SIEMENS

Improving Productivity in the Development of Software-based Systems

SOFTWARE ENGINEERING 2010

February 24, 2010
Paderborn, Germany

Frances Paulisch
Head of the Siemens Software Initiative
Siemens AG, CT T
Munich, Germany

frances.paulisch@siemens.com

Siemens is an the integrated technology company – Corporate Technology is an essential part (networking, multiple-impact,...)

SIEMENS

Customers

Chief Technology Officer (CTO)

- Review innovation strategies
- Drive technology based synergies
- Secure innovation power
- Technology assessments
- Governance and guidance

Sectors / Divisions

Industry

Energy

Healthcare

Regions

Siemens IT Solutions and Services (SIS)
Siemens Financial Services (SFS)

Chief Technology Office (CT O)

- Direct support of CTO

Corporate Research and Technologies (CT T)

- GTFs with multiple impact
- Pictures of the Future
- Accelerators

Corporate Development Center (CT DC)

- Software Development-partner for the sectors

Corporate Intellectual Property and Functions (CT I)

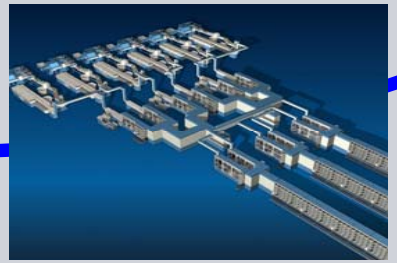
- Intellectual Property services & strategy
- Standardization, environmental affairs
- Global information research

Corporate Technology (CT)

Siemens is one of the World's Largest Software Companies …

SIEMENS

… but is usually not recognized as a "software company" because most of its software is embedded in systems



- Approximately 20,000 software engineers worldwide
- 60% of our sales come from products that contain software
- Software is an integral part of many of our products and systems

SIEMENS

Outline

- **Productivity – the eternal quest**
- **Topic-specific views on productivity or return-on-investment**
  - **Requirements Engineering**
  - **Architecture, Product Line Engineering**
  - **Model-Based Software Engineering, Testing**
  - **Agile Development**
  - **People**
- **Lean Software Development**
- **Closing**

Productivity

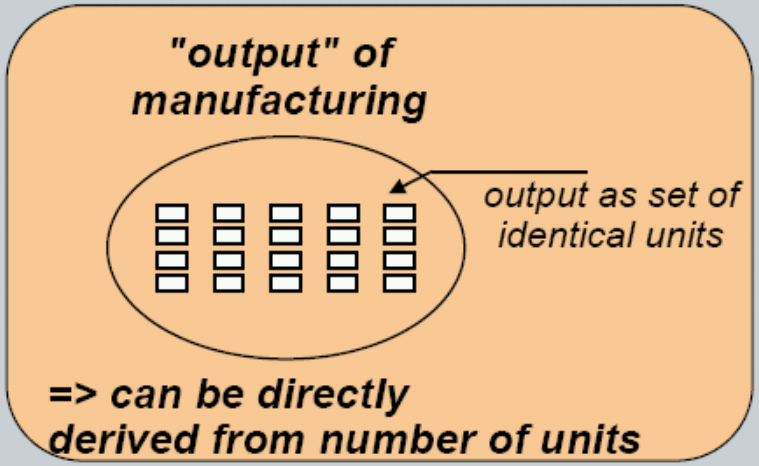
$$\text{Productivity} = \frac{\text{Output (value)}}{\text{Input (cost)}}$$

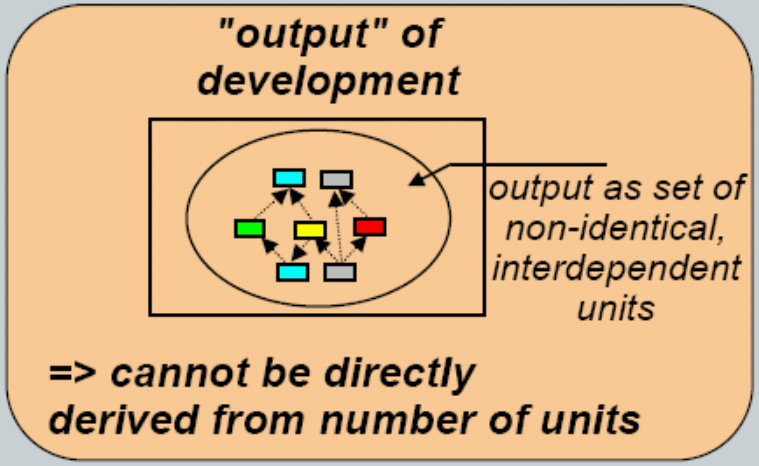
Value includes functionality as well as quality attributes (NFRs) valued by the customer.

