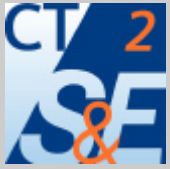
SIEMENS

CORPORATE TECHNOLOGY



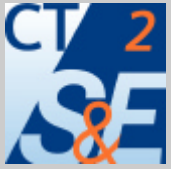
Software &
Engineering
Architecture

eXtreme Programming in Open-Source and Distributed Environments

Michael Kircher

Siemens AG, Corporate Technology

Michael.Kircher@mchp.siemens.de

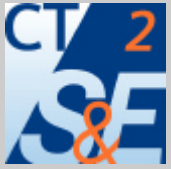


Outline

- **Motivation**
- **Adaptation of XP**
  - Distributed Projects
  - Open-Source Projects
  - Large Projects
- **Examples**
  - ACE & TAO
  - Web-based IDE
- **Patterns as the Metaphor**

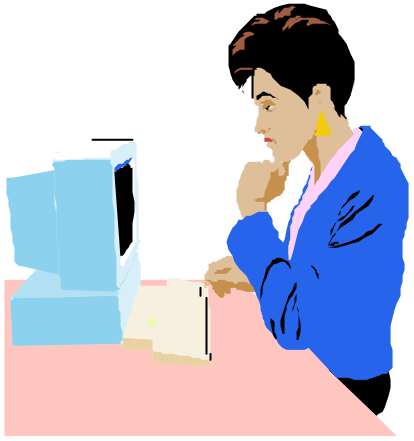
Motivation

- **Increasing number of projects are geographically distributed**
  - Financial consideration
  - Personal reasons of team members
  - No SW development process fully addresses distribution aspects
- **Extra documentation does not help**
  - Does not increase communication efficiency
  - Actually, increases communication overhead
- **No two such projects are identical**
  - Processes by the book are always templates of what actually goes on



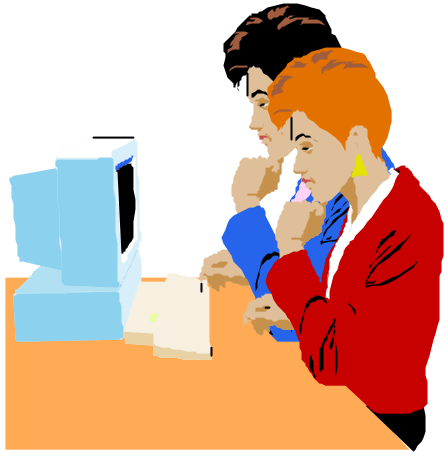
Traditional processes

- **Many traditional processes seem to focus more on intermediate specifications, than they do on the result**



Reducing extra documentation

- **Strengthen direct communication**



- **Strengthen trust within your team and with your customer**

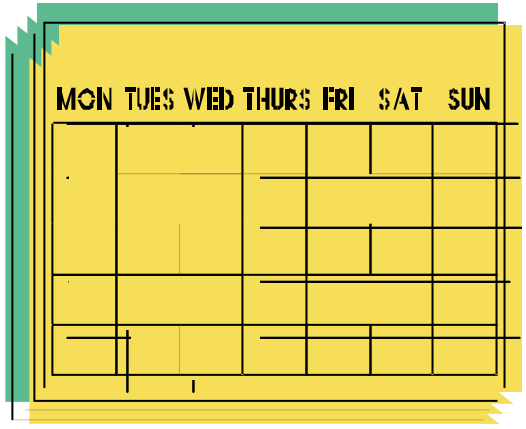
Why many HW managers fail in SW development …

- Many managers believe software can be managed in the same way hardware is managed
- Manufacturing of HW is well known - technology changes very slowly
- ‘Manufacturing’ of SW is supposed to be well known – but technology changes rapidly

Why eXtreme Programming?

- **Traditional processes have two flaws:**
  - "If you plan well enough everything will be going well."
  - "You can prepare for late changes in large system development."

- **Reduce risk from early on**
- **Produce high quality software**
- **Keep your programmers happy**
- **Keep your customer happy**
- **Be prepared for change**
- **Speed-up development**



eXtreme Programming

What is XP?

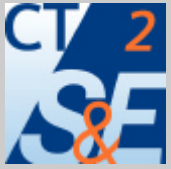
- **Values, Principles, and Practices …**
- **Exciting names: ‘Agile Methodology’**
- **Better to speak of:**
  - XP-influenced processes [Fowler]

Did you know?

