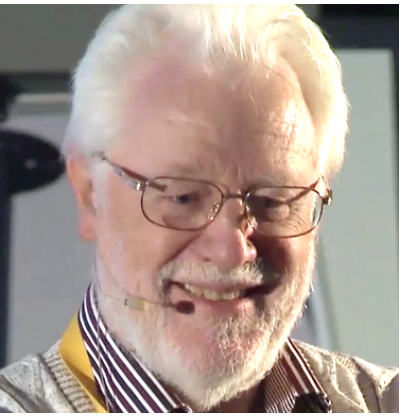
The GREEN WEEK:
Agile *Technical Debt* **Engineering** beats 'Refactoring'



Tom Gilb

Tom @ Gilb . Com

www.Gilb.com

10 Minute Lightning Talk, 5 Nov 2013

Smidig 2013

Technical debt

From Wikipedia, the free encyclopedia

Technical debt

consequences of poor software architecture and software development within a codebase.

Causes of technical debt include

① **Business pressures**
② **Lack of process or understanding**
③ **Lack of building loosely coupled components**,
④ **Lack of test suite**,
⑤ **Lack of documentation**,
⑥ **Lack of collaboration**
⑦ **Parallel**
⑧ **Delayed Refactoring**

Conventional Refactoring

	Technique	Description
1	Code Refactoring (clean-up)	It is intended to remove the unused code, methods, variables etc. which are misleading.
2	Code Standard Refactoring	It is done to achieve quality code.
3	Database Refactoring (clean-up)	Just like code refactoring, it is intended to clean or remove the unnecessary and redundant data without changing the architecture.
4	Database schema and design Refactoring	This includes enhancing the database schema by leaving the actual fields required by the application.
5	User-Interface Refactoring	It is intended to change the UI without affecting the underlying functionality.
6	Architecture Refactoring	It is done to achieve modularization at the application level.

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In agilerecord.com Nov 1 2013

Impact Software Qualities

- "Importantly, the underlying objective behind refactoring is to give thoughtful consideration and **improve** some of the **essential *<Quality> attributes*** of the software."

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In AGILERECORD.COM NOVEMBER 1 2013

Impact Software Qualities

"Key Benefits of Refactoring

From a system/application standpoint, listed below are summaries of the key benefits that can be achieved seamlessly when implementing the refactoring process in a disciplined fashion:

① Firstly, it improves the overall software **extendability**.
② Reduces and optimizes the code **maintenance cost**.
③ Facilitates highly standardized and organized code.
④ Ensures that the system architecture is improved by retaining the behavior.
⑤ Guarantees three essential attributes: **readability**, **understandability**, and **modularity** of the code.
⑥ Ensures constant improvement in the **overall quality** of the system. "

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In agilerecord.com Nov 1 2013

Impact Software Qualities

"Key Benefits of Refactoring

From a syst[illegible] summaries [illegible]essly when imple[illegible]d fashion:

① Firstl[illegible]y.

② Redu[illegible]

③ Facili[illegible]

④ Ensu[illegible] retain[illegible]

⑤ Guar[illegible] **understandability**, and **modularity** of the code.

⑥ Ensures constant improvement in the **overall quality** of the system. "

No numbers given to support this

Refactoring – to Sustain Application Development Success in Agile Environments
by Narayana Maruvada
In agilerecord.com Nov 1 2013

There is a smarter way

- But it means we have to become **real** software *engineers*,

- Not just- - - *softcrafters**

- * coders, devs, programmers.
  - Term coined in
  - "Principles of Software Engineering Management", 1988, Gilb

The Confirmit Case Study 2003-2013

Their product = confirmit

Chief Storyteller =

Trond Johansen

See this case at www.gilb.com

Papers/Cases/Slides, Gilb Library,

value slide w… http://www.gilb.com/tiki-download_file.php?fileId=152

paper What's wrong with Agile… http://www.gilb.com/tiki-download_file.php?fileId=50

Paper on Confirmit http://www.gilb.com/tiki-download_file.php?fileId=32

And (IEEE Software, Fall 2006) by Geir K Hanssen, SINTEF

Customer Successes in Corporate Sector as of 2003 after 8 years legacy code

We gave them a 1 day briefing on our Evo method and Planguage

That's all they needed to succeed!

They were Real engineers

Shift: from Function to Quality

- **Our new focus is on the day-to-day operations of our Market Research users,**
  - **not a list of features that they might or might not like. 50% never used!**
  -
- **After one week we had defined more or less all the requirements for the next version (8.5) of Confirmit.**

EVO Plan Confirmit 8.5 in **Evo Step Impact *Measurement***
4 product areas were attacked in all: **25 USER Qualities** concurrently, one quarter of a year. Total development staff = 13

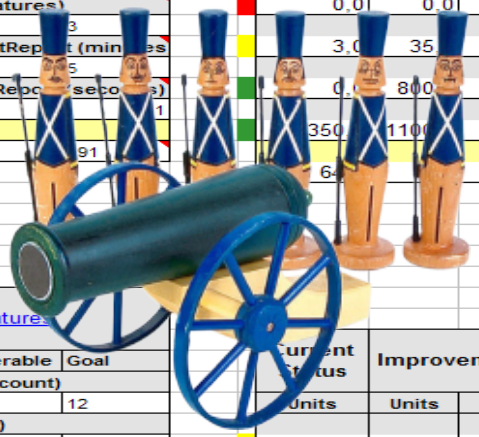
Impact Estimation Table: Reportal codename "Hyggen"

9

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportRep[illegible] (min[illegible]s		
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewRep[illegible] seco[illegible]s)		
1,0	14,0	100,0	15		1
			Development resources		
203,0			0		91

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,0	35	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0,[illegible]	800	100,0	800	0	0
			Runtime.Concurrency (number of users)		
350	1100	146,7	150	500	1000
64			Development resources		
			0		84