Cost includes both direct costs and indirect costs (that lead to less risk, more predictable schedule/budget, early warning signs etc.)

Appropriate approach will depend on your business, organization, project context.

Certainly, you should start with a solid basis.





SIEMENS

Requirements Engineering

- Build the right product (have a good understanding of customer, use prototypes, strive for iterative development).

"The hardest single part of building a software system is deciding precisely what to build." – Frederick P. Brooks

- Don't build too much – beware of "gold plating" and beware of spending effort implementing features that the customer does not want.

"There is nothing so useless as doing efficiently that which should not be done at all. " – Peter Drucker

- Pay particular attention to non-functional requirements (NFRs), these are often overlooked but are very important. Elicit them and realize them in the right order.

"The cheapest, fastest and most reliable components of a computer system are those that aren't there." – Gordon Bell

- Avoid unnecessary complexity (technical vs. management complexity).

"I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone." – Bjarne Stroustrup

- Minimize losses at the interfaces.

Minimize Losses at Horizontal and Vertical Interfaces

SIEMENS

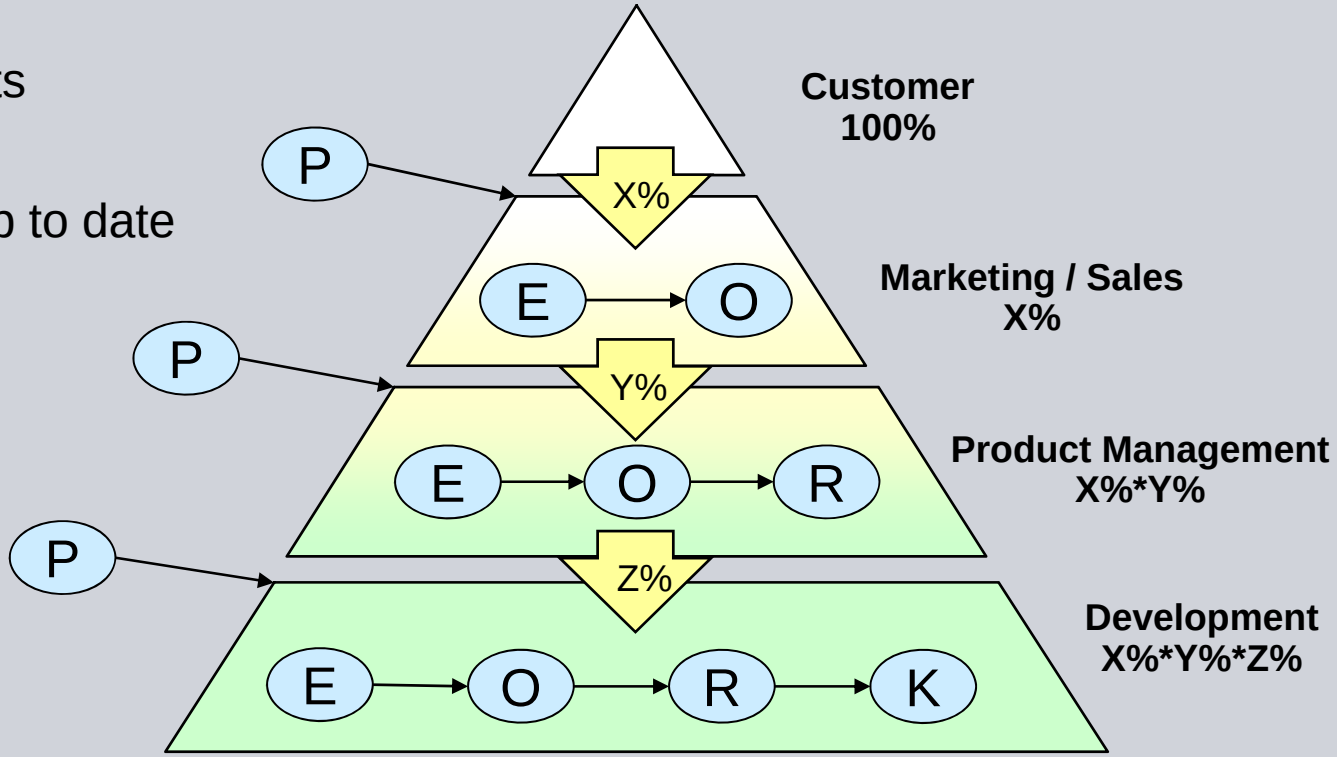
Plan Requirements Process

Elicit Requirements

Organize Requirements

Review Requirements

Keep Requirements up to date



- Elicitation, organization and review of requirements is performed on each level separately in different ways
- Too much loss of requirements information from one level to the next
- Also avoid loss within levels

