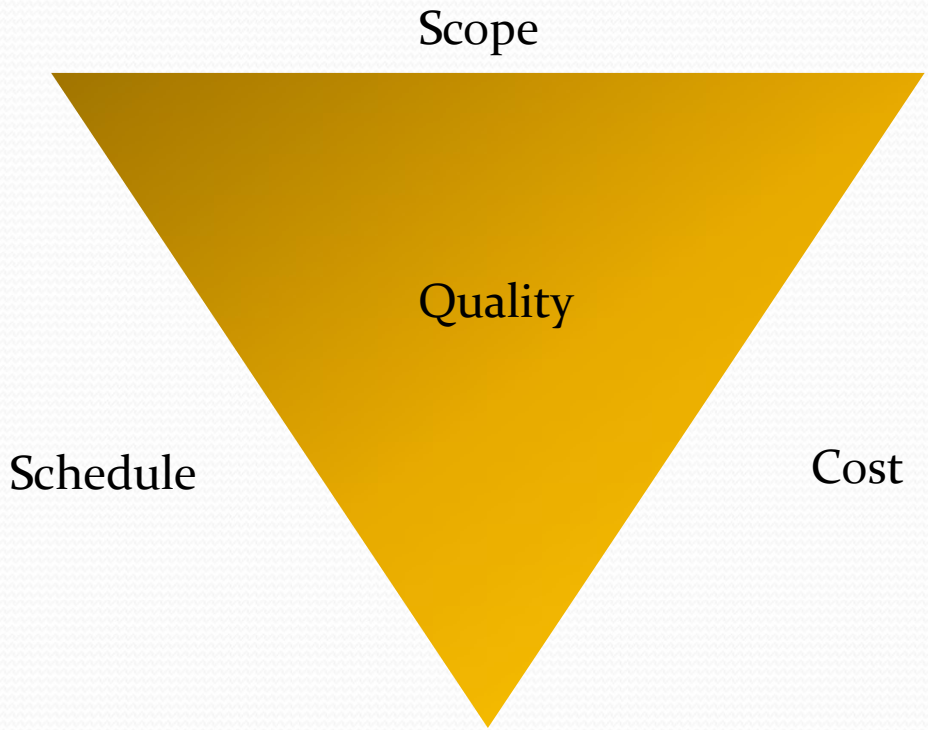
What Agile Is Not

- The Absence of Requirements Elicitation
  - Can be the absence of all-up-front elicitation
- The Absence of Documentation
  - Can be the absence of *most* documents (risk management)
  - Can be the absence of *imutable* contracts (legal rights)
- The Absence of Order
  - A single process is being followed
  - Well defined roles and responsabilities
- Leissez-Faire
  - The team is not abandoned
  - The team is not independent
  - The team is not responsable for business decisions

Part II

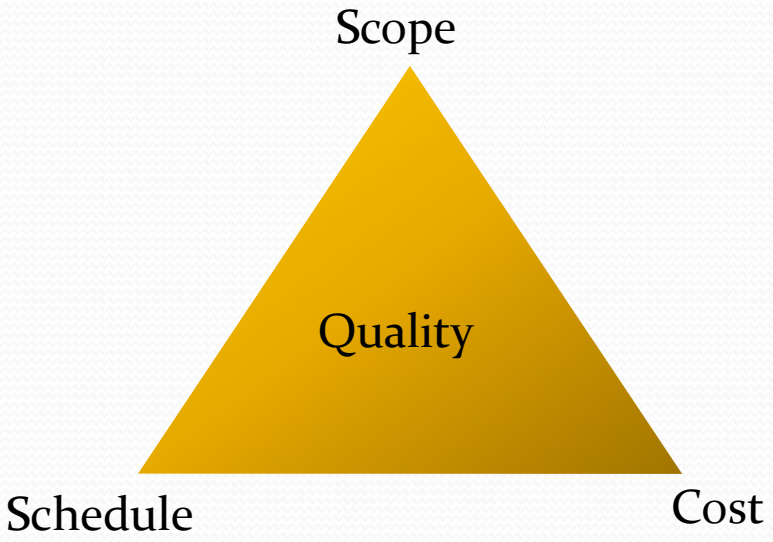
Agile Planning

Project Dimensions

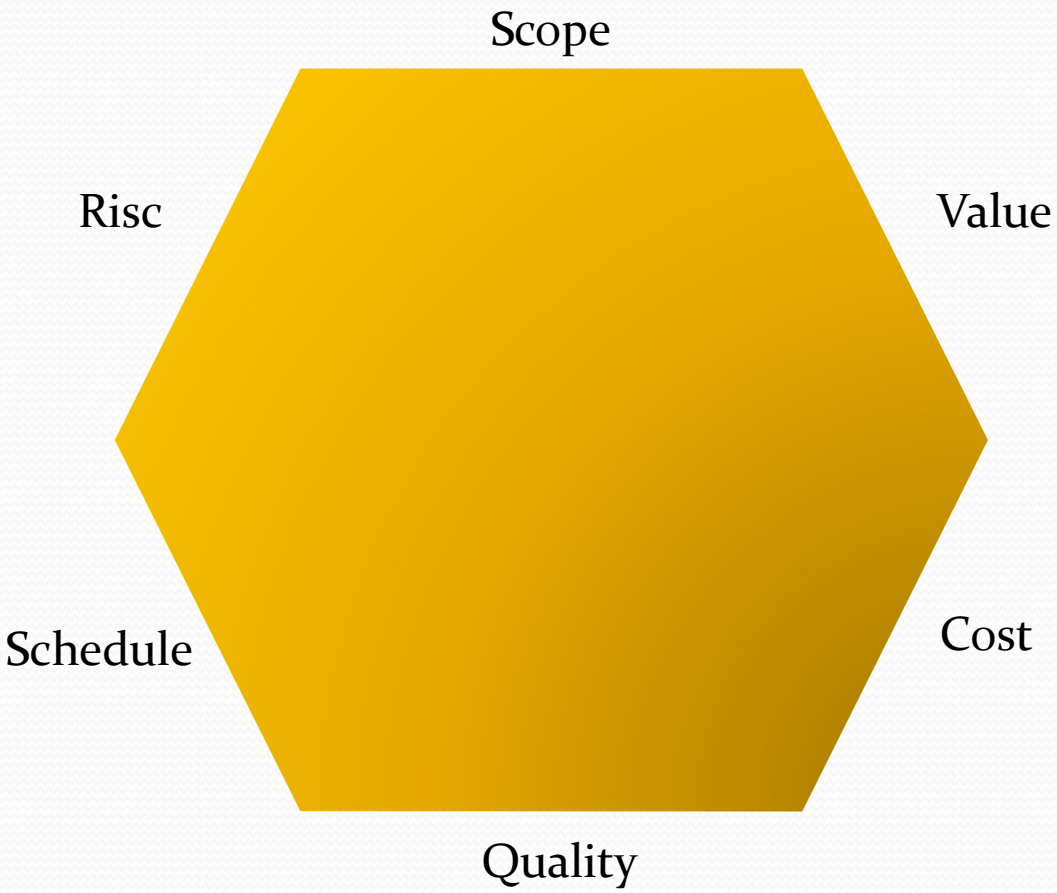


Tradicional View

- Define Scope, Schedule , Cost
  - You can optimize two
- Assume Quality is given
  - If you do not trust them, create a Quality Assurance body
- Assume the Scope is contractual and all Scope is necessary
- Assume Scope will not change
  - If it does annoy them with Change Requests
- Assume Risc is controllable
  - Create a **Stearing Commitee**.

