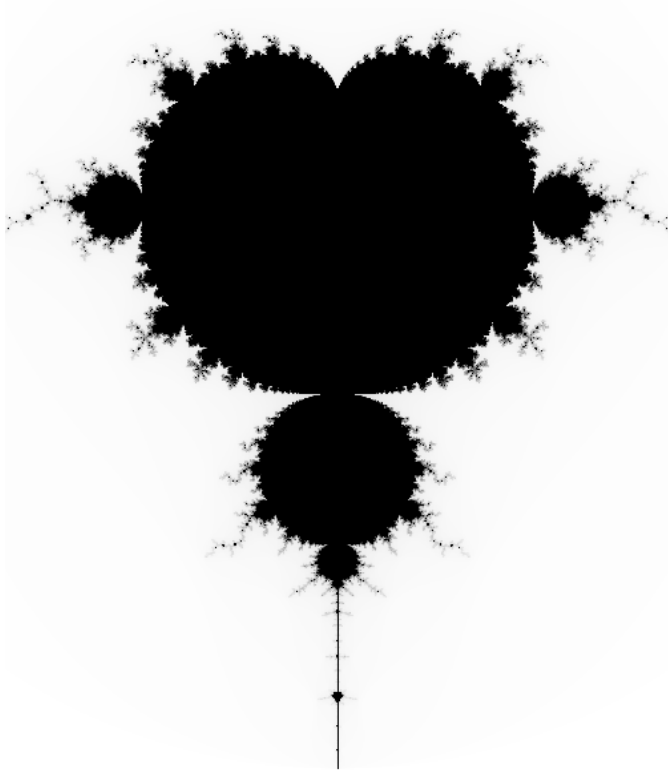


**The reason for human
domination on this
planet is our
extraordinary ability to
adapt to **change**.**

**But why do we try to
fight this ability within
software development?**



Lean Configuration Management



Jens Norin



Agenda

- **What is Configuration Management?**
- **What do agile methods say?**
- **Do they combine?**
- **Lean CM – the recipe**
- **Show me the money**

Softhouse

- **Committed to software technology and business innovation**
- **Experts in software development methods and mobile applications**
- **~20 Certified Scrum Masters**



CM history

- Evolved since the beginning of Software Engineering in the 60's
- Satisfies a need to take control of **changes**
- and handle coordination between individuals and products



CM definitions

- **IEEE – Std-729-1983**

- “Configuration Management is the process of identifying and defining the items in the system, controlling the **change** of these items throughout their lifecycle, recording and reporting the status of items and **change** requests, and verifying the completeness and correctness of items.”

- **CMM**

- “...Software Configuration Management involves identifying the configuration of the software (i.e., selected software works products and their descriptions) at given points in time, systematically controlling **changes** to the configuration, and maintaining the integrity and traceability of the configuration throughout the software lifecycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., the compiler)...”

- **RUP**

- The task of defining and maintaining configurations and versions of artifacts. This includes baselining, version control, status control, and storage control of the artifacts.



SOFTHOUSE

Traditional CM Practices

- Identify configuration items
- Version control of configuration items
- Release management
- Build management
- Control **changes**
- Track status
- Audit and review

CM in Agile Methods

- **Scrum**
 - No explicit mention of CM routines
- **XP**
 - No explicit mention of CM routines
 - However test driven development relies a lot on working CM routines
- **DSDM**
 - Does mention CM as necessary because of the dynamic nature of the method, i.e. frequent deliveries etc.
- **Unified Process**
 - Describe a heavy and rigorous CM process
 - UP is not considered agile in this context

Lean Principles

- **Eliminate waste and Expose problems**
 - Originates from Lean Manufacturing at Toyota
 - Brought to Software Engineering by Mary and Tom Poppendieck
 - According to Lean Principles CM can be considered as waste, since it doesn't add anything to the end product
 - However no CM also means chaos projects
 - Use appropriate amount of process

Agile Manifesto - history

- **17 software development anarchists met 2001 and formed “Agile Alliance”**
- **Representatives from XP, Scrum, DSDM, Crystal, FDD, Pragmatic Programming etc.**
- **Agreed that an alternative to heavyweight, document driven processes had to be considered**
- **Coined “Agile development”**

Agile Manifesto

- 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.

