Fortgeschrittene Themen des Projektmanagement

Prof. Dr. Peter Müller

8. September 2009

Agenda for Today

9:15 – 10:00	Principles
10:00 – 10:45	Earned Value Method
10:45 – 11:00	Break
11:00 – 11:45	Software Estimation
11:45 – 12:30	Risk Management
12:30 – 13:45	Lunch
13:45 – 14:30	Agile Methods
14:30 – 15:15	Scrum
15:15 – 15: 30	Break
15:30 – 16:15	Tools
16:15 – 16:45	Discussion

1. Principles

What is a Project?

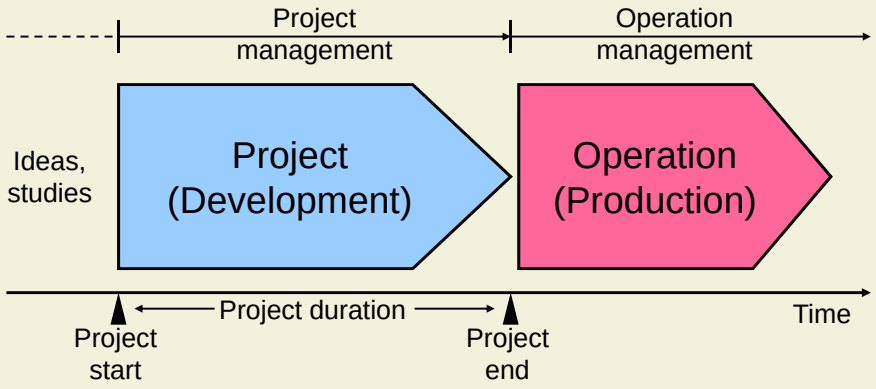
Every project has a definite beginning and a definite end

- Definition:
  A project is a temporary endeavor undertaken to create a unique product or service

The product or service is different in some distinguishing way from all similar products and services

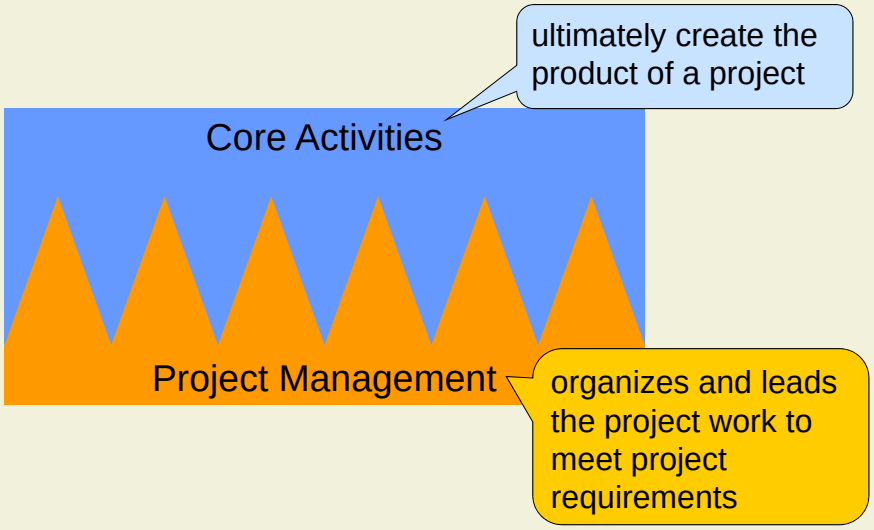
- In contrast: *Operations* are ongoing and repetitive

From Projects to Operations



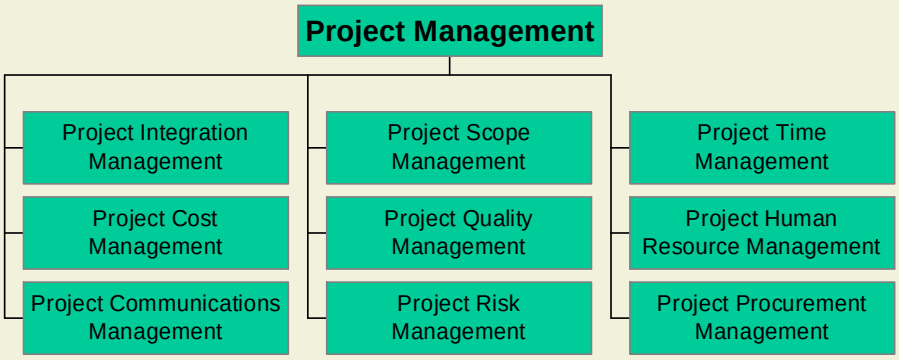
- Applications are neither projects nor operations, but products