"Walking on water and developing software from a specification are easy … if both are frozen."
– Edward V. Berard

Architecture and Product Line Engineering

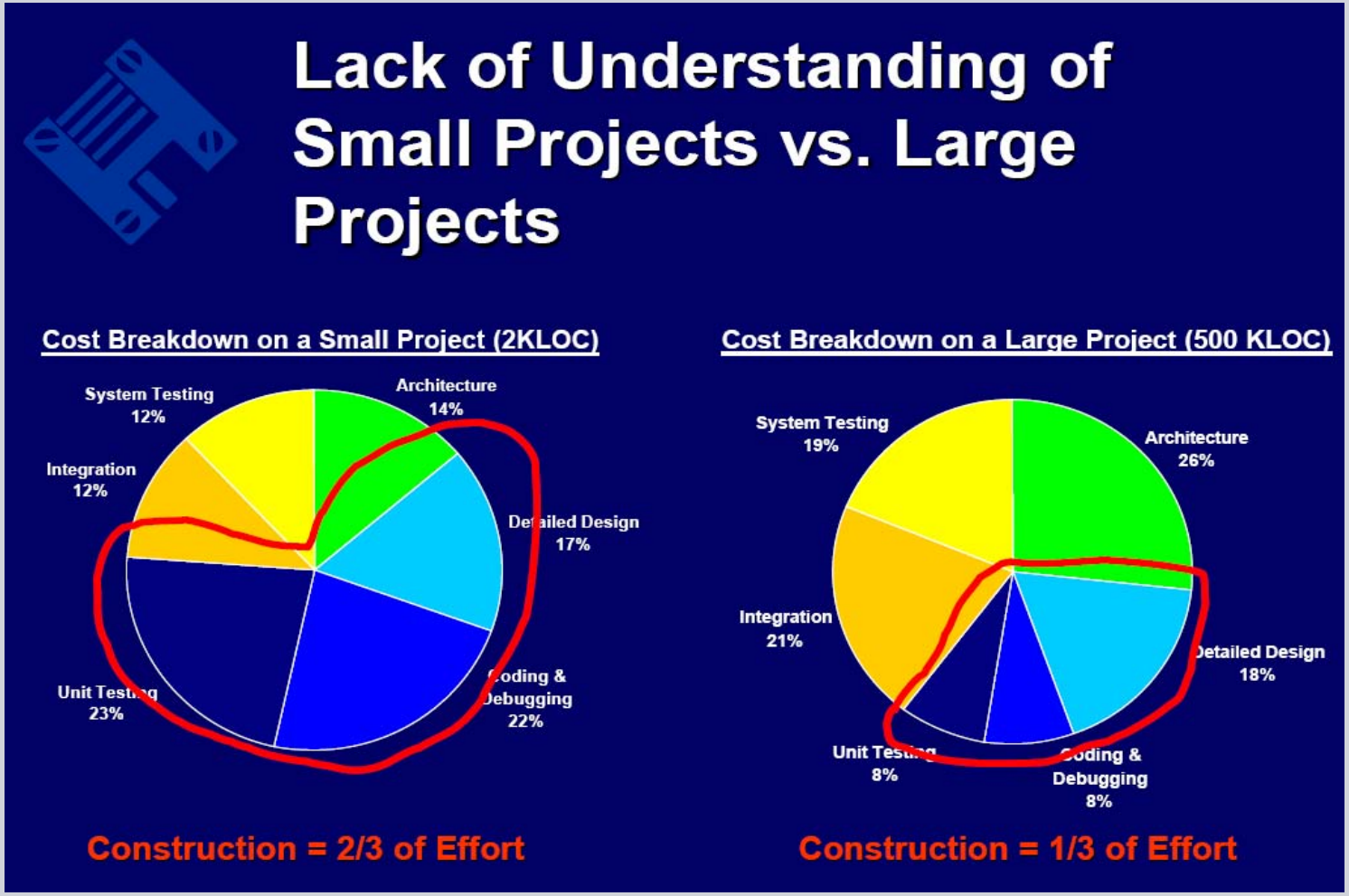
Having an appropriate architecture is key to achieving good productivity.

