zühlke
DIE DENKFABRIK.

Moving from a Plan Driven Culture to Agile Development

Lessons Learned

ICSE 2005, St. Louis, 20-May-2005

Michael Hirsch
Zühlke Engineering AG
hm@zuehlke.com

Topic of this Talk

In scope:

- Experiences and lessons learned in moving from “plan driven” to “agile” development practices

Out of scope:

- Advantages (or disadvantages) of agile development
- When to use an when not to use agile development
- Differences between particular agile methods such as XP, Crystal, Scrum, agile RUP configurations, etc.

Speaker and Company Background

- Active in software development since 1981 in various roles, including developer, architect, project manager and mentor
- Background mostly in control systems, information systems, and embedded systems for various industries
- Partner and process manager with Zühlke Engineering in Zurich / Switzerland
- Zühlke Engineering:
  - Development of custom software and systems
  - Offices in Zurich, Frankfurt and London with approx. 250 employees
  - OOA/OOD/OOP since 1992, incremental and agile development based on RUP since 1998
  - Typical projects are 1 to 20 person years, with teams of 3 to 12 developers

Contents

Fundamental Assumptions

Plan driven

All desired properties of the end product can be known and precisely specified *before* construction of the end product begins.

It follows that projects are predictable and therefore can and should be planned in detail from start to end. A deviation from the plan is a sign of sloppy work in earlier stages of the project.

Analogy:
Building a Bridge

Agile

The desired properties of the end product *can not be known* until at least part of the solution is built.

It follows that projects are in principle unpredictable and the development process must be optimized for situations where the detailed outcome cannot be known in advance.

Analogy:
Exploring where and how to cross a river

Plan Driven versus Agile

Discipline	Plan Driven	Agile
Process Model	Waterfall	Iterative & Incremental
Project Planning	Detailed project plan from start to end, created early in the project	Coarse grained plan for overall project, detailed plans per iteration
Requirements Engineering	Dedicated specification phase Initial requirements are signed off; rigorous change request regime afterwards Comprehensive requirements documents, often part of a contract	Requirements evolve over the course of a project No or very relaxed change request regime Easy access to customer rather than reliance on comprehensive require-ments documentation

Plan Driven versus Agile

Discipline	Plan Driven	Agile
Architecture & Design	Comprehensive architecture and design specs before implementation begins Architecture and design try to accommodate for future extensions	Minimum upfront architecture and design work Architecture is validated iteration by iteration Tradeoff between YAGNI and DOGBITE
Implementation	Programming work concentrated in “Implementation” phase Programming work may be contracted out (preferably to low cost countries) Programmers are assigned to subsystems	Programming work spread out over entire project Pair programming (ideally) Collective code ownership

Plan Driven versus Agile

Discipline	Plan Driven	Agile
Testing	Testing phase at the end of the project Tests are designed and executed by test specialists	Testing spread out over entire project Functional tests are specified and executed by end users
Quality Assurance	Formal QA role QA role is responsible for formal and informal reviews, development process, configuration management, testing, code inspections, ...	No explicit QA role No formal reviews Attitude: quality is the result of how we work here

Contents

How are Stakeholders affected by switching to Agile Development ?

Customer Organization

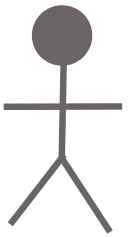
Sponsor

Business Analyst

End User

Development Organization

Project Manager



Developer

Sponsor

- **Plan driven:** Buys a system with fixed functionality for a fixed price which will be delivered on a fixed date (so he/she thinks...)
- **Agile:** Buys a system without knowing all three of functionality, price and delivery date. Requires a lot of trust in development organization.
- **Typical objections against agile development:**
  - I can't buy something if I don't know exactly what it will be and how much it will cost
  - (Can I trust these newfangled agile hackers ?)
- **How to overcome objections:**
  - Appeal to track record of plan driven projects
  - Appeal to analogy with investment in shares
  - Deliver more than promised in all iterations
  - Work like in a glass house

Some Managers perception of plan driven and agile methods

Plan Driven

Some Managers perception of plan driven and agile methods

Plan Driven

Agile

Business Analysts & End Users

- **Plan driven:** Prepare specs at begin of project and then send them off to development team. Occasional questions from development team.
- **Agile:** Constant dialog with development team, participation in iteration planning and assessments, writing and executing test cases => more time needed !
- **Typical objections against agile development:**
  - I have no time
  - The developers should know this
  - (I don't want to take responsibility for system scope)
- **How to overcome objections:**
  - Make sure BAs and EUs have the time they need
  - Appeal to "get it right the first time - less rework"
  - Communicate the customers role *before* the project starts
  - Assign BAs/EUs with authority (and willingness) to make decisions

Minimum Customer Involvement for Agile Development

Discipline	Customer Responsibilities
Requirements	Shared responsibility between customer and development organization. Customers role: (1) Must provide core requirements (2) May or may not participate in documenting requirements (3) Must review written requirements
Test	(1) Write and execute functional tests (2) Provide feedback to every development release
Configuration & Change Mgmt	(1) Issue lightweight feature change requests
Project Management	(1) Assign priorities to requirements, change requests, and bugfixes (2) Participate in planning-, progress- and iteration assessment meetings

Project Manager

- **Plan driven:** Prepares project plan at begin of project. Occasional changes to the plan.
- **Agile:** Prepares a detailed plan for each iteration. Must keep 4 plans in his/her mind: overall project plan, plans of previous, current, and next iteration
  => places higher demands on project managers !
- **Typical objections against agile development:**
  - We can't start unless we have all requirements in detail
  - Agile equals chaotic; agile projects can't be controlled
  - (I hate uncertainties)
  - (I don't like personal communication)
- **How to overcome objections:**
  - Most stated objections are based on misunderstandings
  - Most unstated objections are a sign that this person shouldn't be a project manager, anyways