Core Activities and Project Management

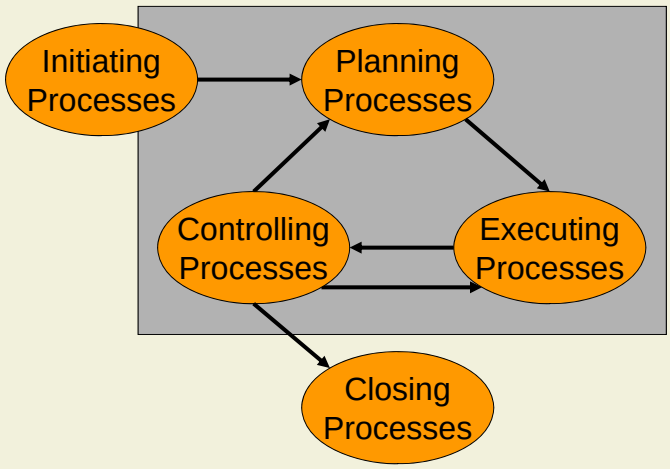


PM Knowledge Areas

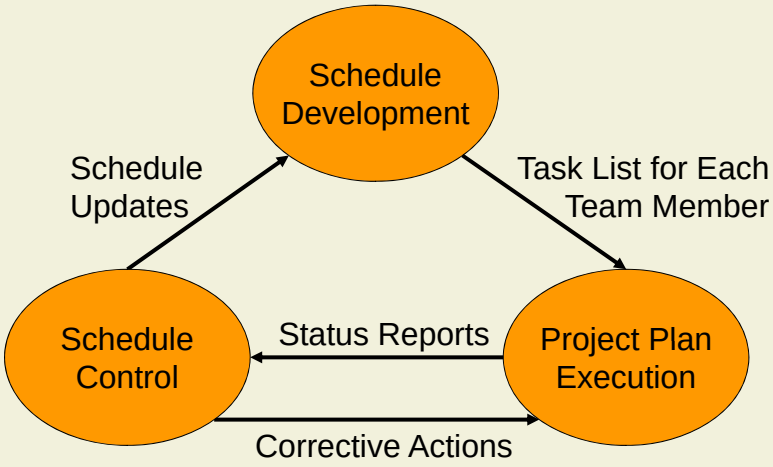
PM activities fall into nine Knowledge Areas



Project Management Life Cycle

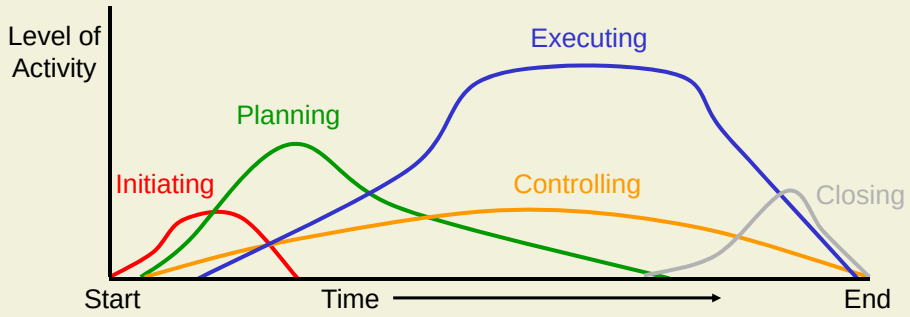


Example: Time Management

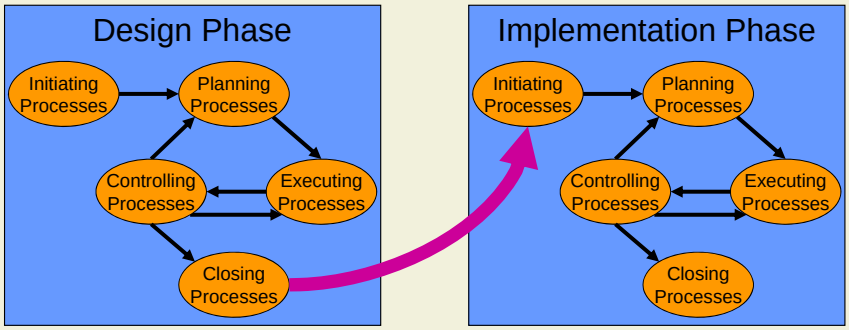


Process Groups

- Project groups are not discrete one-time events
- They overlap and occur at varying levels of intensity within each phase of the project



Interaction between Phases

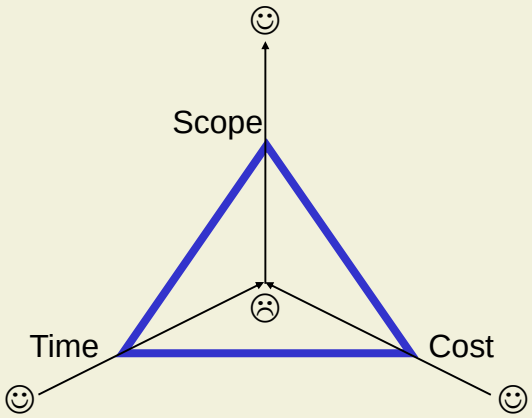


- Input and output of the processes depend on the phase in which they are carried out
- But processes are not limited to one phase (overlaps)