8



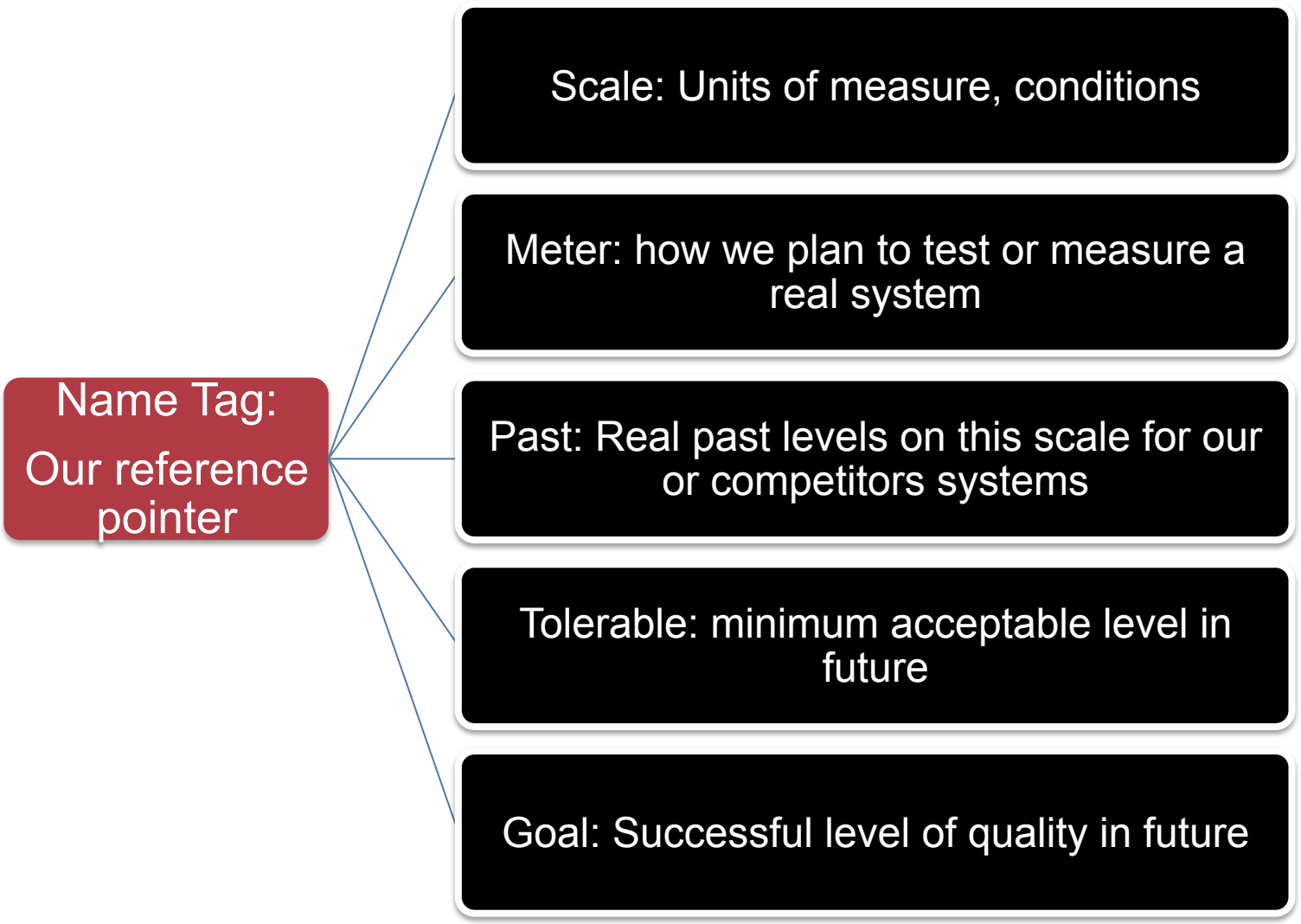
3

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

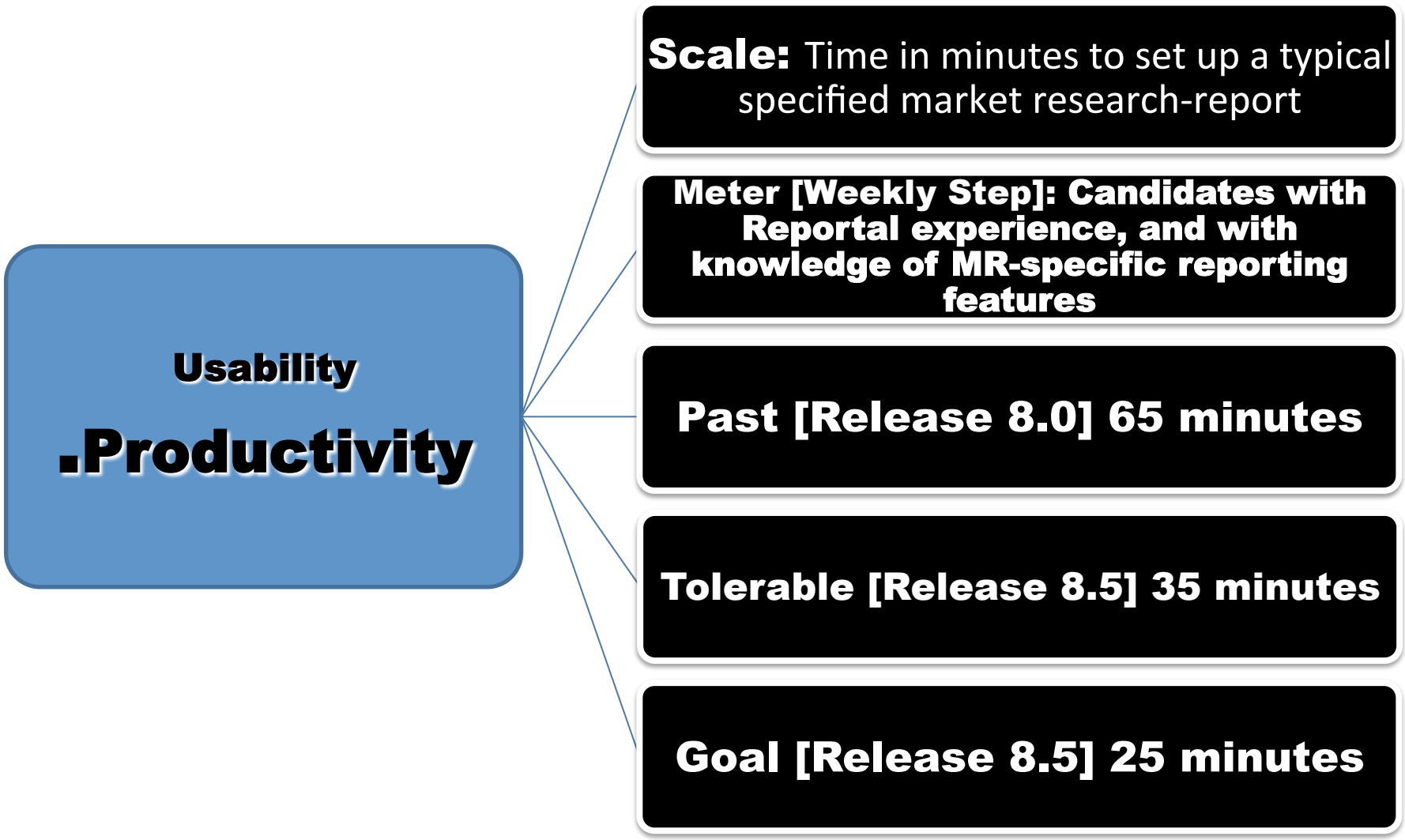
3

Current [illegible]tus	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		48

Each Quality Requirement has this 'Planguage' format



Each Quality Requirement has this 'Planguage' format: Real Example



Real sample of incremental engineering of 1/4th of the 25 qualities at end of week 9 of 12, before world release

Trond Johansen

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Quantified Value Delivery Project Management in a Nutshell
