estimate

SCEA 2010 EST06

estimate • analyze • plan • control

Estimating Issues Associated with Agile Development

Bob Hunt
Vice President, Services
Galorath Incorporated

SEER® by GALORATH

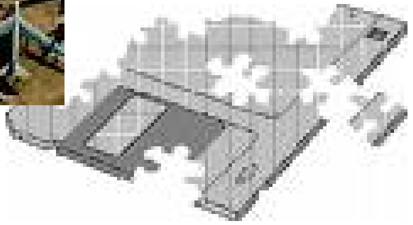
What Is Agile Software Dev?

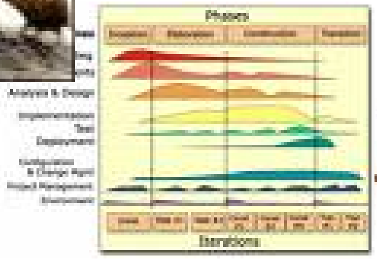
- In the late 1990's several methodologies began to get increasing public attention.
- Each had a different combination of old ideas, new ideas, and transmuted old ideas.
- But they all emphasized
  - close collaboration between the programmer and business experts;
  - face-to-face communication (as more efficient than written documentation);
  - frequent delivery of new deployable business value; tight, self-organizing teams;
  - and ways to craft the code and the team such that the inevitable requirements churn was not a crisis

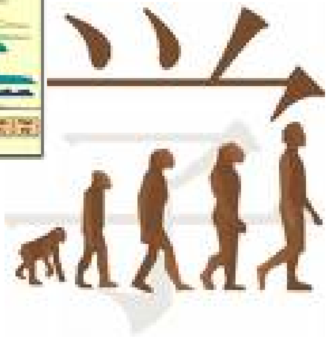
From: Agile Alliance

Methods We've Come to Love

- Waterfall
- Spiral
- Salvage
- Unified
- Evolutionary
- Reengineer
- Modified
- Etc







Welcome to Agile

- But what is it?
- *Depends on the flavor:*
  - LD - Lean Development,
  - ASD - Adaptive Software Development,
  - **Scrum,**
  - **XP - eXtreme Programming,**
  - Crystal methods,
  - FDD - Feature Driven Development,
  - DSDM - Dynamic Systems Development Method,
  - AUP – Agile Unified Process,

  …
  - TBD - you know what that means.

What Do They Have In Common?

- Agile projects are focused on key business values.
  - What does the client REALLY, REALLY – REALLY want.
  - Deliver what the client wants at the end of the project, not what the client wanted at the beginning of the project
- They all contain a Project initiation stage. (AKA Planning)
  - Project scope, constraints, objectives, risks are all officially (ah-hem) - Documented.
- Short (Very short) development of chunks of features/stores/requirements/wants/needs/desires.
  - (AKA – Sprints)
- Constant feedback.
  - The one place where can actually find a short meetings.
- Customer Participation is MANDATORY or no go!
  - Otherwise your flying blind – so what's the point.
- Refactoring – as in – do it again and this time get it right… or better… or WOW!

Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it.

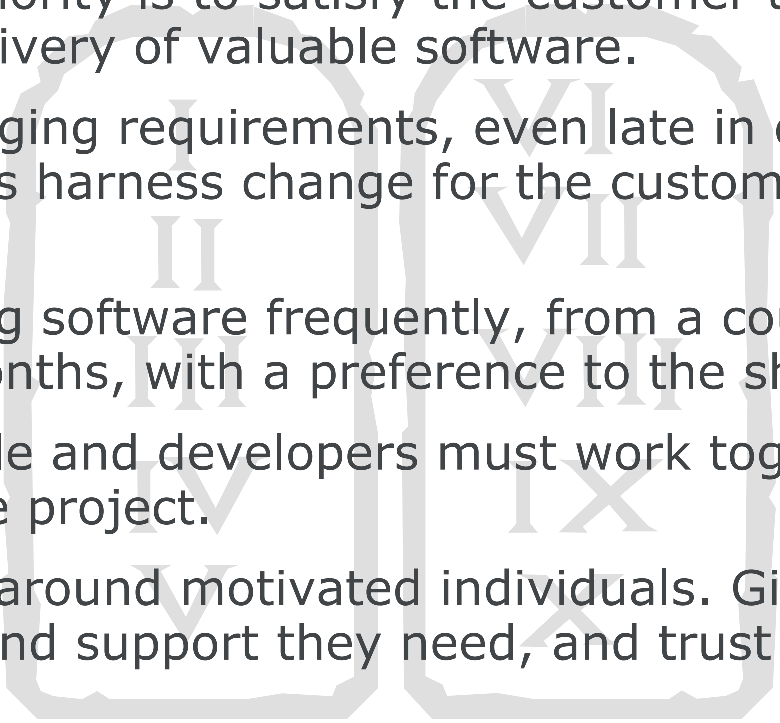
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