Developer

- **Plan driven:** Works on assigned subsystem(s). Heavy reliance on written interface specifications. State of personal work known to project manager (only).
- **Agile:** Heavy reliance on personal communication, state of personal work known to entire team, daily pressure to deliver working code
- **Typical objections against agile development:**
  - 2 - 4 weeks are too short to program something useful
  - (I don't like that everybody on the team knows the state of my work)
  - (I don't like personal communication)
- **How to overcome objections:**
  - Different skill levels of team members are OK, as long as everybody permanently works on improving his / her skills
  - People without a minimum willingness to communicate shouldn't be software developers in first place

Contents

Lessons Learned

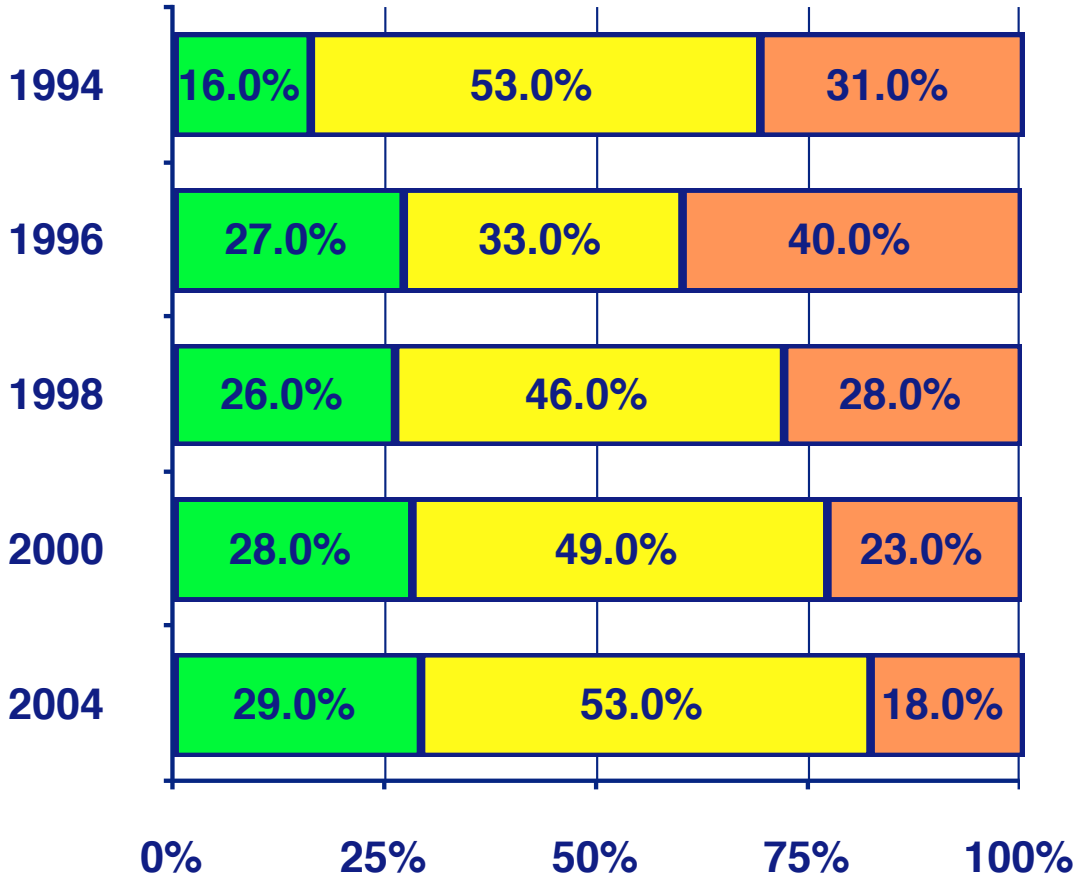
1. The strongest motivator to switch to agile development is failure in previous projects
2. Expect resistance from Software Engineering Process Groups
3. Avoid “single stage” contracts
4. Don’t write off change requests

Lesson 1 - Motivators to Switch

The strongest motivator by far to switch to agile development is failure in previous projects

- Case in point 1: Railway dispatch system
  Project aborted after 20 month of paper-based specification work and the contractor could still not come up with a reasonable estimate of overall project cost
- Case in point 2: Income tax system
  Project aborted after 18 months of “classical” (meaning: very hierarchical) project management and insufficient communication among stakeholders
- Other motivators: tight budgets, tight schedules, lack of alternatives (“last bet”), alternative to offshoring

The CHAOS Authority Approves



Average cost overrun:

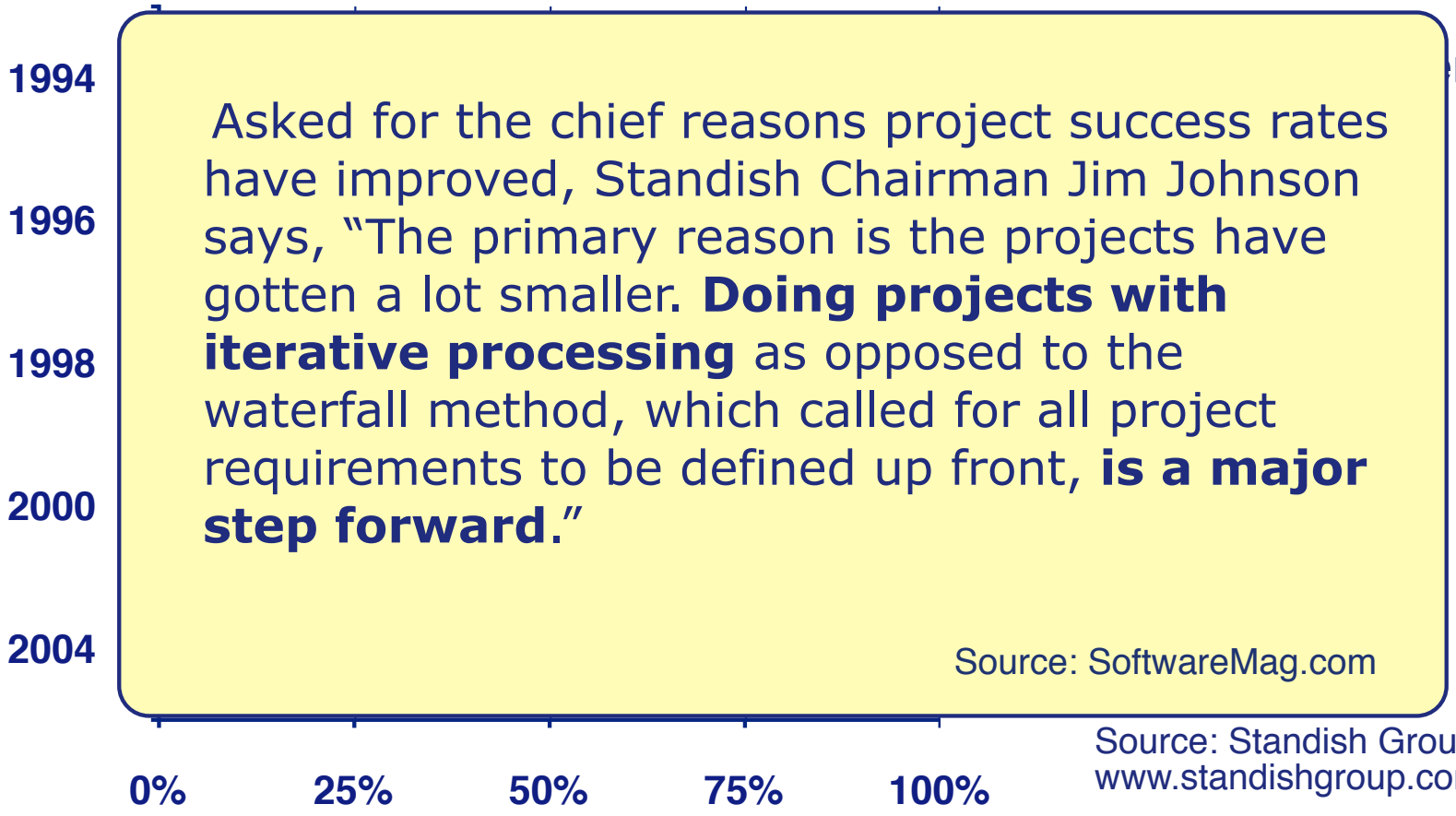
1994: 180%

2004: 43%

Successful
Challenged
Failure

Source: Standish Group
www.standishgroup.com

The CHAOS Authority Approves



Asked for the chief reasons project success rates have improved, Standish Chairman Jim Johnson says, "The primary reason is the projects have gotten a lot smaller. **Doing projects with iterative processing** as opposed to the waterfall method, which called for all project requirements to be defined up front, **is a major step forward**."

Source: SoftwareMag.com

Source: Standish Group
www.standishgroup.com

Lesson 2 - Expect Resistance from SEPGs

Software Engineering Process Groups (SEPGs) in large organization are frequently opposed to agile development

Simply put:

- Large and heavy processes mean large and heavy SEPGs
- Lightweight and agile processes mean lightweight SEPGs with fewer members
- Most SEPG members have no hands-on experience in agile development

How to Overcome Resistance from SEPGs

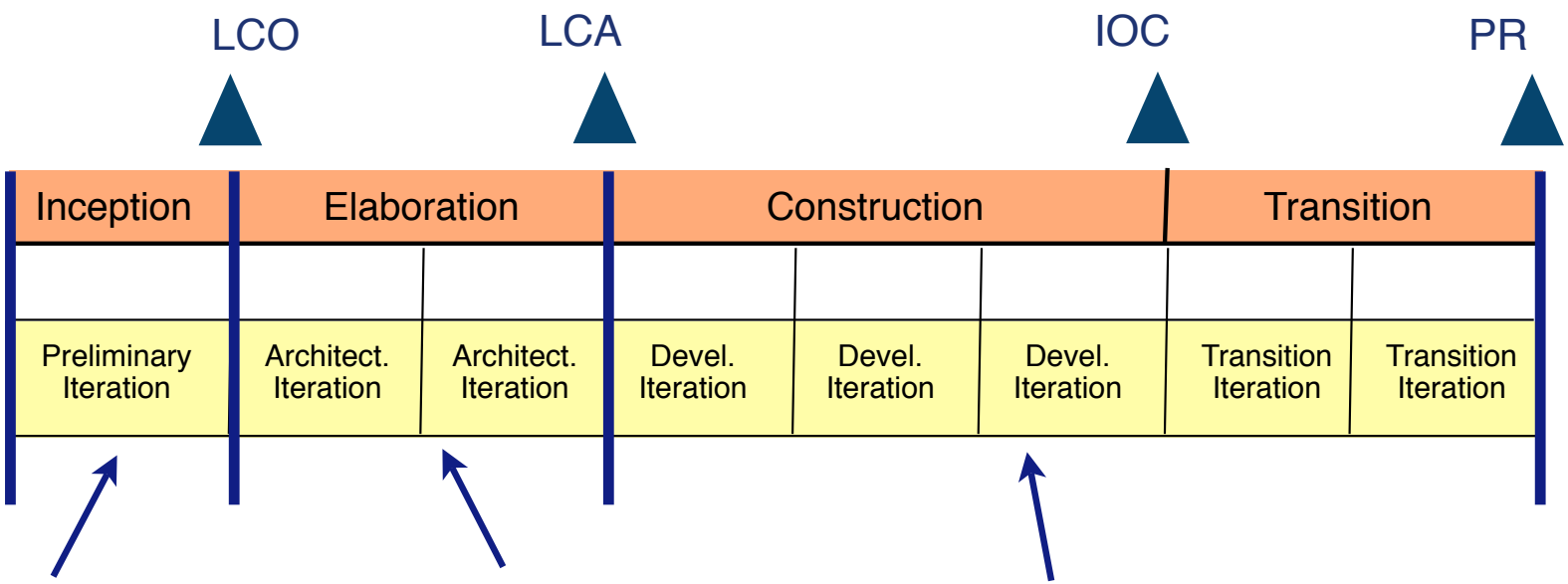
- Resistance is founded in fear of loss of job
- Things to do:
  - Short term:
    Appeal to the “market value” of SEPG members
    What would you rather see in your CV:
    - XP / Scrum / Crystal / RUP specialist, or
    - specialist in company XYZ proprietary process ?
  - Long term:
    Reorganize the SEPG and have it staffed with active project managers and developers who serve part time in the SEPG. Rotate SEPG members over time.

