Lean Software Development

presented by Werner Wild,
EVOLUTION Consulting

Contents

- Origins of Lean Development
  - The Toyota Production System
- Principles of Lean Thinking
  - Eliminate Waste
  - Create Knowledge
  - Defer Commitment
  - Deliver Fast
  - Built Quality In
  - Respect People
  - Optimize the Whole

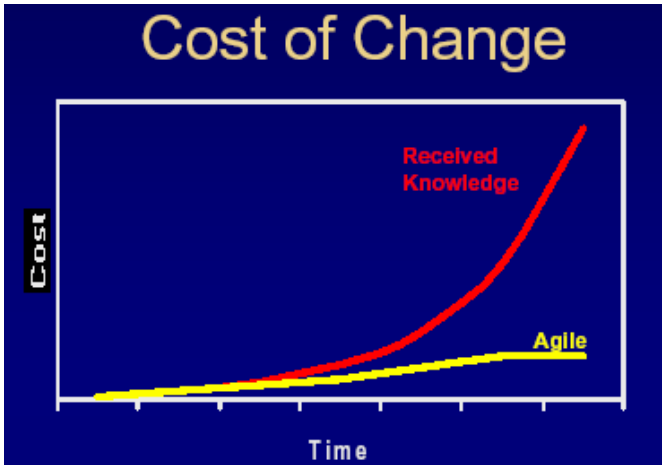
The Toyota Production System

- Just-in-time flow
  - Non stock production
  - Build only what is needed
  - Eliminate everything which does not add value
- Automation
  - Stop if something goes wrong
  - Zero Inspection

Taiichi Ohno

1912-1990

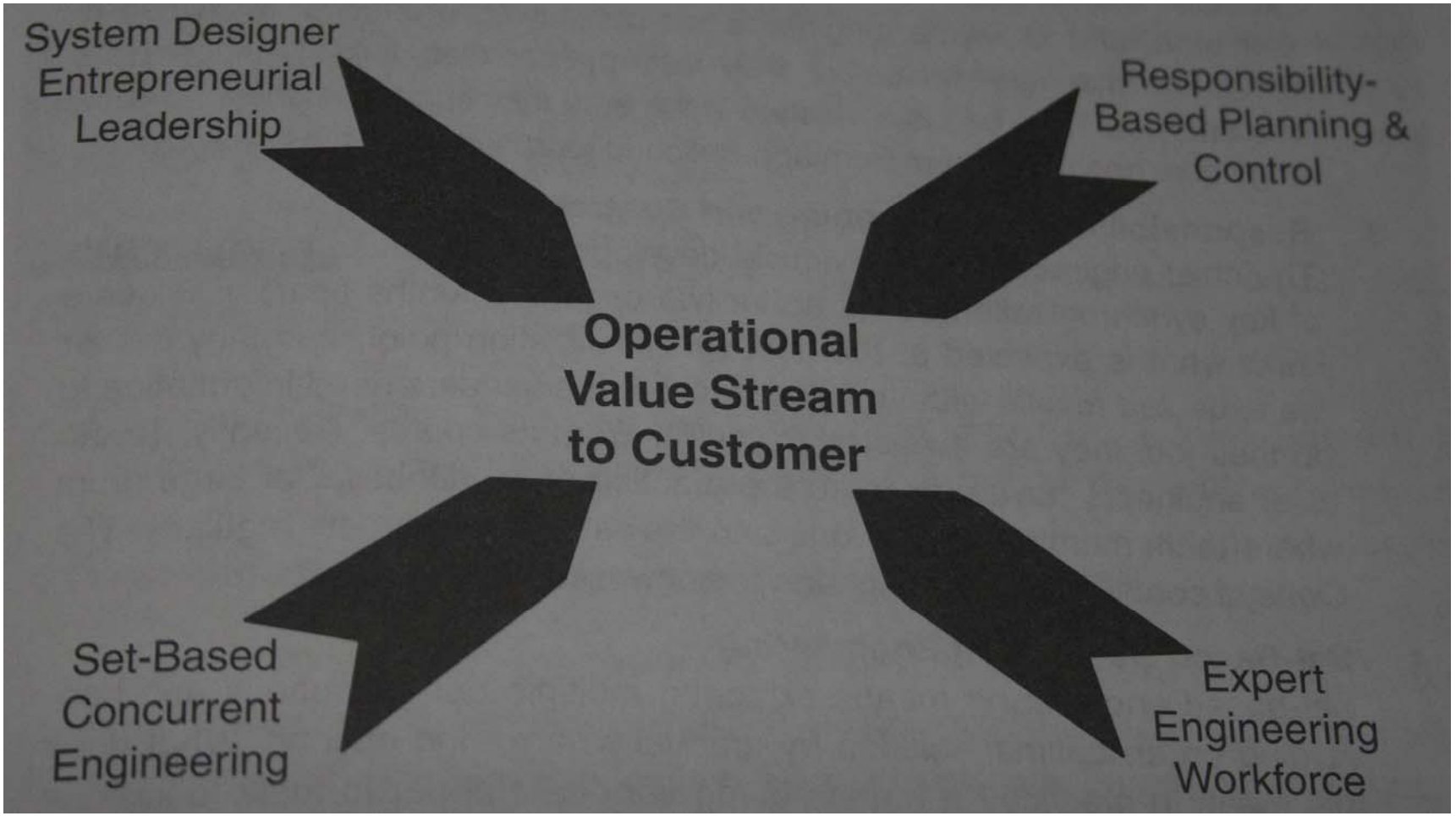
Changing the Mental Model



- Received Knowledge:
  - Die Change is Expensive
  - Don't Change Dies
- Taiichi Ohno
  - Economics Requires Many Dies Per Stamping Machine
  - *One Minute Die Change*