From: Agile Alliance

Principles behind the Manifesto

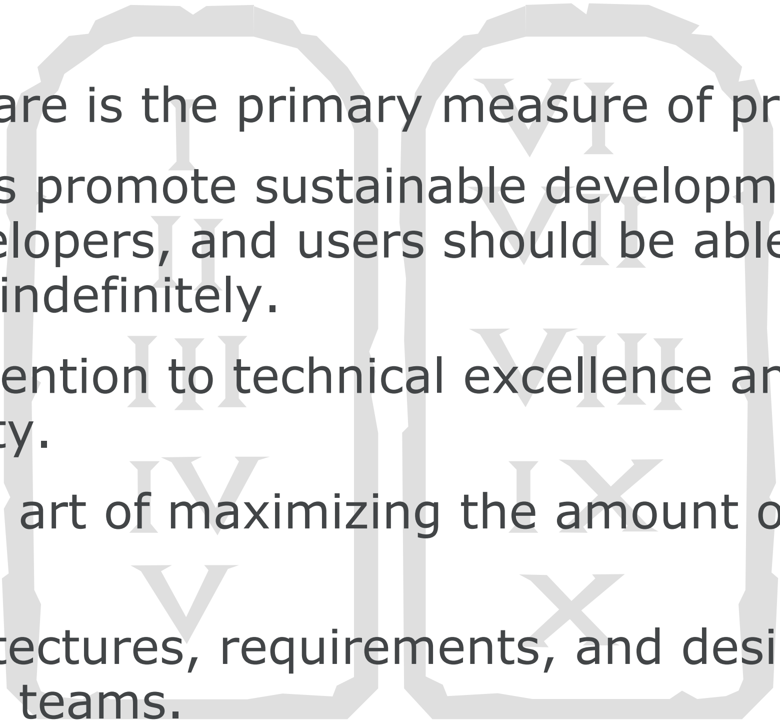


We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

From: Agile Alliance

Principles behind the Manifesto



- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

From: Agile Alliance

Common Myths About Agile

- Silver bullet / magic
  - *Actually very hard work!*
- Has no planning / documentation / architecture
  - *Just the minimum possible*
- Is undisciplined or a license to hack
  - *Disciplined, business driven work*
- Is new and unproven / just a fad / not being used by industry

  leaders
  - *Not anymore. Many large and small organizations using it*
- Only good for small projects
  - *Also used successfully on medium and large projects*

Radical Differences of Agile and Non Agile

Agile

- Prioritize by value
- Self-organizing teams
- Team focus
- Evolving requirements
- Change is natural