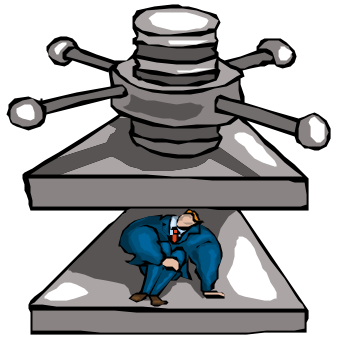
- **XP is good enough to reach CMM level 3 [Beard]**

Adaptation of XP

- **Are we allowed to adapt XP?**
  - Quote: “You are only doing XP, if and only if you do the twelve practices, not more and not less.”
- **Every project is special**
  - Distributed projects
  - Open-source projects
  - Large projects



Distributed Projects

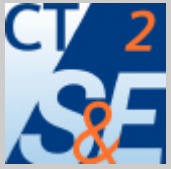


- **Why Distributed Projects?**
  - Constrained by situation
    - Multi-site projects with distributed team members
  - Individual constraints
    - Personal commitments such as doing childcare at home
  - Cost
    - Outsourcing can be much cheaper
  - Mobility
    - Team members stay in contact while traveling
- **Search for SW processes which are designed to support distributed development teams**
  - No process specifically addresses this.
  - Can XP be applied in such projects?

Distributed Projects

Adapting XP to Distributed Projects:

- Distributed eXtreme Programming (DXP)
  - Team members can be highly mobile as well as arbitrarily far apart
  - Applies XP values and principles
  - Adapts XP practices to a distributed team environment
  - Relaxes the assumption of close physical proximity of team members



Distributed eXtreme Programming

Problem: XP practices assuming close physical proximity

Pair Programming

Simple Design

Planning Game

Collective Ownership

Refactoring

Testing

Small Releases

Metaphor

Continuous Integration

On-Site Customer

40-Hour Week

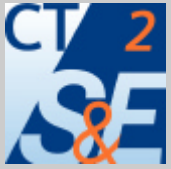
Coding Standards

Software & Engineering Architecture

Distributed eXtreme Programming

Solution: Bypass physical proximity.

Pair Programming	Videoconferencing, application sharing and personal familiarity
Planning Game	Application/Desktop sharing
On-site Customer	'Virtual' on-site customer via videoconferencing and application sharing
Continuous Integration	Remote access to integration machines



Distributed eXtreme Programming

Challenges:

- **Communication**
  - Regular in-person meetings
- **Coordination**
  - Schedules, videoconference appointments
- **Infrastructure**
  - Interoperability, common configuration
- **Availability**
  - Simple rules, consider time zones
- **Management**
  - Regular communication, trust

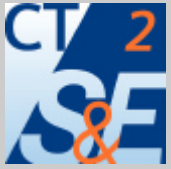
Distributed eXtreme Programming

Opportunities:

- **Integration of team members facing constraints**
- **Convenient customer involvement**
  - E.g. customer need not necessarily be on-site all the time
- **Mobility**
  - E.g. team members stay in contact while traveling

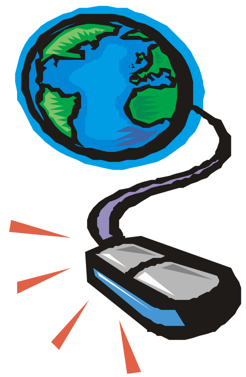
Pitfalls:

- **Missing trust**
- **People do not like to share**
- **Communication overhead too high**
- **Inappropriate infrastructure**



CORPORATE TECHNOLOGY

Distributed eXtreme Programming



DXP Requirements:

- **Connectivity**
  - e-mail,videoconferencing, application sharing
- **Configuration management**
- **Familiarity among team members**
- **Motivation of the team members and strong support of each other**
- **Tolerance for problems while videoconferencing, e.g. poor quality, disconnection, jitter**

Lessons Learned:

- **It works – but not as effective as having physical proximity**
- **Extends the range of XP**

Software &
Engineering
Architecture

Open-Source Projects

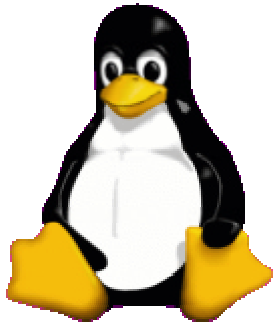
Built-in XP entities:

- **Collective code ownership**
  - Coding standards are a must!
- **'Embrace change' as the basic motivation**
- **Rapid feedback is a natural consequence**

Problem:

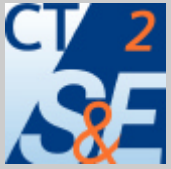
- **Loosely coupled development team as team members join in from all over the world via Internet**
- **Always distributed, as users/developers are distributed**
- **Distinction between business and development is blurred**
- **Many developers, with varying involvement and commitment**

Open-Source Projects



Solution:

- Coding Standards and tests are enforced by a gatekeeper
- Core team ensures main XP practices, like small releases and simple design
- Tight feedback cycle via active encouragement and reward
- Well-maintained e-mail lists with fast responses (feedback)
- Application of DXP for distribution aspects



Large Projects

- **Some software cannot be built by a small team in a reasonable time frame.**
  - Many features
  - High complexity, need for various specialists

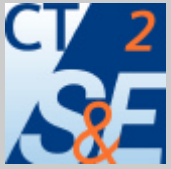
Problem:

- **High communication overhead, as no longer everybody can talk to everybody else**
- **Coordination gets very hard**
- **Often geographically distributed**
- **Developer base changes over time**
  - Staff turnover, loss of competencies

Large Projects

Solution:

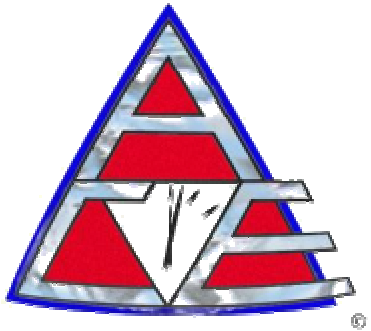
- **Establish a core team and several peripheral teams**
- **Core team starts first and acts as a proxy customer**
- **Make sure to establish XP at local sites**
- **Use DXP practices to interconnect teams**
- **Easy to follow development process**



Example 1: ACE & TAO

Large, Open-Source, Distributed Projects:

- **ACE - Adaptive Communication Environment**
  - Object-oriented network programming toolkit
- **TAO - The ACE ORB**
  - CORBA 2.4 compliant Object Request Broker (ORB) implementation



Example 1: ACE & TAO

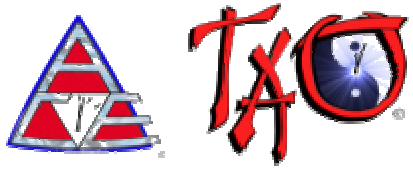
Challenges:

- **Coordination of 600+ users with various involvement**
- **Distinction between business and development is blurred**
  - The core team is tied between both roles
- **Framework development**
  - How to be simple if you need to cover many use cases?
- **High quality software for mission-critical computing**

Supporting facts:

- **Open-source provides a tight feedback cycle with the user community**
- **Open-source is based on collective code-ownership**

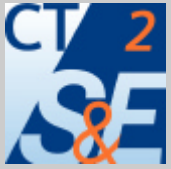
Example 1: ACE & TAO



Solution:

- **Gatekeeper for controlling contributions from user community**
- **Use Bugzilla database and Problem-Report-Forms as story cards**

- **Flexibility regarding communication**
  - Mailing lists for overhearing conversations
  - E-Mails
  - Core team uses pair programming supported by Diet Coke™ and Pizza
- **Real-life applications test weekly beta kits**
- **Framework development:**
  - Leanness is not necessarily a virtue
  - Using standard APIs as advantage instead of a limitation



Example 2: Web-based IDE

Web-based Integrated Development Environment:

- Web-Browser as front end
- Desktop sharing integrated
- Videoconferencing integrated
- Configuration management integrated

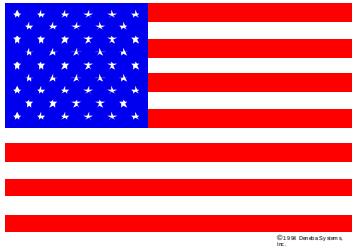
Goal:

- Tool for Distributed eXtreme Programming
- Integration of mobile and remote team members
- Better (off-site) customer involvement
- ‘Pair-programming everywhere’