- Received Knowledge:
  - Code Change is Expensive
  - Freeze Design Before Code
- The Agile Imperative
  - Economics Requires Frequent Change In Evolving Domains
  - *Last Responsible Moment*

Toyota Product Development System



Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Principle 1: Eliminate Waste

- Taiichi Ohno on how the Toyota Production System works:

  All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing this timeline by removing the non value-adding wastes.

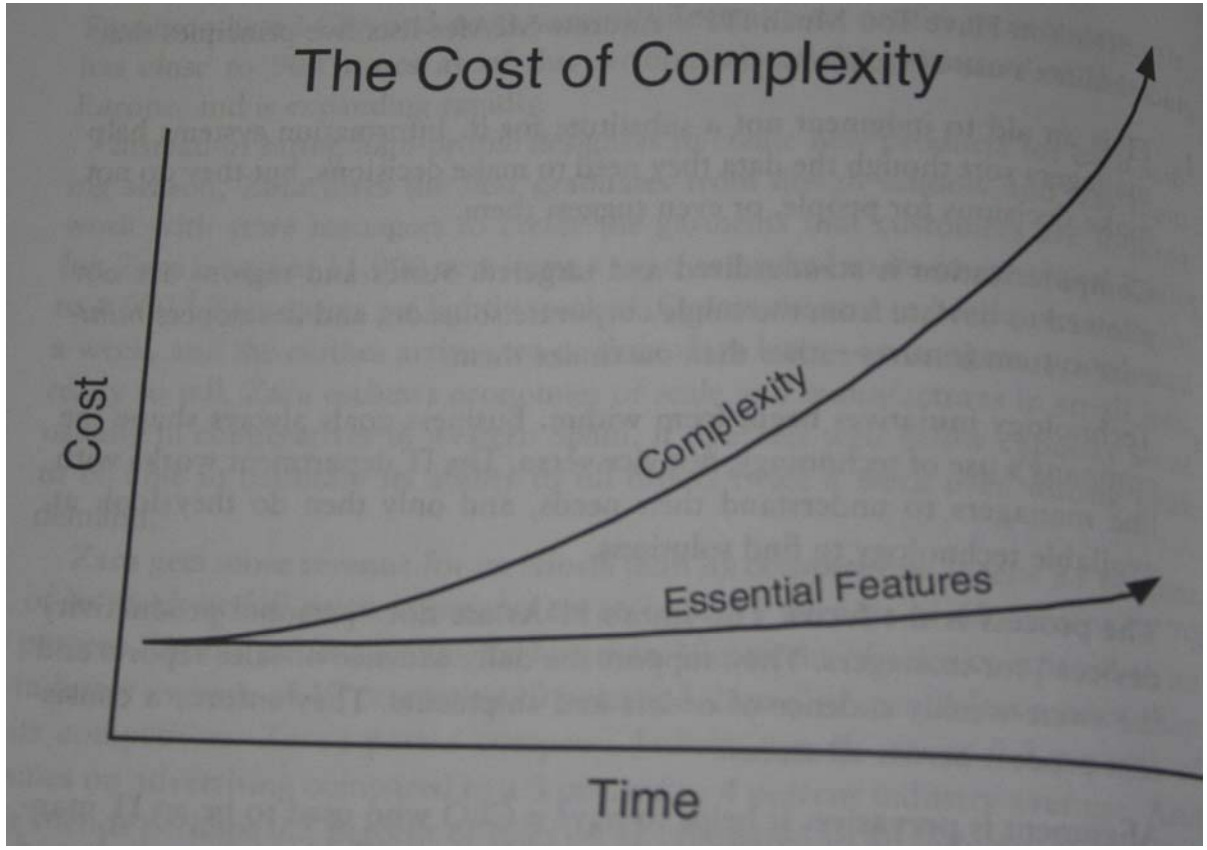
Eliminate Waste

- Waste
  - Anything that does not create value for the customer
  - The customer would be equally happy with the software without it

- Prime Directive of Lean Thinking
  - Create ***Value*** for the customer
  - Improve the ***Value Stream*** by removing non value-adding activities

