CMMI and Scrum at Systematic

Visit from VIA university May 3rd 2011

SYSTEMATIC
software engineering

Program

10.00 Introduction to Systematic

10.30 CMMI – what is it, what did we do

11.00 How does it affect daily work

11.30 Test

12.00 LUNCH

12.30 Departure

Company overview

- Established in 1985
- Approx. 475 employees; 71% with a MSc or Ph.D.
- CMMI Level 5 certified

- Sales partners world-wide
- Customers in 35+ countries
  - 97% of our customers would recommend us to other customers

Systematics business areas

Vision **A leading international company in delivering reliable and simple solutions to people who make critical decisions every day**

Defence

Interoperability,
Command and control
Situational Awareness
Electronic Warfare Support

Healthcare

Clinical Information systems

Intelligence and National Security

Management Information Systems
Border Control
Various other solutions

Integration Services

Finance
Transport
Agriculture

Mission **Simplifying Critical Decision making**

SYSTEMATIC
software engineering

The IT-Knowledge Company

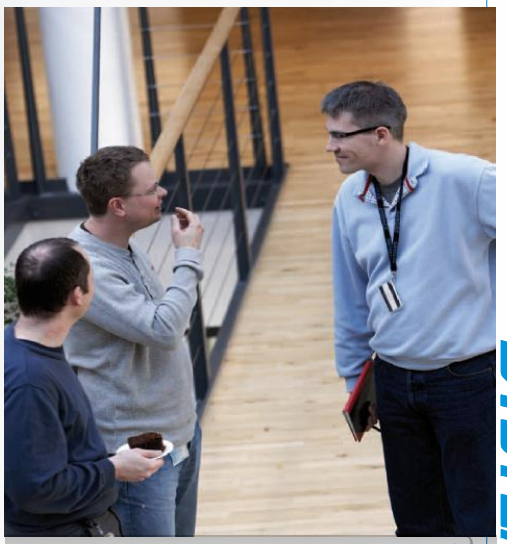
CMMI5

Revision: 1.12

Employees & Competencies

- 72% of our employees are software developers, 8% domain specialists and 20% staff employees
- 58% of our software developers hold a PhD or a master's degree, 21% are bachelors of engineering, and 21% are computer scientists or other
- Average work seniority is 9.5 years and the average age is 36 years
- Each employee on average eats 15 kilos of banana and 10.5 kilos of carrots a year
- Half of the employees ride their bikes to work

"Better train people and risk they leave, than do nothing and they stay"



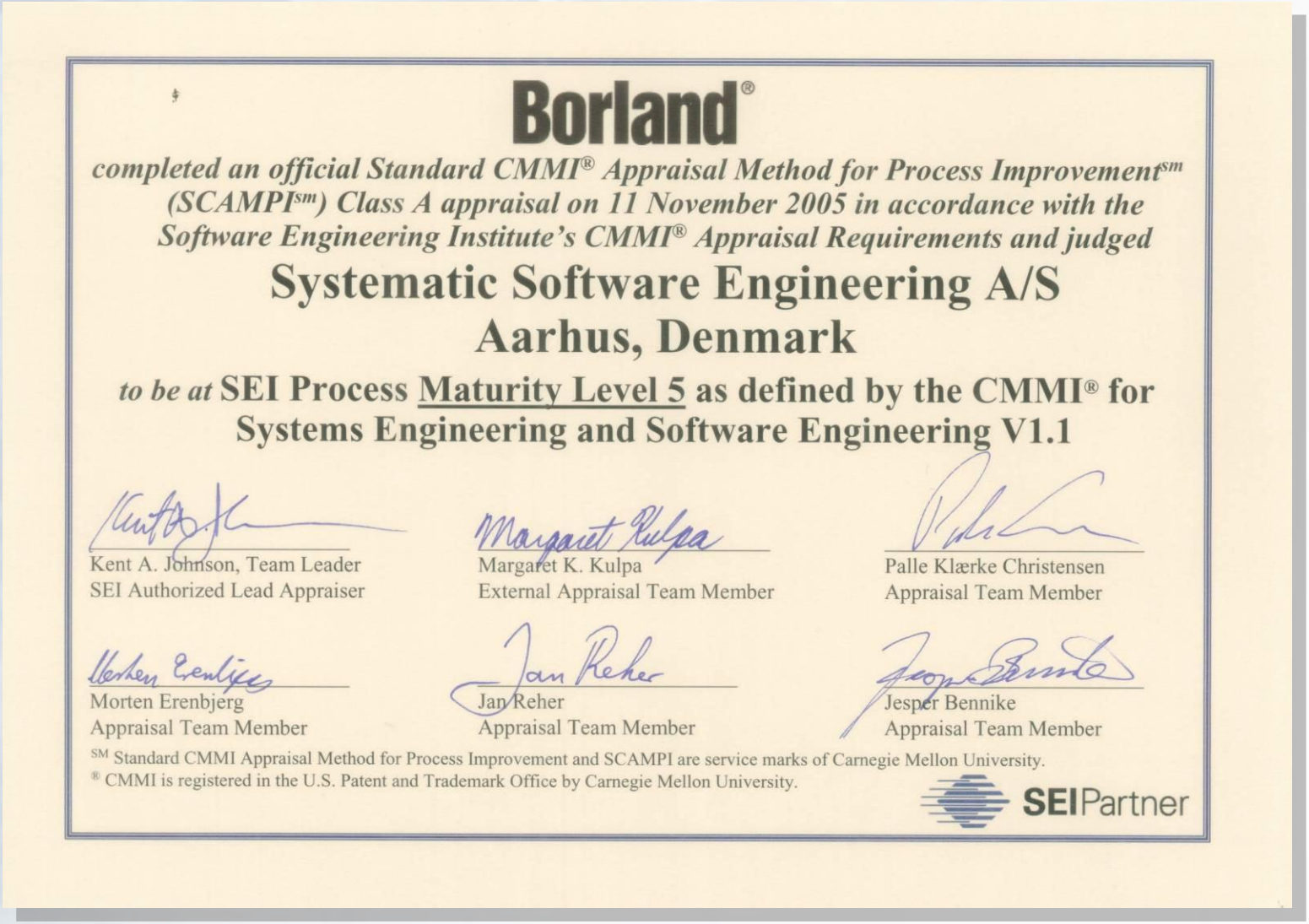
What would you like to hear about?

At 12.30 the time is up!

What would you like to bring with you home?

What is CMMI

Systematic is CMMI level 5



Borland®

completed an official Standard CMMI® Appraisal Method for Process Improvement℠ (SCAMPI℠) Class A appraisal on 11 November 2005 in accordance with the Software Engineering Institute's CMMI® Appraisal Requirements and judged

Systematic Software Engineering A/S
Aarhus, Denmark

to be at **SEI Process Maturity Level 5 as defined by the CMMI® for Systems Engineering and Software Engineering V1.1**

Kent A. Johnson, Team Leader
SEI Authorized Lead Appraiser

Margaret K. Kulpa
External Appraisal Team Member

Palle Klærke Christensen
Appraisal Team Member

Morten Erenbjerg
Appraisal Team Member

Jan Reher
Appraisal Team Member

Jesper Bennike
Appraisal Team Member

℠ Standard CMMI Appraisal Method for Process Improvement and SCAMPI are service marks of Carnegie Mellon University.
® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

SEIPartner

Terminologi

Capability = ”The ability to do something”

- In addition to focus on action 1) Leadership commitment, 2) training, 3) monitoring and 4) quality assurance

Maturity

- Almost always interpreted as ”predictability”

Model = ”Abstraction of reality”

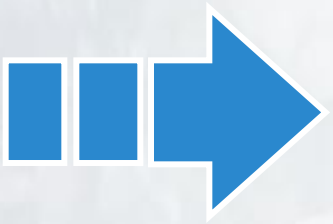
- The model must be adopted to the context of your company
- CMMI provides 1) requirements, 2) expectations and 3) suggestions

Integration

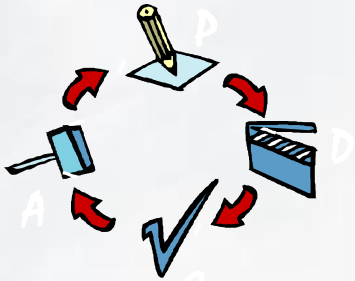
- CMM’s succes resulted in many derivative models
- These models are integrated into a meta model

Why was CMMI invented?

Developed to evaluate suppliers



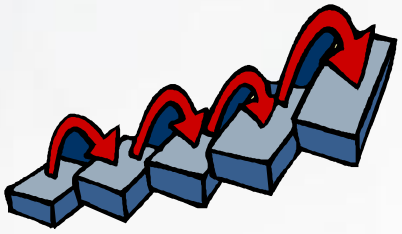
Processconcept is essential



Improvements are done systematicly

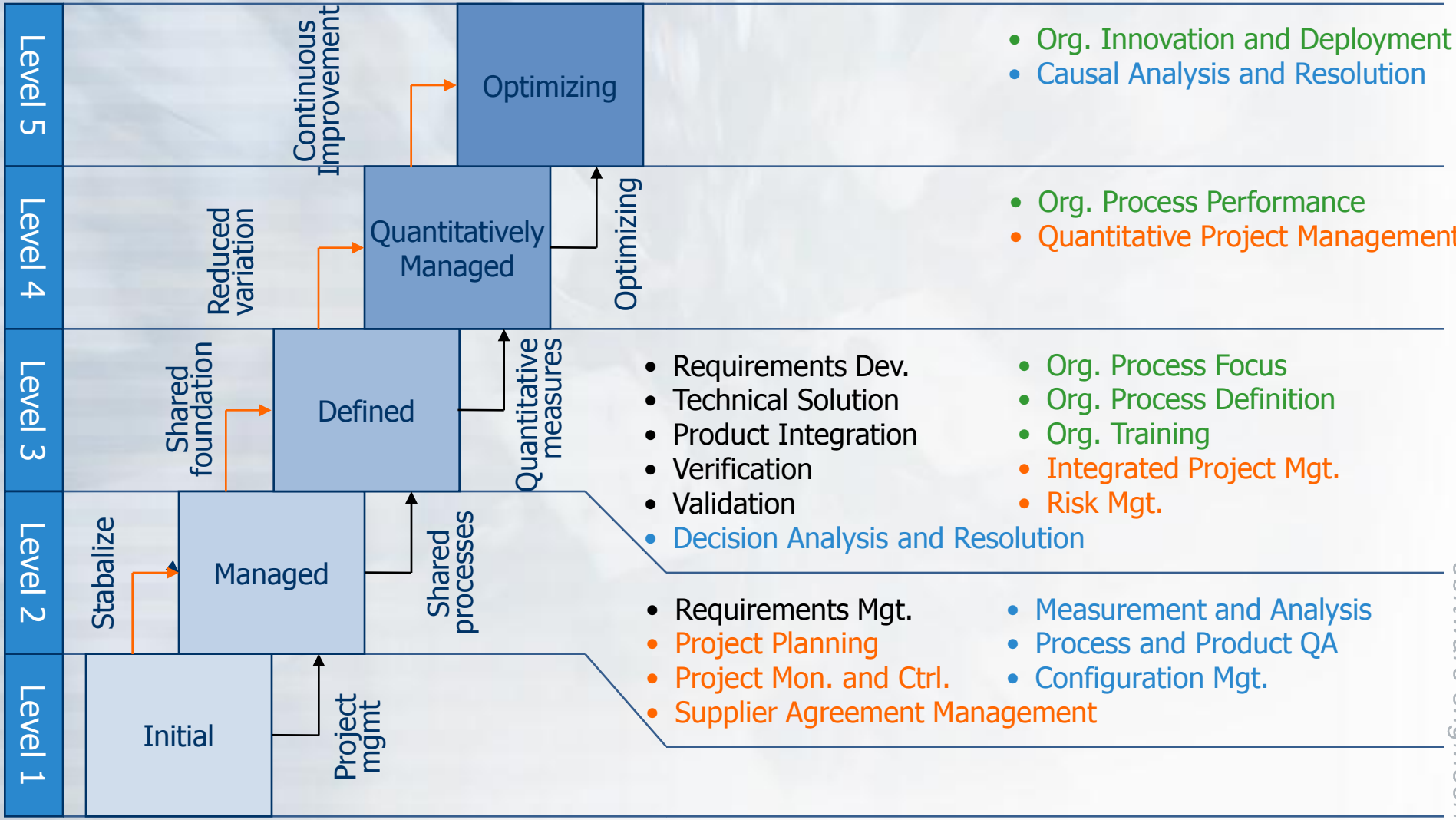
You must learn to crawl before you walk

Objevtivity through measures



Five levels

CMMI specific process areas (PA)



Each PA achieves one or more goals

SG 1 Requirements are managed and inconsistencies with project plans and work products are identified.