Quantified Value Requirements, Design, Design Value/cost estimation, Measurement of Value Delivery, Incremental Project Progress to Date

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements		Goals			Step9			
3								Recoding			
4								ted impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal		%		%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Estimates

Weekly

Testing

Priority

Next week

Warning

metrics based

Cumulative

weekly

progress

metric

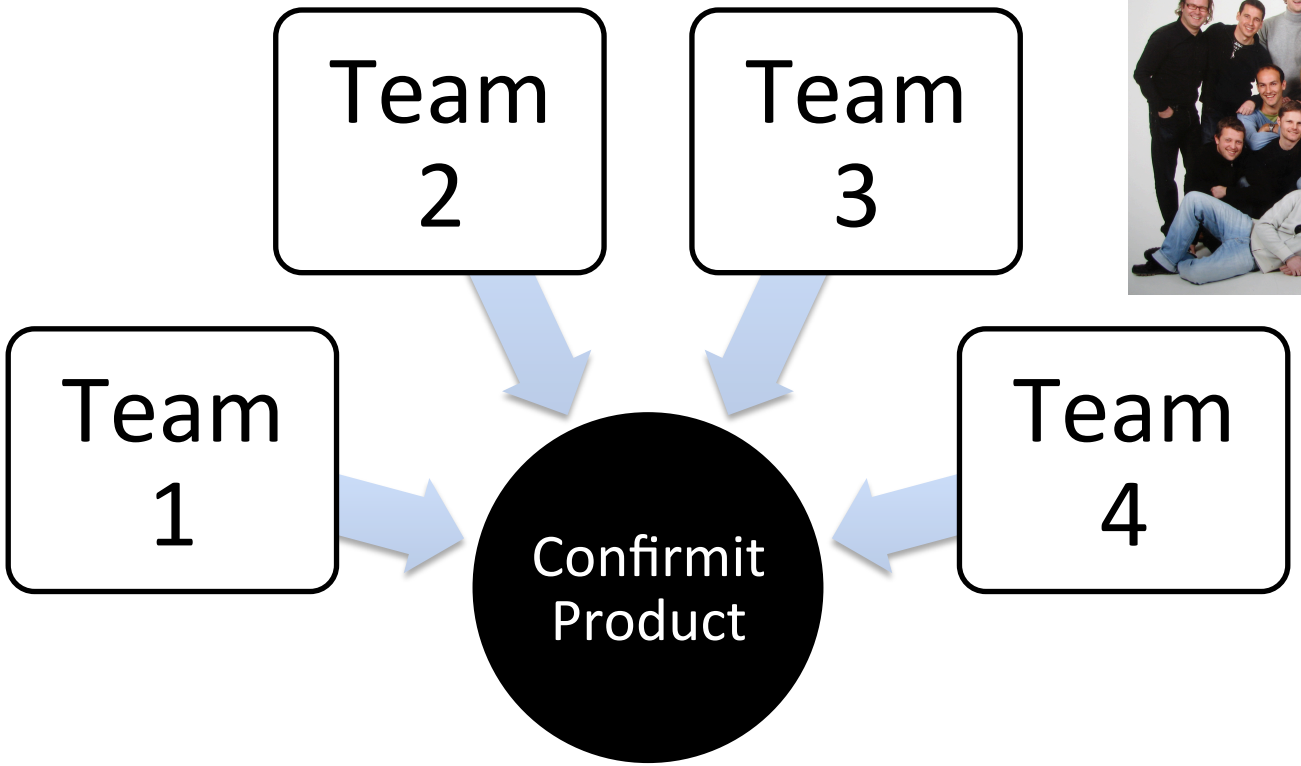
Constraint

Target

Concurrent Quantified 'Empowered Creativity' *

The Software Engineers can **use** ANY design that they *believe* delivers the planned value.