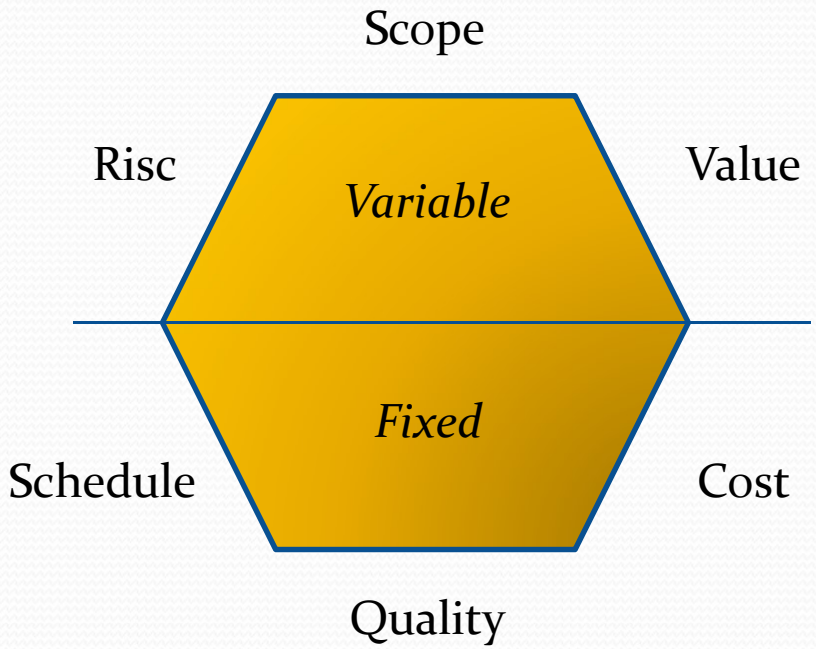


Project Dimensions



Modern View

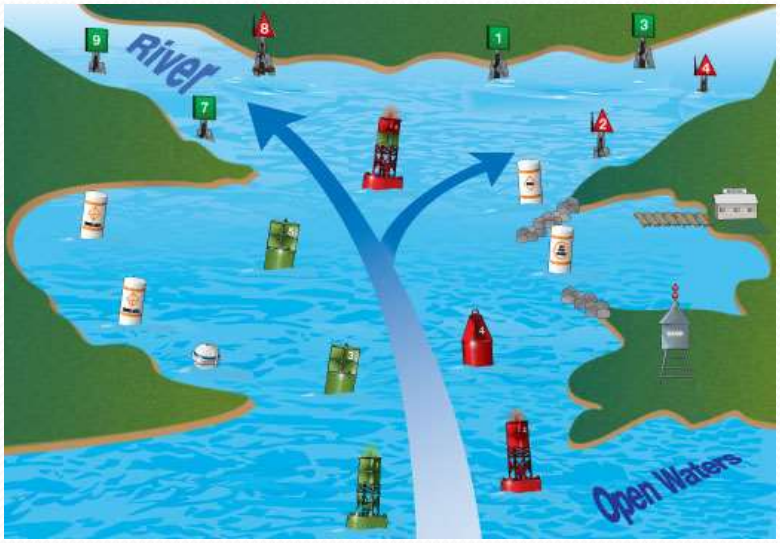
- Mesure all variables with numbers
- Define Schedule, Cost and Quality
- Assume Scope will change
  - Because it will
  - Because Value will change
  - Because Business Objectives will change
- Risc is controlable if measurable
  - Adjust Scope acordingly



Tradicional View

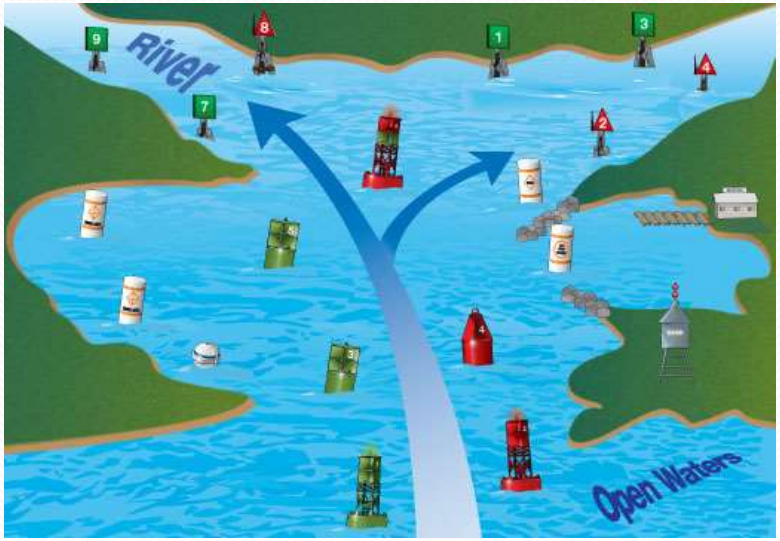
- Target driven
  - Target a Cost
  - Target a Schedule
  - Target a Scope
- Stearing
  - The target does not move
  - You cannot go back
  - You cannot abort
  - The **arrow** cannot be moved by the Steering Commitee to another target on demand

Agile View



- Objective driven
  - Target a Cost
  - Target a Schedule
  - Target a Quality
  - Target an Objective
- Navigation
  - The target does move
  - You can go back
  - You can always abort
  - The **boat** can be moved by the Steering Commitee to another target on demand

Steering vs Navigating



Dimensions in Agile : Schedule

- Define
  - Releases by Time to Market Dates
  - Goals for Releases
  - Dates and Duration of Releases
  - Number of Sprints in Release
- Measure
  - Story Points per Sprint (Velocity)
  - Story Points in Release / Minimum Velocity
- Review Constantly
  - Release Backlog Burndown

Software

Project

Release 1 (Version 1.1)

Sprint 1

Sprint 2

Sprint 3

Release 2 (Version 1.2)