Lesson 3 - Avoid “Single Stage” Contracts

The traditional procurement approach - write requirements specs, invite for tender, select (usually) cheapest offer - does not work for agile development

- This approach does not work because know-how is not efficiently transferred from the requirements writers to the team developing the system
- Worst form of this approach: WTO (world trade organization) contracts:
  - Company which prepared requirements specs is prohibited by law to bid for the project
  - Bidders have no personal contact with requirements writers, questions are formulated and answered in writing only

Contracts for Agile Development



Inception contract
time & material, 5 to 15% of over-all effort

Elaboration contract
fixed price or time & material, 15 to 35% of overall effort

Development contract
fixed price, 50 to 80% of overall effort

Lifecycle model: Rational Unified Process (RUP)

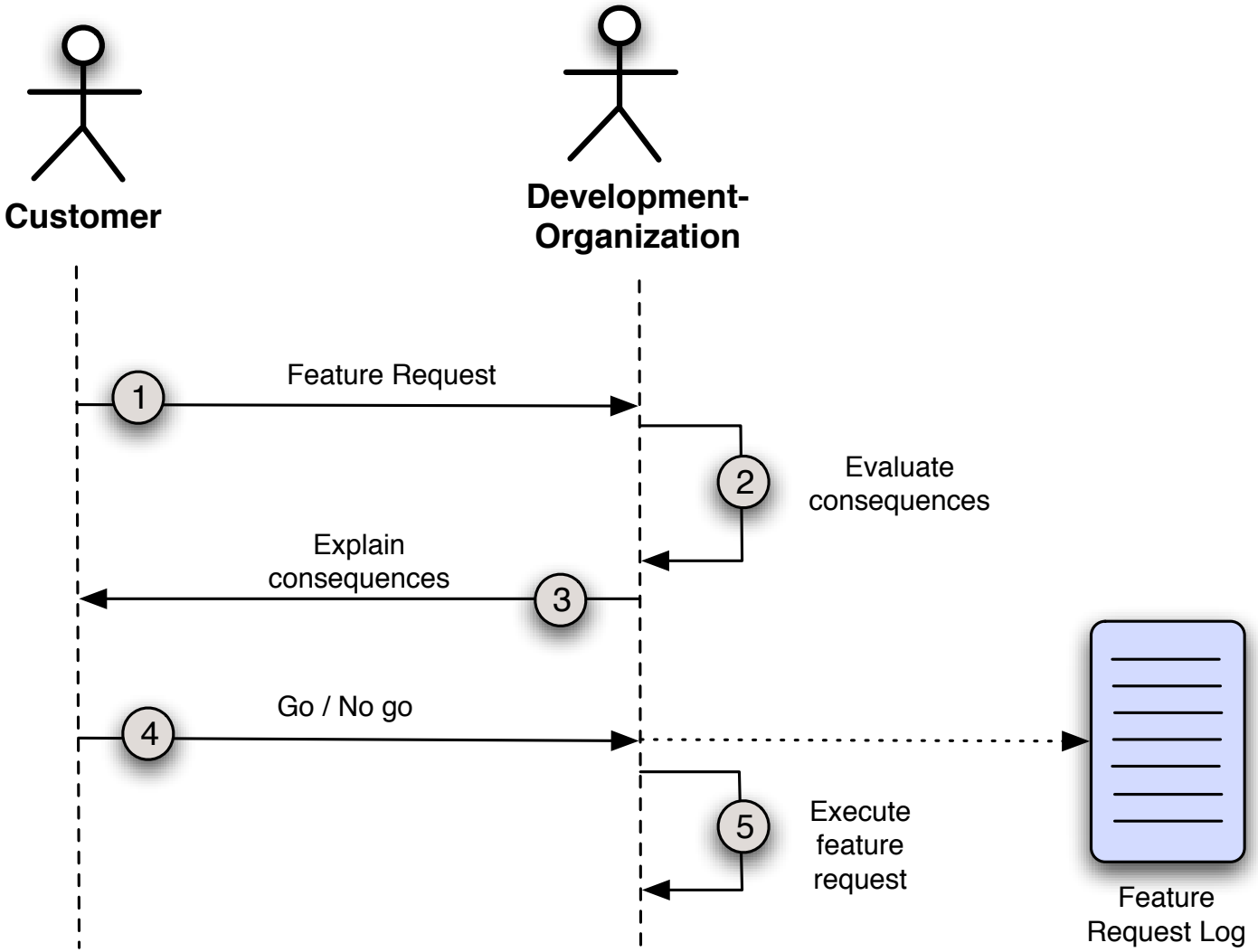
Milestones:
LCO = Lifecycle Objectives / LCA = Lifecycle Architecture / IOC = Initial Operational Capabilities / PR = Product release