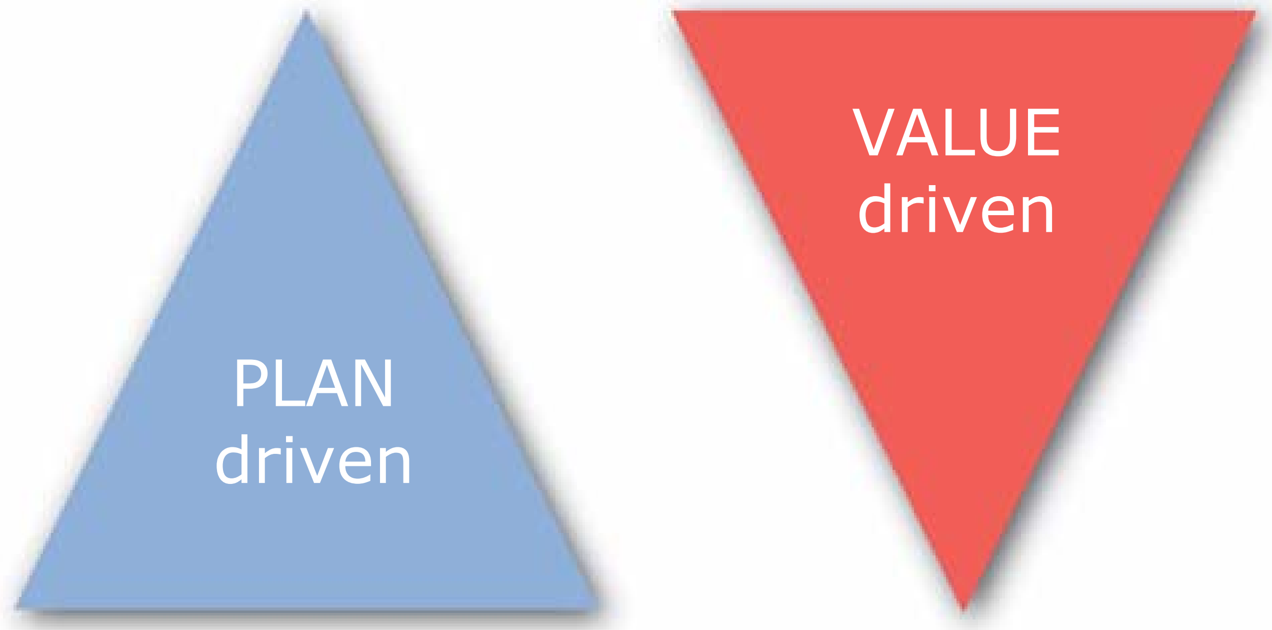
Non-agile

- Prioritize by *dependency*
- *Managed* resources
- *Project* focus
- *Frozen* requirements
- Change is *risky*

May work best when the project is more requirements driven than schedule driven

The Agile Paradigm Shift

	Waterfall	Agile
Fixed	Requirements	Resources Time
Estimated	Resources Time	Features



The plan creates cost/schedule estimates

Release themes and feature intent drive estimates

What do the Models Say?

Comparing Agile to Traditional Development Methods*

Development Type	Schedule Months	Effort Hours	Delivered Defects	Peak Staff	Functions per month
Agile Project	10.9	5145	13.00	4.51	8.81
Waterfall	12.1	6807	20.00	6.00	6.87
RUP	11.8	6020	16.00	4.91	7.77
Spiral	11.9	6066	19.00	4.95	7.71
Object Oriented	12.1	6543	19.00	5.40	7.15

Development Type	Schedule Months	Effort Hours	Delivered Defects	Peak Staff	Functions per month
Agile Project	-	-	-	-	-
Waterfall	12%	32%	54%	33%	78%
RUP	9%	17%	23%	9%	88%
Spiral	9%	18%	46%	10%	88%
Object Oriented	11%	27%	46%	20%	81%

* Client Server Platform, Transaction Processing Application, using Commercial High Standards
Project Size set to 250 Function Points. Calculated Using SEER for Software

Scrums and Sprints

- Scrum Size:
  - 1 to 10 people
- Sprint Length
  - 1 wk to 1 Month
- Story Points per Sprint
  - 6 to 9 Use Case Points per Sprint

Three Estimating Approaches

- Simplistic approach based on averages
- Moderate approach based on established “velocity”
- Adapting modern Models to Agile

Level 1 – Simplistic Approach

- The simplest model can be defined as the number of Sprints times the number of folks in a Scrum
- Scrum Size (SS) (1 to 10 people) times number of Sprints times length of Sprints (Sp time) (0.25 to 1.0 months)
- SS X Sp time X # Sprints

Level 2 – Structured Approach

1. Express Requirement in Story Points
2. Use a process to rank Story Points (small, medium, large; Fibonacci Series; Planning poker)
3. Estimate and/or document the velocity (# story points per time period) at which the scrum team can work
4. Spread the Sprints over time to develop time pha

Level 3 - Automated Model Approach

- The “Parameter” settings within automated models can be adjusted to estimate costs and schedule

Mapping Agile Principles to SEER

- Welcome changing requirements, even late in development …
- Deliver working software frequently, from a couple of weeks to a couple of months…
- Business people and developers must work together daily throughout the project …
- Build projects around motivated individuals…
- … self-organizing teams.

- Requirements
  - Volatility Hi/Hi+/Vhi
  - Complete Low
  - Definition Formality Vlow+/Low-/Low
  - After Baseline – YES
- Iterations
  - Start with Iterative Development Kbase
- Resources
  - Dedication Nom/Nom/Nom
  - Location Nom/Nom/Nom
- Staff Loading
  - Ehi+
- Personnel Capabilities
  - Analyst Hi/Hi+/Vhi-
  - Programmers Hi/Hi+/Vhi-

Mapping Agile Principles to SEER

- ..conveying information to and within a development team is face-to-face conversation.
- … team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
- Agile processes promote sustainable development.