SP 1.1 Obtain an Understanding of Requirements

SP 1.2 Obtain Commitment to Requirements

SP 1.3 Manage Requirements Changes

SP 1.4 Maintain Bidirectional Traceability of Requirements

SP 1.5 Identify Inconsistencies between Project Work and Req's

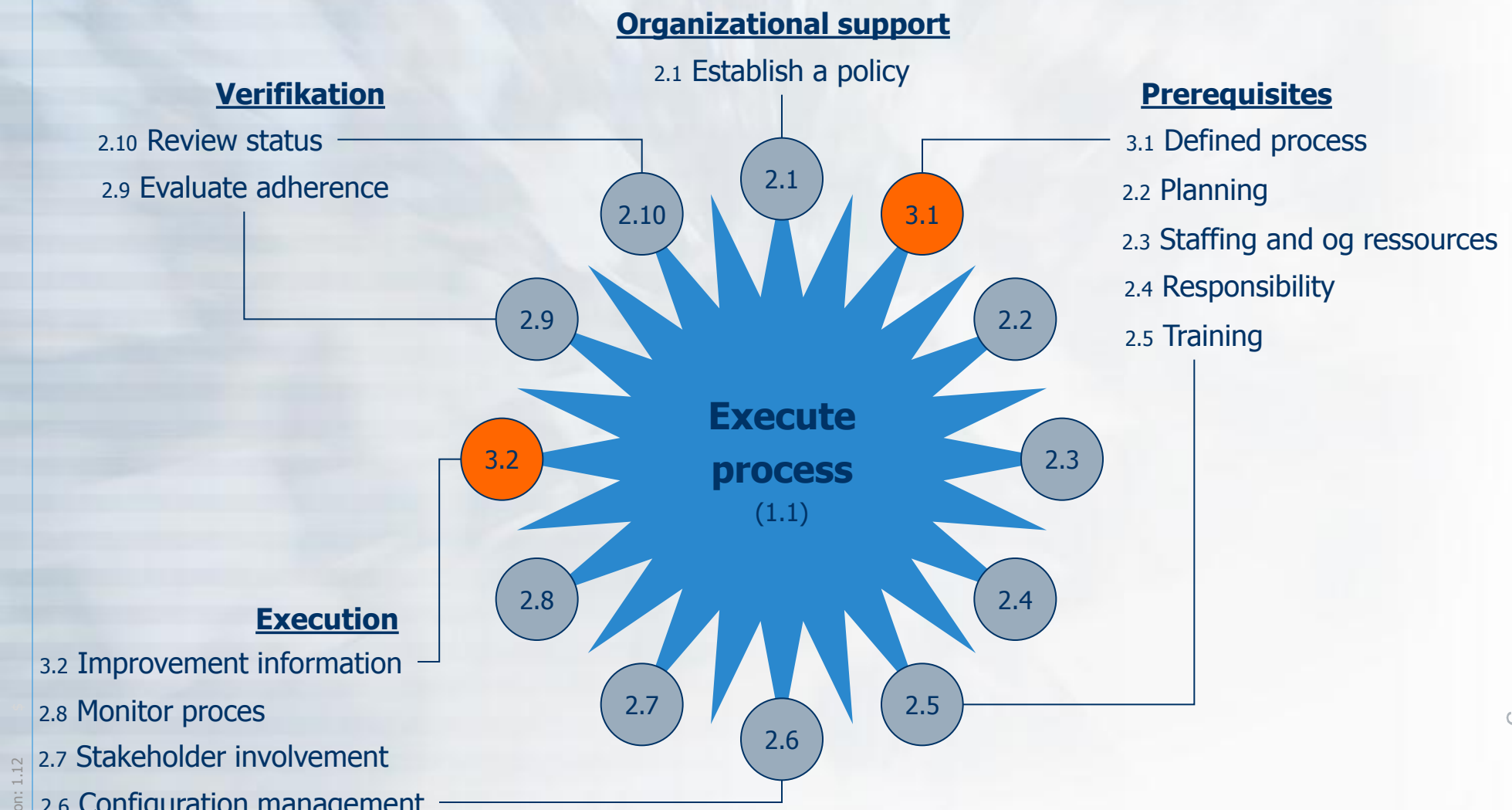
SubP1.1-1. Establish criteria for distinguishing appropriate requirements providers.

SubP1.1-2. Establish objective criteria for the acceptance of requirements.

SubP1.1-3. Analyze requirements to ensure that the established criteria are met.

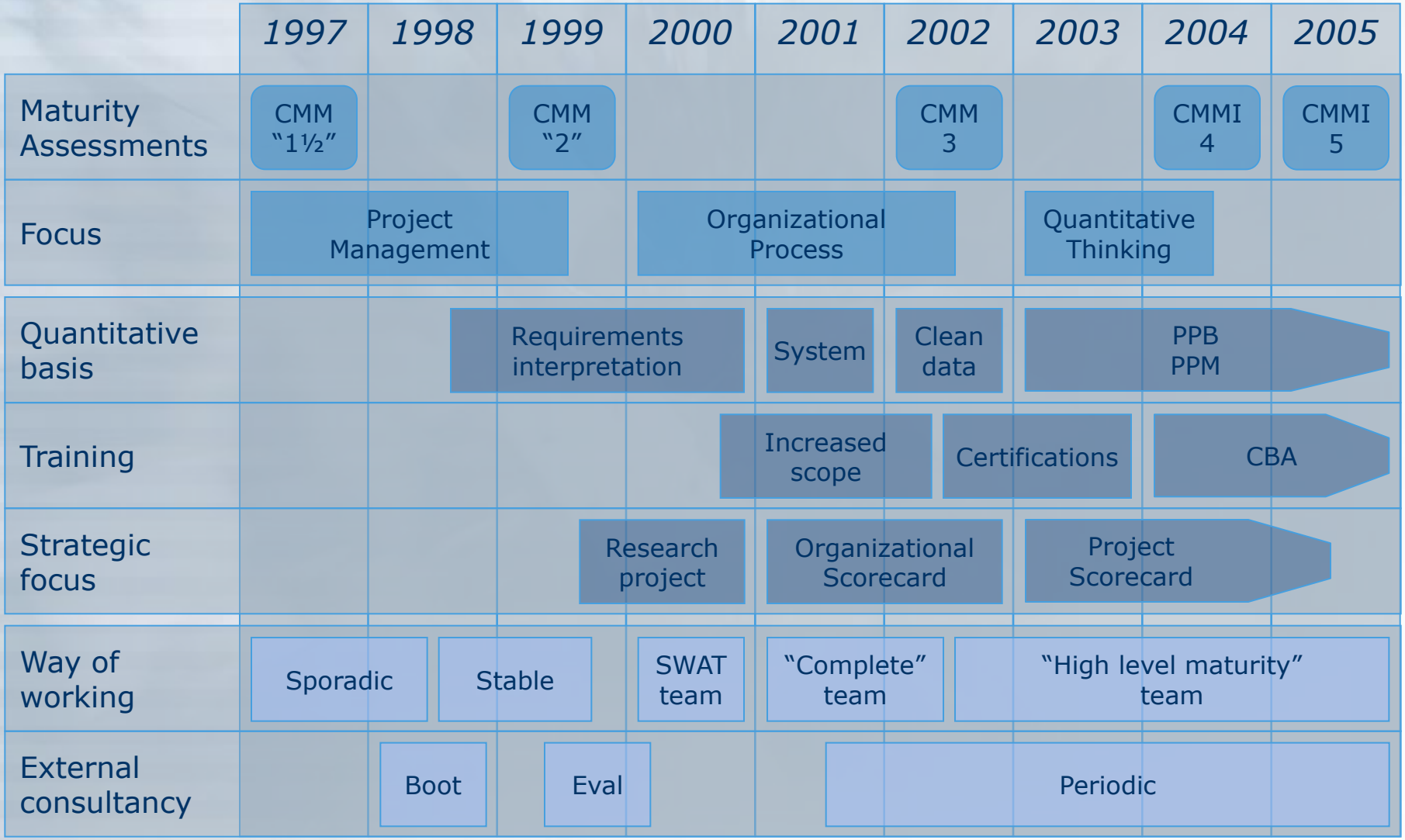
SubP1.1-4. Reach an understanding of the requirements with the requirements provider so the project participants can commit to them.

All PA applies a set of generic practices



CMMI at Systematic

Timeline of Process Improvement



Statements after achieving CMMI level 3

It will change your culture

- "The amount of overtime has been reduced and people are more satisfied"
- "Compared to 3-5 years ago, projects are better at estimating and planning"

New practices are vulnerable

- "Quantitative Management is new to the organization. It's at its very beginning."

Change will call for more change

- "Better integration between organizational tools supporting processes"
- "Would like to see more testing going on along with design, more automated test, regression tests etc."

How CMMI 5 affects daily work today?

Standards and models

QMS

PDP

- CMMI, ISO, AQAP, …

- BM
- SSP

- Project way of working
- Based on QMS
- Adjusted to the projects characteristics (tailored)

- Important steps of key activities are documented from actual practice
- Descriptions assume a certain level of knowledge to the process
- Proces is appr. 8 workflows and 15 2-3 page descriptions of disciplines

CMMI and Agile

CMMI level 5 and beyond

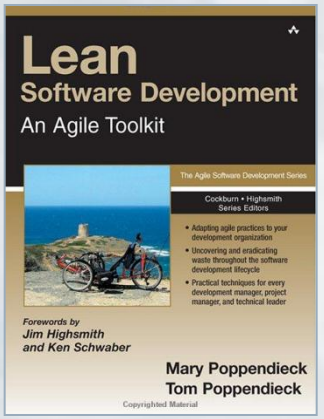
- Agile2007
- SEPG 2007
- HICCS 2008

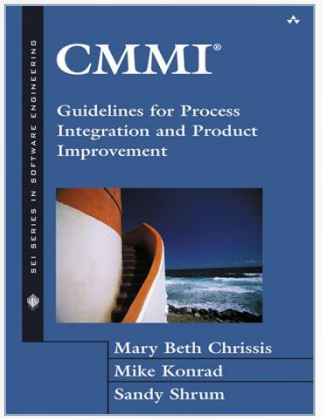
Scrum and CMMI Level 5: The Magic Potion for Code Warriors