Sprint 1

Sprint 2

Sprint 3

Sprint 4

Software

Project A

Project B

Project C

Software

Release 1

Sprint 1

Sprint 2

Sprint 3

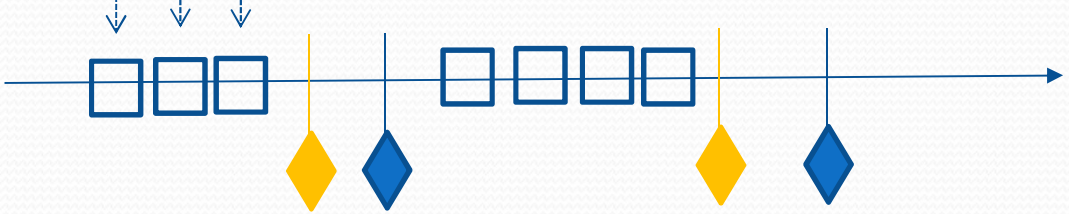
Release 2

Sprint 1

Sprint 2

Sprint 3

Sprint 4



Dimensions in Agile : Cost

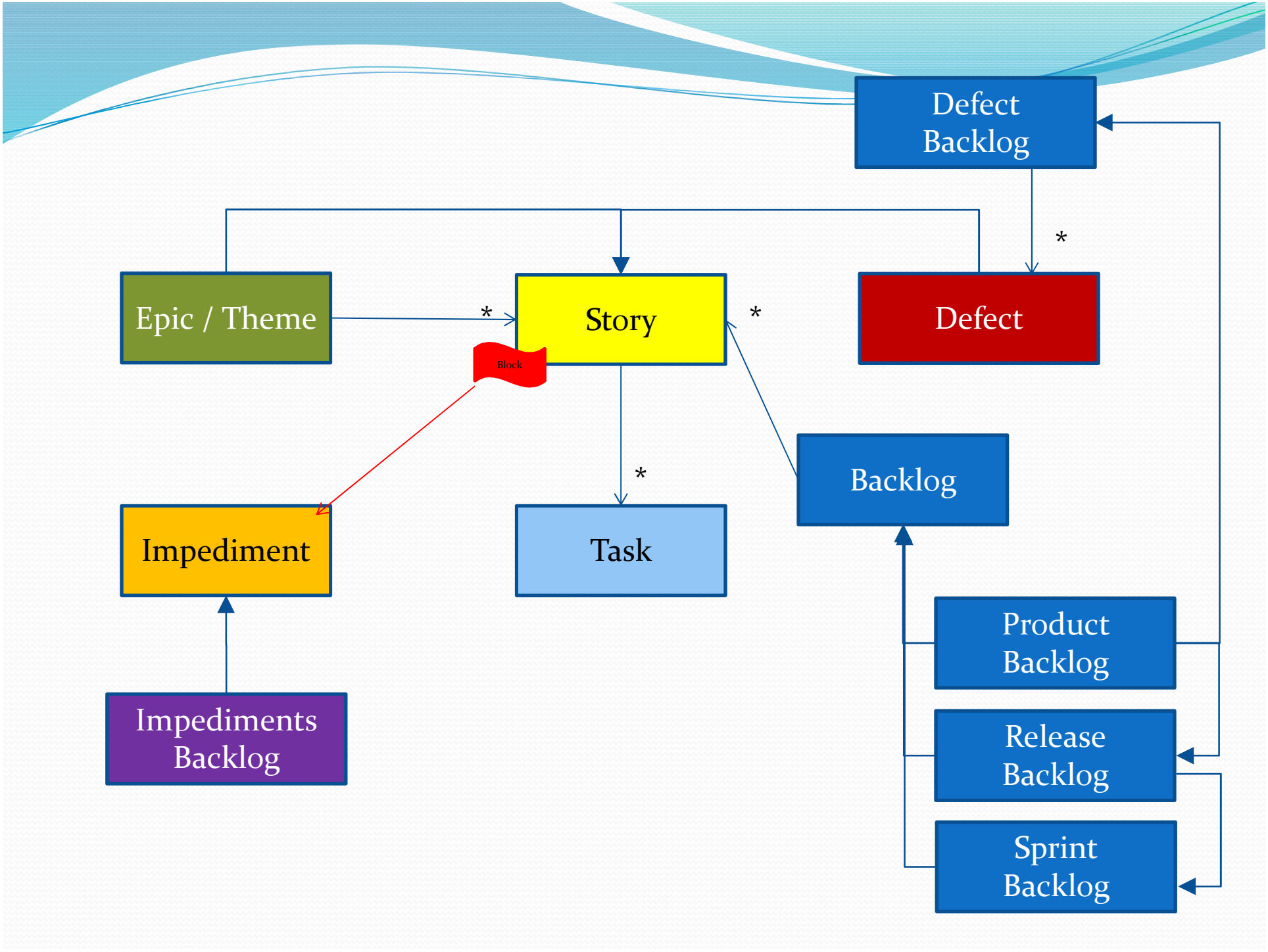
- Define
  - Cost = $Cost_{per\ Relase} \times Nº\ of\ Releases$
  - Cost Per Release = $\left(Cost_{per\ Sprint} + Overhead\right) \times Nº\ of\ Sprints_{in\ Release}$
  - Velocity = Story Points Burned per Sprint
- Measure
  - Focus Factor = Fixed Budget / Velocity
  - Nº of Sprints Needed = Backlog Size / Velocity
  - Cost = $\frac{Cost\ Per\ Sprint}{Fixed\ Sprint\ Budget} \times \frac{Backlog\ Size}{Focus\ Factor}$

Agile Contract

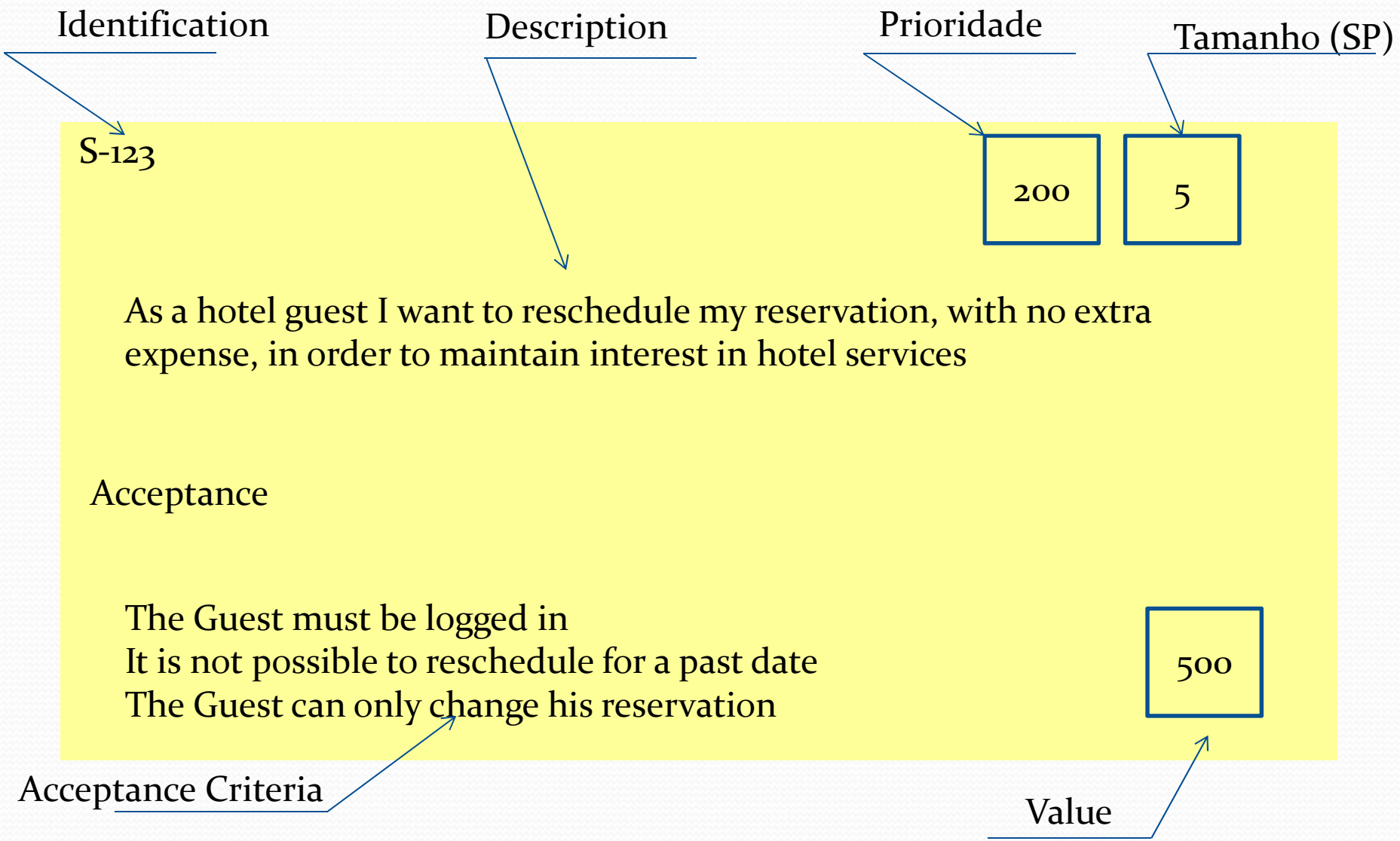
- Customer collaboration over contract negotiation
  - “over” not “instead”
  - Contract is legally mandatory
  - “Negotiation” comes from problems with promises made about schedule and scope, that do not exist in agile
- Agile Contract
  1. Define an umbrella contract that defines the stakeholders involved and the general rules of engagement
     - Does not define scope, or time limit
  2. Append “Service Orders” for an appropriated time period (1 Release , 2 Releases, etc..)