Project Success

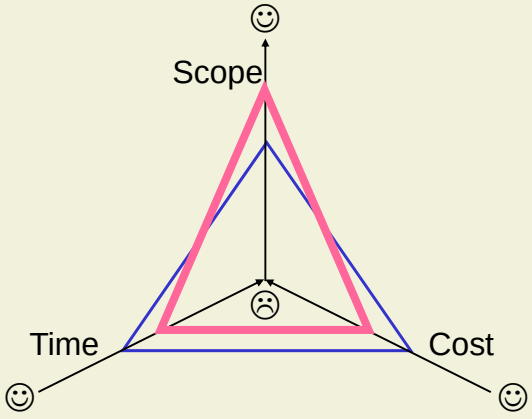
- Definition:
  A project is successful if the specified results are delivered in the required quality and within the predetermined time and resource limits.
- Computer scientists tend to focus on scope and quality only
  - The development of a technically perfect application is not a success if the cost exceeds the price clients are willing to pay
  - Excellent project results often are worthless if they come too late (temporary market windows, external deadlines)

The Triple Constraint



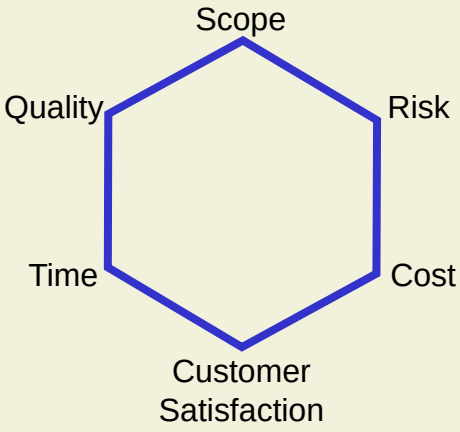
- Project objectives are **equally important**
- Actions in one project area usually affect other areas

The Triple Constraint



- **Tradeoffs** among objectives must be **managed**
- **Priorities** are set by customers and management

More Competing Objectives

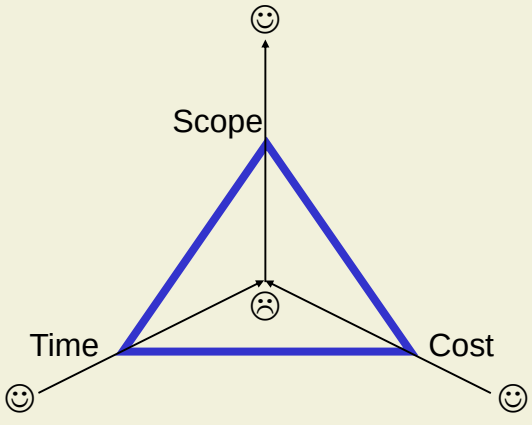


Assumptions

- Definition:
  Assumptions are factors that, for planning purposes, are considered to be true, real, or certain
- Assumptions affect all aspects of project planning, and are part of the progressive elaboration of the project
- Project teams frequently identify, document, and validate assumptions as part of their planning process
- Assumptions generally involve a degree of risk

2. Earned Value Method

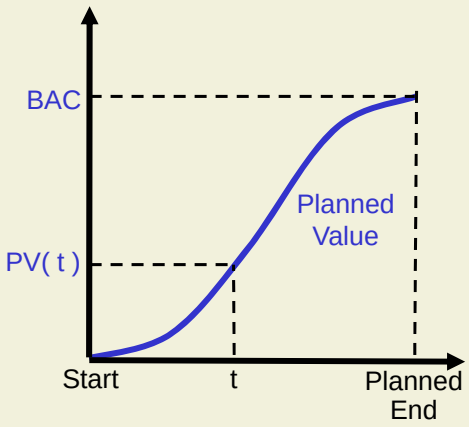
The Triple Constraint



- Project objectives are **equally important**
- Actions in one project area usually affect other areas

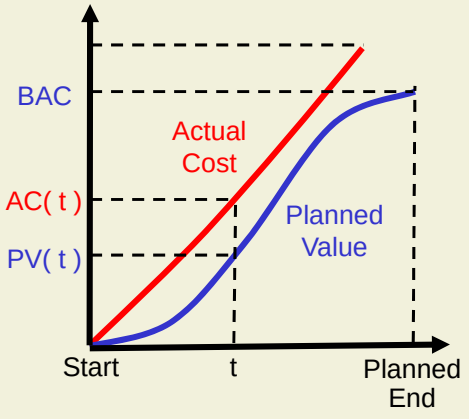
Planned Value (PV)

- The **cumulative** sum of the **approved** cost for activities **scheduled**
- Corresponds to the **cost baseline**
- **Budget at completion** is the estimated baseline total cost: BAC = PV(end)



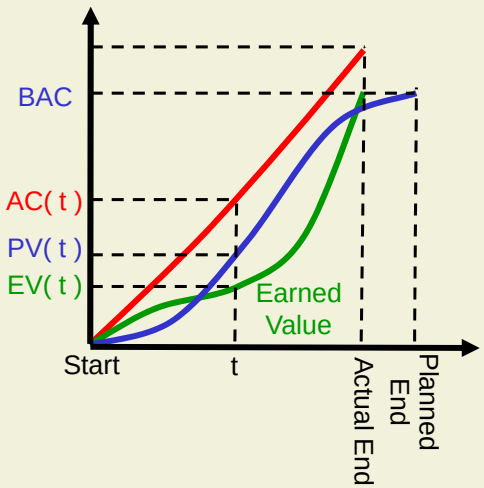
Actual Cost (AC)