Example 2: Web-based IDE

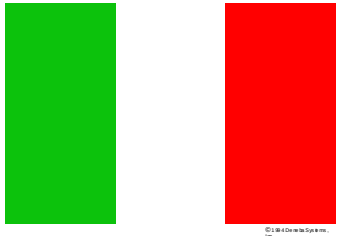
Set-up:

Prashant
Delhi,
Notebook,
Dial-up < 32kBit



Michael
Munich,
Notebook,
ISDN

David
Pittsburgh,
Desktop PC,
T1



Angelo
Catania,
Notebook,
Dial-up < 56 kBit

CORPORATE TECHNOLOGY

CT SE 2

Software & Engineering Architecture

Example 2: Web-based IDE

Lessons Learned:

- **Communication was quite effective**
  - However, it cannot fully replace physical proximity
- **Configuration management is essential**
- **No major problems in using application sharing and videoconferencing**
  - Bandwidth was mostly sufficient for application sharing and voice channel; video requires at least 64kB/s

Future:

- **Bigger need for DXP as development cycle times and budgets shrink**

XP Practice: Metaphor

- **What is the metaphor in XP?**
  - "The system metaphor is a story that everyone – customers, programmers, and managers - can tell about how the system works." [Kent Beck]
  - Weakly defined, a lot of room for (mis-)interpretations.

- **What is the benefit?**
  - Team speaks a common, precise language
  - Avoid misunderstandings
  - Establish a common vision

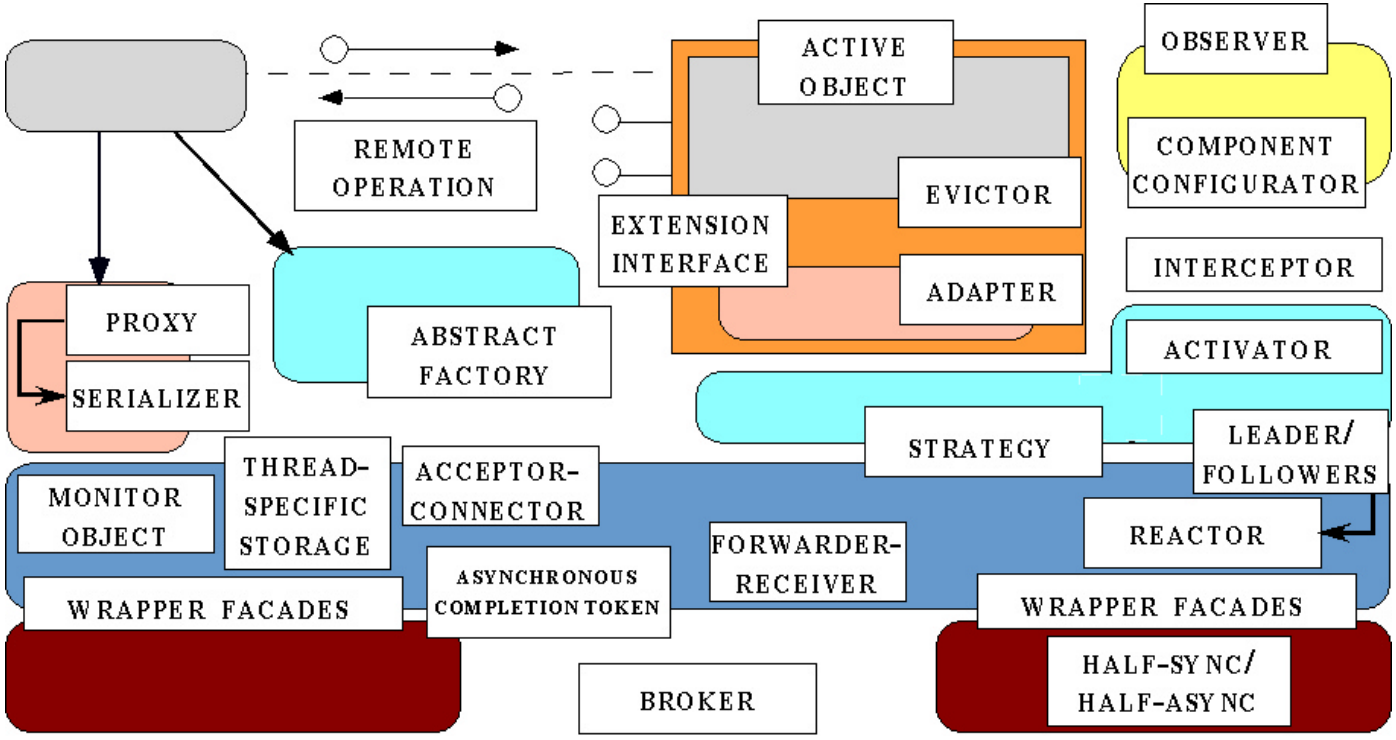
- **What are the limits?**
  - Some projects don't offer good metaphors.
  - Being too entrenched, not open to different/cross-cutting approaches