Jeff Sutherland, Ph.D.
Patientkeeper Inc.
jeff.sutherland@computer.org

Carsten Ruseng Jakobsen
Systematic Software Engineering
crj@systematic.dk

Kent Johnson
AgileDigm Inc.
kent.johnson@agiledigm.com

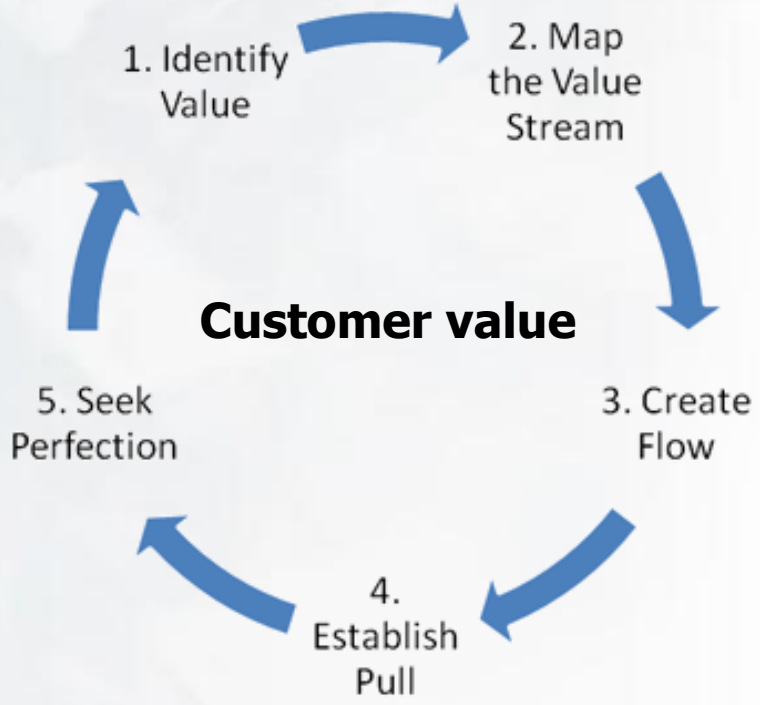




Faster than agile stronger than CMMI

CMMI and Lean combined

Lean (principples) - CMMI (processes) - Agile (practice)



- **Lean –** principples (what)
  - Value, people, culture
- **CMMI** - processes (where)
  - Process og organization
- **Agile** - Practice (How)
  - State of art praksis

CMMI and Lean – what does it mean?

- Do the CMMI disciplines in an agile iterative way

- **Scrum**
  - Iterative development, typically planned in 1 month sprints
  - Self organized feature teams
- **Story based development**
  - Scope decomposed to features and stories
  - Described method supported with inspection checklists

In order to deliver working software by the end of each 1-month sprint, a number of disciplines must be established

- **Automated test**

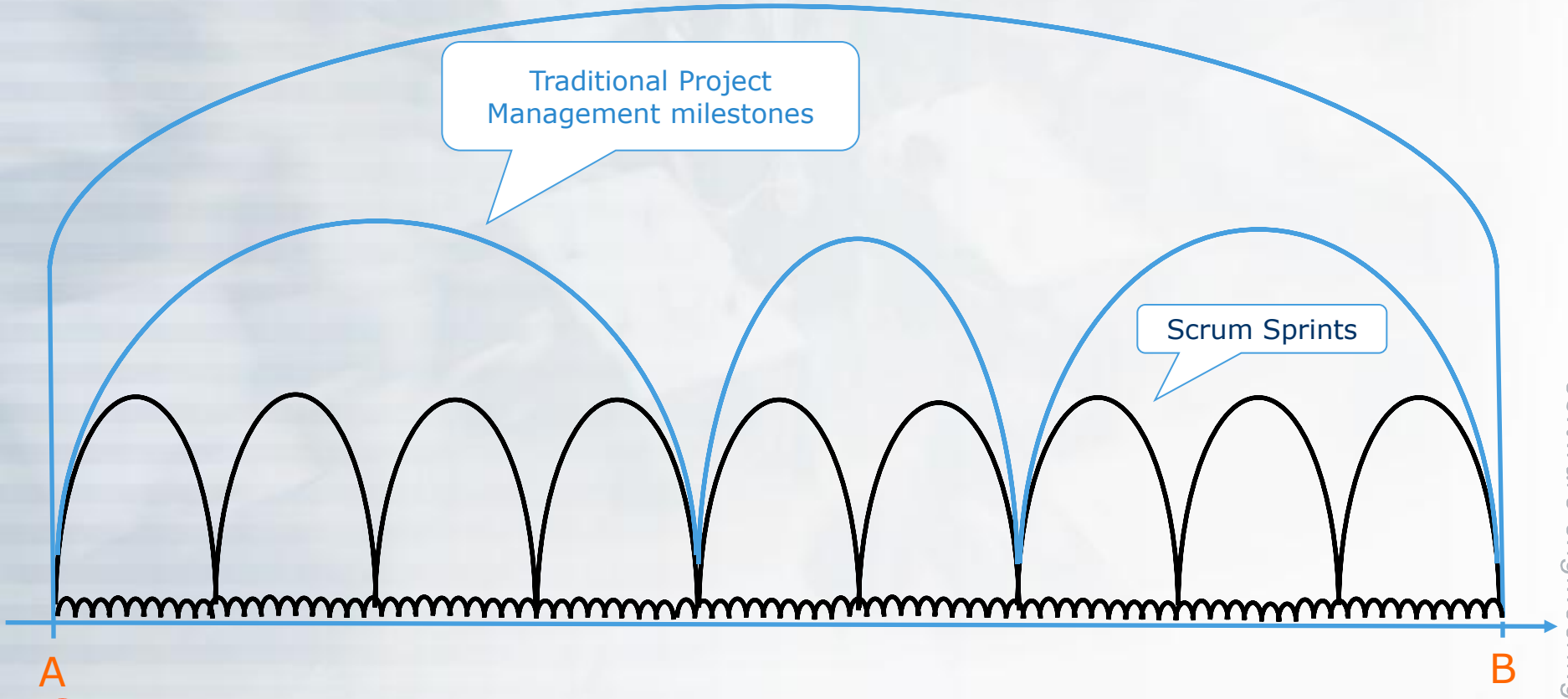
  No time for manual testing
- **Testdriven development**

  Defects must be found and adressed efficiently
- **Continous Integration**

  Integration problems must be found and adressed efficiently

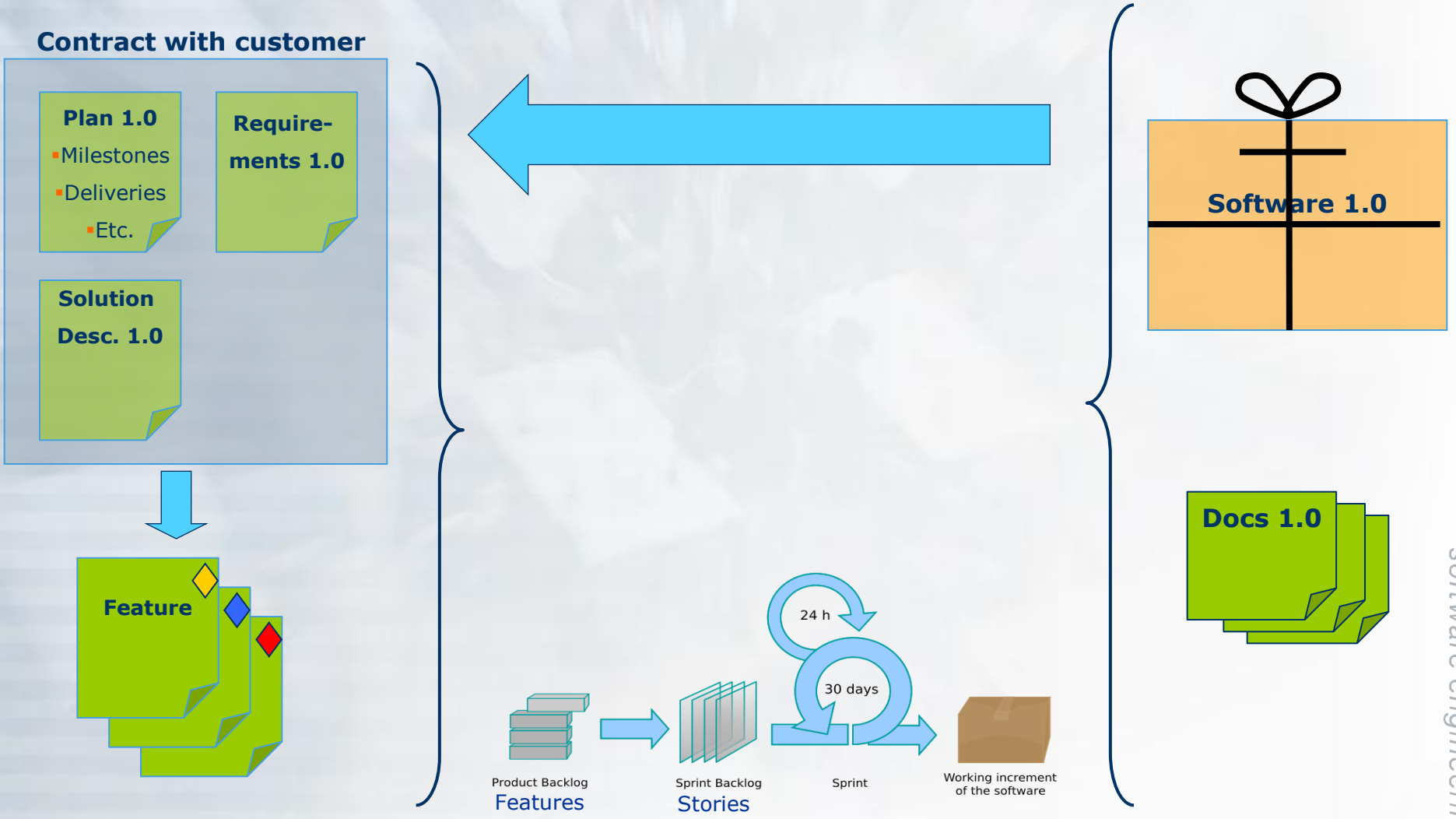
Project Management and Scrum

Different levels of planning linked by the Product Backlog

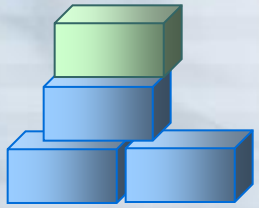


How we work

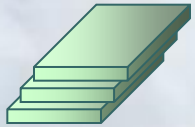
How to decompose work in a contract



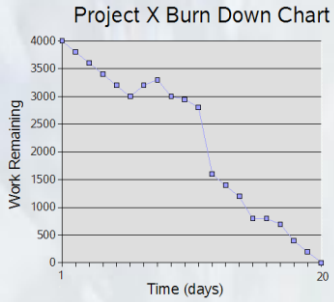
SCRUM Meetings

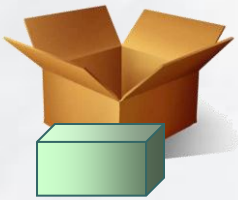


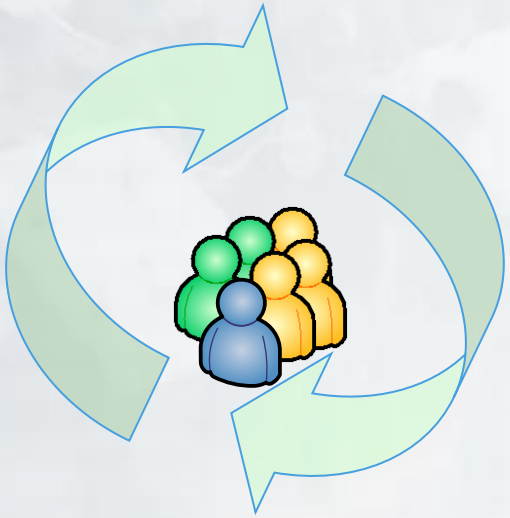
Product Backlog

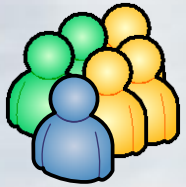


Sprint Backlog









Product Backlog Selection

1

Sprint Planning Meeting

2

Daily Scrum Meetings

3

Sprint Review

4

Sprint Retrospective

Daily Scrum

We work in co-located teams using Scrum

Lean A3 Project Management Status

Project status - Agriculture – Februar 2008

▶	**Schedule & Progress**		
	▶	Ability to Deliver (milestones)	
	▲	Corrective Actions	
	▲	Issue Responsiveness	
	▶	Sprint Management	
▶	**Resources and Cost**		
		Earned Value Management	N/A
	▶	Estimeringsevne	
	▲	Udfaktureringsgrad	
	▲	Sales forecast	
	▶	Effort Distribution	TBD
▶	**Product Size and Stability**		
		Requirements Management (scope)	N/A
▶	**Product Quality**		
	▶	(SQA) (throughput, latency & performance)	
	▶	Defect Insertion	N/A
	▶	Product Integration (fra IS)	8,97
▶	**Process Performance**		
	▼	(Process Compliance - SAS)	2,71
	▶	Configuration Audit	
	▶	DAR	
	▶	CAR	
	▶	QAS	
	▶	Review (fra IS)	
	▶	Inspection	TBD
	▶	Unit Test	98%
	▶	Test	TBD
▶	**Customer Satisfaction**		
	▶	Customer Product Satisfaction	TBD
▶	Risk		