- Total **cost incurred** for the project up to a specified date
- The **actual** or **real** cost of work performed
- Contains both direct and indirect cost



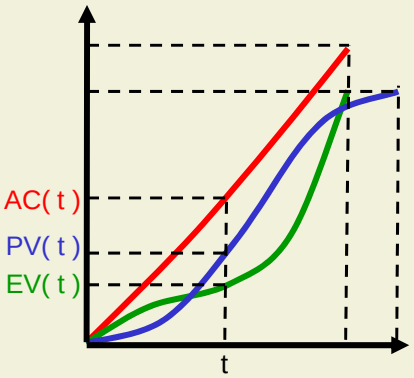
Earned Value (EV)

- The sum of **approved cost estimates** for activities **completed** up to a specified date
- An activity is completed if PV=EV, regardless of the actual cost

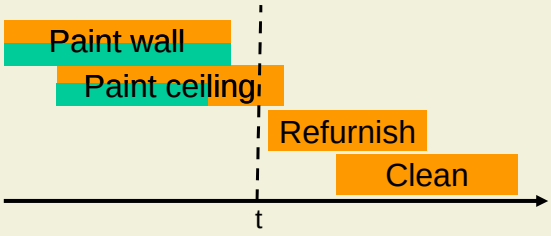


Earned Value Method

- Expresses effort, cost, and time as **monetary value**
  - PV(t): Worth of the activities scheduled (planned)
  - AC(t): Cost spent
  - EV(t): Worth of the activities performed
- Compares the amount of work planned to what was actually accomplished to **determine cost and schedule performance**



Example



Activity	PV(t)	AC(t)	EV(t)
Paint wall	800	1000	800
Paint ceiling	400	300	300
Total	1.200	1.300	1.100

Cost Performance Index (CPI)

- Compares **budgeted cost** of work performed to **actual cost**
- Indicates the **efficiency** of the project

$$CPI = \frac{EV}{AC}$$

- How much do we get out of one Franc we spend?

Activity	PV(t)	AC(t)	EV(t)
Paint wall	800	1000	800
Paint ceiling	400	300	300
Total	1.200	1.300	1.100

$$CPI = \frac{1.100}{1.300} = 85\%$$

Schedule Performance Index (SPI)

- Compares **work performed** to **work planned**

$$SPI = \frac{EV}{PV}$$

- How fast does the project progress in relation to how fast it is expected to progress?

Activity	PV(t)	AC(t)	EV(t)
Paint wall	800	1000	800
Paint ceiling	400	300	300
Total	1.200	1.300	1.100

$$SPI = \frac{1.100}{1.200} = 92\%$$

Calculated Estimate at Completion

$$CEAC_1 = \frac{BAC}{CPI}$$

- Budget modified by performance
  - If the current variances are **typical for the future**

$$CEAC_2 = AC + BAC - EV$$

- Actual to date plus remaining budget
  - If the current variances are **atypical for the future**

$$CEAC_3 = AC + ETC$$

- Actual plus a new estimate for remaining work
  - If the original estimate was **fundamentally flawed**

Interpreting EV-Indicators

- Typically, indicators are **stable after 20%** of the project duration

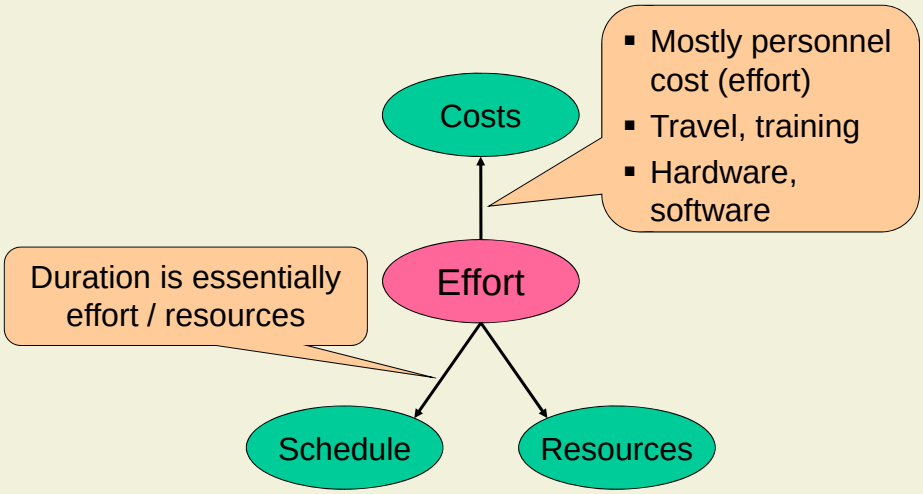
- CPI > 1: Project is in budget
- CPI < 1: Project is over budget
- SPI > 1: Project is ahead of schedule
- SPI < 1: Project is behind schedule

Golden Rules of Earned Value

- Rule 1: Earned value should be verified by **physically examining** the **work product** associated with the activity
- Rule 2: For unfinished activities, earned value estimates are usually just a guess. Apply one of the following rules consistently
  - **50/50 Rule**: A task is considered 50% complete when it begins and 100% only when it is completed
  - **20/80 Rule**: A task is considered 20% complete when it begins and 100% only when it is completed
  - **0/100 Rule**: A task does not get credit for partial completion, only for full completion

3. Software Estimation

Estimations in Software Projects



Estimation Exercise

- How many passenger planes does Lufthansa have?
  - Not counting regional subsidiaries
- How can we approach this problem systematically?