And *keep* what *really* works



* Empowered Creativity: Term coined by Trond Johansen, Confirmit, 2003

EVO Plan Confirmit 8.5 in **Evo Step Impact *Measurement***
4 product areas were attacked in all: **25 Qualities** concurrently, one quarter of a year. Total development staff = 13

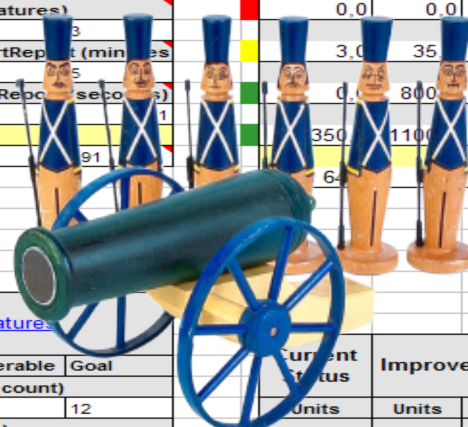
Impact Estimation Table: Reportal codename "Hyggen"

9

Current Status	Improvements		Reportal - E-SAT features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Intuitivness (%)		
75,0	25,0	62,5	50	75	90
			Usability.Consistency.Visual (Elements)		
14,0	14,0	100,0	0	11	14
			Usability.Consistency.Interaction (Components)		
15,0	15,0	107,1	0	11	14
			Usability.Productivity (minutes)		
5,0	75,0	96,2	80	5	2
5,0	45,0	95,7	50	5	1
			Usability.Flexibility.OfflineReport.ExportFormats		
3,0	2,0	66,7	1	3	4
			Usability.Robustness (errors)		
1,0	22,0	95,7	7	1	0
			Usability.Replacability (nr of features)		
4,0	5,0	100,0	8	5	3
			Usability.ResponseTime.ExportRep[illegible]t (min[illegible]es		
1,0	12,0	150,0	13	13	5
			Usability.ResponseTime.ViewRep[illegible] (sec[illegible]s)		
1,0	14,0	100,0	15	[illegible]	1
			Development resources		
203,0			0		91

Current Status	Improvements		Survey Engine .NET		
Units	Units	%	Past	Tolerable	Goal
			Backwards.Compatibility (%)		
83,0	48,0	80,0	40	85	95
0,0	67,0	100,0	67	0	0
			Generate.WI.Time (small/medium/large seconds)		
4,0	59,0	100,0	63	8	4
10,0	397,0	100,0	407	100	10
94,0	2290,0	103,9	2384	500	180
			Testability (%)		
10,0	10,0	13,3	0	100	100
			Usability.Speed (seconds/user rating 1-10)		
774,0	507,0	51,7	1281	600	300
5,0	3,0	60,0	2	5	7
			Runtime.ResourceUsage.Memory		
0,0	0,0	0,0		?	?
			Runtime.ResourceUsage.CPU		
3,[illegible]	35[illegible]	97,2	38	3	2
			Runtime.ResourceUsage.MemoryLeak		
0,[illegible]	800[illegible]	100,0	800	0	0
			Runtime.Concurrency (number of users)		
350[illegible]	110[illegible]	146,7	150	500	1000
			Development resources		
64[illegible]			0		84

8



3

Current Status	Improvements		Reportal - MR Features		
Units	Units	%	Past	Tolerable	Goal
			Usability.Replacability (feature count)		
1,0	1,0	50,0	14	13	12
			Usability.Productivity (minutes)		
20,0	45,0	112,5	65	35	25
			Usability.ClientAcceptance (features count)		
4,4	4,4	36,7	0	4	12
			Development resources		
101,0			0		86

Current Status	Improvements		XML Web Services		
Units	Units	%	Past	Tolerable	Goal
			TransferDefinition.Usability.Efficiency		
7,0	9,0	81,8	16	10	5
17,0	8,0	53,3	25	15	10
			TransferDefinition.Usability.Response		
943,0	-186,0	######	170	60	30
			TransferDefinition.Usability.Intuitiveness		
5,0	10,0	95,2	15	7,5	4,5
			Development resources		
2,0			0		48

3

Evo Weekly Value Delivery Cycle: Built on HP Evo

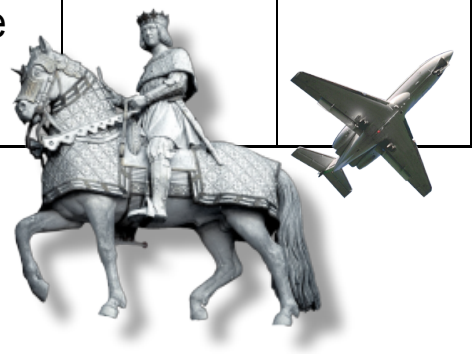
	Development Team	Users (PMT, Pros, Doc writer, other)	CTO (Sys Arch, Process Mgr) Petter M	QA (Configuration Manager & Test Manager) Trond J
Friday	✓ PM: Send Version N detail plan to CTO + prior to Project Mgmt meeting ✓ Attend Project Mgmt meeting: 12:00 ✓ Developers: Focus on genereal maintenance work, documentation.		✓ Approve/reject design & Step N ✓ Attend Project Mgmt meeting: 12-15	✓ Run final build and create setup for Version N-1. ✓ Install setup on test servers (external and internal) ✓ Perform initial crash test and then release Version N-1
Monday	✓ Develop test code & code for Version N	✓ Use Version N-1		✓ Follow up CI ✓ Review test plans, tests
Tuesday	✓ Develop Test Code & Code for Version N ✓ Meet with users to Discuss Action Taken Regarding Feedback From Version N-1	✓ Meet with developers to give Feedback and Discuss Action Taken from previous actions	✓ System Architect to review code and test code	✓ Follow up CI ✓ Review test plans, tests
Wednesday	✓ Develop test code & code for Version N			✓ Review test plans, tests ✓ Follow up CI
Thursday	✓ Complete Test Code & Code for Version N ✓ Complete GUI tests for Version N-2			✓ Review test plans, tests ✓ Follow up CI