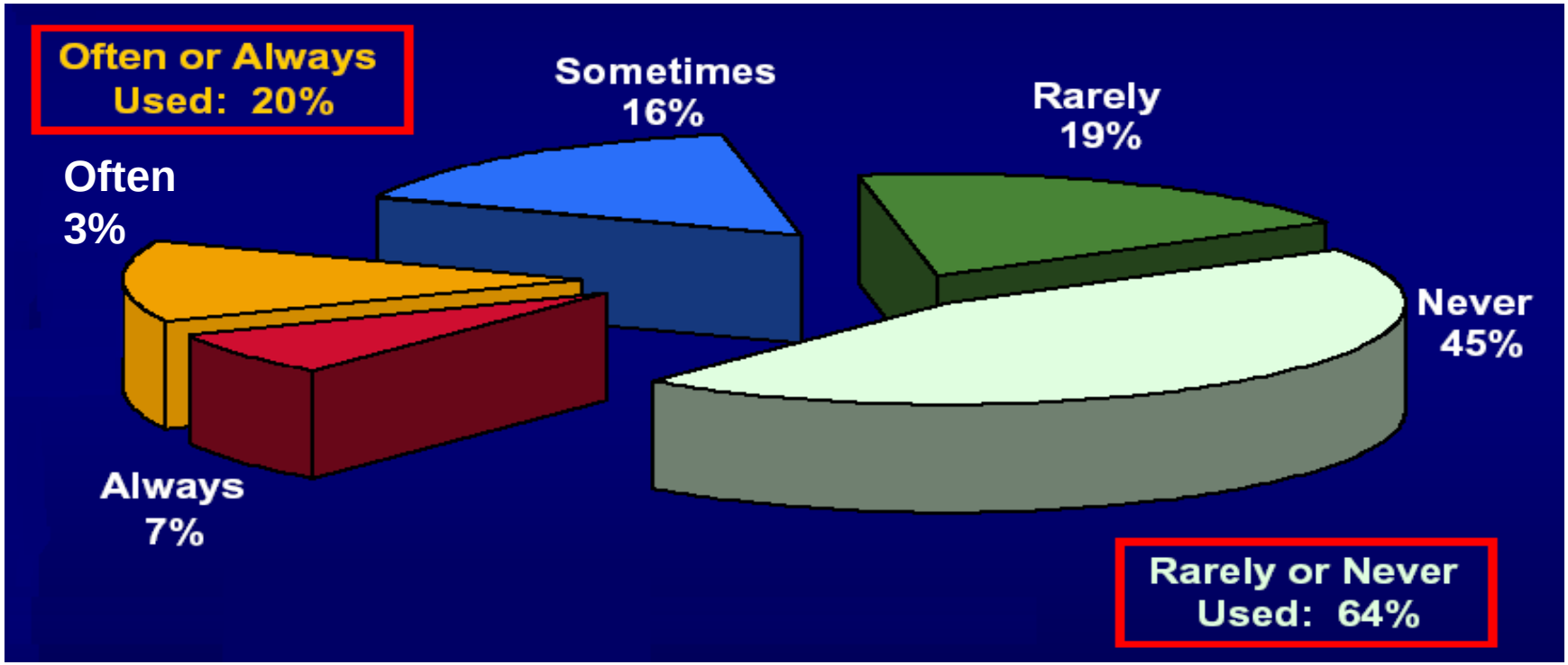
Complexity

- Complexity is the biggest source of waste



The biggest Source of Waste

Features and Functions Used in a Typical System

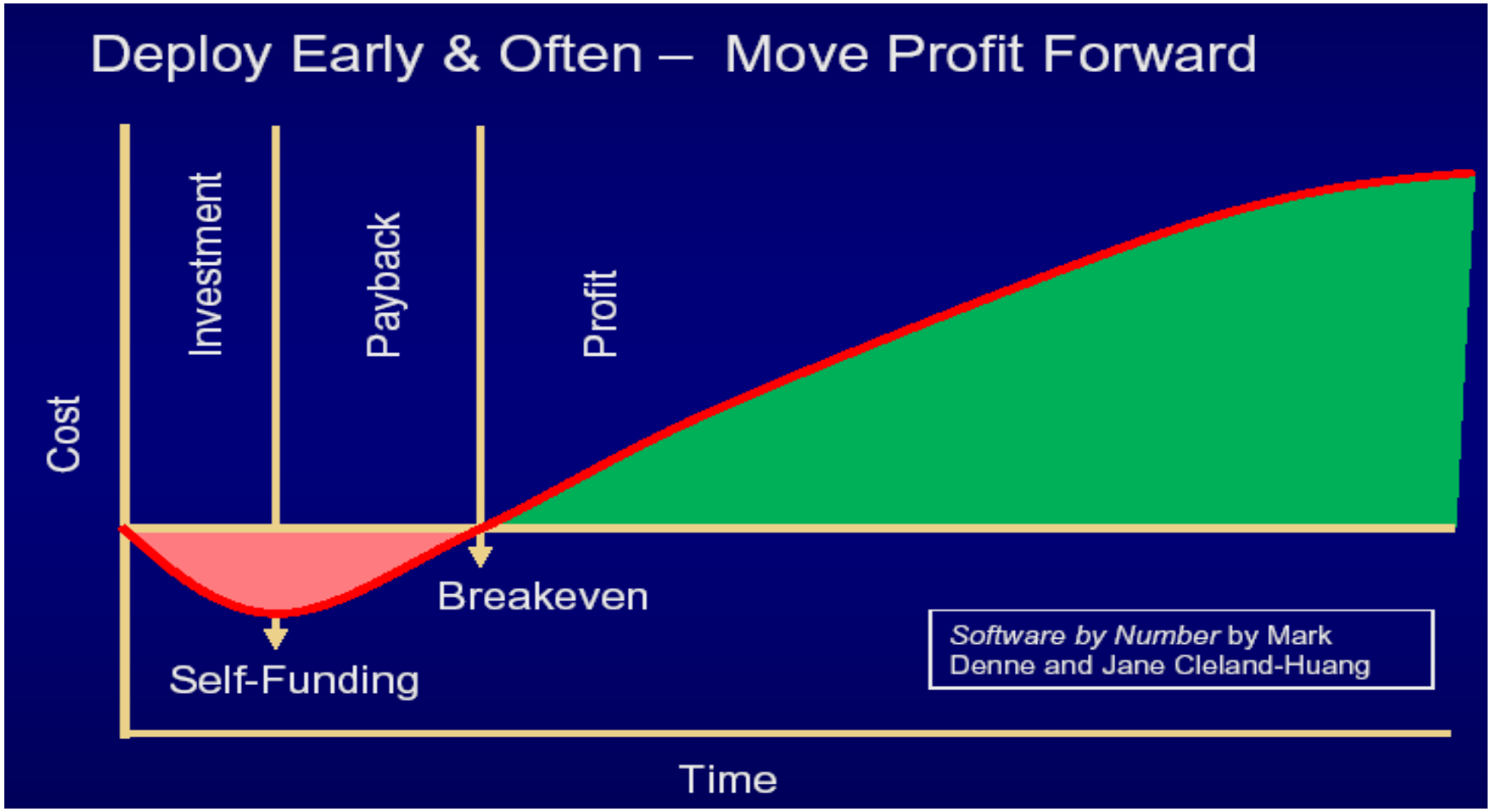


Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Write Less Code

- Justify every feature
  - Limit the features and the functions that make it into the code base
  - Every feature should proof that if will create more economic value than its lifecycle costs
- Minimum useful feature sets