→

- Development Support
  - Practices Use Hi+/Hi+/Hi+
  - Automated Tools Hi/Hi/Hi
  - Dev Sys Volatility Low/Low/Low
  - Process Volatility Low/Low/Low

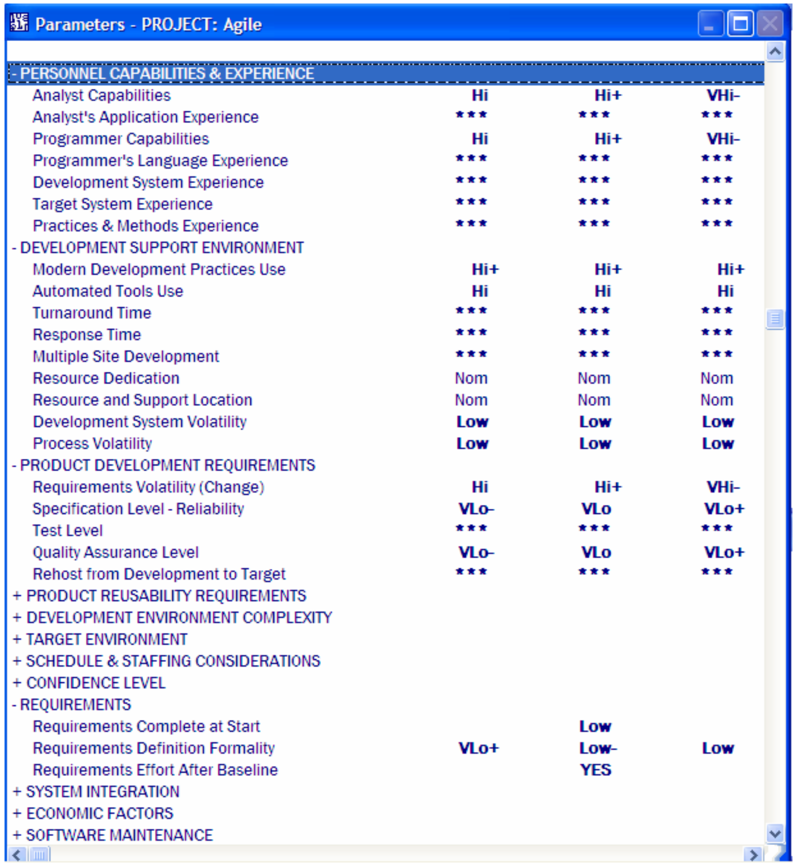
- Working software is the primary measure of progress.
- Continuous attention to technical excellence and good design

→

- Quality Assurance
  - Level VLow-/VLow/VLow+

Agile Class Kbase

- Create an Agile "Class" Kbase with Recommended Setting
- Modify Class to Fit Organization
  - Assess teams interactivity and motivation
  - Teams familiarity process
  - What is the real role of QA
  - Do they really have everyone in place
- Revisit the Class after one or two Iterations


Parameters - PROJECT: Agile

- PERSONNEL CAPABILITIES & EXPERIENCE			
Analyst Capabilities	Hi	Hi+	VHi-
Analyst's Application Experience	***	***	***
Programmer Capabilities	Hi	Hi+	VHi-
Programmer's Language Experience	***	***	***
Development System Experience	***	***	***
Target System Experience	***	***	***
Practices & Methods Experience	***	***	***
- DEVELOPMENT SUPPORT ENVIRONMENT			
Modern Development Practices Use	Hi+	Hi+	Hi+
Automated Tools Use	Hi	Hi	Hi
Turnaround Time	***	***	***
Response Time	***	***	***
Multiple Site Development	***	***	***
Resource Dedication	Nom	Nom	Nom
Resource and Support Location	Nom	Nom	Nom
Development System Volatility	Low	Low	Low
Process Volatility	Low	Low	Low
- PRODUCT DEVELOPMENT REQUIREMENTS			
Requirements Volatility (Change)	Hi	Hi+	VHi-
Specification Level - Reliability	VLo-	VLo	VLo+
Test Level	***	***	***
Quality Assurance Level	VLo-	VLo	VLo+
Rehost from Development to Target	***	***	***
+ PRODUCT REUSABILITY REQUIREMENTS			
+ DEVELOPMENT ENVIRONMENT COMPLEXITY			
+ TARGET ENVIRONMENT			
+ SCHEDULE & STAFFING CONSIDERATIONS			
+ CONFIDENCE LEVEL			
- REQUIREMENTS			
Requirements Complete at Start		Low	
Requirements Definition Formality	VLo+	Low-	Low
Requirements Effort After Baseline		YES	
+ SYSTEM INTEGRATION			
+ ECONOMIC FACTORS			
+ SOFTWARE MAINTENANCE			