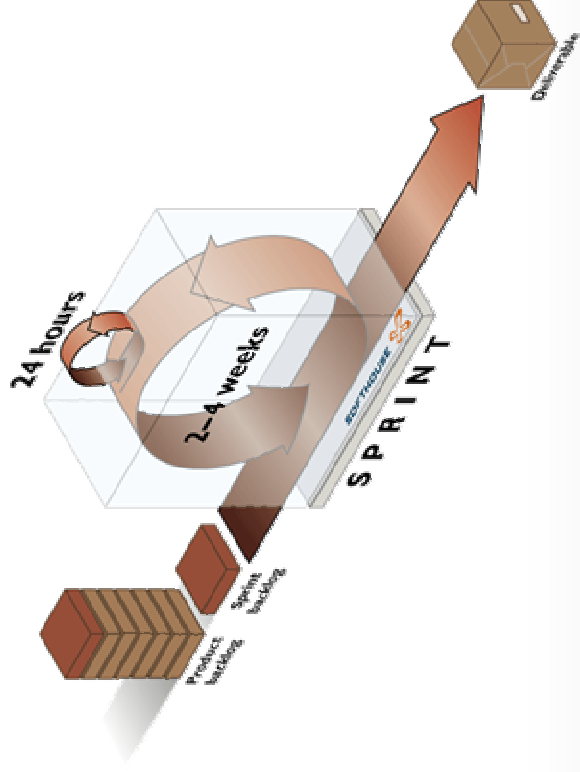


Roles in a traditional project

- **Project manager**
- **Requirement Analyst**
- **Architect**
- **Developer**
- **Tester**
- **Deployment Manager**
- **CM**
- **Etc...**

Roles in a Scrum project

- **Product owner**
- **Scrum Master**
- **Team members – a self organizing team**



Are Traditional CM and

Agile development

• Traditional CM handle **changes** by **compatible team**

- Agile development handle **changes** by responding and adapting to them
- There also seem to be little focus on CM in the agile community
- Can these two disciplines really co-exist?
- Does one make the other obsolete?
- Or even impossible?

Important since!

CM in the agile community

- **Make no mistake about it – CM is more important than ever!**
- **Embrace **change** – is not possible without appropriate CM routines and tools**
- **CM routines should support and enhance agile methods**

Agile practices that require

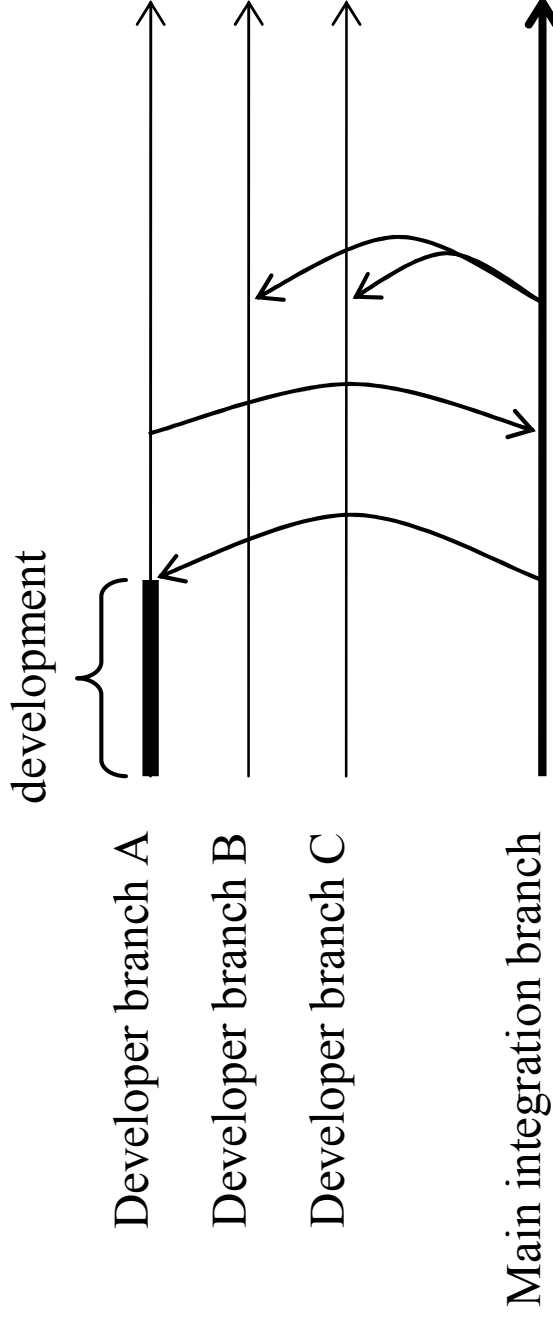
CM support Agile practices taken from:

- Engineering practices from XP
- Project practices from Scrum

“Embrace Change”

- Is a common phrase within the agile community
- Handling **changes** is a key concept when describing Configuration Management

Copy/merge model



Continuous Integration

- Tools that automatically builds, baselines, tests and reports code **changes**
- Triggers as soon as someone has integrated a code **change**
- Provides an automatically updated health status of the code at all times including:
 - Code quality
 - Build warnings (no errors allowed)
 - Automated tests

Automatic testing

- **Tools that are triggered when there is something to test, and is automatically performing:**
 - **Regression tests**
 - **Unit tests**
 - **Business acceptance tests**
 - **And even GUI tests**
- **Taken one step further will give Test Driven Development (TDD)**

Collective Code Ownership

- **Strong code ownership**
 - Breaks code into modules
 - Assigns each module to an owner
 - Only allows owner to make changes
- **Weak code ownership**
 - Breaks code into modules
 - Assigns each module to an owner
 - Allows anybody to make changes
- **Collective code ownership**
 - No individual ownership of modules
 - Code base is owned by the team
 - Anyone make changes anywhere

Refactoring

- Frequent **changes** and collective code ownership eventually degenerates the code
- Repeatedly refactor the code to keep the structure and architecture simple

Iterative and incremental

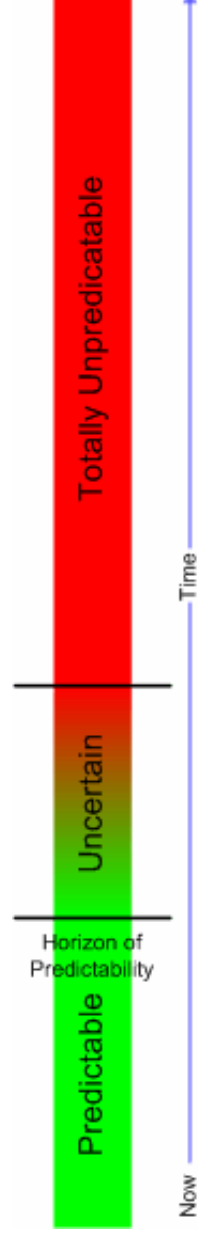
- **development**
Split project into many smaller parts/deliveries/iterations
- ...that adds up incrementally for each iteration

Retrospective

- **What did we do well?**
- **What did we learn?**
- **What should we do different the next time?**
- **What still puzzles us?**