Empirical Estimation: Expert Judgment

- Estimate is based on **experience and historical data**
- Involve experts in
  - Development techniques
  - Application domain
- Most **common technique** in practice

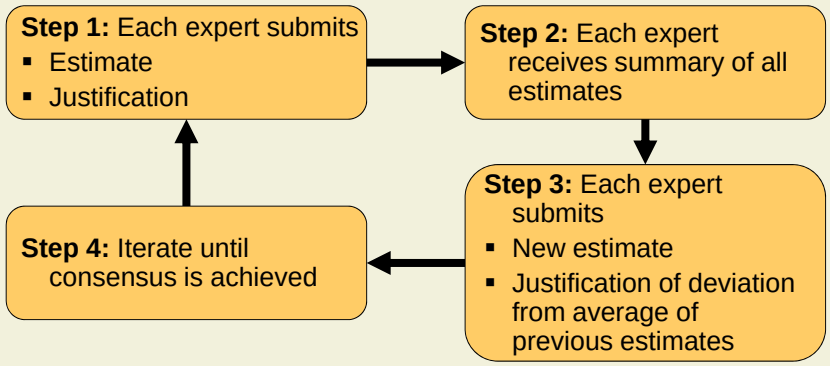
Top-Down Estimation

- Estimation by **analogy**
  - Comparison with **similar projects**
  - Analysis of differences
  - Typical example: SAP introduction

Pros
- Quicker and less expensive than other methods
- Can be done early in the project

Cons
- Underestimation of difficult technical problems likely
- No detailed justification of estimate
- Be aware of scalability problems!

Top-Down Estimation: Delphi Method



- **More accurate** than ordinary expert judgment
  - Eliminates outliers
- **More expensive** to produce

Bottom-Up Estimation

- Estimation by **decomposition**
  - Estimating the effort for **individual work packages**
  - Cost and accuracy depend on size of the work packages

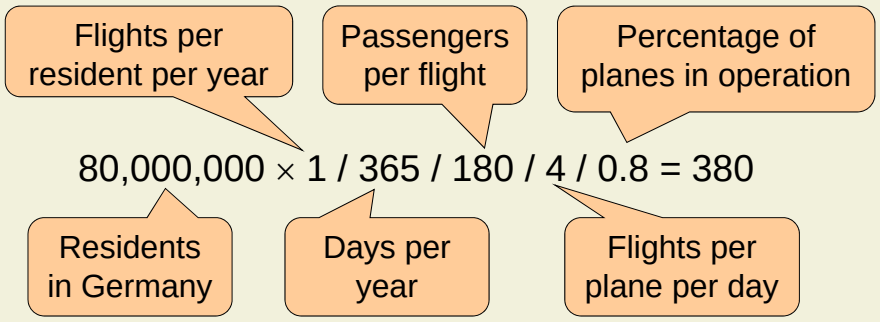
Pros
- See “cons” of top-down estimation

Cons
- Underestimation because effort does not grow linearly (due to complexity, etc.)
- Underestimation of integration effort
- Requires initial system design

Program Evaluation and Review Technique

- Goal: Manage probabilities with simple statistics
- Approach: Ask several experts for three estimates
  - **Optimistic, Likely (mode), and Pessimistic**
- Important formulas
  - Mean $M = (O + 4 \times L + P) / 6$
  - Deviation $V = (P – O) / 6$
- Assumptions
  - Project effort is normally distributed (more than 20 work packages)
  - Work package efforts are statistically independent (ignores single underlying cause of delay)

Algorithmic Estimation



- Algorithmic estimation is based on
  - **Cost model**, represented by formula
  - **Measurement of size** (passengers, destinations, etc.)
  - **Parameters** (size of planes, planes in operation, etc.)