  - Divide software into minimum useful features and deploy these one set at a time, highest priority first
  - Allows to use the software much faster

Minimum Useful Feature Sets



1st step is Seeing Waste

Manufacturing	Software Development
In-Process Inventory	Partially Done Work
Over-Production	Extra Features
Extra Processing	Relearning
Transportation	Handoffs
Motion	Task Switching
Waiting	Delays
Defects	Defects

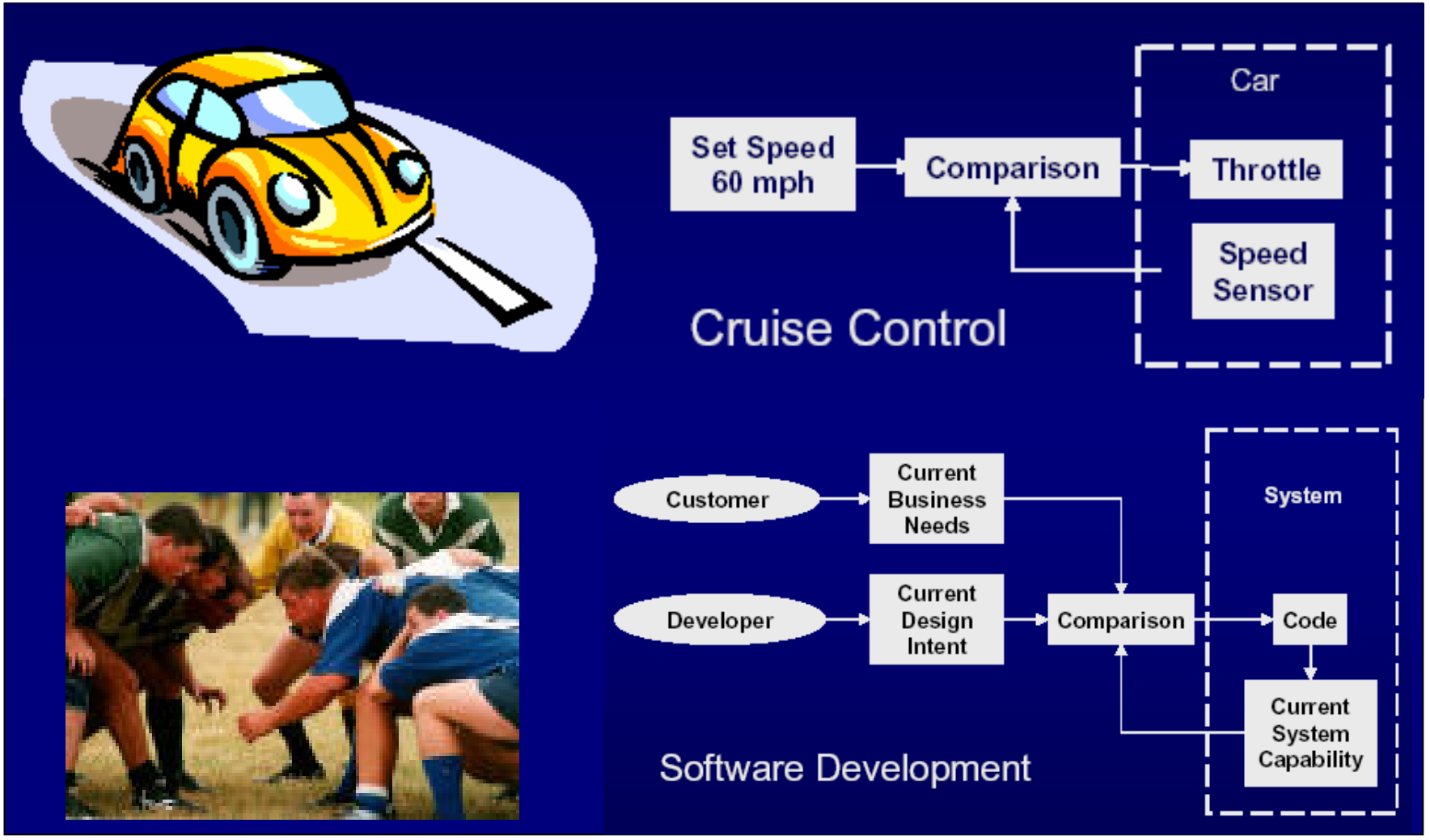
Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Taking Feedback into Account