Dimensions in Agile : Scope

- Define
  - By Creating Stories
  - Putting Stories in Backlog
- Measure
  - Size in Story Points
- Review Constantly
  - Priority
  - Dependency
  - Value


Defect Backlog
*
Epic / Theme
*
Story
*
Defect
Block
*
Backlog
Impediment
Task
Product Backlog
Release Backlog
Sprint Backlog
Impediments Backlog

Story Taxomony



INVEST (in) Ready Stories

- I – Independent
  - One Story must not depend on others (decoupling)
- N – Negotiable
  - Not all stories are necessary
  - Not all stories are ASAP
- V – Valuable
  - Must improve business ROI
  - Must benefit the user in some aspect
- E – Estimable
  - Must be possible to estimate its size
- S – Small / Simple
  - As small and simple as possible to help implementation
- T – Testable
  - Must be verifiable by the user
  - Better if can be automatically, and repeatedly, verified

Story Taxomony

Actor

Goal

Business Value

S-123

200

8

As a hotel guest I want to reschedule my reservation, with no extra expense, in order to maintain interest in hotel services

Acceptance

The Guest must be logged in
It is not possible to reschedule for a past date
The Guest can only change his reservation

500

Pre and Pos Conditions + Alternative Paths

Software

Release 1

Sprint 1

Sprint 2

Sprint 3

Release 2

Sprint 1

Sprint 2

Sprint 3

Sprint 4

Product Backlog

Release Backlog

Sprint Backlog

Sprint Backlog

Sprint Backlog

Release Backlog

Sprint Backlog

Sprint Backlog

Sprint Backlog

Sprint Backlog

The Story Size

- Size is a new property not found in traditional environments
- Size a measure of work
- Size is used to estimate of “how big the story is” and not “how long is necessary to implement it”
- Size is related to
  - Definition of “Done”
    - Needed documentation , quality , etc..
  - Number of tasks necessary
  - Difficulty of tasks necessary
  - Complexity of Business Rules
- Size is not a quantity of Risk, nor a quantity of Time

Determine Story Point Meaning

1. Ideal Day
   - 1 SP = 1 Ideal Day for a Senior Developer
2. Relative Size
   - N SP = Size of a Reference story or story group
   - 1 SP = Size of a story that is 1/N of the reference story
   - Normally N = 5
3. Calculated Size
   - Based on a cost methodology like Function Point Analysis, User Case Points, etc…
   - This approach falls rapidly to an all-elicitation-up-front

Goodhart's Law

"When a measure becomes a target,
it ceases to be a good measure"

- Do not use Story Points contractually
- Story Points are an "internal" measure

Dimensions in Agile :Quality

- Define
  - Definition of “Ready”
  - Definition of “Done”
  - Acceptance Criteria (global and per story)
  - Adherence to Standards (User Experience, Coding, Documenting , etc…)
- Measure from
  - Size of Defect Backlog
  - Automatic Metrics (Sonar)
- Review Constantly
  - Too many constrains make Velocity slow down
  - Too few constrains make Defect Backlog bigger

Quality Assurance

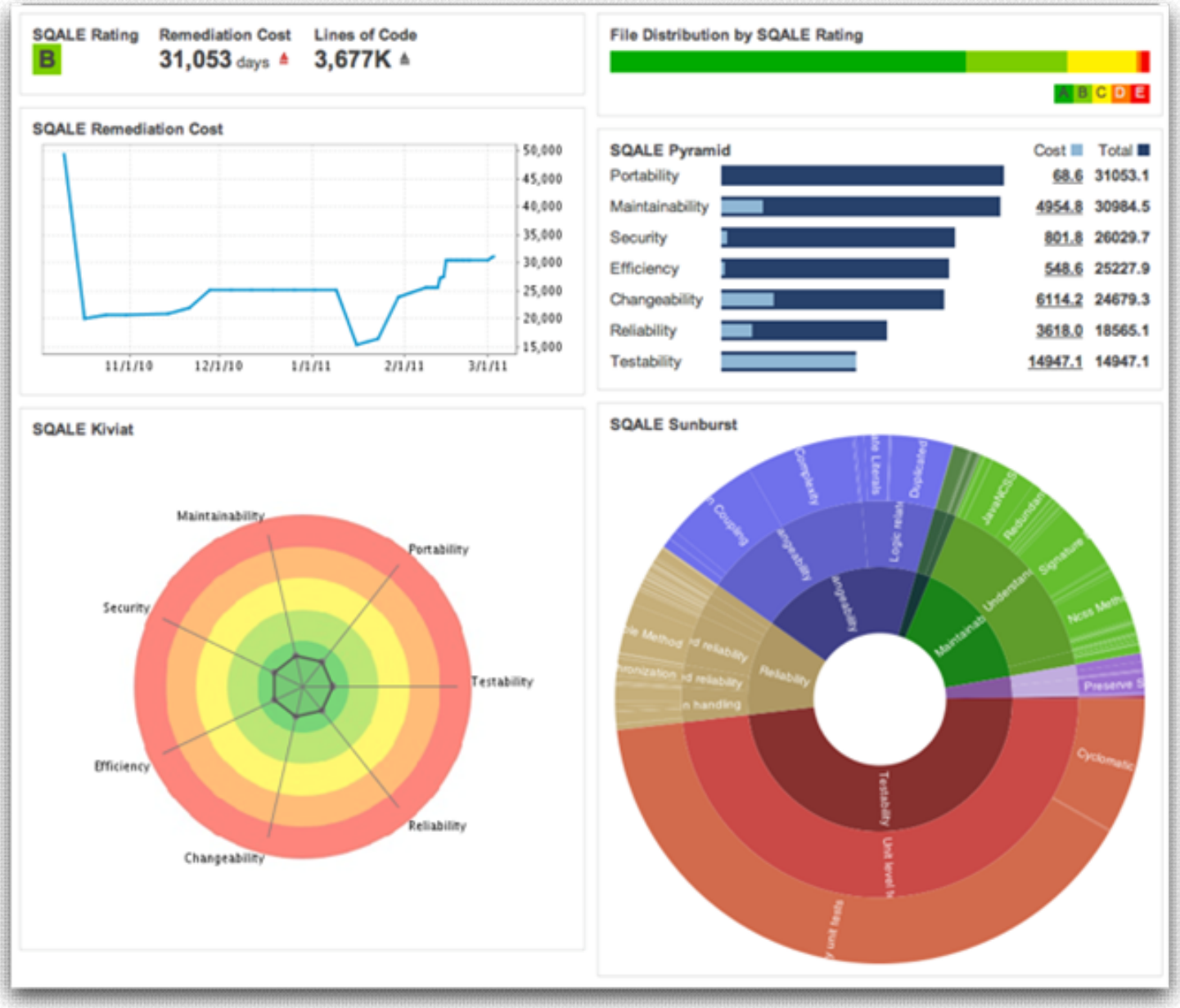
Experiments

- Manual
  - Crystalized
- Low Reproducibility
- Slow
  - Integration
  - Environment
  - Data
- Red-Green Cycle
  - Correct or incorrect
  - No degree of correctness
  - Other Metrics are hard to include
- Acceptance Criteria
  - One single special case
  - Focus on “happy path”

Tests

- Automatic
  - Organic Growth
- High Reproducibility
- Slow to Fast
  - Depending on what to test
  - Unit => Fast
  - Integration => Slow
- Red-Green Cycle
  - Correct or incorrect
  - With degree of correctness
  - Includes Other Metrics
- Acceptance Criteria
  - All cases
  - Include corner cases


Dashboards
Projects
Measures
Issues
Settings
Log in
Search
Helicopter View
Activity
Java Projects
Javascript Projects
Languages Panel
TOOLS
Dependencies
Compare
sonarqube
Sonar as a Service
for your project with
CloudBees
All Projects
SQALE Rating
B
5,280.4 days to
Remediation Cost
69,102.9 days
Lines of Code
10,637K
All Projects
Issues
524,089
Rules compliance
87.1%
Blocker 889
Critical 3,019
Major 441,509
Minor 20,256
Info 58,416
Forges
Name
LOCs
SQALE Rating
Forges 8,114,396 B
Apache 4,149,049 A
Others 1,984,743 B
JBoss 560,876 B
OW2 535,057 B
Sourceforge 367,201 A
Codehaus 257,051 A
GoogleCode 137,790 A
OPS4J 71,501 A
SpringSource 51,128 A
9 results
All Projects
Size: Lines of code Color: Rules compliance 0.0% 100.0%
JDK 7
Apache CloudStack
Apache CXF
Silverpeas Core
Apache Jackrabbit
Apache Jena
OpenEJB
XWiki Platform -
Jahia Project Root
ActiveMQ
Apache Isis
Jetspeed 2
Camel
Wicket Parent
Jenkins main
Struts 2
Bonita
Nuxeo ECM Projects
ApacheDS Build With
OpenJPA Parent POM
JOnAS
Apache Tomcat
Apache Sling
Cayenne
Sigmah
org.qi4j
Jetty :: Project
Hudson
Castor
Guava
Maven
Apache
Sonar
All
PMD
All Projects
Lines of code: 10,637,079
Duplicated lines: 1,537,042
Unit tests: 551,730
04:10
2010
2011
2012
2013