Patterns as the Metaphor

- **In some projects it seems impossible to come up with a shared story, e.g.**
  - In framework development, as the target domain might be well suited.
  - Implementing standard APIs, e.g. CORBA
- **Basic idea:**
  - **"XP is an effective means of communication".**
  - **"Patterns are effective communication".**
- **Patterns and Pattern Languages exist as a common language between developers.**

Patterns as the Metaphor - Example

- **Patterns used in the Real-Time CORBA Object Request Broker TAO**
  - "We need to enhance the implementation of the Leader/Follower model with a new strategy."

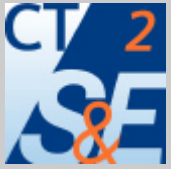


Patterns as the Metaphor

- **Patterns are used to communicate …**
  - Architecture, overall responsibilities and relationships
  - Design, interaction between collections of classes
    - Allow for reusable, and expandable designs
  - Implementation idioms, e.g., guarded locking
- **Though they …**
  - Require that everybody knows how to speak the common language
  - Might be of less use in communication with the customer
  - Might not be simple enough - might be too abstract to new programmers

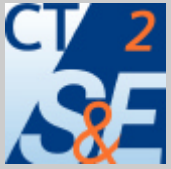
XP in Large Companies

- **Very cautious usage – Not Invented Here Syndrome**
- **'Process people' don't embrace it**
  - "Isn't it about programming?" ☹
- **Developers have to do it in storerooms.**
- **How to get started?**
  - Teams implementing subsystems have about the right size
  - Take the best and mix it with your traditional methodology – but carefully.
- **Start small and learn driving**



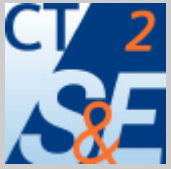
Other Adaptations

- **Adaptation [Collins]**
  - Treat XP “by the book” as your training wheels.
  - Constantly, introspect and retrospect
- **Scaling of XP [Crocker]**
  - Loosely coupled teams
  - Team coordination layer
- **Scaling down - Micro-XP [Adrian]**
  - Scaling down to a single person
  - Sacrifice pair programming and collective code ownership



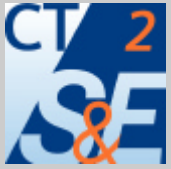
XP as a Management Philosophy

- **Manage with the XP “mind-set”**
- **Learn from you employees**
- **Get quick feedback from your project**
- **Communicate your visions early**
- **Start implementing instead of discussing and planning**
- **Formulate simple, clear goals**
- **‘Break Through Strategy’ [Schaffer]**
  - Try out and verify early to further deploy



References

- M. Kircher, P. Jain, A. Corsaro, and D. Levine, *Distributed Extreme Programming*, XP2001, Italy, May 21-23, 2001
- Jim Highsmith, *Agile Methodologies*, XP 2001, Italy, May 21-23, 2001
- Agile Alliance, http://www.agilealliance.org, 2001
- M. Kircher, D. Levine, *The XP of TAO*, 1st International Conference on eXtreme Programming and Flexible Processes in Software Engineering, Cagliari, Italy, June 21-23, 2000
- F. Adrian, *micro-eXtreme Programming (mXP): Embedding XP Within Standard Projects*, XP2001, Italy, May 21-23, 2001
- R. Crocker, *The 5 reasons XP can't scale and what to do about them*, XP2001, Italy, May 21-23, 2001
- M. Fowler, *Variations On a Theme of XP*, http://www.martinfowler.com, 2001
- R. Schaffer, *Breakthrough Strategy*, Harper Business, 1988
- G. Succi, M. Marchesi, *Extreme Programming - Examined, Addison-Wesley, 2001*
- C. Beard, *A look at Extreme Programming*, SEPG 2001, New Orleans, March 14, 2001



Thank you for listening.

Any questions?

Always remember:
Do the simplest thing that could possibly work ;-)