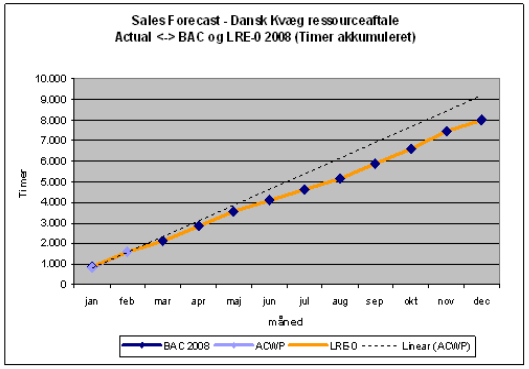
Comments from Project Manager:

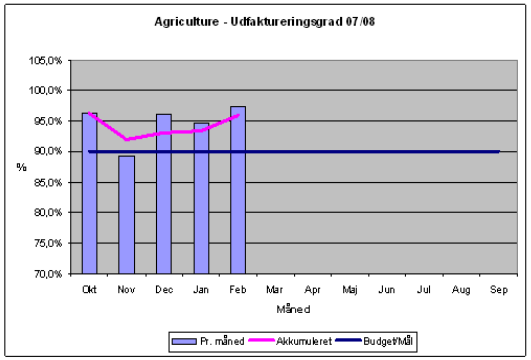
- Corrective actions: 2 åbne, 1 overdue:
  1. SQA-møde med LC afholdes 8.03.08
- SQA: se kommentar under corrective actions
- Product Integration. Processen er ikke i control. Der er iværksat analyse af målingerne. Der har været konkrete problemer hvor KVK og HNC kom i konflikt på byggeserver, hvilket er fikset. Det skal analyseres om der er andre problemer.
- SAS: 2,71 er på institutionalisering, QPM er på 3,00. Øvrige procesområder ligger mellem 4 og 5 med et gns på 4,47. Projektet skal sikkert have waivers på nogle områder (i gang). Derudover er afsat en manddag pr uge for at fikse SAS-huller

Risk top	
Fejl i HN-"drift" som følge af uplanlagte ændringer i 3. parts værktøjer/snitflader	Indtruffet
Fejl ved idriftsættelse pga at prodtest- og prod-miljøer ikke er ens.	Mitigeret
Manglende performance og skalering	Mitigeret
Vidensdeling med KAP – UE/Ark	Mitigeret

Indicators

	Mar	Apr	Maj	Jun	Jul	Aug	Sep
PL	1	1	1	1	1	1	1
SSE-Developer	5	5	5	5	5	5	5
LC-Developer	0,5	0,2	0,1	0,1	0	0	0





Comments from Project Manager:

Faktureringsgrad og omsætning er foran planen. Der har ikke været sprintafslutning sidst sidste opgørelse, hvorfor der ikke er nyt på den front.

Udvikling af Foderkontrol v.1 til produktionsafprøvning leveres d. 7.03.08. Det er aftalt at inkludere stories fra den planlagte v2 i v1 for at øge udbyttet af produktionsafprøvningen. Den samlede LRE for v1+v2 lander knap 20% under BAC. Vi har analyseret årsager til den lave LRE. Hovedkonklusioner er:

- Det er mere effektivt at udvikle en samlet mængde ny funktionalitet end en række spredte observationer
- Vi har ikke haft så mange releases, hvilket har reduceret antal timer til generelle projektaktiviteter
- Meget tæt samarbejde med LC, hvor funktionalitet konstant blev "rettet ind"
- Meget autotest i nyudvikling
- Den story-baserede udvikling med inspections mener vi er en gevinst.

Vi har netop afleveret estimat på den næste opgave til KVK – Foderplan til Ungdyr. Det samlede estimat er på 777 timer.

Den nye SAS er virkelig god. Nu kan man identificere hvad der faktisk forventes – dvs det er ikke længere en individuel fortolkning. Vi har nogle huller, som vi vil fokusere på: den kommende tid. Der er afsat 1 manddag pr uge til at rette op på vore huller. Derudover skal/kan vi få waiver på nogle områder, hvilket vil øge vores score – der skal dog kun gives waiver på de områder, hvor projektet ikke får værdi af (del)processen – dvs vi skal ikke have waiver, bare fordi vi kan få waiver.

Kundens udvikler (PML) er stoppet hos SSE, dog vil han forsat komme ca ½ dag pr uge, så længe der er behov ift videnoverdragelse. JDM sidder pt alene på HN-opgaver, men PD skal på opgaverne så snart han har løst nogle BL3-opgaver. Der er en dialog med kunden om at allokere endnu en person på BL3, da der er planlagt en del opgaver.

Der kommer sandsynligvis en udvikler fra Tomlab/Sverige i næste uge. Han skal arbejde på ændring og optimering af deres optimeringskerne. Han skal sidde her for at der kan være tæt dialog om ændring af interfaces i Norfor. Forventeligt skal han være her 5-7 arbejdsdage, dvs frem til påske.

Internt er projektets største udfordring forsat at projektet mangler en erfaren teknisk "førerhund"/Lead developer efter at ØHE er overgået til andet projekt. Udvikling på HN og Bch lider under det, hvor vi ikke har den forventede/tidligere fremdrift.

Actions/notes:

Date **Signature**

Quality of story

- Before development of a story is started developer will ensure understanding of scope and design acceptance test
- During development the story is inspected by an inspector
- The inspector is often a lead developer.
- When a story is completed, a final inspection ensures completeness

- **Checklist made by the developers**

Story Completion Checklist

Story:	
Feature:	
Developers:	
Inspected by:	

Activity	Work Product(s)	Completed	Inspected	# Defects
Story scope and estimate reconsidered		☐	☐	/
Development environment established		☐	☐	/
Requirements drafted		☐	☐	/
User interface drafted		☐	☐	/
Technical design drafted		☐	☐	/
User documentation drafted		☐	☐	/
Test design drafted		☐	☐	/
Requirements complete		☐	☐	___
Existing manual tests identified		☐	☐	/
Tests drafted (and coordinated with the Test Designer)		☐	☐	/
Code drafted		☐	☐	/
Manual tests complete		☐	☐	___
Automated tests complete		☐	☐	___
Test design complete		☐	☐	___
Code complete. (User interface inspected by UE)		☐	☐	___
Manual tests executed and passed		☐	☐	___
Technical design documentation complete		☐	☐	___
User interface documentation complete		☐	☐	___
User documentation complete		☐	☐	___
Story integrated (and integration tests passed)		☐	☐	___

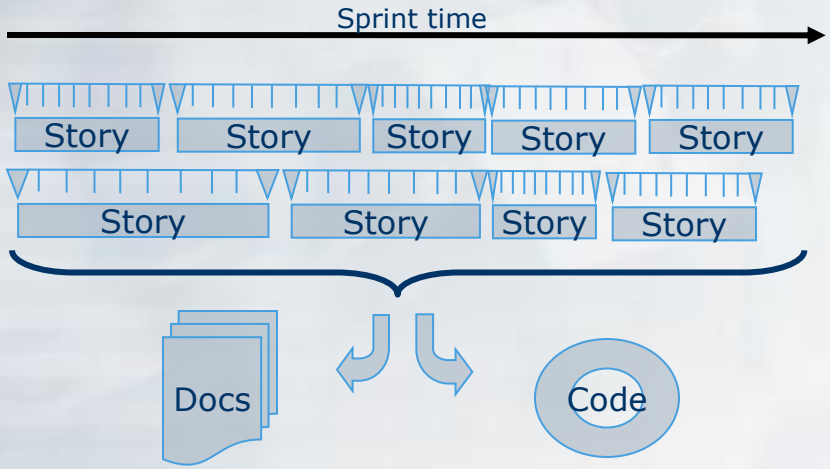
Story complete		
Date	Developers	Inspector

- At start of story, cross out non-applicable activities and add extra activities.
- The "Work Product(s)" column is optional.
- Fill out checkboxes with ✓ or ÷. Do not pass a line until all activities before it are completed and inspected. During inspections, note number of defects found and mitigated in each completed activity.
- When all checkboxes are filled out with ✓or ÷, sign the checklist and give it to the Team Leader.

SSE04529/CHK029 $Revision: 1.4 $ $Date: 13 Jul 2006$ Page 1 of 1

Quality of product

- Implementing stories drives establishing or updating documents and code.
- Depending on the type of document one or more reviews are required
- Similarly the project will have a criteria for what code is subject to review



Document Reviews

-Concept

-Quality

Code Reviews

Customer Review Report

[SSE/03477/RER/040]

Completed before review	Completed after review	Completed after corrections

Project No.	Review Title				Review Date
03477	Dokumentation af det leverede it-system				08-10-2007
Documents To Be Reviewed					
Document No.	Document Title / Section	Rev.	# Pages	# Defects	Corrected Rev.
SSE/03477/USM/001	Brugervejledning til det samlede system Section: All	1.1	20	0	

Participants								
Name/Initials	Function/Role						Focus	Signature
	Lead	Prod	Revw	Corr	Veri	Cust		
crj	■	□	□	□	□	□	Komplethed i forhold til krav	
jar	□	□	■	□	□	■	Anvendelighed	
jjc	□	□	■	□	■	□	Tværgående sammenhænge	
mha	□	■	□	■	□	□	Konsistens til løsning	

Conclusion	
□	Accepted as is
■	Conditionally Accepted
□	Not Accepted
□	Review not completed

Statistics	
Time spent preparing for review	Time spent conducting review
120 min(s)	240 min(s)

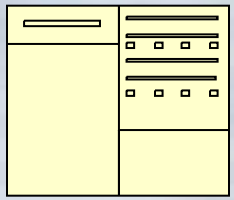
Verification of Corrections		
Name/Initials	Date	Signature
jjc		

Continuous integration

Specific Goal	Agile Practice
Prepare for Product Integration	1.1 – Establish Automatic Build Env. 1.2 – Establish Check-In Procedure and Criteria 1.3 – Test-Driven-Development (TDD)
Ensure Interface Compatibility	2.1 – Integrate Modules/Classes/ Components as Soon as Completed 2.2 – The Unit/Integration Tests Have to be 100% Successful
Assemble Product Components and Deliver the Product	3.1 – Continuous Integration 3.2 – Daily/nightly build 3.3 – Incremental Releases

Example: Continous integration

Status on the projects shared build server is constantly monitored and reported by CruiseControl