- Waterfall approach is based on the idea that knowledge in the form of “requirements” exists prior and separate from coding
- Software development is a knowledge-creation process; successful software products emerge
- Early design does not take ongoing feedback that comes from actually building software into account

Principle 2: Create Knowledge



Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Defer Commitment

- The longer we defer decisions, the more we can learn
- Challenge to make decisions Just-in-time
- 2 Approaches for learning
  - Building a system which is change tolerant
    - Refactoring
  - Building several options
    - Set-Based Design

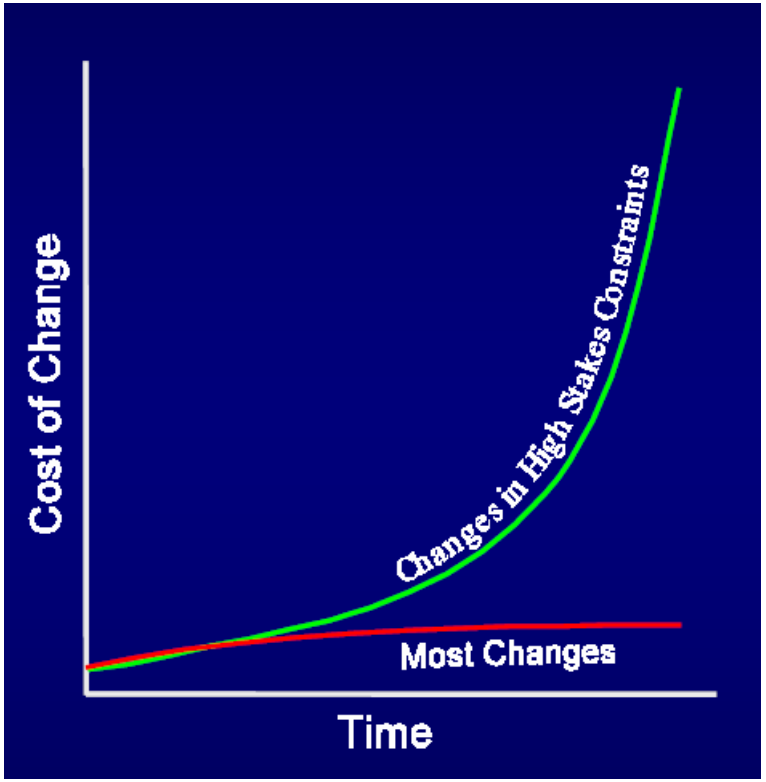
Set-Based Design

- Is based upon the idea of building several options
- Leader with technical expertise knows where to maintain options
- At the **last responsible moment** the option that gives the best overall solution is chosen
- Appropriate approach for making high-impact, irreversible decisions
- Example: Red Eye Reduction