Modify Activity Naming Scheme

- Out of the Box

Activity	Weight
– Sys Req.	1%
– SW Req.	4%
– Prel. Design	9%
– Det. Design	15%
– Code/Unit Test	24%
– Integ. Testing	28%
– Program Test	3%
– Sys IOT&E	15%

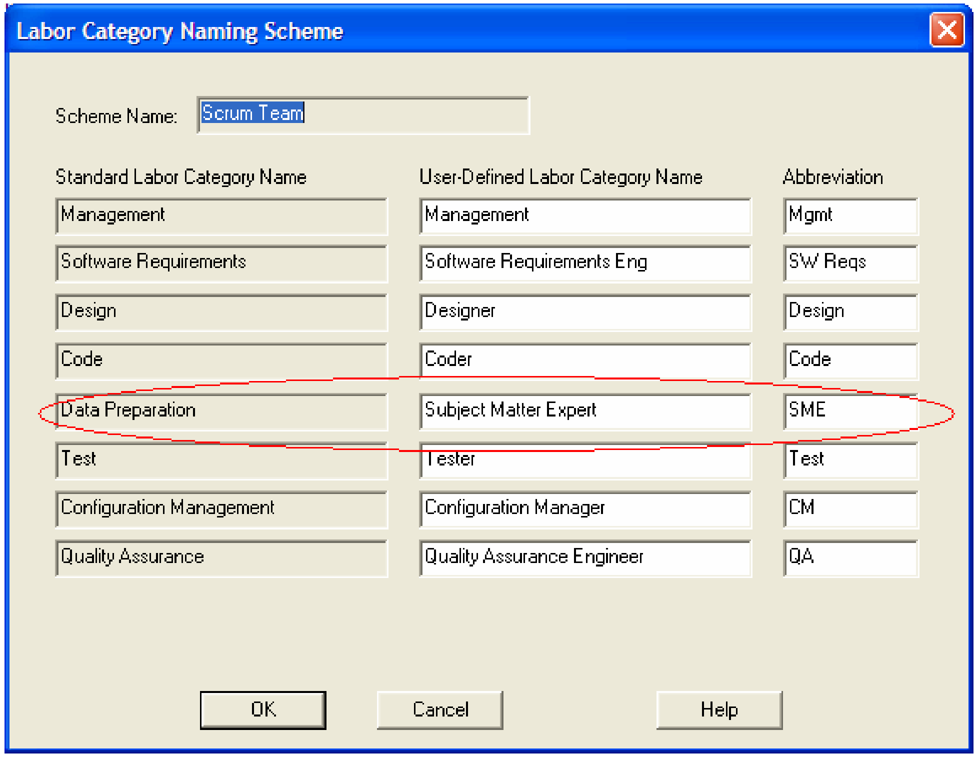
- Agile

Activity	Weight
– Planning	1%
– Arch/Design	4%
– Test Des/Dev	9%
– Feature Coding	15%
– Unit Testing	24%
– Integ. Testing	28%
– Delivery Review	3%
– Sys Test/Release	15%

Change Activities Names To Reflect <u>Their</u> Agile Process
Different Agile Methods Have Different Names