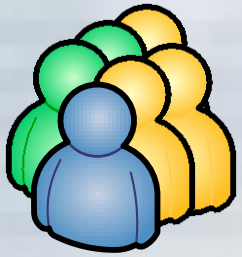


Buildserver

Test and continous integration

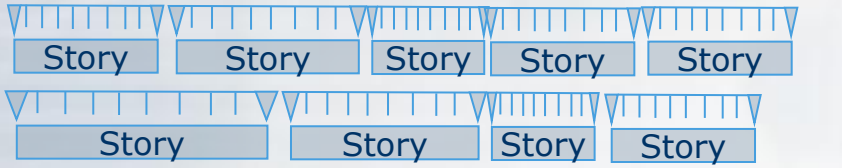
Criteria for moving code from developer to build-server:

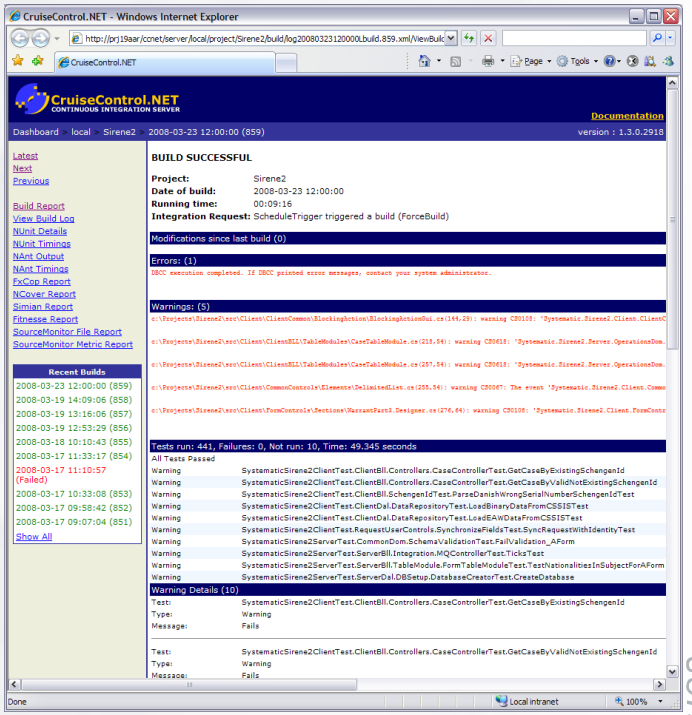
- Unit test works locally
- Code is accepted by FxCop

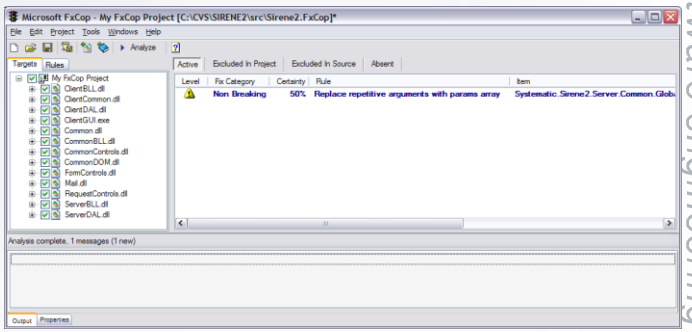


Team

Sprint time

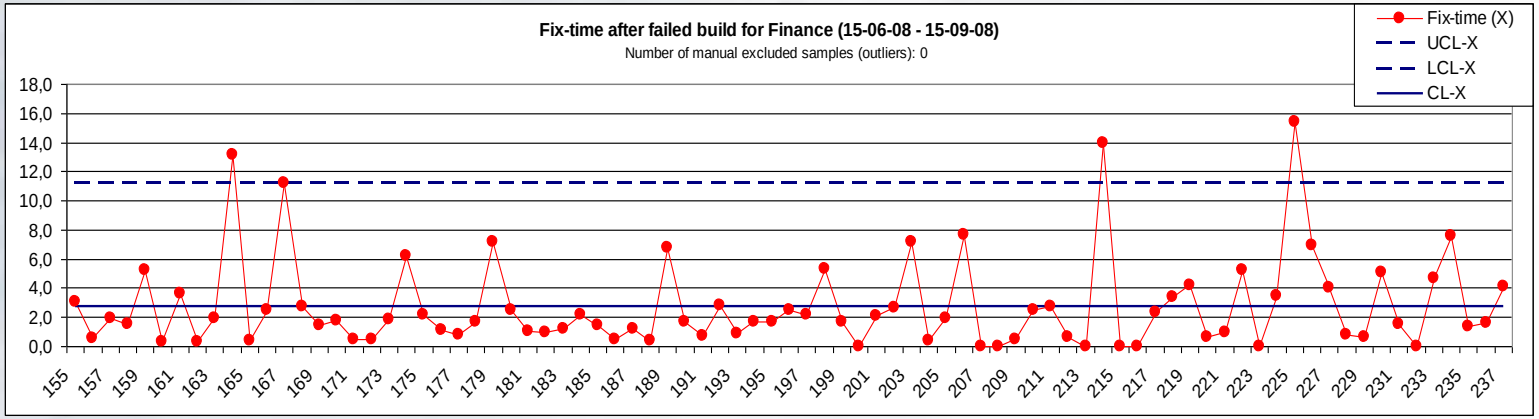






Continous Integration

- Low fix-time drives high quality and speed
  - To improve the fix time between failed builds
    - First step: stabilize the build process
    - Second step: create and use PPBs and PPMs
    - Third step: continuously improve the process



Risk Management and Audit

- Risk Management proactively removes impediments
  - Has it owns meetings
- Audit helps sustain good habits
  - The integrity of releases are verified upon release
  - Integrity of configuration universe is verified periodically

$SE/06574/WPE/0009 $Revision: 1.2 $ $Date: 25 Jan 2008$

Work Product Evaluation Report

WPE Identification

Project Name:	eFPI
Project Number:	06574
WPE ID:	0009

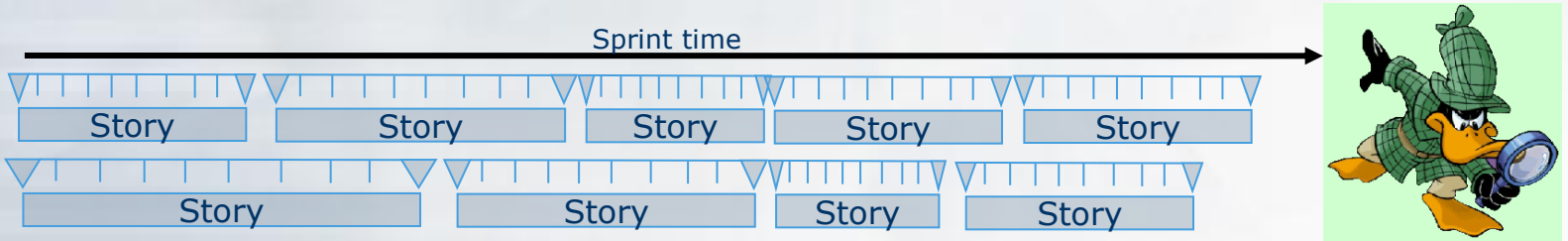
Performed by

Signature Project Manager	Date
Signature Internal Auditor	Date

WPE Conclusion

The overall conclusion of the WPE is (tick only one):

WPE Conclusion	Description	Comments
☒	Accepted as is	See WPE result and improvement suggestions in the "Summary" section
☐	Conditionally accepted	See the conditions for fulfilling the WPE in the "Summary" section
☐	Not accepted	A new WPE shall be performed
☐	WPE not completed	See disposition in the "Summary" section



Release

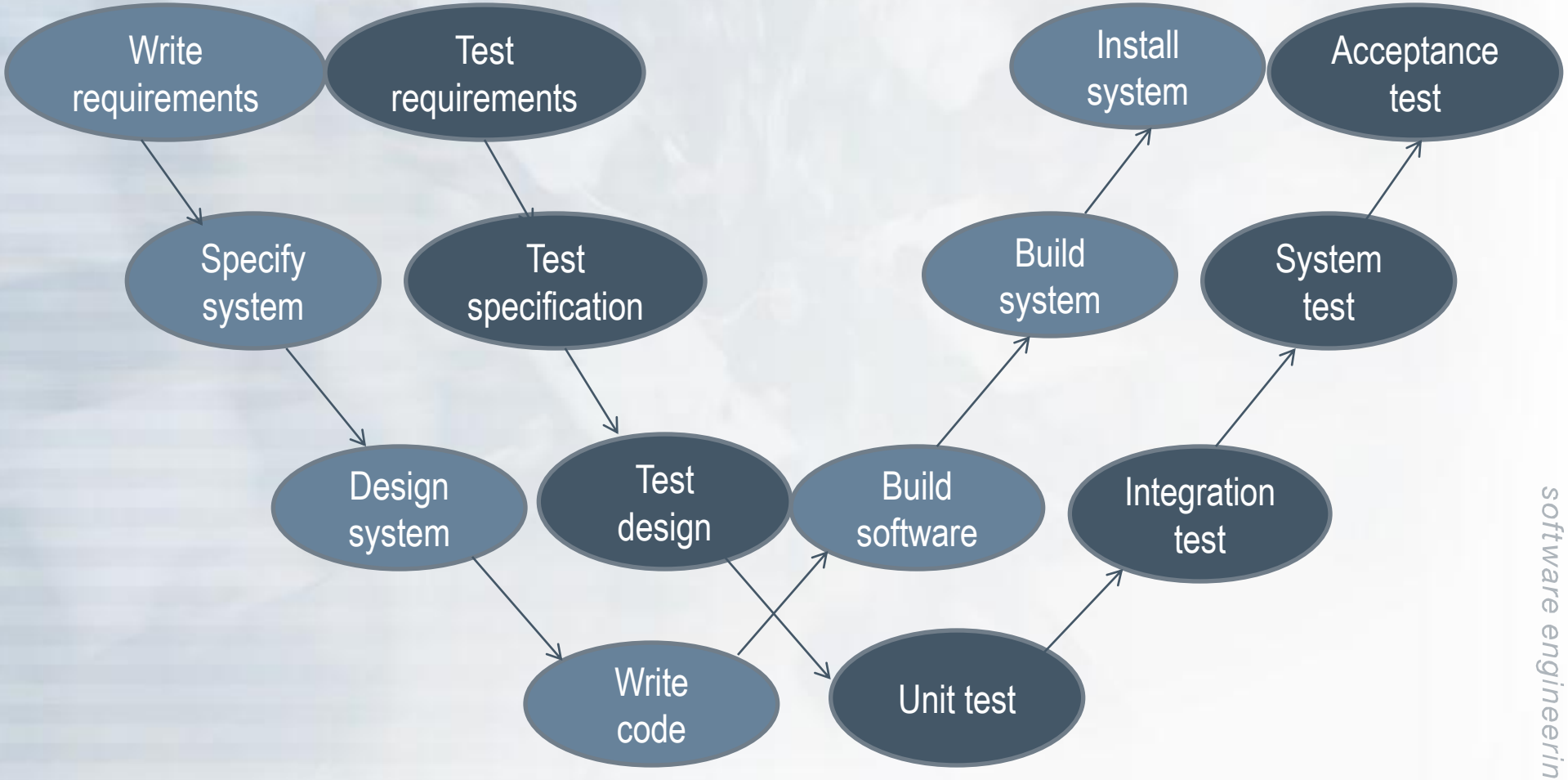
Testing at Systematic

What is the cost of a defect?