Refactoring Day

Evo's impact on Confirmit product qualities 1st Qtr

- Only 5 highlights of the 25 impacts are listed here

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000



confirmit

Release 8.5

Initial Experiences and conclusions (TJ)

- **EVO has resulted in**
  - **increased motivation and**
  - **enthusiasm amongst developers,**
  - **it opens up for *empowered creativity***
- **Developers**
  - **embraced the method and**
  - **saw the value of using it,**
  - **even though they found parts of Evo difficult to understand and execute**

confirmit

Trond Johansen

TJ Conclusions -

- **"The method's positive impact on Confirmit product qualities has convinced us that**
  - **Evo is a better suited development process than our former waterfall process, and**
  - **we will continue to use Evo in the future.**
- **What surprised us the most was**

the method's power of focusing on delivering value for clients versus cost

Trond Johansen

ACTUAL RESULTS IN SECOND 12 WEEKS OF USING THE NEW METHOD

Evo's impact on Confirmit 9.0 product qualities

Product quality	Description	Customer value
Intuitiveness	Probability that an inexperienced user can intuitively figure out how to set up a defined Simple Survey correctly.	**Probability increased by 175%**
Productivity	Time in minutes for a defined advanced user, with full knowledge of 9.0 functionality, to set up a defined advanced survey correctly.	Time reduced by **38%**

Product quality	Description	Customer value
Productivity	Time (in minutes) to test a defined survey and identify 4 inserted script errors, starting from when the questionnaire is finished to the time testing is complete and is ready for production. (Defined Survey: Complex survey, 60 questions, comprehensive JScripting.)	Time reduced by **83%** and error tracking **increased by 25%**

MORE ACTUAL RESULTS IN SECOND 12 WEEKS OF USING THE NEW METHOD

Evo's impact on Confirmit 9.0 product qualities

Product quality	Description	Customer value
Performance	**Max number of panelists that the system can support without exceeding a defined time for the defined task, with all components of the panel system performing acceptable.**	**Number of panelists increased by 1500%**
Scalability	**Ability to accomplish a bulk-update of X panelists within a timeframe of Z sec.**	**Number of panelists increased by 700%**
Performance	**Number of responses a database can contain if the generation of a defined table should be run in 5 seconds.**	**Number of responses increased by 1400%**

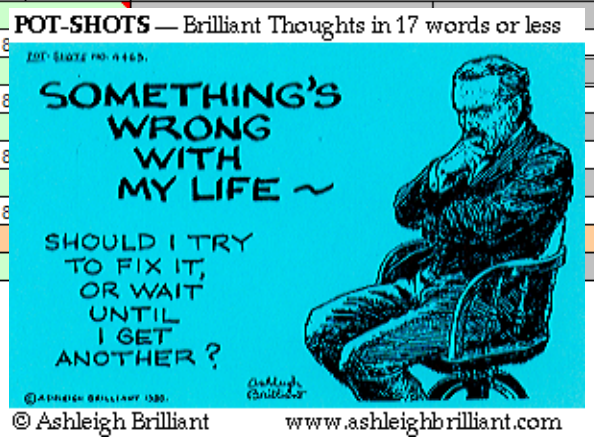
A bright idea: based on experience

- So, Confirmit was getting amazing results for the user, customer, and system level attributes they targeted
- And someone on the team realized…
  - What about us devs and testers
  - We are stakeholders too!
  - Refactoring (1 day a week) was NOT working well.
- Let us try to engineer the qualities that we need into the system
- The same way we engineer the user qualities into the system

Code quality – ”green” week, 2005
“Refactoring by Proactive Design Engineering!”

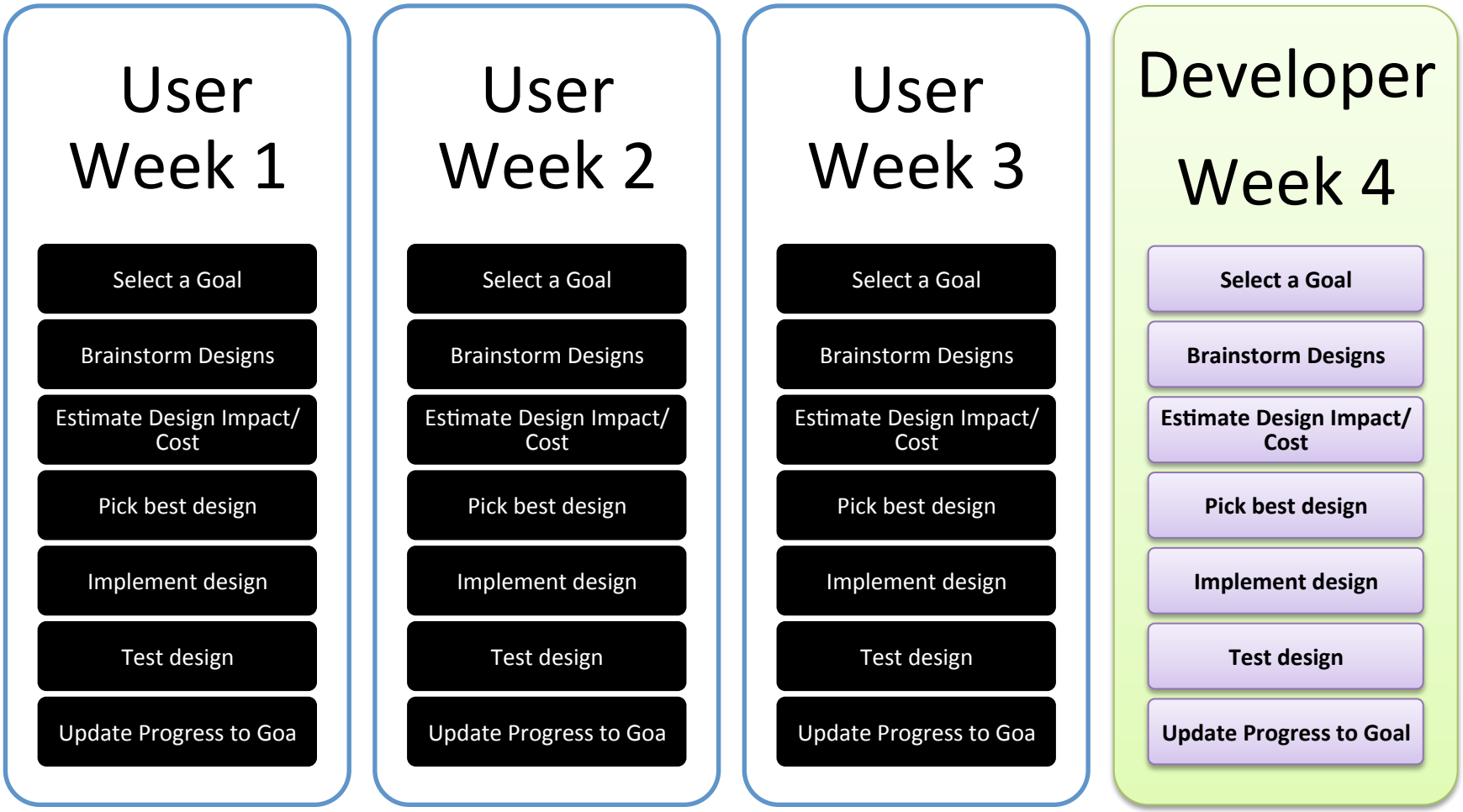
- **In these ”green” weeks, some of the deliverables will be less visible for the end users, but more visible for our QA department.**
- **We manage code quality through an Impact Estimation table. TJ**