SQALE Rating
B
Remediation Cost
31,053 days
Lines of Code
3,677K
File Distribution by SQALE Rating
A B C D E
SQALE Remediation Cost
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
11/1/10
12/1/10
1/1/11
2/1/11
3/1/11
SQALE Pyramid
Cost
Total
Portability 68.6 31053.1
Maintainability 4954.8 30984.5
Security 801.8 26029.7
Efficiency 548.6 25227.9
Changeability 6114.2 24679.3
Reliability 3618.0 18565.1
Testability 14947.1 14947.1
SQALE Kiviat
Maintainability
Portability
Security
Testability
Efficiency
Reliability
Changeability
SQALE Sunburst

Dimensions in Agile : Value

- Define
  - Relative Scale between stories (Priority Scale)
  - By Order in Backlog
- Measure
  - Benefit : how bad if removed [1 to 9]
  - Penalty : how badly would be missed [1 to 9]
  - Value = Benefit + Penalty
- Review Constantly
  - Attractiveness = $\frac{v}{t+r}$

Dimensions in Agile : Risk

- Define
  - Risk Exposure = Probability x Loss
  - Loss = Money, Time or Story Points
  - Sources of Risk
    - Story, Release, Project, Software
      - Ex: Risk of Dependent Systems are incompatible during integration
    - People, Scope, Quality, Cost, Schedule
      - Ex: Risk of Team members dropping out
      - Ex: Risk of incorrect calculations affecting company billing
- Measure
  - Size of Defects Backlog / Velocity (min)
  - Size of Stories Blocked by Impediments in Implements Backlog
  - Value of Stories Blocked by Impediments in Implements Backlog

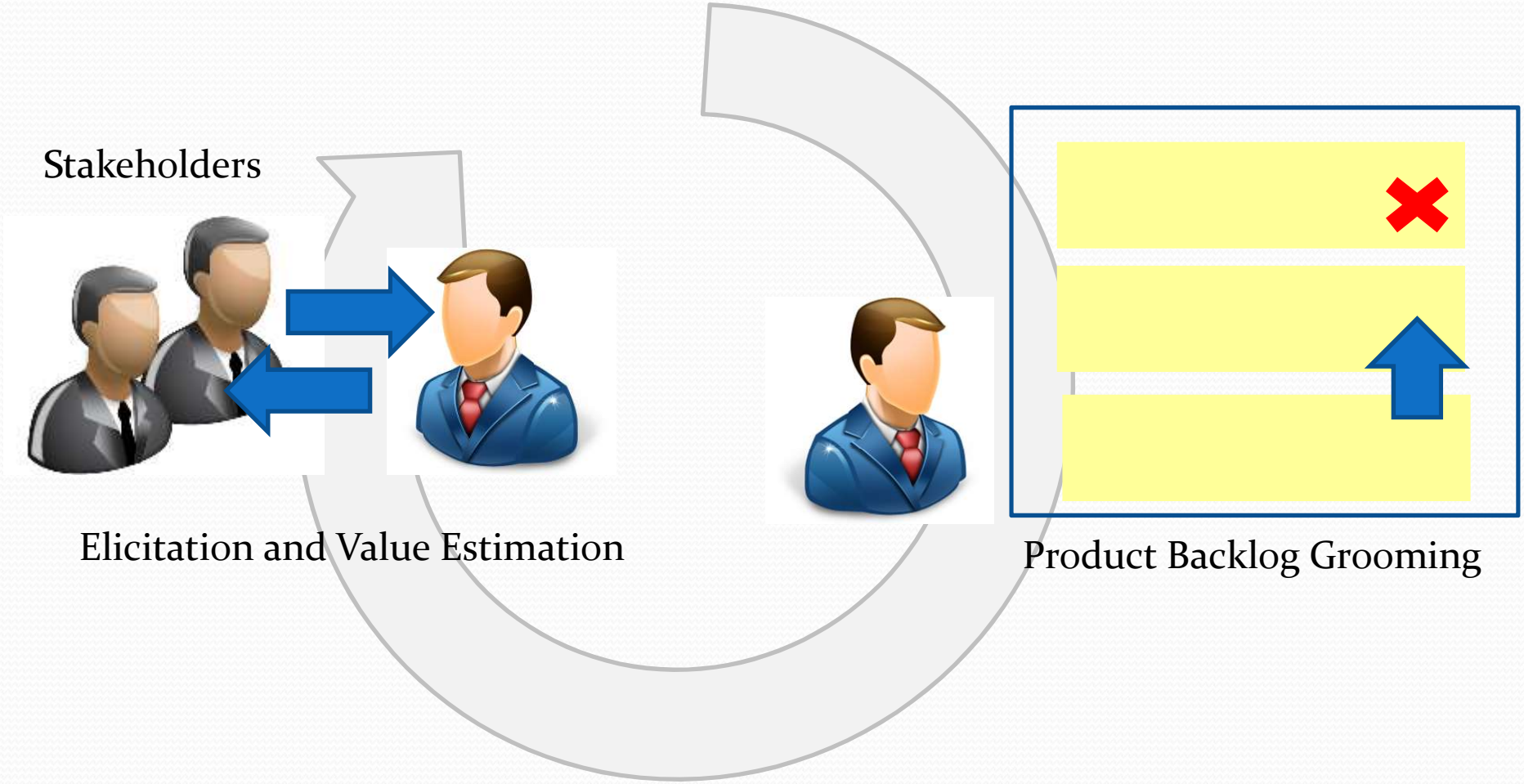
The Product Owner (PO)

- Controls the Product
- Mediates between
  - Stakeholders and the Team
  - Stakeholders and Stakeholders
- Enriches the Product in order to optimize the Return On Investment
  - The Gain / Cost / Time
- The PO is not a project manager (PM). A product may need several different PM. The PO is the “teams custumer”

Product Team

- Is not the Dev Team. It's the "arms" of the PO
- May include business specialist and advisors
- Focus on collecting information from stakeholders
  - Product Usability, FAQ, Suggestions, Features not used
- Focus on measuring stories Value
  - Benefit, Penalty, Priority
- Focus on getting the stories to "Ready" state
  - Focus on creating acceptance criteria
  - Corner Cases
  - Business Model consistency (interaction of rules)