"If I had eight hours to chop a tree,
I would use six hours to sharpen the axe."
– Abraham Lincoln

- Build on existing basis where feasible
- Be able to recognize when reuse is suitable and when not suitable
- Adequate documentation of architecture (e.g. key design principles) so that others (e.g. those later doing maintenance) can understand the reasoning.
- Avoid complexity where feasible (see next two slides)

The development of large, complex projects tend to have a much lower productivity than smaller ones

SIEMENS



Source: Steve McConnell, www.construx.com

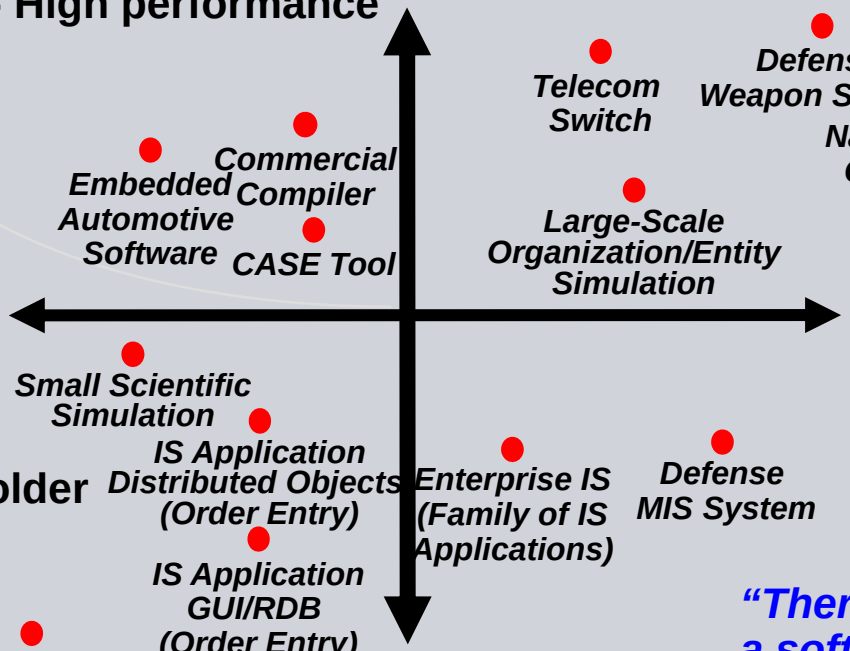
Two Dimensions of Software Complexity: Technical and Management (Source: Grady Booch)

SIEMENS

Higher technical complexity
- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

Lower management complexity
- Small scale
- Informal
- Single stakeholder
- "Products"

Higher management complexity
- Large scale
- Contractual
- Many stakeholders
- "Projects"



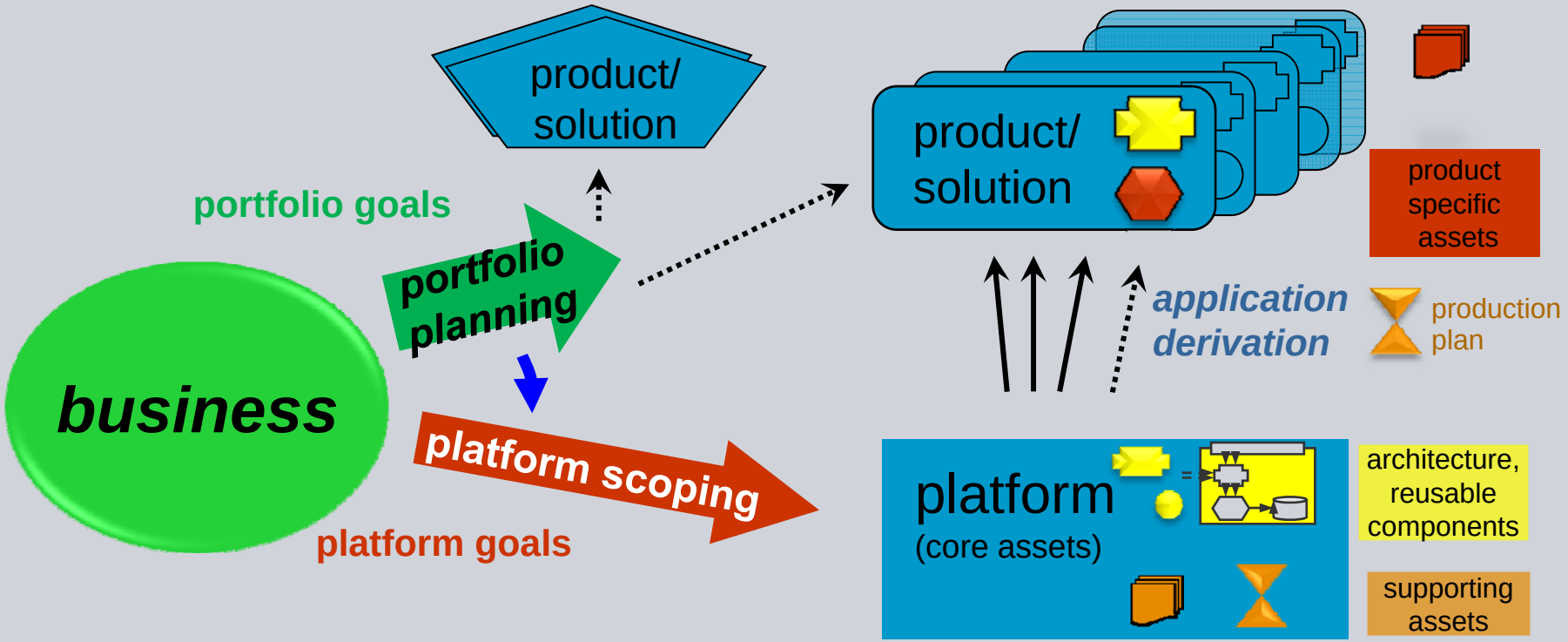
Lower technical complexity
- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