Current Status		Improvement	Goals			Step 6 (week 14)		Step 7 (wee	
	Units		Past	Tolerable	Goal	Estimated Impact	Actual Impact	Estimated Impact	A
	100,0	100,0	0	80	100			100	
Speed									
	100,0	100,0	0	80	100	100	100		
Maintainability.Doc.Code									
	100,0	100,0	0	80	100	100	100		
InterviewerConsole									
NUnitTests									
	0,0	0,0	0	90	100				
PeerTests									
	100,0	100,0	0	90	100			100	
FxCop									
	0,0	10,0	10	0	0				
TestDirectorTests									
	100,0	100,0	0	90	100			100	
Robustness.Correctness									
	2,0	2,0	0	1	2	2	2		
Robustness.BoundaryConditions									
	0,0	0,0	0						
Speed									
	0,0	0,0	0						
ResourceUsage.CPU									
	100,0	0,0	100						
Maintainability.Doc.Code									
	100,0	100,0	0						
SynchronizationStatus									
NUnitTests									

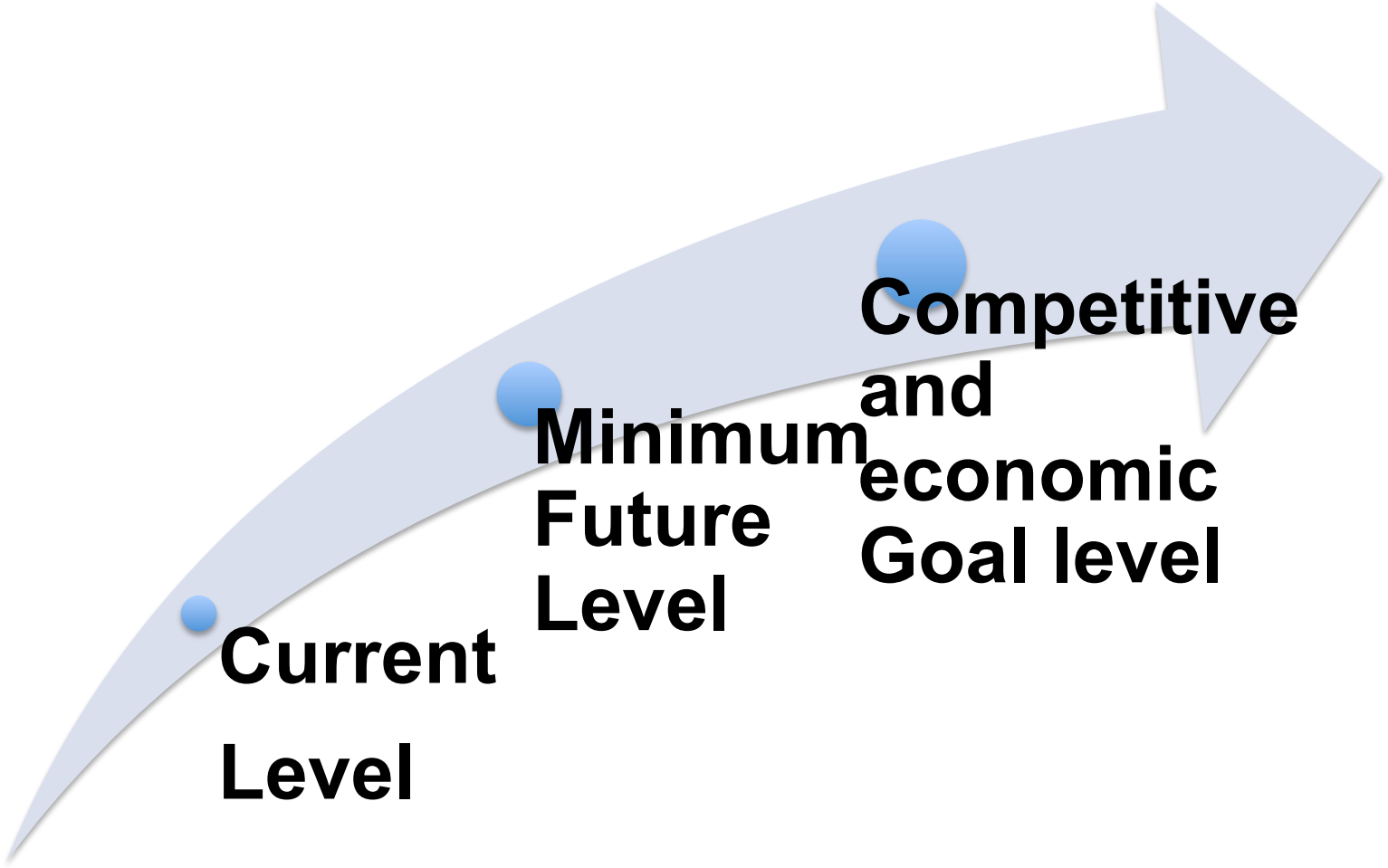


- Speed
- Maintainability
- Nunit Tests
- PeerTests
- TestDirectorTests
- Robustness.Correctness
- Robustness.Boundary Conditions
- ResourceUsage.CPU
- Maintainability.DocCode
- SynchronizationStatus

The Monthly 'Green Week'



Raising the Levels of Maintainability like ‘Mean Time To Fix a Bug’



Raising the Levels of Maintainability
Multiple Attributes of Technical Debt

Broader 'Maintainability' Concepts
ALL *quantified,* with a defined Scale of measure in CE-5

- Performance
  - Quality
    - Availability
      - Reliability
      - Maintainability
      - Integrity
        - Threat
        - Security
    - Adaptability
      - Flexibility
        - Connectability
        - Tailorability
          - Extendibility
          - Interchangeability
      - Upgradeability
        - Installability
        - Portability
        - Improveability

1. **The Conscious Design Principle**:

- "Maintainability must be *consciously* designed into a system:
  - failure to **design** to a set of levels of maintainability
  - means the **resulting maintainability** is both *bad* and *random*."