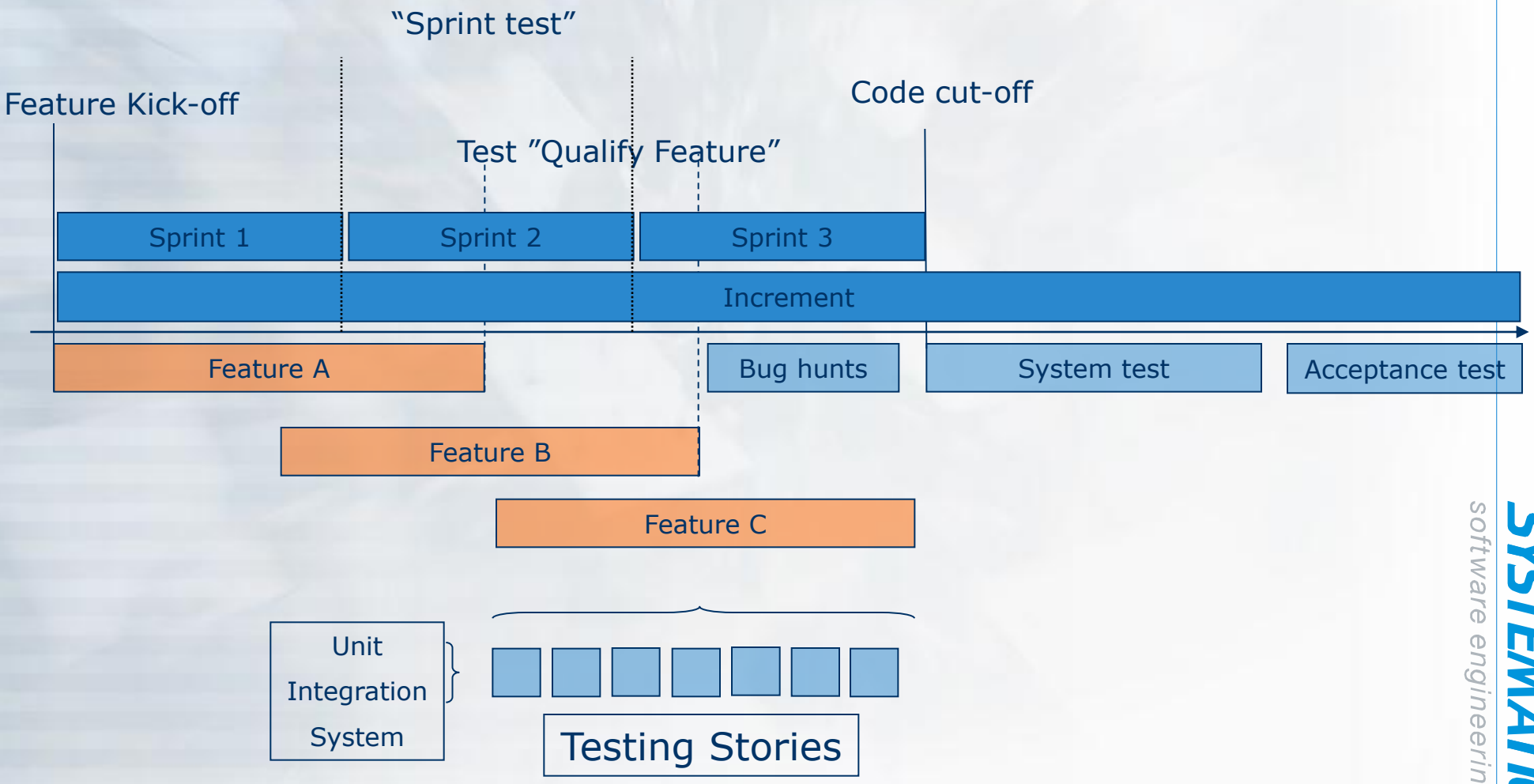
		Found				
		Req.	Arch.	Coding	System test	Post-release
Introduced	Req.	1x	3x	5-10x	10x	10-100x
	Architecture	-	1x	10x	15x	25-100x
	Coding	-	-	1x	10x	10-25x

Source: McConnell, Steve (2004). *Code Complete* (2nd edition ed.).

Test during feature development



Testing in an increment



Planning the feature test

Test masterplan

- Describes in general terms how the project will test

Test plan

- Describes how we will test a delivery
- Focus on system- and acceptance testing

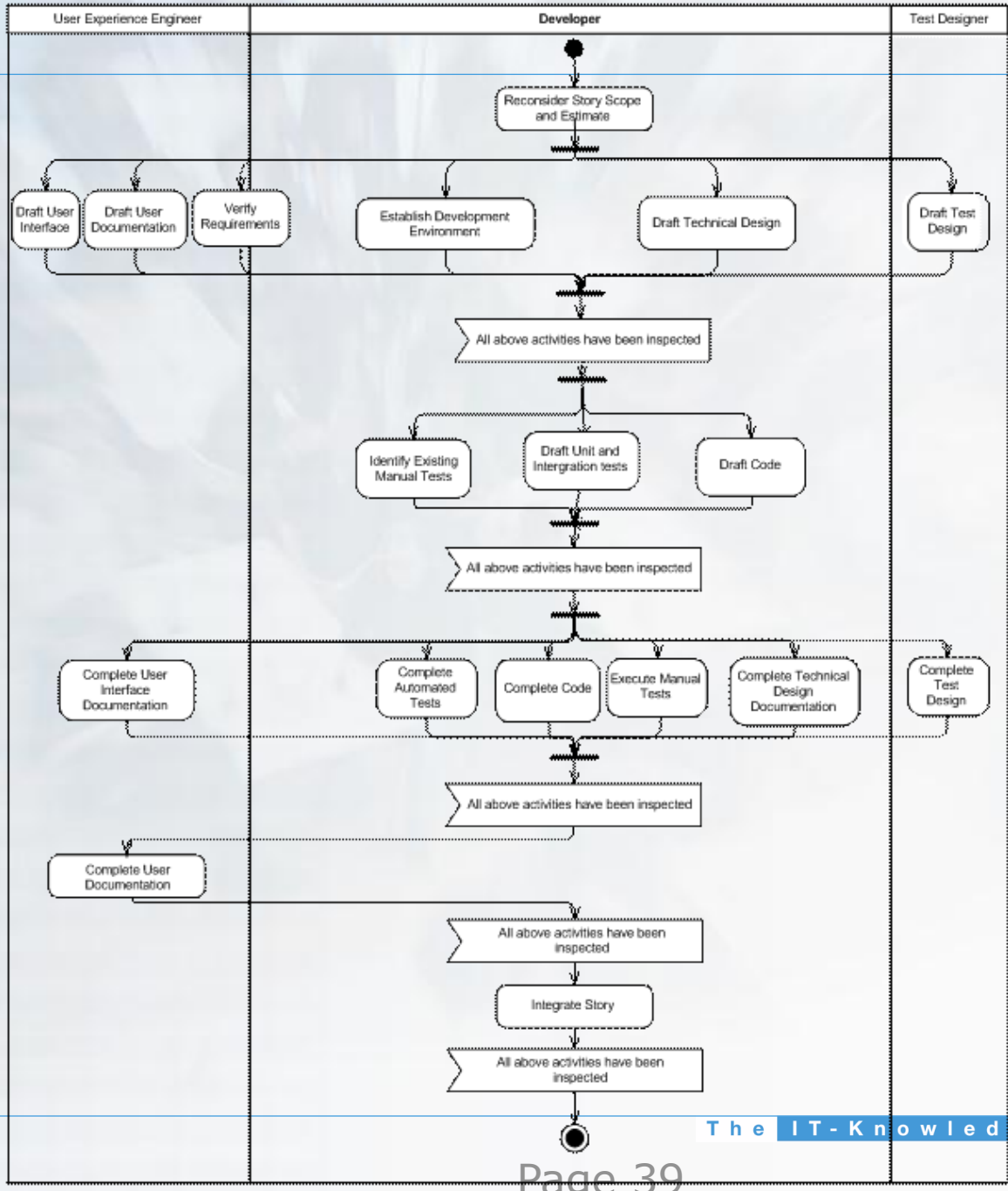
Test strategi/approach

- Describes how we will test a feature
- Ready before feature implementationen starts

Test design

- Describes what will be tested in a feature (including every story)
- Is drafted and discussed before writing story code

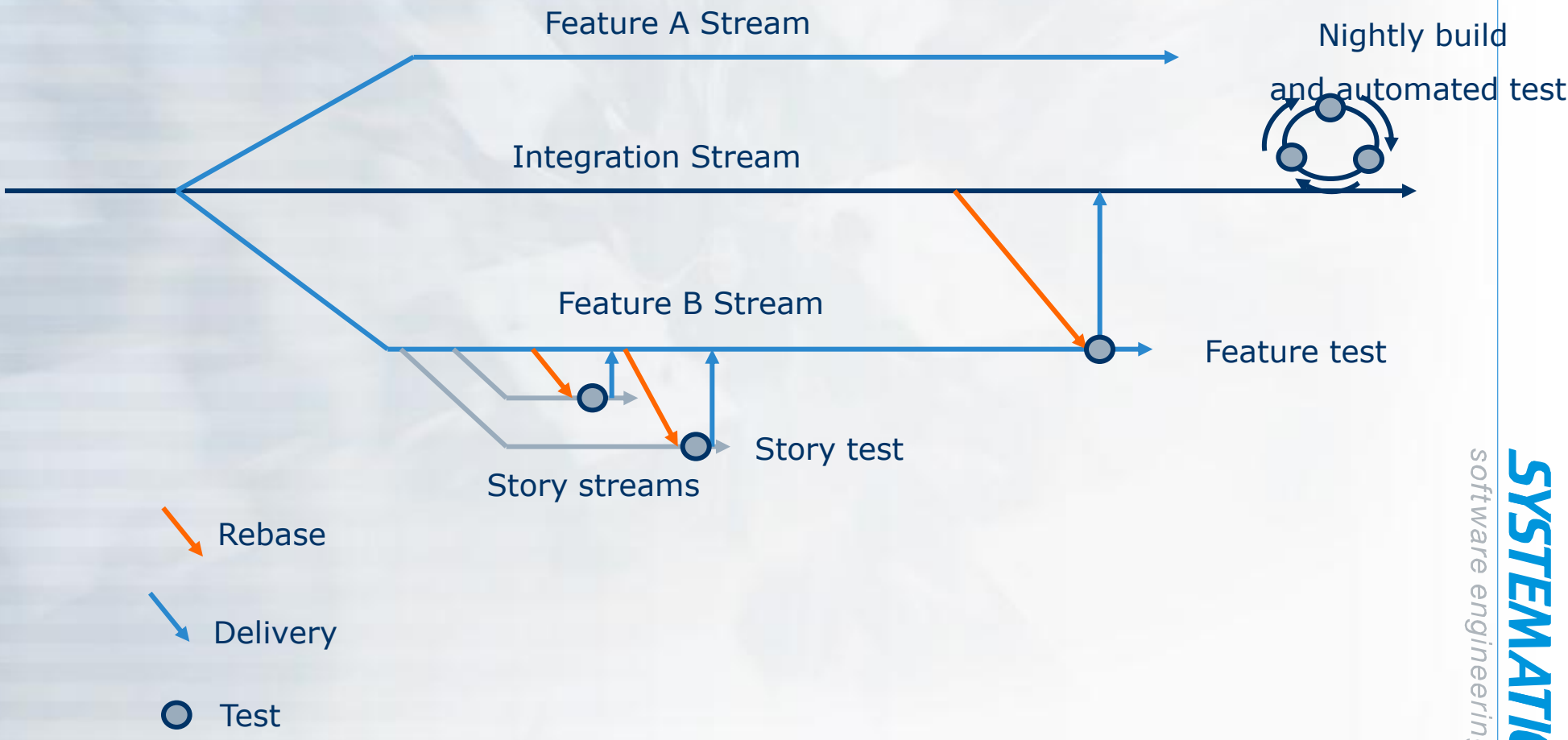
The Story


User Experience Engineer
Developer
Test Designer
Reconsider Story Scope and Estimate
Draft User Interface
Draft User Documentation
Verify Requirements
Establish Development Environment
Draft Technical Design
Draft Test Design
All above activities have been inspected
Identify Existing Manual Tests
Draft Unit and Intergration tests
Draft Code
All above activities have been inspected
Complete User Interface Documentation
Complete Automated Tests
Complete Code
Execute Manual Tests
Complete Technical Design Documentation
Complete Test Design
All above activities have been inspected
Complete User Documentation
All above activities have been inspected
Integrate Story
All above activities have been inspected

Qualify feature

1. Execute all relevant system and acceptance tests for the feature in the test environment
2. Report all found defects to the feature team
3. Correct defects (or report them to obs. system)
4. Execute all tests necessary to verify corrections on new build
5. Deliver the feature to the integration stream

Continuous Integration and Build Process



The Result

- 42% less defects in the acceptance (FAT) test. That is either not introduced or solved in the coding phase.
- For large projects: indication that performance is increased significantly
- For all projects: Better employee satisfaction, better quality and more satisfied customers.