Lesson 4 - Don’t Write Off Change Requests

Completely discarding change request processes would mean to throw out the baby with the bath water

- Without some form of change log, it is impossible to account for effort, cost and schedule at the end of a project
- Traditional “heavyweight” change request processes are not practical for agile development
- A lightweight change request process (“feature request process”) does not add overhead to a project and accounts for the turnarounds of a project

Lightweight Feature Request Process



Contents

Summary

- The biggest difficulty in moving from plan driven to agile development is dealing with objections, resistance and simply fear of the involved stakeholders
- Expect 1 to 4 years to make the transition, depending on the size of the organization
- Agile development is best viewed as an evolution of plan driven development and not as a radical departure from previous practices as seen by some advocates of agile development

Productivity Figures from Real World Agile Projects

Some Projects at Zühlke, 1999 to 2003

Project	Technology	Total Effort [PD]	Total LOC	Total Classes	Total Use Cases	FPs deliv-ered	Productivity [LOC/PY]	[FP/PY]
Airport MIS	Java J2EE, Swing	327	15'000	202	12	430	9'175	260
ATM Software	Win2k native, C++	2'940	128'000	2'180	16	2'415	8'705	165
Workflow Engine	Java J2SE, Swing	252	17'700	188	11	505	14'045	400
Product Data Mgmt System	Java J2EE, Web client	461	40'000	470	12	1'140	17'315	495
Protocol Converter	Java J2SE	215	26'400	507	9	725	24'560	675

PD = Person Days / PY = Person Years / FP = Function Points / LOC = (effective) Lines of Code

Thank you for your attention !

Questions ?

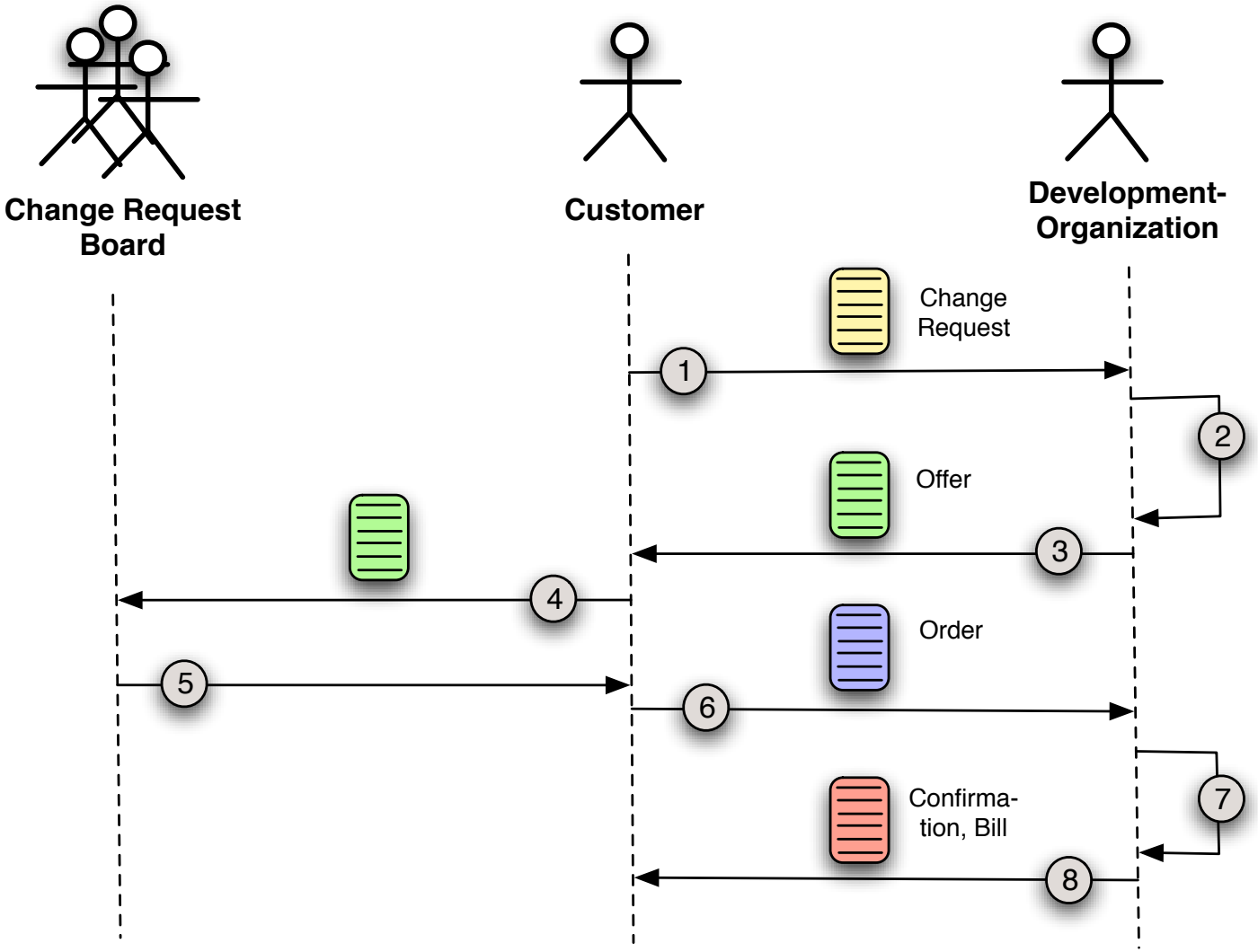
Michael Hirsch
Zühlke Engineering AG
hm@zuehlke.com

some things are perfect,
others need engineering.