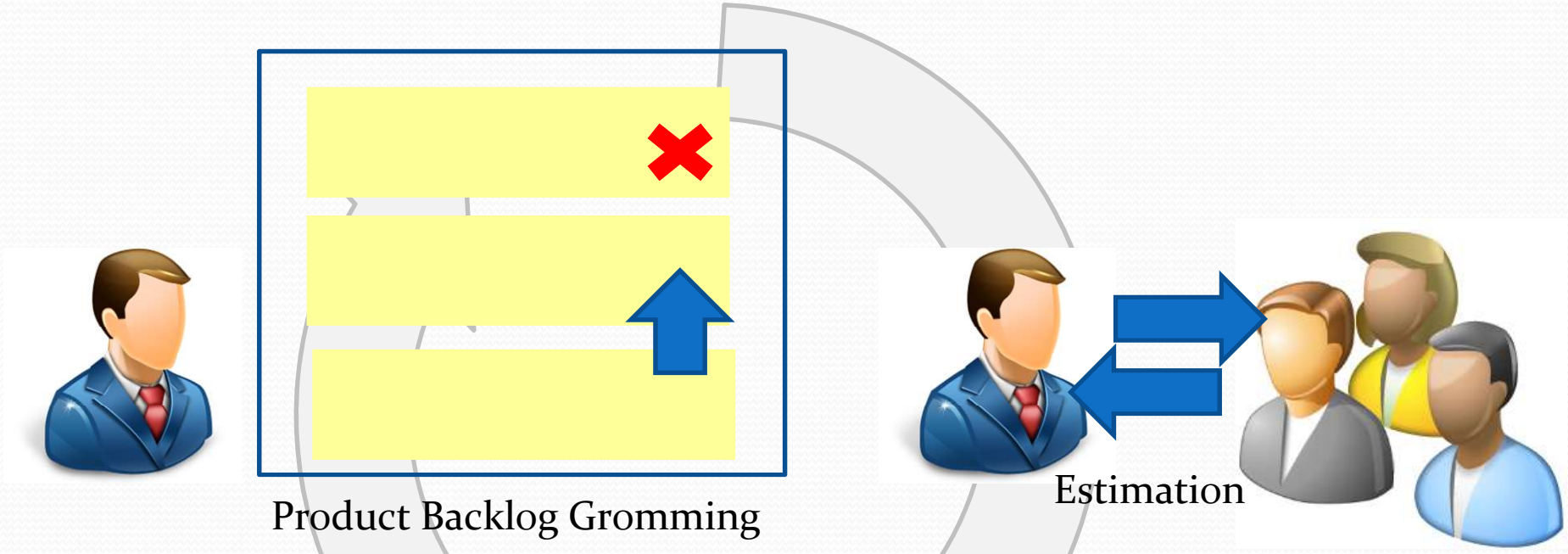
Elicitation and Ready State



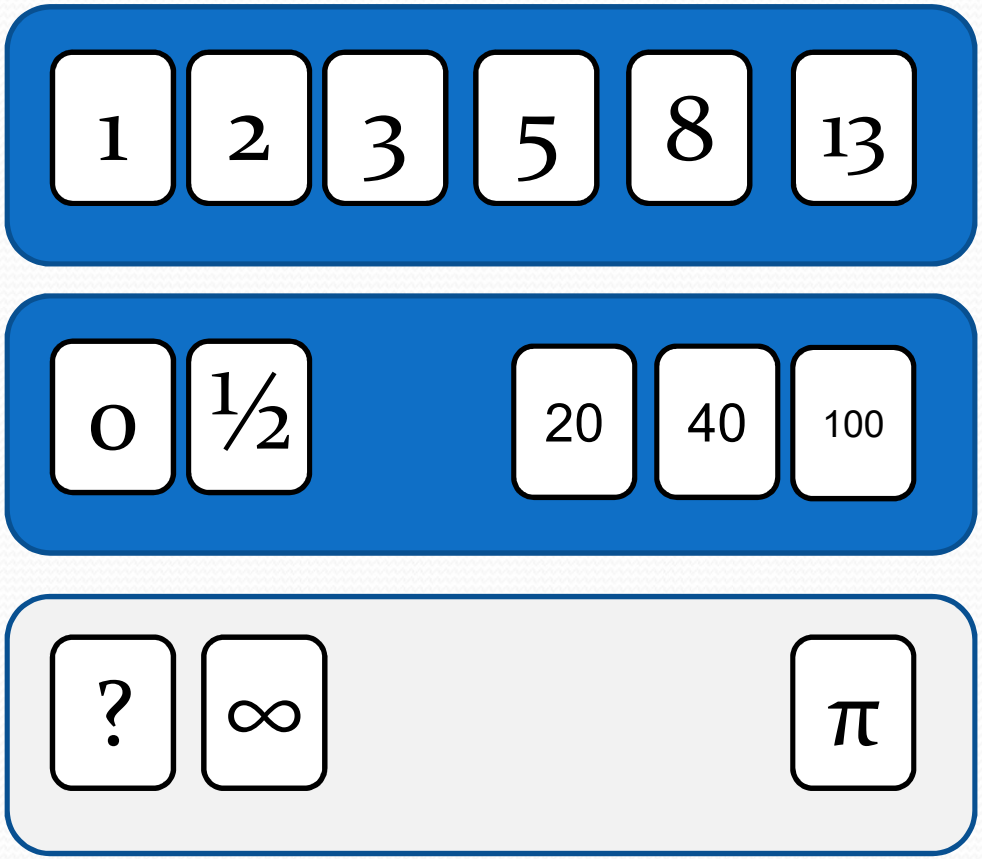
Backlog Grooming

- Add Story
  - Collect actors, objectives and acceptance
- Remove Story
  - Log reasons for removal
- Split Story
  - Transform one story in two or more stories
- Merge Stories
  - Transform a collection of stories in a single story
- Prioritize Story
  - Change relative story priority
- Move story Between backlogs
  - From Product backlog to Release Backlog
  - From Release A to Release B

Size Estimation



Define Story Scale



- Define Scale
  - Fixed
  - Discrete
- Elicit Scope Size
  - From Team
    - Planning Poker
  - From Requirements
    - Points per …
    -

Planning Poker

- Repeat for each story
  1. Product Owner explains story
  2. Team members can ask questions to PO
  3. Team members select size secretly
     - No Anchoring , No Dialog
  4. Team members show selections simultaneously
     - If the same number is selected that is the story size
     - Else , members select size secretly again
       - Team members show selections simultaneously
         - If The same number is selected that is the story size
         - Else, members with higher and lower points explain selections. Resume point 2.

Estimate With Planning Poker

- As a Hotel Guest I want to create a Reservation for a period of time. Receive confirmation by email
- As an Hotel Clerk I want to list free Rooms in a range of dates, in order to create a VIP Reservation.
- As an anonymous user I want to login providing my username and password.
- As an anonymous hotel guest I want to create a login account, in order to manage my reservations.
- As a hotel guest I want to access my room's bill.

Preparing a Release

1. Define Goals for the release
2. Define Number of Sprints
   - This indirectly define schedule
3. Elicit stories that achieve the Goal
   - Put them in the Release Backlog
4. Estimate Size (if absent)
5. Estimate Value (if absent)
6. (Re)Prioritize
   - Move to next release if necessary
   - Move to backlog if no release can be defined
7. Adjust Goal(s) if necessary (resume 3)

Preparing a Sprint

1. PO Defines Goals for the Sprint
2. Team Defines Sprint Budget
   - Justifications can be offered
3. PO adds stories to sprint until budget is reached
   - Cannot explode budget
   - PO revisits stories goals and acceptance criteria
4. PO adds slack stories
   - 2 or 3 points worth depending on sprint size and previous velocities

Sprint Budget

- Consider Number of Team members
- Consider Each members Allocation (full time, part-time)
  - Team members can work in other projects
  - Consider vacations
  - Consider personal matters (1 day off to see the doctor)
- Consider previous Velocity
  - Consider the velocity of previous sprints
  - Consider the variation of velocities
  - Do not consider using the best velocity (false optimism)

Estimate Possible Velocity/Budget

- When SP = 1 Ideal Day
  - Consider how many developers
  - Consider days in Period
  - Consider time allocation (A , in days)
  - Consider team member focus factor (F, in %)
    - Normal : 6/8 (= productive hours / available hours)
  - Possible Velocity = P * $\sum_{i=1}^{N} A_i F_i$