- © Tom Gilb (2008, INCOSE Paper)
  - http://www.gilb.com/tiki-download_file.php?fileId=138

The 'Maintainability' Generic Breakdown into Sub-problems

1. Problem Recognition Time.

   How can we reduce the time from bug actually occurs until it is recognized and reported?

2. Administrative Delay Time:

   How can we reduce the time from bug reported, until someone begins action on it?

3. Tool Collection Time.

How can we reduce the time delay to collect correct, complete and updated information to analyze the bug: source code, changes, database access, reports, similar reports, test cases, test outputs.

4. Problem Analysis Time.

   Etc. for all the following phases defined, and implied, in the Scale scope above.

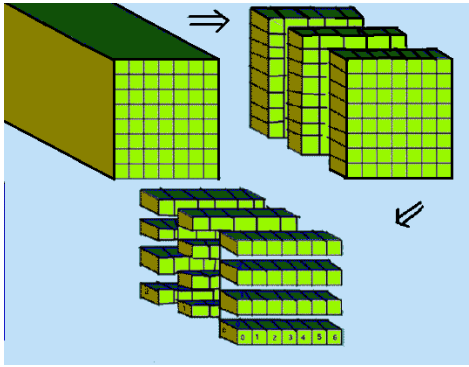
5. Correction Hypothesis Time

6. Quality Control Time

7. Change Time

8. Local Test Time

9. Field Pilot Test Time



10. Change Distribution Time

11. Customer Installation Time

12. Customer Damage Analysis Time

13. Customer Level Recovery Time

14. Customer QC of Recovery Time

Source: Competitive Engineering Ch 5
& Ireson (ed.) Reliability Handbook, 1966

An Example of Specifying 1 Attribute in 'Planguage'

Restore Speed:

***Type**: Software Quality Requirement.* ***Version**: 25 October 2007.*

***Part of**: Rock Solid Robustness*

***Ambition**: Should an error occur (or the user otherwise desire to do so), the system shall be able to restore the system to a previously saved state in less than 10 minutes. <-6.1.2 HFA.*

Scale: Duration from Initiation of Restore to Complete and verified state of a defined [Previous: Default = Immediately Previous]] saved state.

Initiation: defined as {Operator Initiation, System Initiation, ?}. Default = Any.

Goal [Initial and all subsequent released and Evo steps] 1 minute?

Fail [Initial and all subsequent released and Evo steps] 10 minutes. <- 6.1.2 HFA

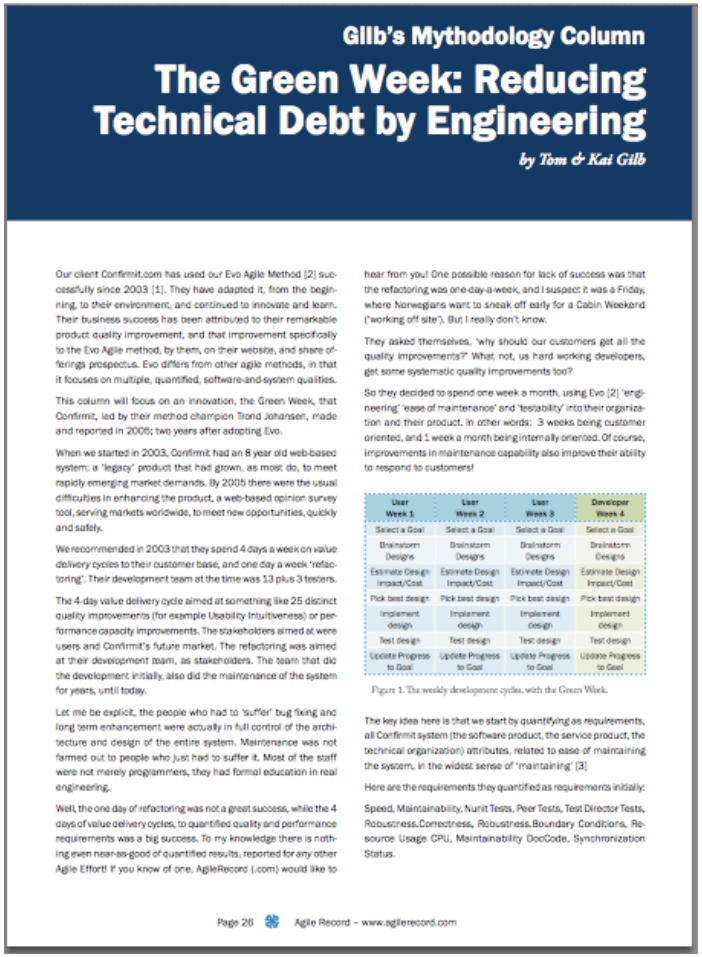
Catastrophe: 100 minutes.