Modify Labor Category Naming

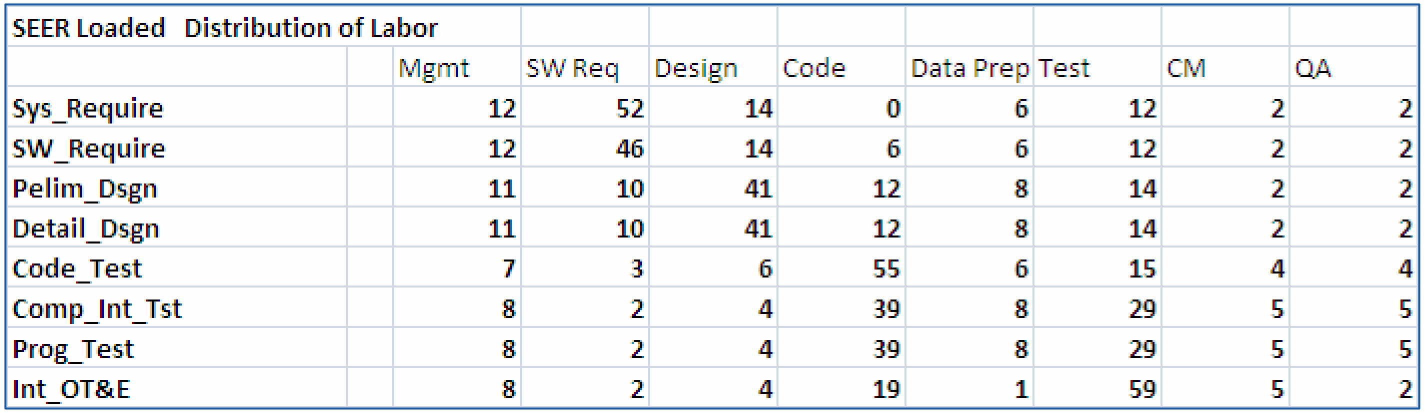
- Most Labor Category Names are the Same
- Change Data Manager to SME
  - The SME is an integral part of the team
  - Need to account for the SME hours


Labor Category Naming Scheme

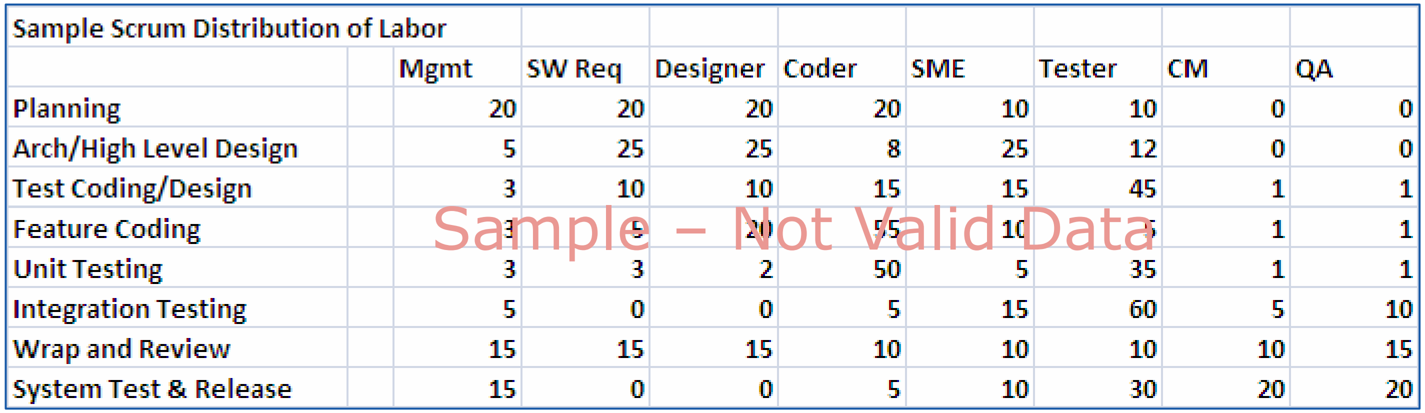
Scheme Name: Scrum Team

Standard Labor Category Name	User-Defined Labor Category Name	Abbreviation
Management	Management	Mgmt
Software Requirements	Software Requirements Eng	SW Reqs
Design	Designer	Design
Code	Coder	Code
Data Preparation	Subject Matter Expert	SME
Test	Tester	Test
Configuration Management	Configuration Manager	CM
Quality Assurance	Quality Assurance Engineer	QA