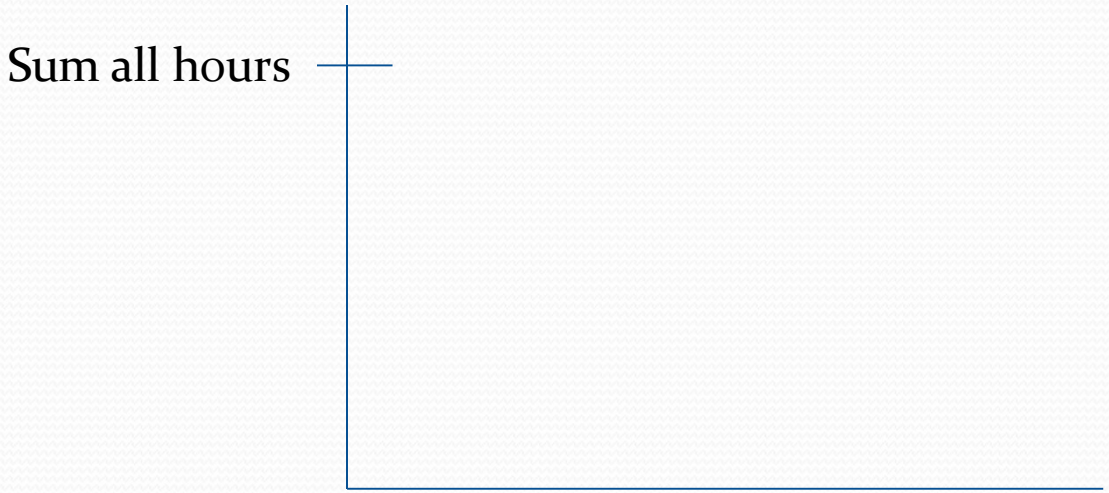
- When SP =/= 1 Ideal Day
  - Run some sprint and measure the real Velocity
  - Extrapolate the possible Velocity using the measurements

Velocity and Focus Factor

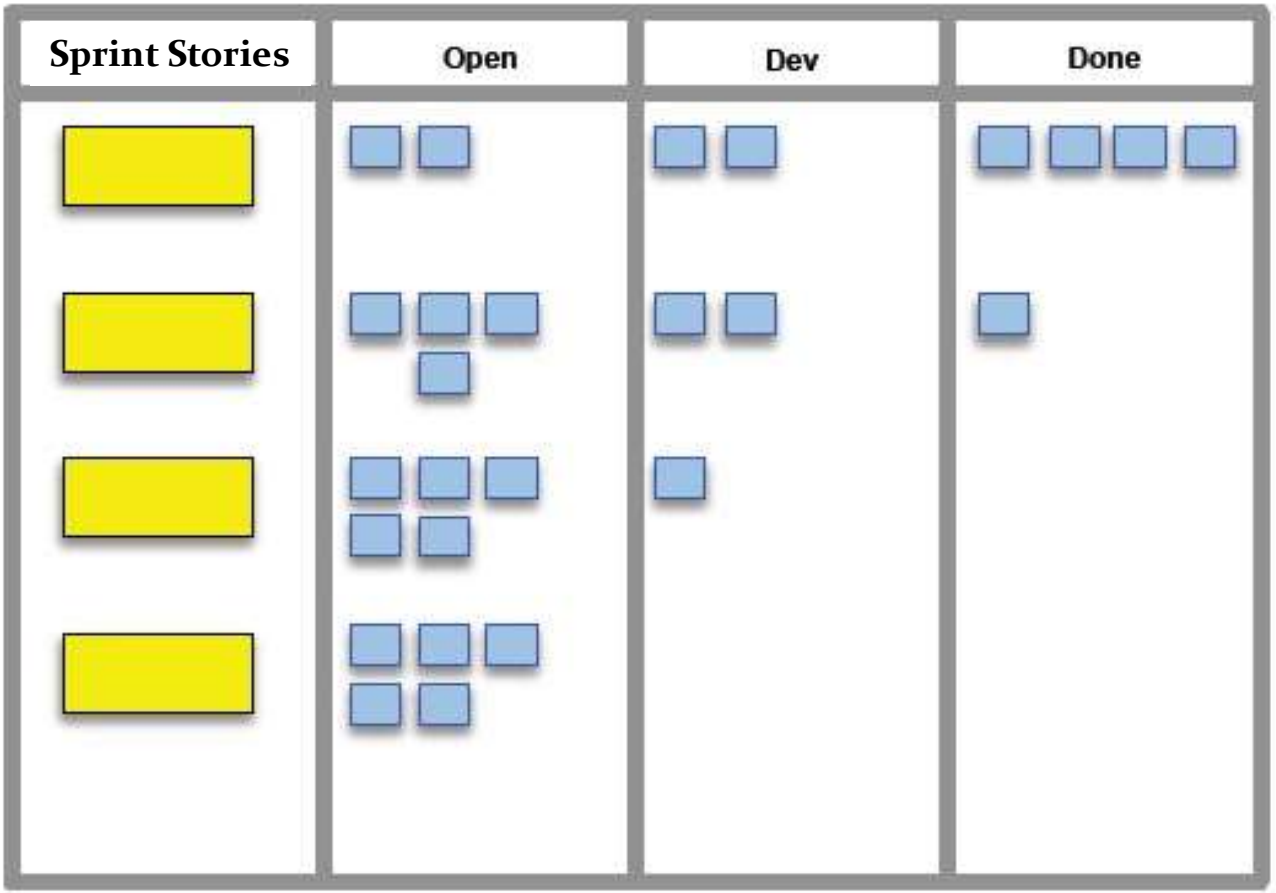
- Sprint Velocity – Sum of Sizes of the Stories marked "Done" at the end of a sprint
  - Definition of "Done" is predefined before a sprint and influences the size of the story.
- Sprint Budget – Quantity of Story Points the Team considers appropriated for the Sprint Backlog before it is defined
  - Normally the Velocity of previous Sprint
- Sprint Factor of Focus = Velocity / Budget
  - Should be 100% or more.
  - Less than 75% represents severe impediments

Sprint Work Breakdown

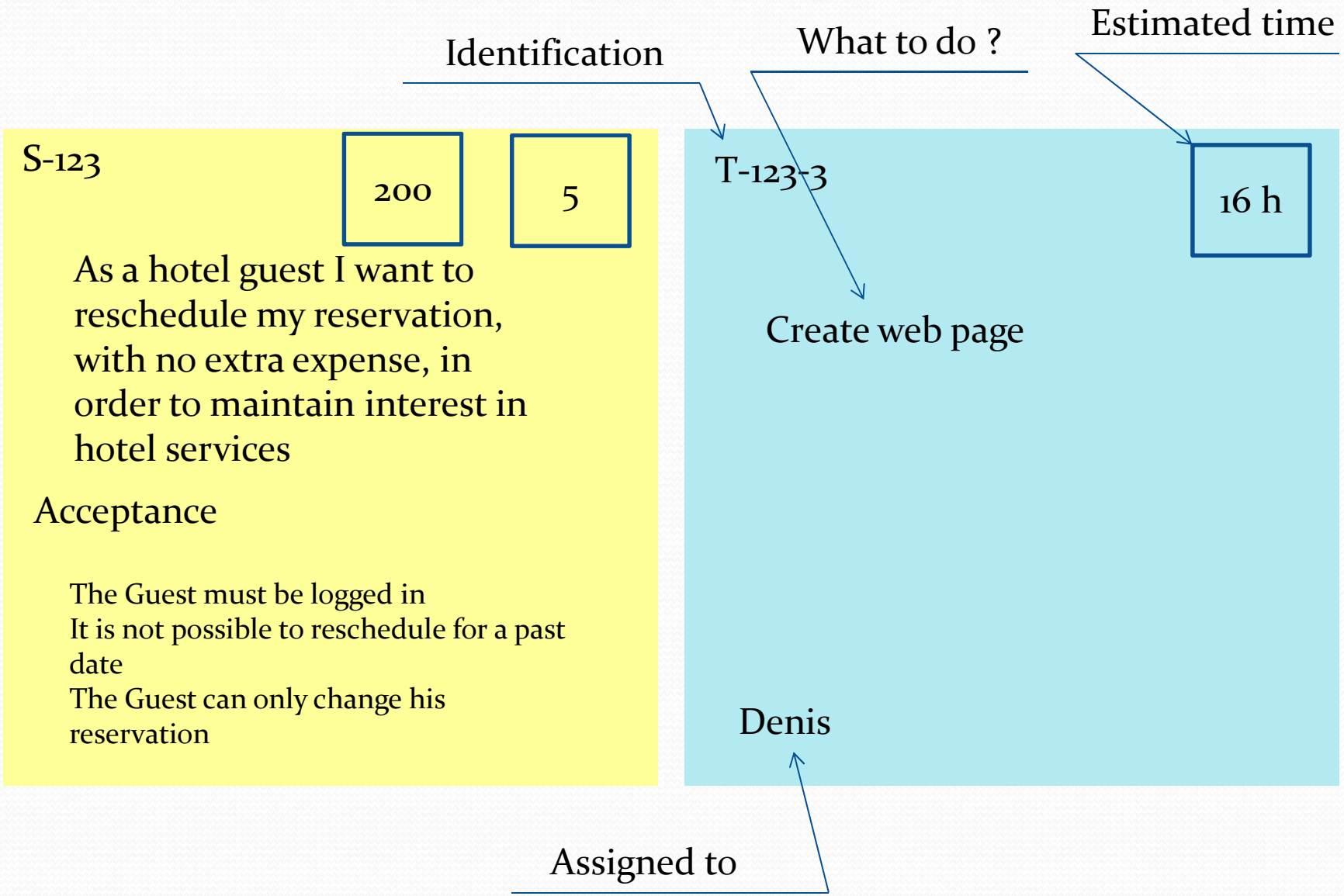
- Team members break stories in tasks
- Each task containing a duration estimation in hours
- Each task virtually assigned to a team member
- Sum of all task durations initializes Burndown Chart