"There are two ways of constructing a software design; one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."
– C.A.R. (Tony) Hoare

SIEMENS

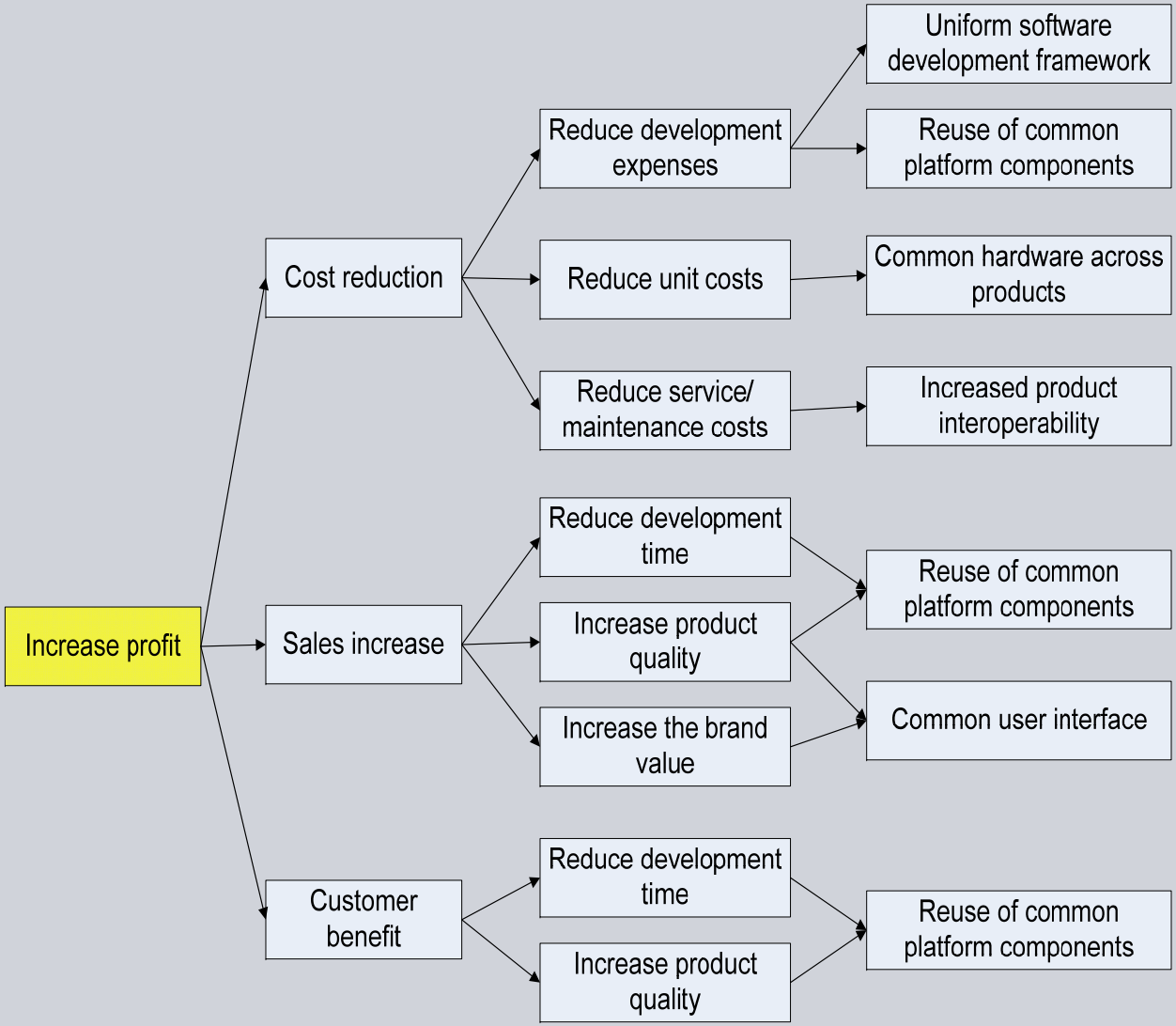
Product Line Engineering (PLE)

Product line engineering is an approach for **optimizing economic benefits** through the **pro-active, constructive reuse** of assets for customer-specific products/solutions. The aim is to increase product quality and decrease development effort and cost by exploiting commonality among products/solutions.