zühlke
DIE DENKFABRIK.
www.zuehlke.com

Additional Material

Typical Traditional Change Request Process



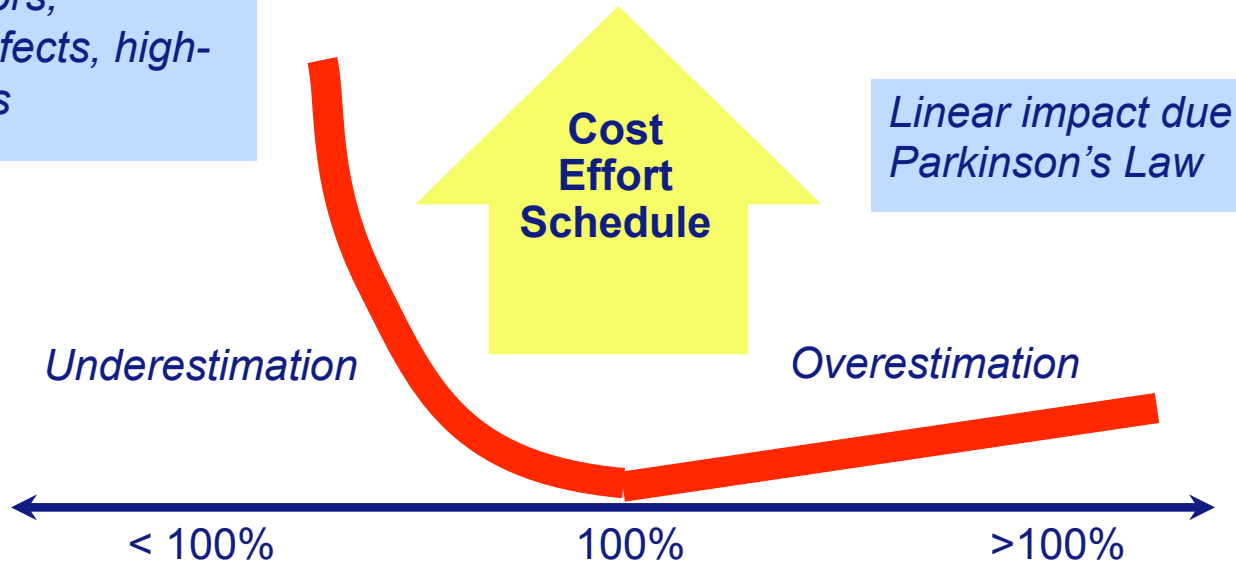
Lesson 5 - Good Estimation Matters

Most organizations have a bad track record concerning estimation accuracy. The introduction of agile methods is a good opportunity to improve the estimation process, too.

- There is a systematic tendency to underestimate projects
- Underestimation (and overestimation) are not “free”
- Bad estimation practices are encouraged by typical company specific plan driven development processes
- The agile community has a more realistic view on the limits of estimation accuracy