How to test…

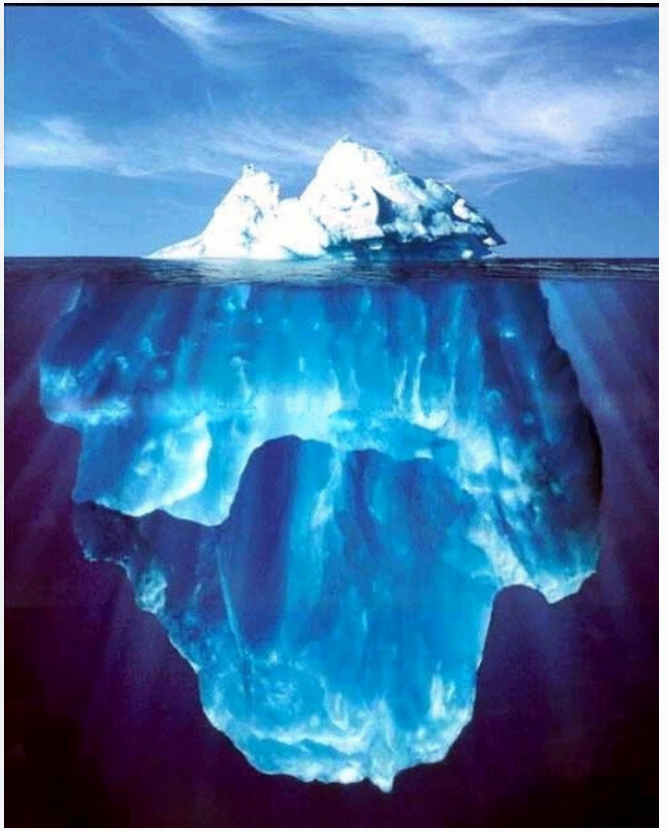
One size doesn't fit all. Depends on:

- Contract
- Customer
- Users
- Product/project type
- Architecture
- Platform
- Schedule
- Ressources
- Phase in the development
- Etc.

Risk-driven testing

Probability x impact = exposure

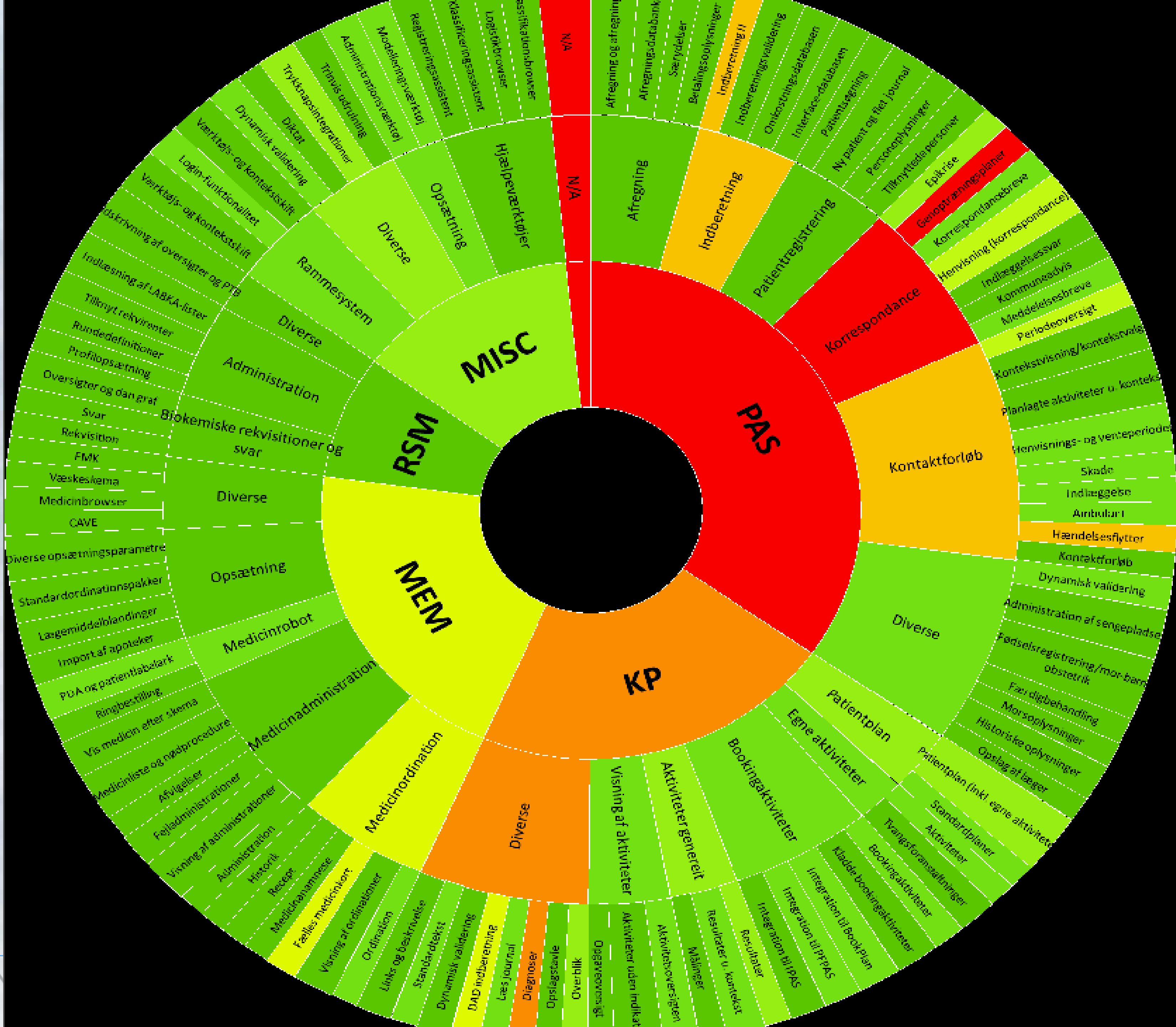
- Product vs. project risk
- Tells us where to focus our test effort
- Goal: Optimal test within the given time frame
- Risk changes constantly
- Both functional and non-functional risk



Risk analysis

Risikovurdering, PAS

Område	Funktionalitet/egenskaber	Impact	Probability	Exposure	Kommentar
Afregning					Ingen ændringer indenfor afregning
	Afregning og afregningsfejl	4	1	4	
	Afregningsdatabanken	3	1	3	
	Særydelser	3	1	3	
	Betalingsoplysninger	3	1	3	
Indberetning					
	Indberetning II	3	4	12	PAS146
	Indberetningsvalidering	3	3	9	
	Omkostningsdatabasen	2	1	2	
	Interface-databasen	3	2	6	
Patientregistrering					
	Patientsøgning	5	1	5	
	Ny patient og flet journal	5	1	5	
	Personoplysninger	2	1	2	
	Tilknyttede personer	2	1	2	
Korrespondance					
	Epikrise	3	2	6	obs 82450
	Genoptræningsplaner	3	4	12	PAS157 TSP0253 og TSP0254
	Korrespondancebreve	2	2	4	
	Henvisning (korrespondance)	4	2	8	PAS167
	Indlæggelsessvar	1	1	1	
	Kommuneadvis	1	2	2	pga ny kommuneliste til Genoptræningsplan - sikring af at den i
	Meddelelsesbreve	3	1	3	
Kontaktforløb					
	Periodeoversigt	5	2	10	PAS159 + obs 96351(TSP0084)
	Kontekstvisning/kontekstvalg	5	2	10	
	Planlagte aktiviteter u. kontekst	3	4	12	JBM har rettet en defect som er omfattet af det kommende CAF denne og Integrations til PFPAS have samme exposure
	Henvisnings- og venteperioder	4	2	8	PAS159 (TSP0092)
	Skade	4	2	8	



Effective test I

1. Test early
2. Test risk based
3. Focus: Know the purpose of the test
   - Requirement verification?
   - Validation against customer needs?
   - Smoke test (common user scenarios)?
   - Finding errors?
   - Risk assesment?
   - Learning?
   - Something else?
4. Focus: Know what you are testing
   - Limit the scope (funktionality, use cases, quality aspect etc.)
   - One thing at a time

Effective test II

4. Attack the code from as many different angles as possible:
   - Collect good test ”attacks” and angles
   - Learn formal test techniques
   - Do pair-testing
   - Do team-brainstorming sessions
   - Do a bug-hunt
   - Don’t rely just on scripted tests
   - …anything to break the habit

Effective test III

5. Remember the big picture:
   - Functionality across stories
   - Interaction with other features
   - Requirements
   - Operative scenarios
   - Non-functional abilities

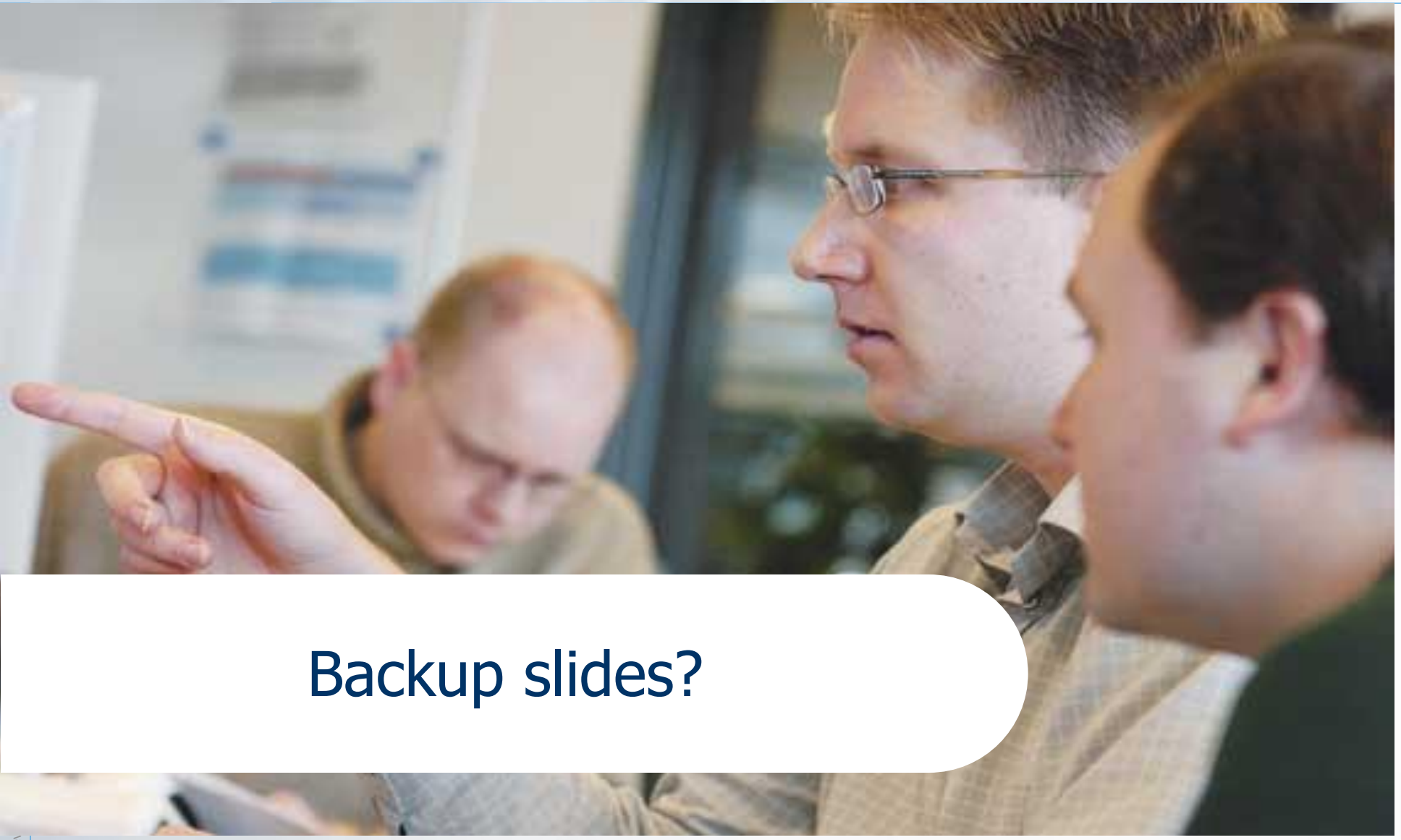
6. Positive test is not enough – remember negative test:
   - Look for problems and you'll find them
   - Mind set: "How do I break this?". Be malicious.
   - Challenge hidden assumptions
   - Don't assume rationel user behaviour
   - Stress the system (large amounts of data, complex data, long sequences, concurrent use etc.)