Sample business drivers for Product Line Engineering

SIEMENS



Model-driven Software Engineering

- Take advantage of using higher level of abstraction (increased efficiency, fewer errors, clear basis for discussion with customer etc.)
- Take advantage of automation, e.g. test automation
- Model-based testing

1. The effort spent in creating the first models to derive test cases automatically is roughly the same as if one created the test cases manually
2. During the creation of the models often errors are detected in the requirement specification
3. The investment is small compared to the benefit of finding the defects.

SIEMENS

Agile Development

Business Impacts

- Strong focus on business value for the customer
- Predictable time to market through time-boxed, short iterations
- Improved customer satisfaction, stronger ability to incorporate early and continuous customer feedback
- Higher motivation of engineering teams through trust and ability to self-organize
- Better estimation accuracy through repeated planning activities on different levels of granularity
- Improved quality through test-driven development and continuous integration
- Possibility to launch an early release to the marketplace
- More team productivity and stability through cross-functional and self-organizing teams who continuously improve their way of work
- Allows the teams to focus clearly on the short-term goals (the “sprint”) and protects from distractions
- Has much more ability than a waterfall approach to allow changes (backlog)

“There's no point in being exact about something, if you don't even know what you're talking about” – John von Neumann

SIEMENS

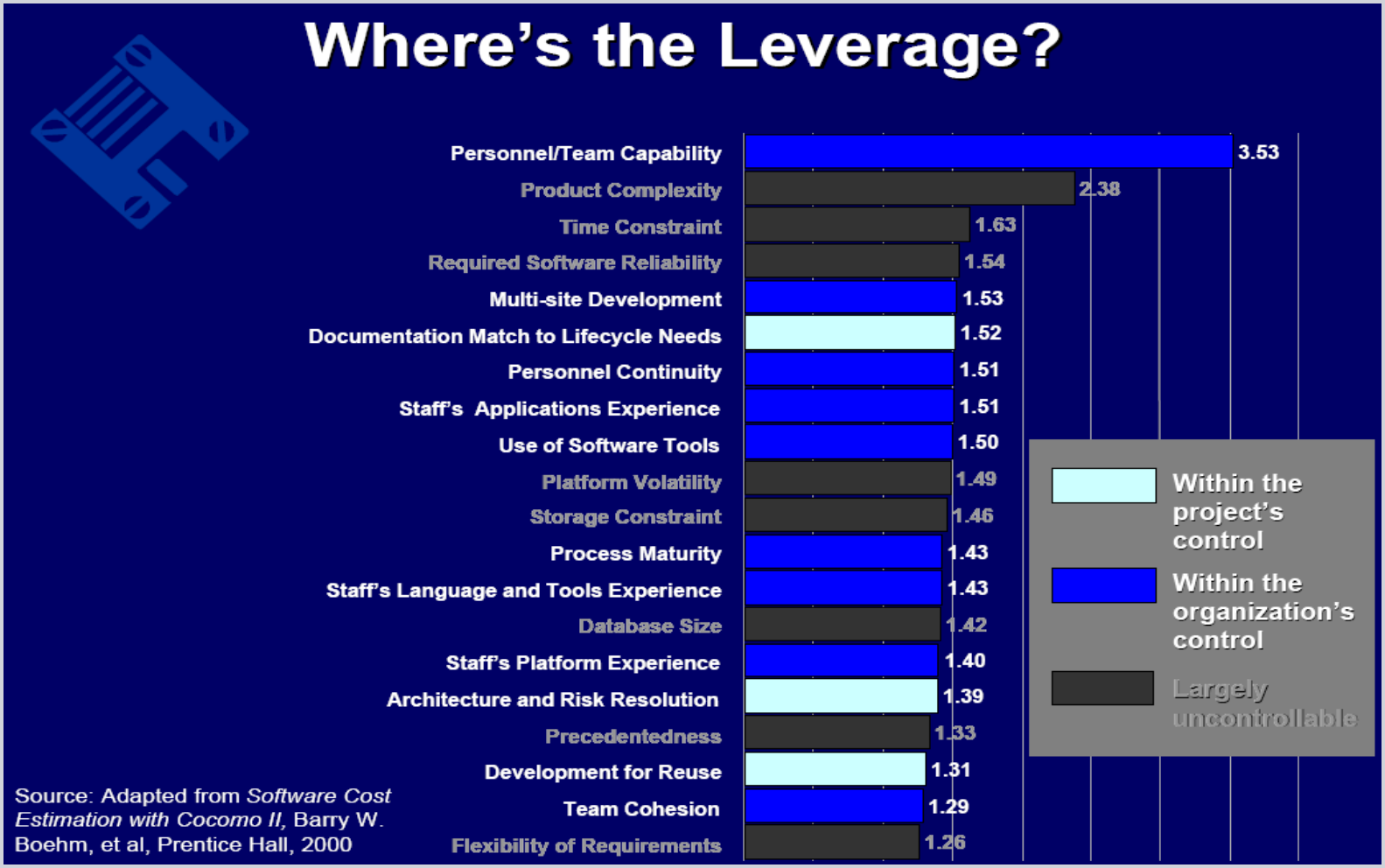
People

- Recent HBR Study by Amabile/Kramer entitled “What really motivates workers” shows that among the 5 factors (*recognition, incentives, interpersonal support, making progress, clear goals*) the key for knowledge workers is not *recognition*, but *making progress*.
- Avoid “silo” thinking (e.g. throwing requirements “over the wall” to developers) but instead work together truly as a team
- Encourage trust and empower teams.
- Literature Summary in July/August 2008 issue of “IEEE Software” (http: www. computer.org/software) entitled “What Do We Know about Developer Motivation?” showed list of motivating factors and states*:.“…Managers must provide challenging problem-solving tasks, explicitly recognize quality work, and give developers autonomy to do their jobs.”*

It also points out that recent trends in software engineering (agile development, global development) make the human aspects increasingly important.

Influence Factors broken down by levels. People and Teamwork are a very important factors

SIEMENS



Lean Software Development
“Think Big, Act Small, Fail Fast, Learn Rapidly”

SIEMENS

We need a holistic view, not only individual improvements.

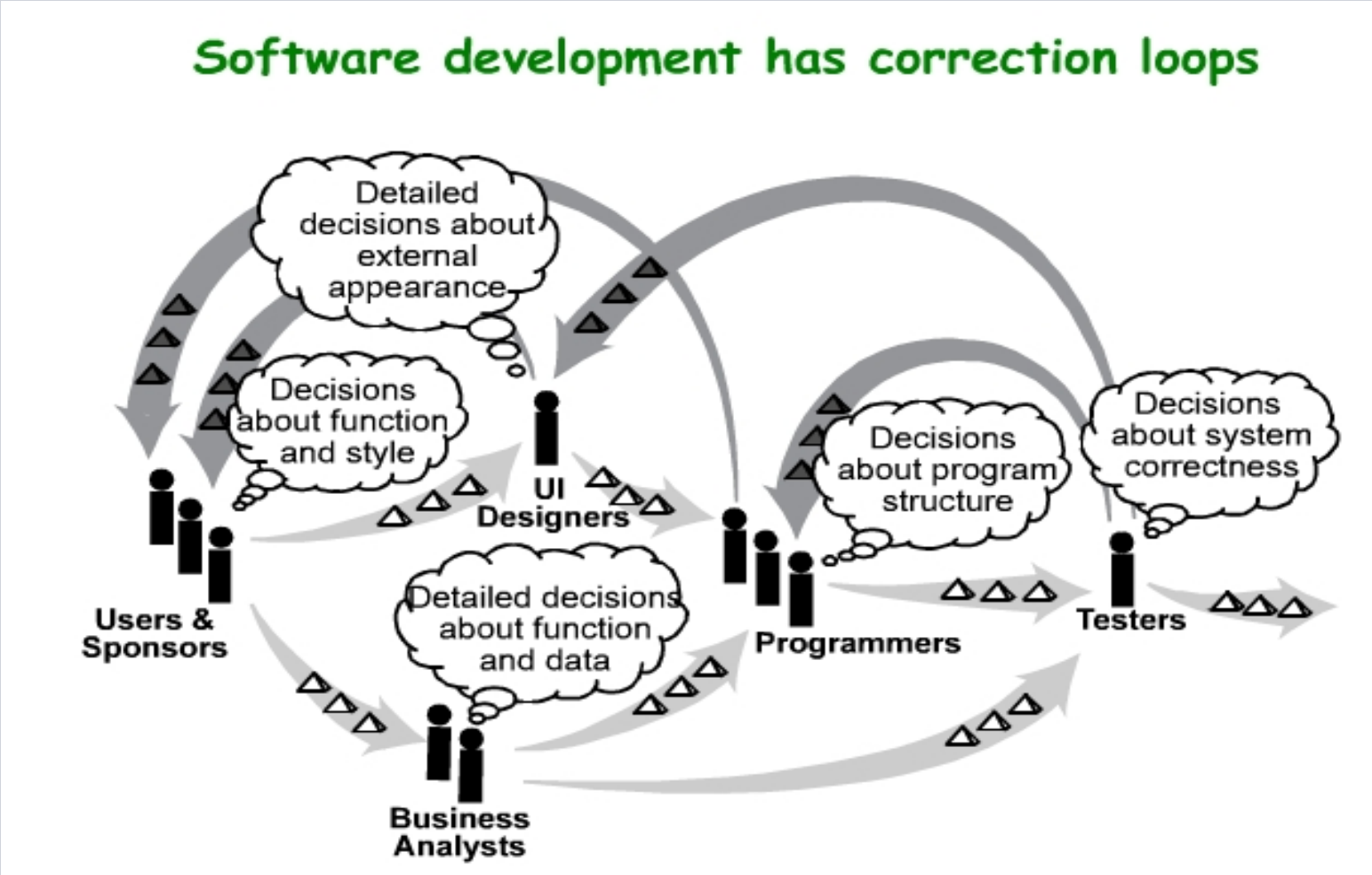
- A modern approach needs a holistic view on the value chain throughout the entire lifecycle and across organizational structures as well as the integration of partners
- Identify and eliminate “waste” as a means to increase “productivity”
- A cross-functional approach generates new perspectives for optimization.
- Strong connection of principles and methods to the employees.
- We need a sustainable culture change.