Identify Labor Distribution



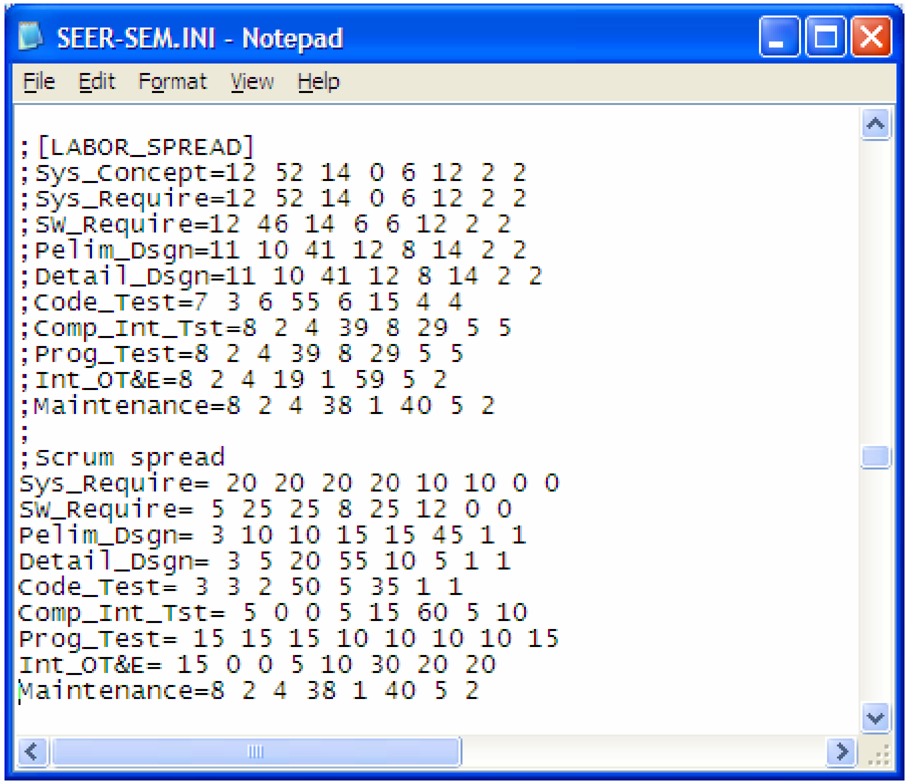
SEER Loaded Distribution of Labor	Mgmt	SW Req	Design	Code	Data Prep	Test	CM	QA
Sys_Require	12	52	14	0	6	12	2	2
SW_Require	12	46	14	6	6	12	2	2
Pelim_Dsgn	11	10	41	12	8	14	2	2
Detail_Dsgn	11	10	41	12	8	14	2	2
Code_Test	7	3	6	55	6	15	4	4
Comp_Int_Tst	8	2	4	39	8	29	5	5
Prog_Test	8	2	4	39	8	29	5	5
Int_OT&E	8	2	4	19	1	59	5	2



Sample Scrum Distribution of Labor	Mgmt	SW Req	Designer	Coder	SME	Tester	CM	QA
Planning	20	20	20	20	10	10	0	0
Arch/High Level Design	5	25	25	8	25	12	0	0
Test Coding/Design	3	10	10	15	15	45	1	1
Feature Coding	3	5	20	55	10	5	1	1
Unit Testing	3	3	2	50	5	35	1	1
Integration Testing	5	0	0	5	15	60	5	10
Wrap and Review	15	15	15	10	10	10	10	15
System Test & Release	15	0	0	5	10	30	20	20

Simple Excel Spreadsheet To Redistribute Labor by Category and Process Activity

Modify the SEER-SEM.ini File



```
;[LABOR_SPREAD]
;Sys_Concept=12 52 14 0 6 12 2 2
;Sys_Require=12 52 14 0 6 12 2 2
;SW_Require=12 46 14 6 6 12 2 2
;Pelim_Dsgn=11 10 41 12 8 14 2 2
;Detail_Dsgn=11 10 41 12 8 14 2 2
;Code_Test=7 3 6 55 6 15 4 4
;Comp_Int_Tst=8 2 4 39 8 29 5 5
;Prog_Test=8 2 4 39 8 29 5 5
;Int_OT&E=8 2 4 19 1 59 5 2
;Maintenance=8 2 4 38 1 40 5 2
;
;Scrum spread
Sys_Require= 20 20 20 20 10 10 0 0
SW_Require= 5 25 25 8 25 12 0 0
Pelim_Dsgn= 3 10 10 15 15 45 1 1
Detail_Dsgn= 3 5 20 55 10 5 1 1
Code_Test= 3 3 2 50 5 35 1 1
Comp_Int_Tst= 5 0 0 5 15 60 5 10
Prog_Test= 15 15 15 10 10 10 10 15
Int_OT&E= 15 0 0 5 10 30 20 20
Maintenance=8 2 4 38 1 40 5 2
```