Let's Vote

1. How many of you would **prefer** to keep doing conventional 'softcrafter' refactoring; even if the results were not measurable

2. How many of you think you **ought** to try to engineer measurable software maintainability results into your systems
   - Even if your boss is not smart enough to ask you, or support you doing it?

Further Reading: AgileRecord.com



Gilb's Mythodology Column

The Green Week: Reducing Technical Debt by Engineering

by Tom & Kai Gilb

Our client Confirmit.com has used our Evo Agile Method [2] successfully since 2003 [1]. They have adapted it, from the beginning, to their environment, and continued to innovate and learn. Their business success has been attributed to their remarkable product quality improvement, and that improvement specifically to the Evo Agile method, by them, on their website, and share offerings prospectus. Evo differs from other agile methods, in that it focuses on multiple, quantified, software-and-system qualities.

This column will focus on an innovation, the Green Week, that Confirmit, led by their method champion Trond Johansen, made and reported in 2005; two years after adopting Evo.

When we started in 2003, Confirmit had an 8 year old web-based system; a "legacy" product that had grown, as most do, to meet rapidly emerging market demands. By 2005 there were the usual difficulties in enhancing the product, a web-based opinion survey tool, serving markets worldwide, to meet new opportunities, quickly and safely.

We recommended in 2003 that they spend 4 days a week on value delivery cycles to their customer base, and one day a week 'refactoring'. Their development team at the time was 13 plus 3 testers.

The 4-day value delivery cycle aimed at something like 25 distinct quality improvements (for example Usability Intuitiveness) or performance capacity improvements. The stakeholders aimed at were users and Confirmit's future market. The refactoring was aimed at their development team, as stakeholders. The team that did the development initially, also did the maintenance of the system for years, until today.

Let me be explicit, the people who had to 'suffer' bug fixing and long term enhancement were actually in full control of the architecture and design of the entire system. Maintenance was not farmed out to people who just had to suffer it. Most of the staff were not merely programmers, they had formal education in real engineering.

Well, the one day of refactoring was not a great success, while the 4 days of value delivery cycles, to quantified quality and performance requirements was a big success. To my knowledge there is nothing even near-as-good of quantified results, reported for any other Agile Effort! If you know of one, AgileRecord (.com) would like to hear from you! One possible reason for lack of success was that the refactoring was one-day-a-week, and I suspect it was a Friday, where Norwegians want to sneak off early for a Cabin Weekend ("working off site"). But I really don't know.

They asked themselves, 'why should our customers get all the quality improvements?' What not, us hard working developers, get some systematic quality improvements too?

So they decided to spend one week a month, using Evo [2] 'engineering' 'ease of maintenance' and 'testability' into their organization and their product. In other words: 3 weeks being customer oriented, and 1 week a month being internally oriented. Of course, improvements in maintenance capability also improve their ability to respond to customers!

User Week 1	User Week 2	User Week 3	Developer Week 4
Select a Goal	Select a Goal	Select a Goal	Select a Goal
Brainstorm Designs	Brainstorm Designs	Brainstorm Designs	Brainstorm Designs
Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost	Estimate Design Impact/Cost
Pick best design	Pick best design	Pick best design	Pick best design
Implement design	Implement design	Implement design	Implement design
Test design	Test design	Test design	Test design
Update Progress to Goal	Update Progress to Goal	Update Progress to Goal	Update Progress to Goal

Figure 1. The weekly development cycles, with the Green Week.

The key idea here is that we start by quantifying as requirements, all Confirmit system (the software product, the service product, the technical organization) attributes, related to ease of maintaining the system, in the widest sense of 'maintaining' [3]

Here are the requirements they quantified as requirements initially:

Speed, Maintainability, Nunit Tests, Peer Tests, Test Director Tests, Robustness.Correctness, Robustness.Boundary Conditions, Resource Usage CPU, Maintainability DocCode, Synchronization Status.

Page 26 Agile Record – www.agilerecord.com

End

- Free Book Offer:
  - Email me Tom@Gilb.com
  - Subject 'BOK' or BOOK
  - I'll send you a link to
    - The **Competitive Engineering** Book
    - And, other papers related to **Green Week** and **Reducing Technical Debt,** Including this Lecture's slides
    - *No obligations, and you will NOT be put on a mailing list of any kind.*

End

- Free Book Offer:
  - Email me Tom@Gilb.com
  - Subject 'BOK' or BOOK
  - I'll send you a link to
    - The **Competitive Engineering** Book
    - And, other papers related to **Green Week** and **Reducing Technical Debt,** Including this Lecture's slides
    - *No obligations, and you will NOT be put on a mailing list of any kind.*

End

- Free Book Offer:
  - Email me Tom@Gilb.com
  - Subject 'BOK' or BOOK
  - I'll send you a link to
    - The **Competitive Engineering** Book
    - And, other papers related to **Green Week** and **Reducing Technical Debt,** Including this Lecture's slides
    - *No obligations, and you will NOT be put on a mailing list of any kind.*

End Slide!