Algorithmic Estimation of Software

- Basic cost model

$$\text{Effort} = A \times \text{Size}^B \times m(X)$$

- Size: Some measurement of the software size
- A: Constant factor that depends on
  - Organizational practices
  - Type of software
- B: Usually lies between 1 and 1.5
- X: Vector of cost factors
- m: Adjustment multiplier

Cost Models

$$\text{Effort} = A \times \text{Size}^B \times m(X)$$

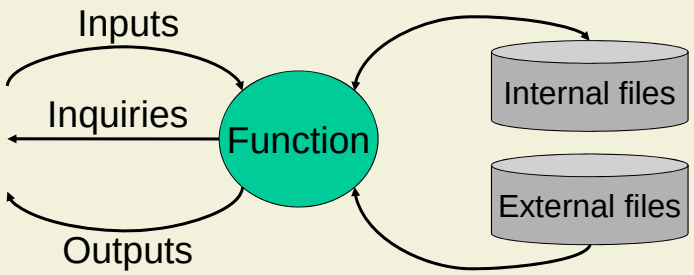
- Cost models
  - Define a way to determine the size
  - Define cost factors X
  - Provide defaults for parameters A, B, m (based on hundreds of projects)
- Important examples
  - Function point analysis
  - Constructive cost model (COCOMO)

Measuring Size: Lines of Code

- Software size can be measured in lines of source code
  - Most commonly used metric
- **Difficult in early phases** of the project (before design is known)
  - Reuse, make-or-buy decisions
- **Influenced** heavily by choice of **programming language**
- Should only be **used indirectly**

Function Point Analysis

- Size is estimated based on **requirements**



Functions

- **Inputs**
  - Forms, dialogs, messages, XML documents
- **Outputs**
  - Web pages, reports, graphs, messages, XML documents
- **Inquiries** (input/output combinations)
  - Simple web inputs, generally producing a single output
- **Logical internal files** (controlled by the program)
  - Tables, views or files in database
- **External files** (controlled by other programs)
  - Tables or files used from other systems or databases

Complexity of Functions

- Determine **complexity** of each function

Input	Simple	Average	Complex
Data elements	1-5	6-10	>10
Checking	Formal	Formal, logical	Formal, logical, requires DB access

- **Weight** each function according to complexity

Factor	Simple	Average	Complex
Inputs	3	4	6
Outputs	4	5	7
Inquiries	3	4	6
Ext. files	7	10	15
Int. files	5	7	10

Cost Factors

- Data communications
- Distributed processing
- Performance
- Heavy use
- Transaction rate
- Online data entry
- Complex interface
- Online data update
- Complex processing
- Reusability
- Installation ease
- Operational ease
- Multiple sites
- Facilitate change

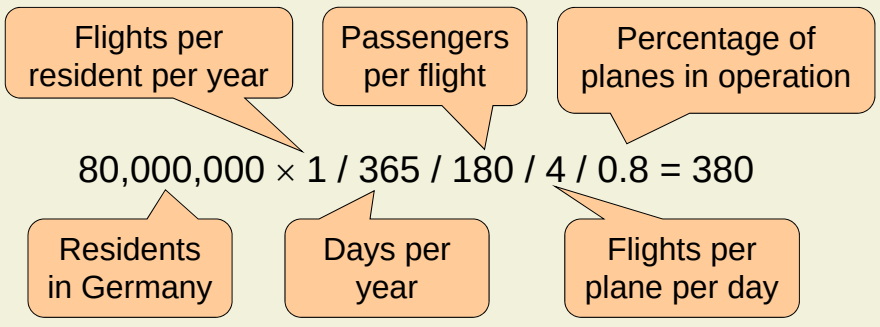
- Rate each element from 0 – 5
  - 0: no influence
  - 1: insignificant influence
  - 2: moderate influence
  - 3: average influence
  - 4: significant influence
  - 5: strong influence
- **Technical complexity factor**
  - TCF = 0.65 + 0.01 × sum
  - Varies between 0.65 and 1.35

Function Point Computation

	Simple	Average	Complex
Inputs	6 x 3 = 18	2 x 4 = 8	3 x 6 = 18
Outputs	7 x 4 = 28	7 x 5 = 35	0 x 7 = 0
Inquiries	0 x 3 = 0	2 x 4 = 8	4 x 6 = 24
Ext. files	9 x 5 = 45	0 x 7 = 0	2 x 10 = 20
Int. files	5 x 7 = 35	2 x 10 = 20	3 x 15 = 45
Unadjusted function points (UFP)			304
Technical complexity factor (TCF)			1.15
Adjusted function points			**350**

- **Adjusted function points**: FP = UFP × TCF

Calibration



- Assume that model (formula) is correct
- **Calibrate** model based on **comparable** airlines
- Estimate number of residents in the country

Determining Effort and Size

- Empirical value for effort
  - Or use a table

$$\text{Effort} = FP^{1.4} / 150$$

- Empirical value for size
- Huge differences in productivity
  - Factor 10-20 between individual programmers
  - Factor 4 between companies

Language	Level	Statements per UFP
Assembler	1	320
C	2.5	125
C++	6.5	50
Perl	15	25
Pascal	3.5	90
Visual Basic 3	10	30
Excel	50	6

Function Point Analysis: Discussion

Pros

- Based on requirements (instead of code size)
- Can be applied in early project phases
- Can be calibrated (for company, project type)
- Counting standards by "International Function Points User Group"
- Technology-independent

Cons

- Estimation of overall effort (not per phase)
- Tailored towards functional decomposition (rather than OO)
- Tailored towards information systems
- Needs calibration to produce reliable results

Estimation Techniques: Discussion

Empirical Estimation
- Accurate if experts are experienced
- Experts can be strongly biased (over-optimism)