Defer Commitment

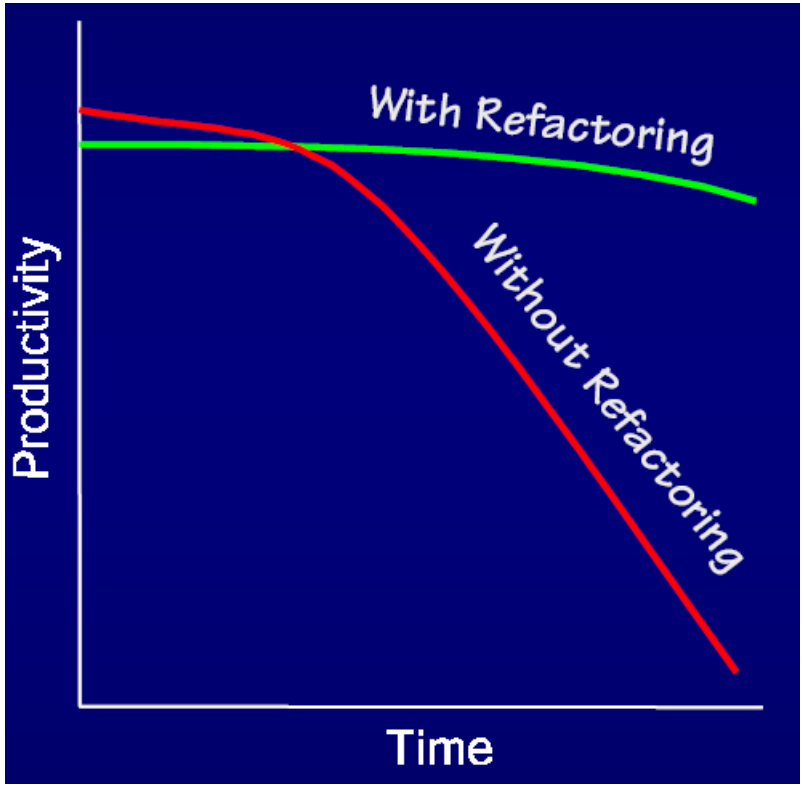
Two Kinds of Change

- High Stakes Constraints
  - Examples:
    - Language
    - Layering
    - Usability
    - Security
    - Scalability
  - Rule:
    - Only a Few
    - At a High Level
- Most Changes
  - Keep the Cost Low!



Refactoring

- Building a change tolerant code base which allows to adapt to changes
- Mitigates the risk and minimizes the cost of complexity in the code base
- Requires automated testing, continuous integration and stop-the-line



Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Principle 4: Deliver Fast

- The most disciplined organizations are those that respond to customer requests
  - Rapidly
  - Reliably
  - Repeatedly

- Software Development Maturity
  - The speed at which you reliably and repeatedly convert customer requests to deployed software
  - Quality of a software development process can be measured by measuring the average end-to-end cycle time of the development process

Queues

- Cycle Time
  - Average End-to-End Process Time
    - From Entering The Terminal
    - To Arriving at the Gate
- Time Spent in a Queue is Wasted Time
- The Goal: <u>Reduce Cycle Time</u>