Adaptive planning

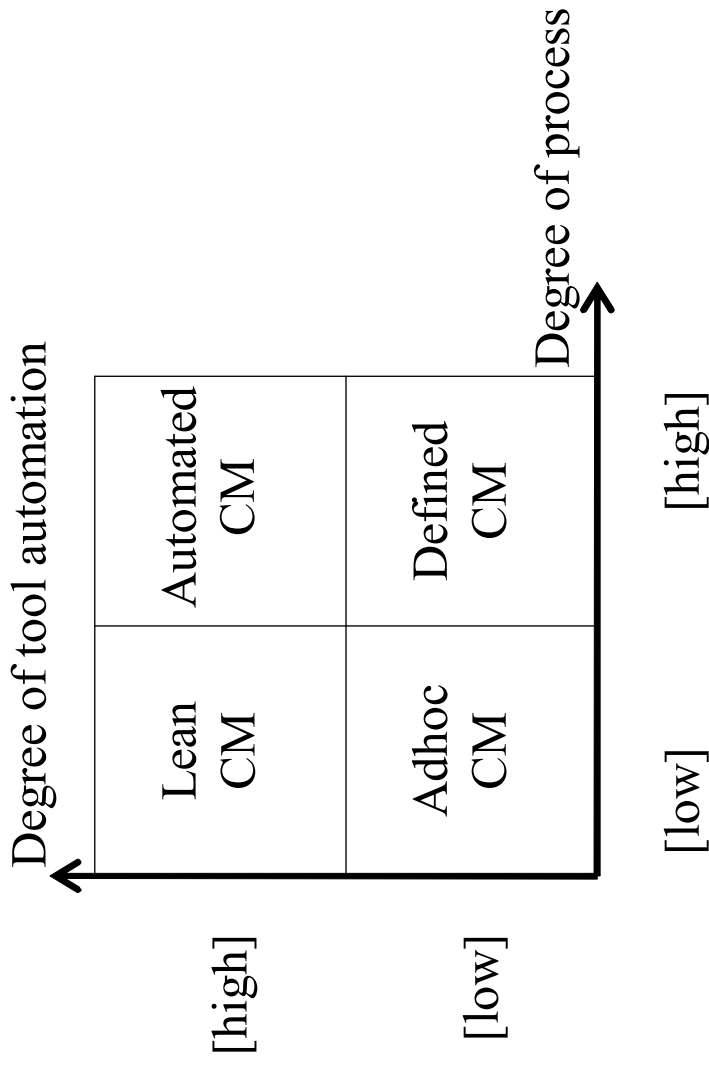
- **Feedback-driven instead of forecast-driven**
- **Receptive to change instead of trying to be predictive**
- **Horizon of predictability**



Frequent releases

- **Short feedback loops to customers**
- **Being able to respond quickly to changes**
- **A lean process enables short iterations**

CM performance



Important slide!

Ad Hoc CM

- **Lean process**
- **No tool automation**
- **Small organisation**
- **Potential CM problems**
- **Productive teams**

Defined CM

- **Heavy process**
- **No tool automation**
- **CMM, RUP or some other heavy document driven process**
- **Dedicated CM role**
- **Controlled changes and CCB routines**
- **Unproductive teams**

Automated CM

- **Heavy process**
- **Tool automation**
- **Heavy process is effectively blocking possible benefits from tool automation**
- **Unproductive teams**

Lean CM

- **Lean process**
- **Tool automation**
- **Share same values as Agile methods and Lean principles**
- **Disqualifies RUP and CMM**
- **Hyper productive teams**

Important since!

Creating Value with Lean

CM. To increase ROI we must increase productivity by:

- Reduce software development effort
- Streamline development processes
- Increase customer value

Important since!

Reduce software

development effort
• Reduce development costs

- Eliminate investments in features not valuable to customers
- *Frequent deliveries*

Streamline development

processes

• Streamline the process

- Make better use of resources
- *Remove the dedicated CM role*
- *Cancel formal CCB routine and introduce backlogs of scrum*
- *Cancel CM audits*
- *Continuous Integration*
- *Automatic testing*
- *Collective code ownership*

Increase Customer Value

- Find out what represents value to the customer
- Often the customer doesn't know this themselves
- *Close collaboration with the customer*
- *Frequent deliveries*

Conclusion

- **Cancel the formal CCB**
- **Cancel the CM audit**
- **Automate by introducing continuous integration and automatic testing**
- **Remove the dedicated CM**
- **Transfer CM skills by pairing**

Important slide!

Thanks for listening!

Be sure to visit my blog
<http://intellijens.se>

IntelliJens