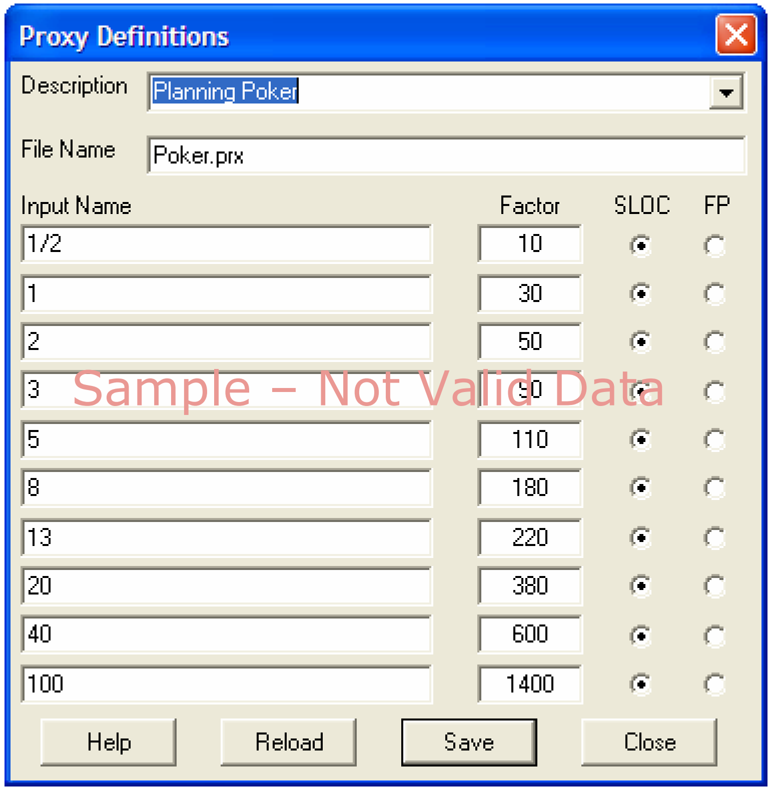
- Identify the New Labor Distribution
- Open the SEER-SEM.INI File
- Comment out the Old Distribution with ";"
- Insert the new Distribution
- Save INI File
- Restart SEER-SEM

Building an Agile Proxy

- Different Front Ends to Agile
  - The "requirements" are collected using various methods. Here are just a few examples:
    - Features
    - User Stories
    - Classes
    - Use Cases
    - Analogies
- Whatever Is Used – Quantify the Effort
  - Example: How much effort is needed to BUILD one average size "User Story"
  - A Popular Approach to Estimating Effort for a User Story or Feature is called "Planning Poker"

Example: Planning Poker Proxy

- Individual stories are presented for estimation.
  - After a period of discussion, each participant chooses from his own deck the numbered card that represents his estimate of how much work is involved in the story under discussion.
  - All estimates are kept private until each participant has chosen a card. At that time, all estimates are revealed and discussion can begin again.
- Build a Proxy to Map to the “Plan Poker” distribution.


Proxy Definitions

Description: Planning Poker

File Name: Poker.prx

Input Name	Factor	SLOC	FP
1/2	10	●	○
1	30	●	○
2	50	●	○
3	80	●	○
5	110	●	○
8	180	●	○
13	220	●	○
20	380	●	○
40	600	●	○
100	1400	●	○

http://www.planningpoker.com/

Wrap Up - Estimating Agile

- Remember – Emphasis is on delivered “stuff” not effort or even schedule.
  - Whatever’s not finished is moved to next Iteration
- Start with the Iterative Method then load the Agile Class
  - Keep the Class consistent with the team
- Change naming schemes to match method
  - Change both the Activity and applicable Labor names
- Calculate and redistribute labor by activity
  - This data is available from project metrics
- Build a proxy to estimate effort
  - Agile teams are used to measuring effort per “XXX”

Final Note

- When building an estimate for an Agile Method One needs to follow one important rule:

  - Stay "Agile"

References

- Becoming Agile in an imperfect world; Grteg Smith and Ahmed Sidky; ©2009
- Agile Estimating and Planning; Mike Cohn; © 2009
- Succeeding with Agile Software Development Using Scrum; Mike Cohn; © 2009

For more information:

Please contact me at:

Bob Hunt

Bhunt@galorath.com

703-201-0651

www.galorath.com

Designed to Meet the Challenges of Today's IT Projects