Queuing Theory

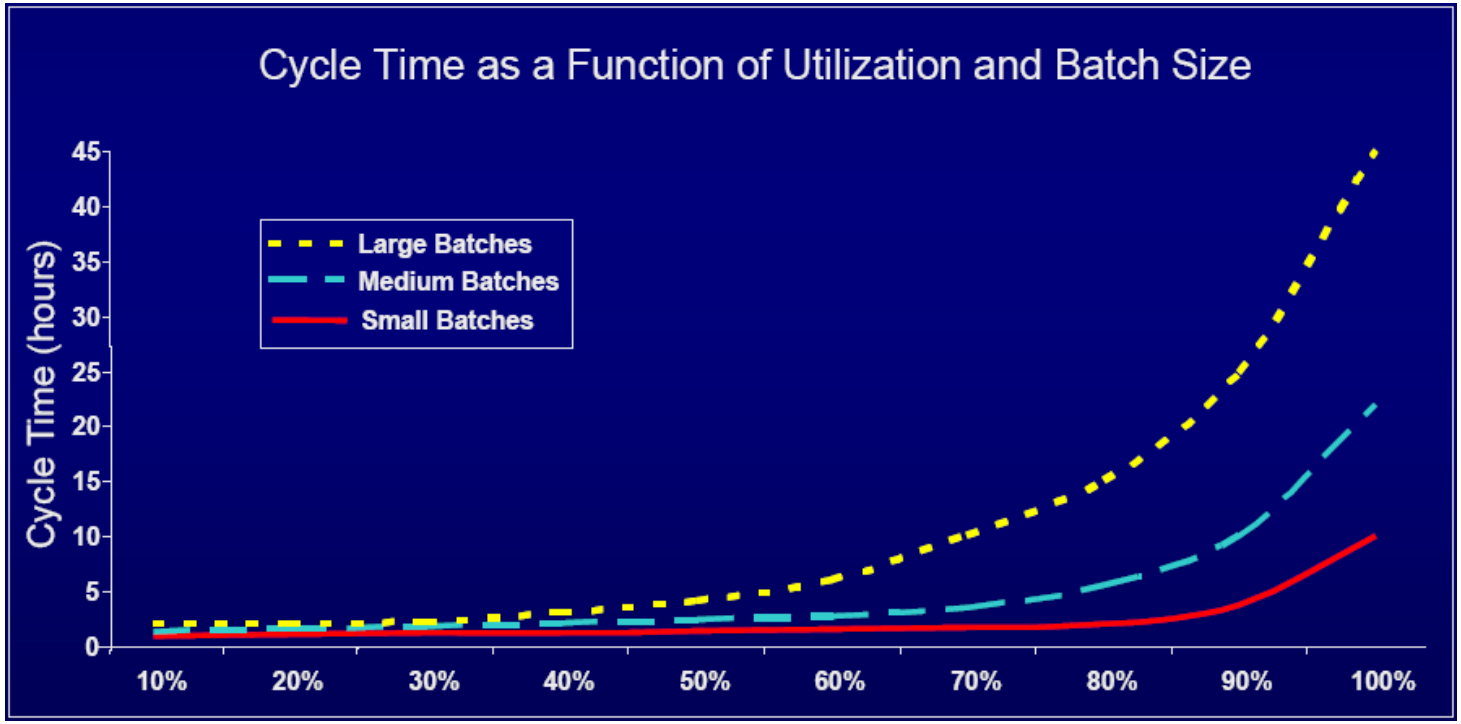
- Little's Law

$$\text{Cycle Time} = \frac{\text{Things in Process}}{\text{Average Completion Rate}}$$

- 2 ways to reduce cycle time
  - Do things faster
  - Reduce the number of things in the process

Queuing Theory Lessons

- Small Batches Move Faster
- Slack Resources Decrease Cycle Time

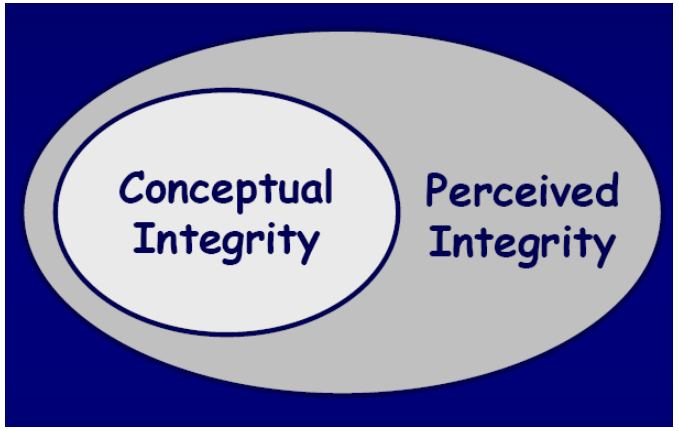


Principles of Lean Thinking

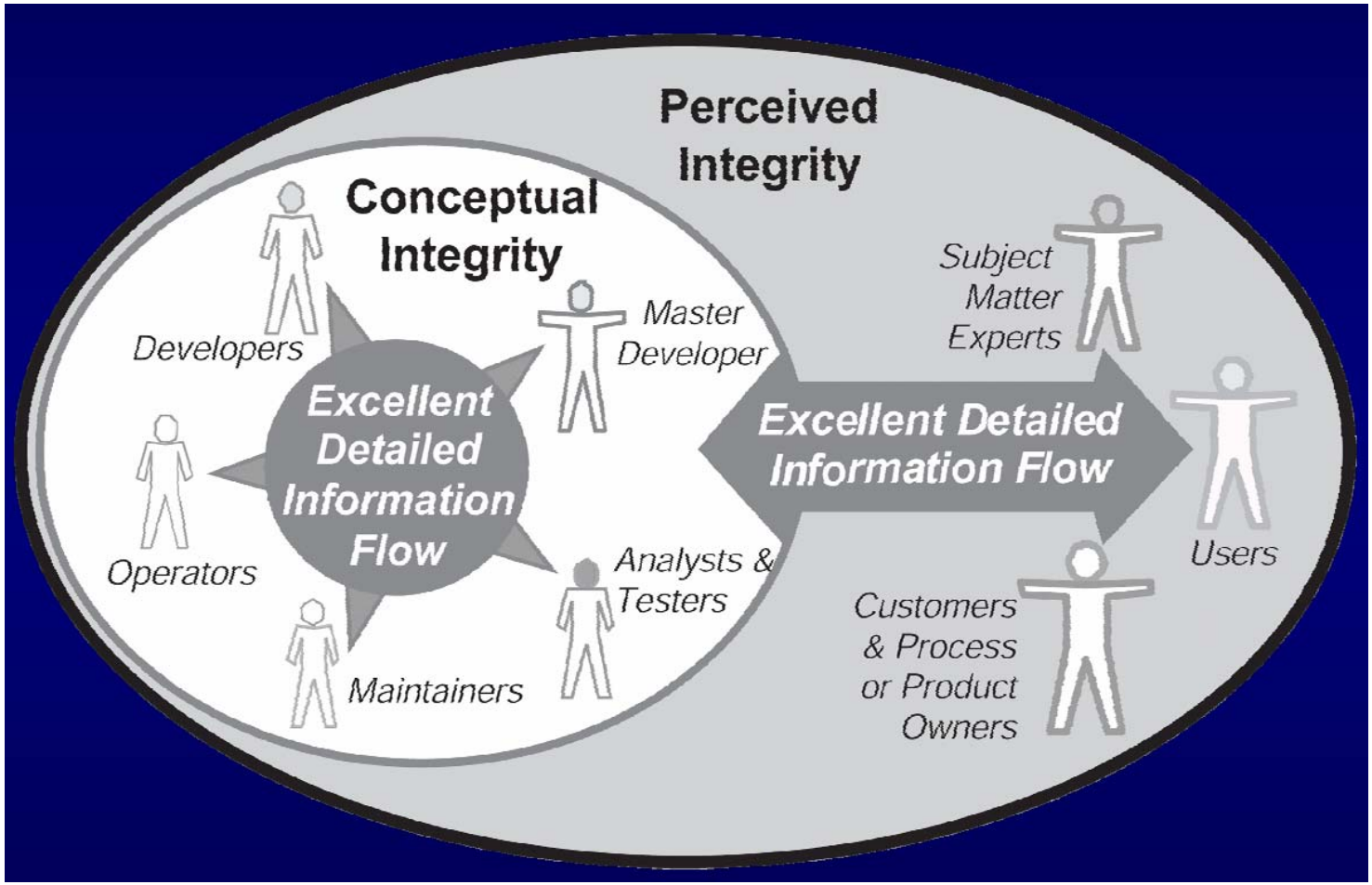
1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Software Integrity