“Do the right things and do them right” implies a mix of not only customer pull (as is common in lean), but also technology push.

Lean in development is significantly different from lean in manufacturing

Lean Software Development is about flow of information/decisions

SIEMENS



Source: Alistair Cockburn, Keynote Agile 2009

SIEMENS

Lean Software Development, Key Concepts

Eliminate waste	Activities that consume resources, but do not result in generation of customer value are called “waste”. *Waste can be further split into "necessary waste" and (unnecessary) "pure waste". Especially avoid pure waste.*
Build quality in	Build quality in from the start, not test it in later. When a defect is found, stop the line, find its cause, and fix it.
Create knowledge	The development process must encourage systematic learning but we also need to systematically improve that process.
Defer commitment	Schedule irreversible decisions for the last possible moment, when much information is available.
Deliver fast	It is not the biggest that survives, but the fastest. Speed is gained by achieving continuous flow and demand-driven development (pull).
Respect People	Find an entrepreneurial leader and an expert technical workforce and let them do their own job. Managers respect the people and provide support.
Optimize the Whole	Optimize the whole “concept to cash” chain, not local “silos”.

Lean Software Development: Types of "Waste"

(activities that consume resources, but do not result in generation of customer value.)

SIEMENS

- **Waiting:** e.g. through unclear responsibilities, insufficient process/tool integration, lack of parallelization, insufficient infrastructure (e.g. computer/network too slow)
- **Over-Processing:** unnecessary or too-detailed process steps
- **Defects:** Re-work (i.e. having to do an activity a second time, for example due to finding defects or inadequate involvement of all relevant stakeholders)
- **Transportation:** Inadequate transportation of information across interfaces (e.g. manual transfer of information between roles due to incompatible processes or tools, lack of common understanding, hunting for information)
- **Over-Production:** e.g. delivery of features that the customer does not need, overly-complex products, too many variants instead of doing systematic reuse
- **Inventory:** creation of artifacts that are not used downstream (e.g. effort invested in the definition, effort estimation, review of requirements or features that are not realized)
- **Motion:** unnecessary transfer of persons due to inadequate relationship between the roles (e.g. lack of direct access to necessary information, having to multi-task between too many projects, insufficient communication between sites).

SIEMENS

Closing Words / Guiding Principles

Summary of most of the guiding principles. These are also part of our “Software Curriculum” project (paper at ICSE 2010).

ICSE 2010, May 2-8, 2010
Cape Town, South Africa

http://www.sbs.co.za/ICSE2010/

1. Architecture is the key throughout the whole lifecycle as well as across releases.
2. Structure the system to avoid unnecessary complexity, and to actively enable and support multi-site development
3. Build on existing basis where feasible and be able to recognize when such re-use is not suitable
4. Strive for transparency, clear well-grounded decisions
5. The product manager must act as the owner of the main requirements
6. Pay particular attention to non-functional requirements (NFRs),
7. Be prepared and able to handle changing requirements, but be aware about the risk of late changes
8. Work iteratively, strive to identify and resolve technical and business risks early
9. Work together truly as a team, avoid “silo” thinking, be willing and able to speak and understand the other roles and disciplines
10. Do not underestimate the importance of soft skills, these are of great and growing importance. (Real) communication is key (especially for large projects).

Thanks for your attention.
Questions?

SIEMENS