Consequences of Estimation Errors

Non-linear impact due to planning errors, upstream defects, high-risk practices



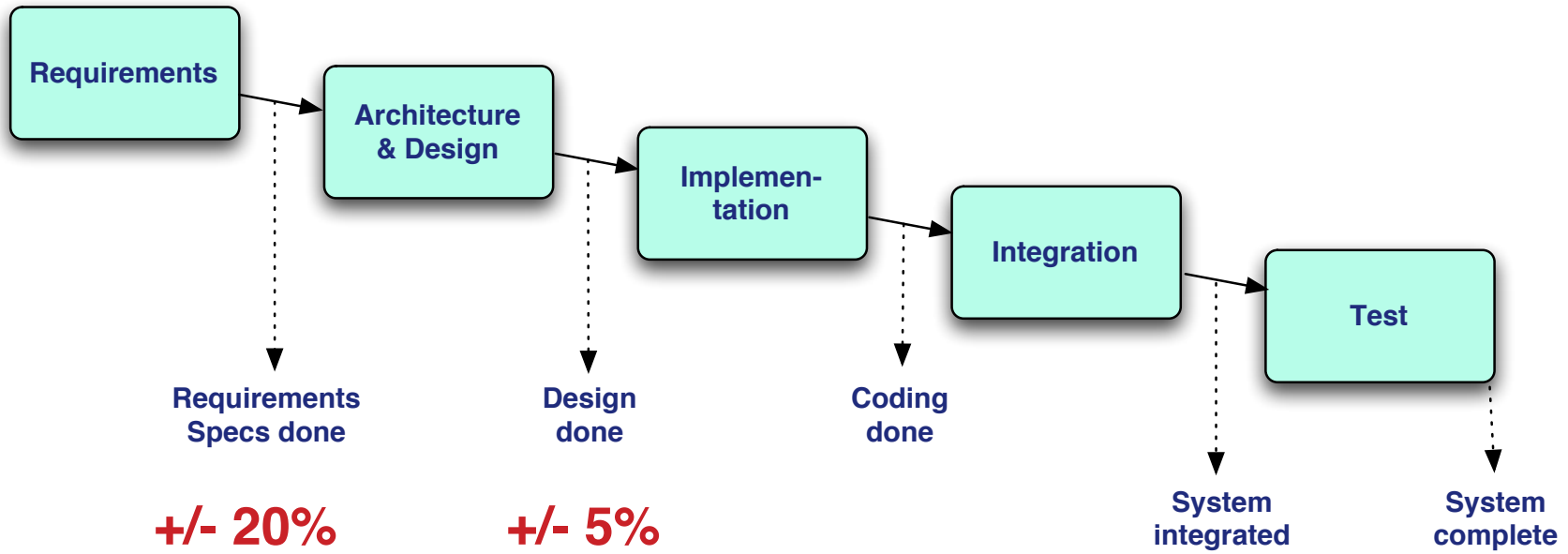
Linear impact due to Parkinson's Law

Target as a Percentage of Nominal Estimate

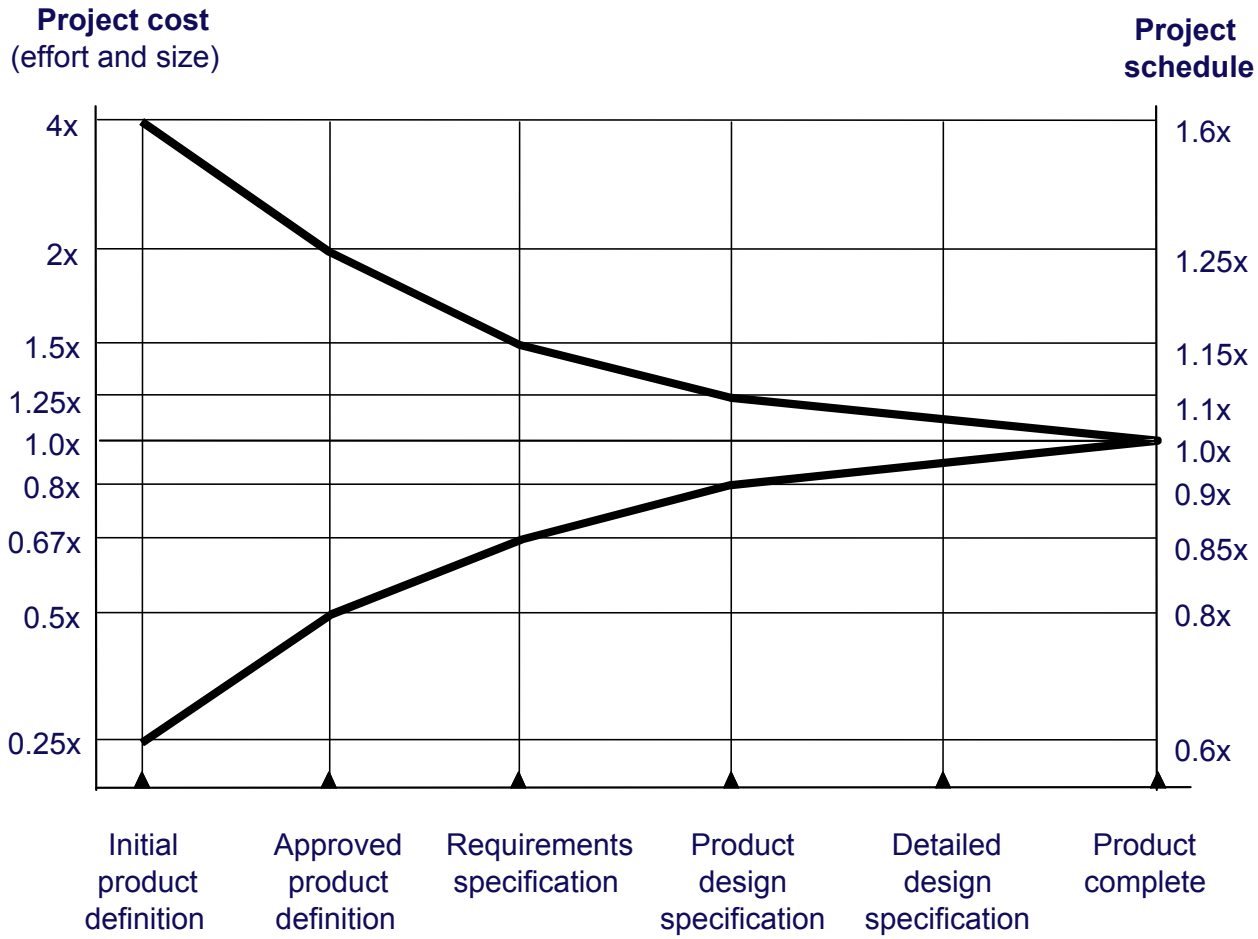
Source: Construx

The Illusion of Predictability

Requirements for accuracy of estimates in typical company specific plan driven process models:



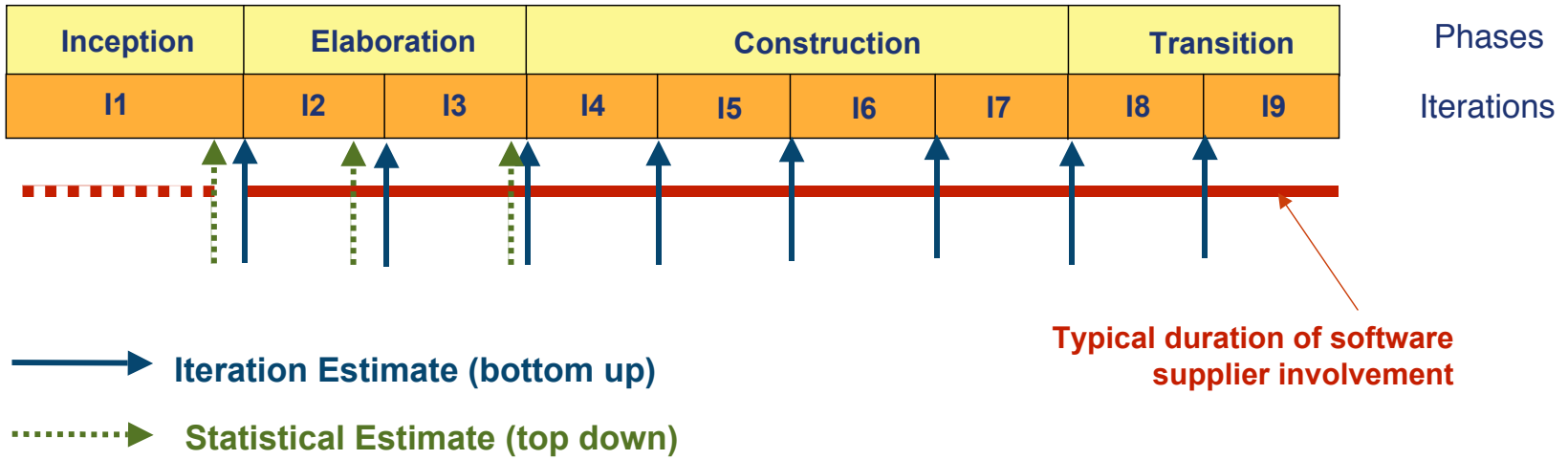
Limits of Estimation Accuracy



Source: (Boehm 1995)

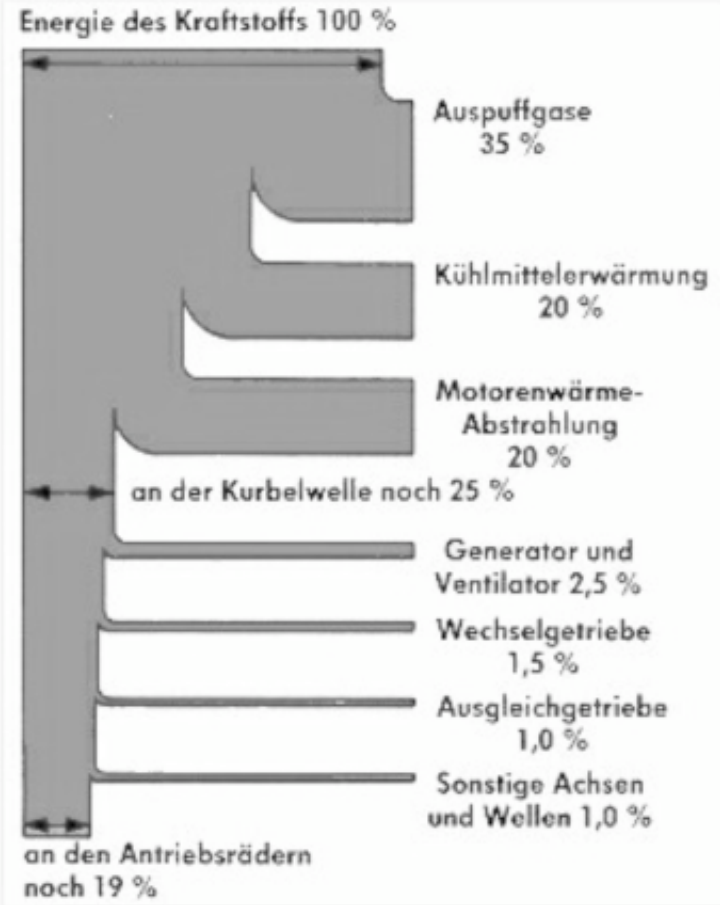
Estimation Process for Agile Projects

- Create estimates with statistical methods (e.g. COCOMO) early in a project, with “best case” and “worst case” scenarios
- “Bottom Up” estimate for each iteration



Life cycle model: Rational Unified Process (RUP)

Is Agile Development Here to Stay ?

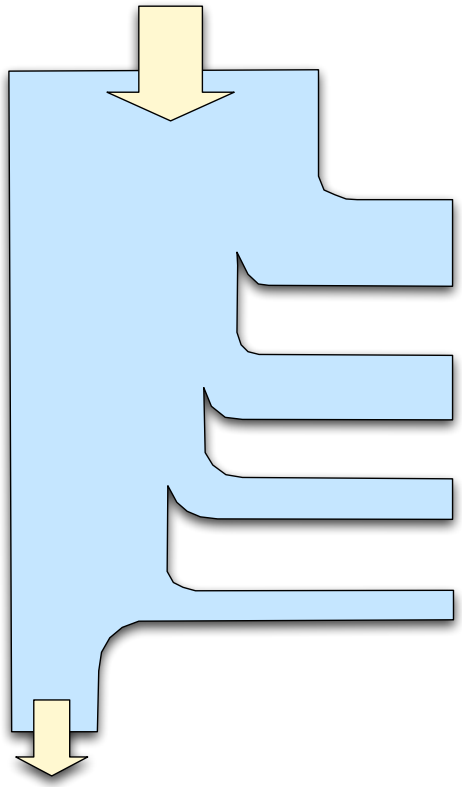


Energy Balance of a typical car engine

Of 100% of the energy in the fuel, only 19% is available at the wheels, the rest is wasted

The “Energy Balance” of Agile Development

Time and money invested into project



Negotiating contracts

Fighting over change requests

Too much documentation

Overspecifying requirements

Time left for building software

The “Energy Balance” of Agile Development