- Perceived (External) Integrity
  - The totality of the system achieves a balance of function, usability, reliability and economy that delights customers.
- Conceptual (Internal) Integrity
  - The system's central concepts work together as a smooth, cohesive whole.



Integrity Comes From Excellent, Detailed Information Flow



Leadership

- Successful development efforts are highly correlated with the existence of a champion who
  - deeply understands the customers' job and the technology
  - is responsible for making key product decisions and who is accountable for the result of those decisions

Leadership

- Several models for implementing the role of a champion
  - Chief Engineer (at Toyota)
    - Responsible for the business success of a vehicle family
    - Understands the Target Customer and what they will value
    - In-depth knowledge of vehicle design
  - Leadership Teams
    - Two leaders: One technical lead and one with deep market understanding (e.g., Intuit)
    - Three leaders: Leaders from sales, product engineering and development (e.g., Honda)
  - Shared Leadership
    - Team of experts work together to come to a common understanding of what customers would value

Complete Teams

- Complete products are developed by complete teams
- Example “Intuit”
  - Team included not only people from software engineering, but everyone necessary to put the product on the marked
  - Checked with management team on a regular basis to report on status, receive guidance and to get approval to continue to the next checkpoint
  - Except for guidance the team made both product and process decisions on its own

Built Quality In

- Built quality into the code from the beginning, not test it in later
- To really achieve quality control conditions so as not to allow defects in the first place
  - In test-driven development unit tests and acceptance tests are written before the code
  - Defect tracking systems are queues of partially done work

Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

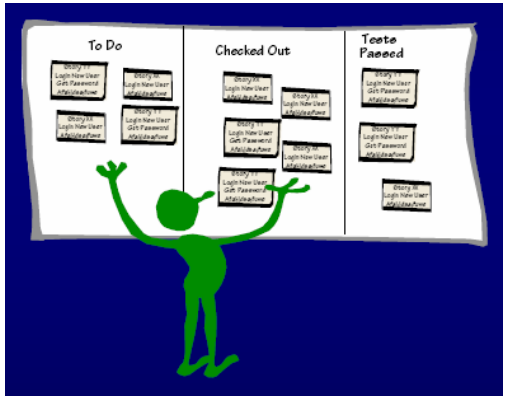
Responsibility-Based Planning and Control

- Instead to tell people what to do, create an environment where they can figure it out by themselves

- Switch from dispatching to enabling
  - **Dispatching**: planning every step for someone else's job
  - **Self-directing**: setting up an environment that people can figure out how to do their job without being told

Visual Workspace and Self-Directing Work

- **Kanban** to let people know what to do next
  - Index cards
- **Andon** to make problems visible so they can be addressed immediately
  - Lava lamps
- **Dashboard** to see the overall progress of their work
  - Burn-down charts
  - Visible charts



Principles of Lean Thinking

1. Eliminate Waste
2. Create Knowledge
3. Defer Commitment
4. Deliver Fast
5. Build Quality In
6. Respect People
7. Optimize the Whole

Measurement

- A lean organization optimizes the whole value stream (from customer order to deployment)
- Projects performance often measured based on cost, schedule and scope which does not take quality and customer satisfaction into account
- Lean measurements
  - Cycle Time
  - Customer Satisfaction
  - Financial Return

References

- Mary and Tom Poppendieck: An Agile Toolkit for Software Development Managers. Addison Wesley, 2003.

- Mary and Tom Poppendieck: Implementing Lean Software Development. From Concept to Cash. Addison Wesley, 2006.

Acknowledgements

Some of the slides are based on the Tutorial “Lean Software Development” by Mary & Tom Poppendieck and were created by Barbara Weber.

I would like to thank Mary & Tom & Barbara for their kind permission to use their slides in this presentation.

Thanks for your Attention !