7. …finally remember: Bugs are social creatures

Test automation

- Project specific strategies
- Automatic and manual test supplement each other. Both are relevant in most cases
- Automatic test is central in agile development
- TDD – up to the individual developer
- In the end, automatic vs. manual test is a matter of pure cost/benefit
- A poor test doesn't become better by being automated
- Please don't be too impressed by coverage metrics

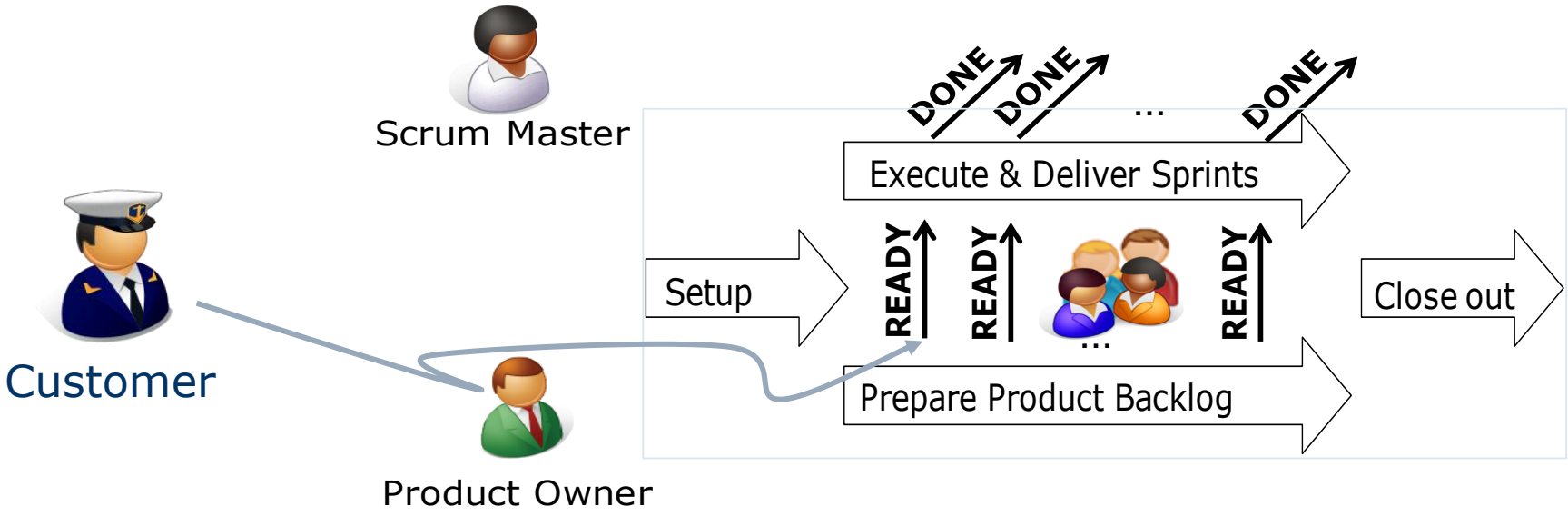
Questions?



Backup slides?

Understanding Scrum success

READY and DONE is simple to understand but hard to do



Customer participation is a great challenge compared to traditional contracts

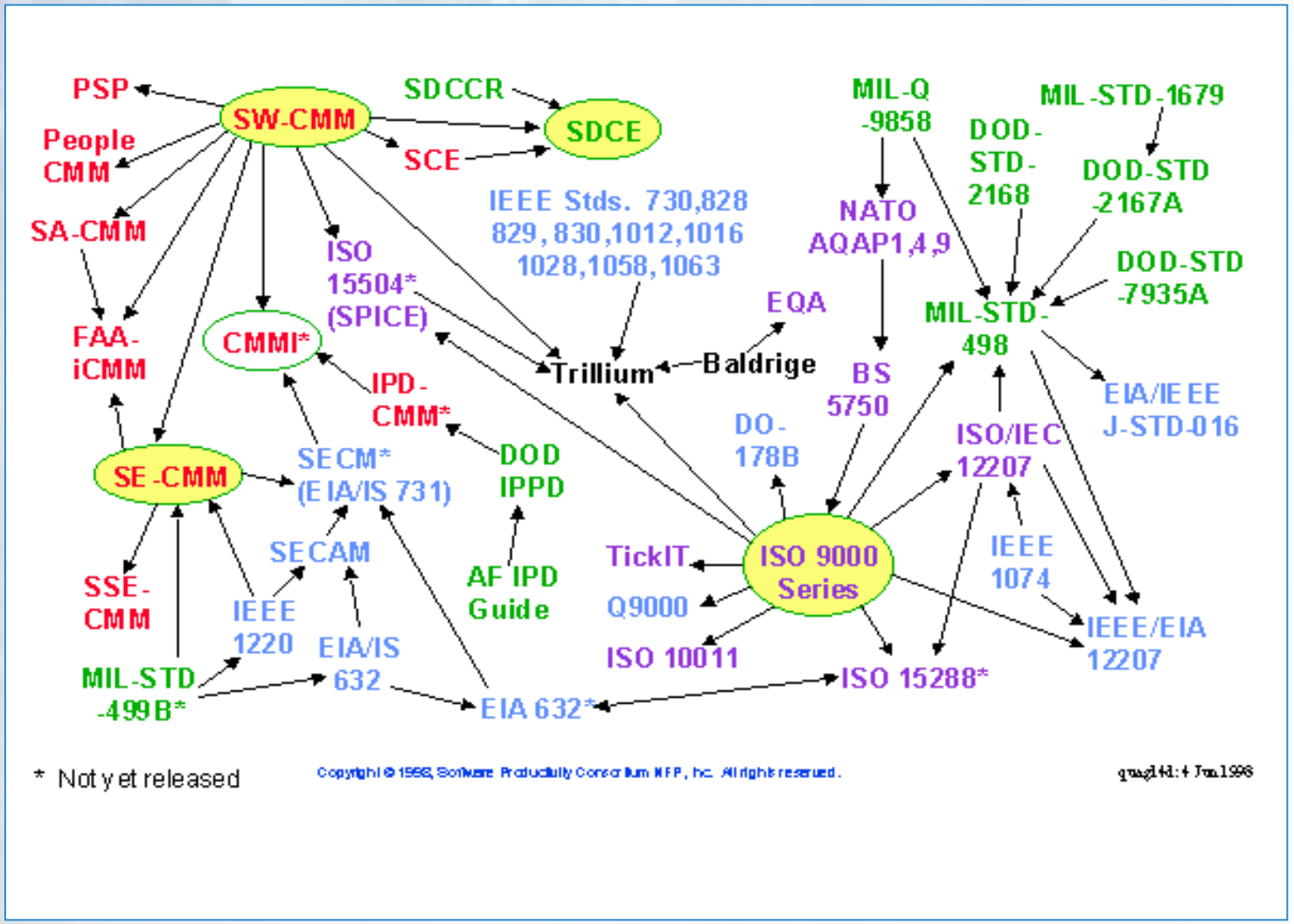
Sources for CMMI

CMMI kilder

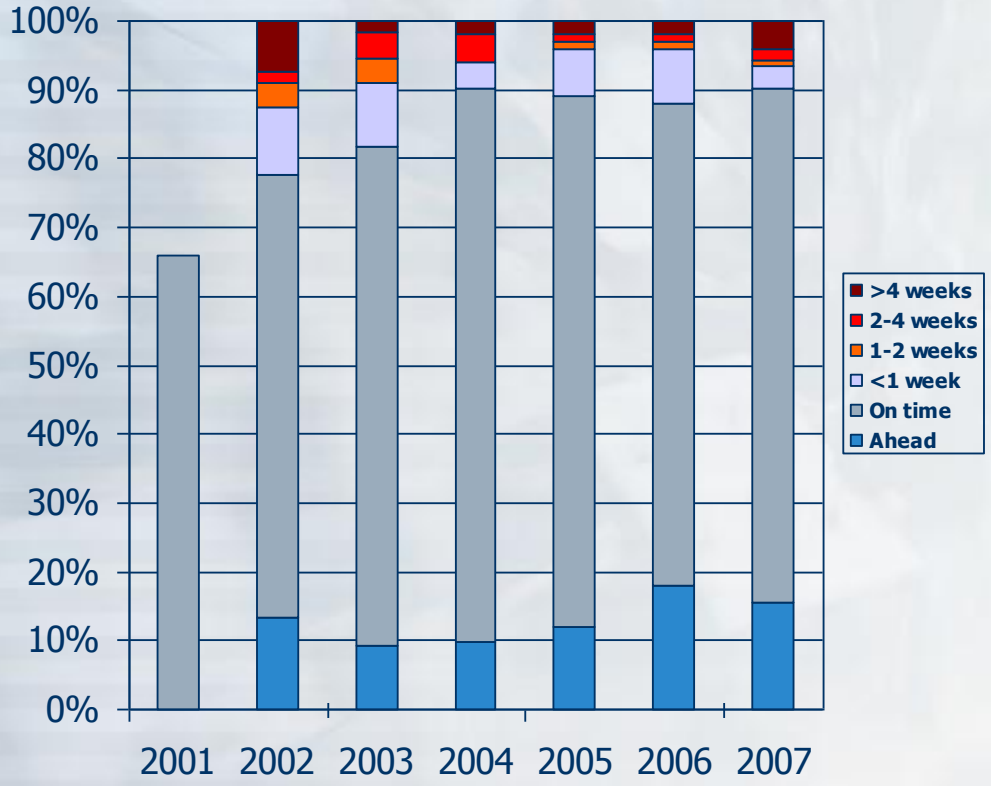
Capability Maturity Model for Software V2, draft C (SW-CMM)

EIA Interim Standard 731, System Engineering Capability Model (SECM)

Integrated Product Development Capability Maturity Model, draft V0.98 (IPD-CMM)



CMMI improved delivery on time and quality



- Our ability to deliver on time has increased from 66% in 2001 to 90% in 2007
- Main reasons for delay are bad estimation or late changes to requirements

- Quality also increased:
- From Q3 2002 to Q4 2003 code review was increased from 17,8% to 41,2% of newly written code
- Today we use a combination inspection and review