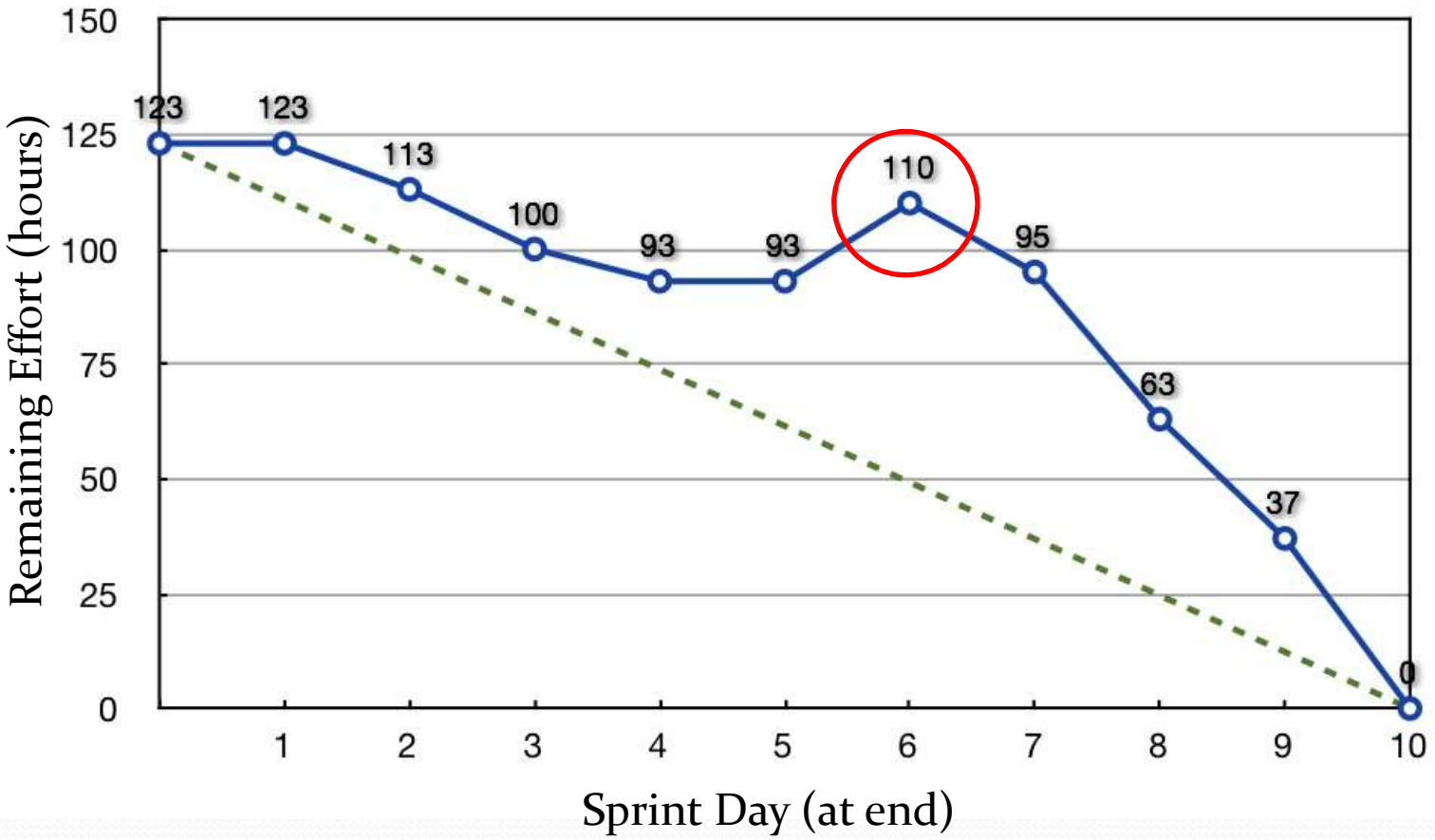
Kanban Board (Execution)



Task Taxomony



Burndown Chart (Execution)



Sprint Review

- Teams shows software working
  - According to acceptance criteria
  - Team shows impediments
- Product Owner marks stories as “Done”
  - The stories not done go back to the top of the backlog
- Measure Velocity
  - Velocity = Sum of Size of Stories Done
- Product Owner and Teams dialog about impediments
  - Impediments are put in the Impediments Backlog

Release Review

- Product Owner and Stakeholders
  - Dialog about Impediments backlog, Defect Backlog and new Stories
  - Dialog about Sprint Velocity and estimations for future sprints and impacts on the Release
    - Some grooming will be necessary at this point
  - Establish stories for Sprint N+2
    - The next sprint should already be prepared
    - The Release Review should be one or more sprints ahead

Final Notes

- Agile is Adaptive, but not instantaneously Adaptive
  - The time between when a story is created and implemented should be more that one sprint
- Agile is not *ad doc,* go horse , cowboy style
  - Involves measuring and using measurements to decide
  - Measurements are often dynamic, on the spot or automated
- Quality is the main drive of sustainability
  - Value must be delivered often, but that is not possible if defects take all the space in the backlog
- Agile is not easy
  - PO must constantly communicate with Stakeholders, not only at the end of a sprint. The PO must be several step ahead of Team. Story supply must be permanent.
  - Team should be self-managed and that is a cultural challenge

References

- Agile Estimation and Planning
  Mike Cohn Boehm, Richard Turner, Prentice Hall
- Balancing Agile and Discipline
  Barry Boehm and Richard Turner – Addison-Wesley
- Software Requirements
  Karl E. Wiegers , Microsoft Press
- Software Requirement Patterns
  Stephen Withall, Microsoft Press
- Agile Software Management with Scrum
  Ken Schwaber, Microsoft Press
- CHAOS Report
  http://www.projectsmart.co.uk/docs/chaos-report.pdf
  http://www.csus.edu/indiv/v/velianitis/161/ChaosReport.pdf
- Agile Science
  http://www.agilescience.org/

References

- Principle of Least Astonishment
  http://en.wikipedia.org/wiki/Principle_of_least_astonishment
- The Clean Coder
  Robert C. Martin et al, Prentice Hall
- The Pragmmatic Programmer: From Journeyman to Master
  Andrew Hunt, David Thomas, Addison-Wesley
- What Does "Going Agile" Really Mean?
  http://www.solutionsiq.com/resources/agileiq-blog/bid/58901/What-Does-Going-Agile-Really-Mean
- Which Factors Affect Software Projects Maintenance Cost More?
  http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3610582/
- Agile Principles
  http://agilemanifesto.org/principles.html