Algorithmic Estimation
- Very accurate if model is calibrated
- Calibration is very difficult and expensive
- Estimation is expensive

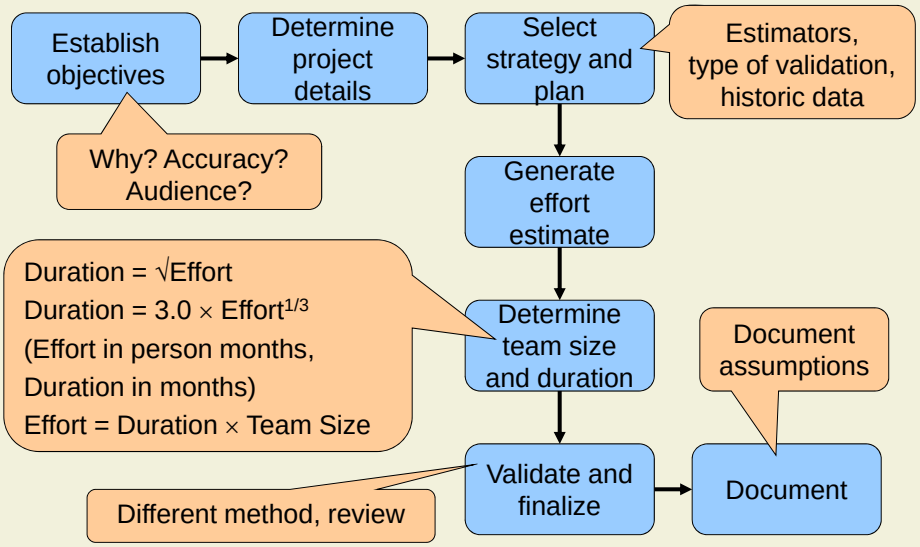
- Empirical studies
  - Do not show that uncalibrated algorithmic estimation is, in general, more accurate
  - Show that algorithmic estimation is more accurate than experts who do not have important domain knowledge

Other Estimation Strategies

Parkinson's Law
- Work expands to fill the time available
  - Gold plating
- Effort is determined by available resources
- Important for team management

Pricing to win
- Cost is estimated to whatever the customer is willing to spend
- Common strategy to win projects
- Features are negotiated later, constrained by agreed costs
- Costs are fixed, not requirements

Estimating Process



4. Risk Management

Risk

- Definition:
  An uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective

- Risks have three components
  - A possible future event (uncertainty)
  - Probability of the occurrence of that event (likelihood)
  - Impact of that event (consequence)

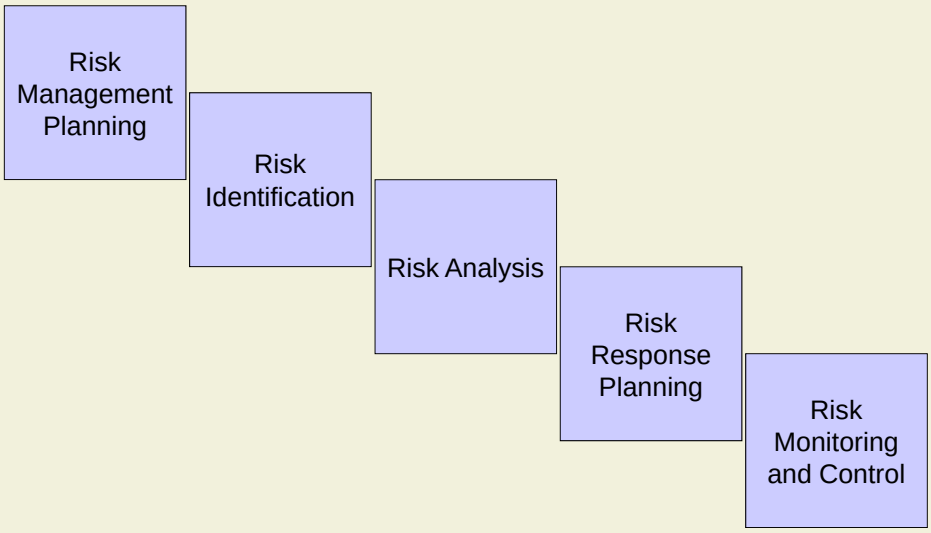
"Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns — the ones we don't know we don't know."

Donald Rumsfeld, 2003

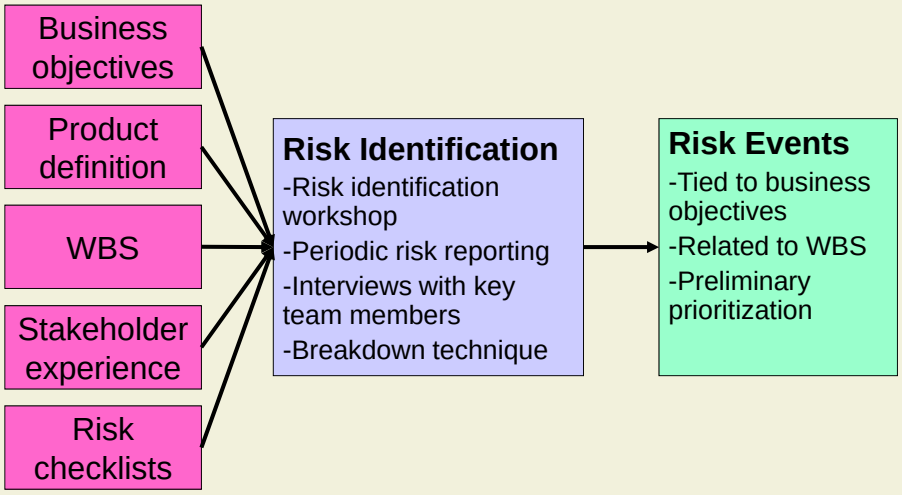
Risk Classification

- Known risks
  - Unclear requirements
  - Inexperienced team
- Unknown risks: Foreseen based on experience
  - Difficult communication with customer
  - Fluctuation within team
- Unknowable risks: Cannot be foreseen
  - Half of the team gets fish poisoning at first social event
  - Earthquake wipes out production plant

Risk Management

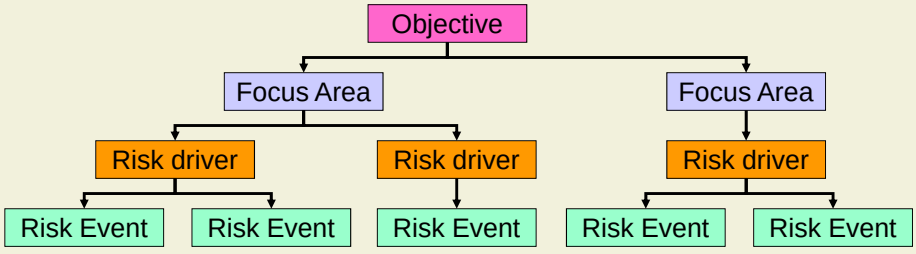


Risk Identification

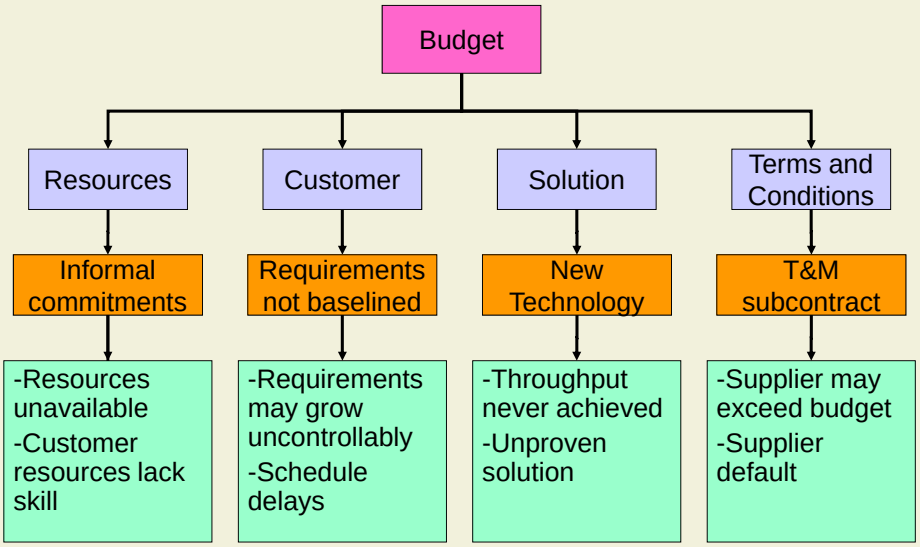


Breakdown Technique

- Identify risks systematically
  - Project objectives: Win, Budget, Satisfy
  - Focus area: A breakdown of the project's potential sources of risk
  - Risk driver: A condition that increases the probability that a risk event will be present



Breakdown Example



Risk Analysis

- Often called risk assessment or risk evaluation
- Determine
  - **Probability** of the risk to occur
  - **Impact** on the project objectives in case the risk occurs
  - **Severity** (Severity = Probability x Impact)
- Identify risks to be mitigated

- Qualitative analysis
- Quantitative analysis
  - Based on estimates and simulations

Probability Criteria

Qualitative Rating	Quantitative Rating	Description
Very High	>84%	Almost assured to happen
High	60-84%	Likely to happen
Medium	35-59%	Somewhat likely to happen
Low	10-34%	Not very likely to happen

Impact Criteria

Risk Rating	Description
Very High	Has potential to cause cancellation of the project
High	Likely to cause significant disruption to schedule, increase in cost, or degradation of performance
Medium	Has potential to cause some disruption to schedule, increase in cost, or degradation of performance
Low	Has little potential to cause some disruption to schedule, increase in cost, or degradation of performance

Severity of Individual Risks

Severity		Impact			
		Very High	High	Medium	Low
Probability	Very High	Unacceptable	Very High	High	High
	High	Very High	High	High	Medium
	Medium	High	High	Medium	Medium
	Low	High	Medium	Medium	Low

Risk Ranking

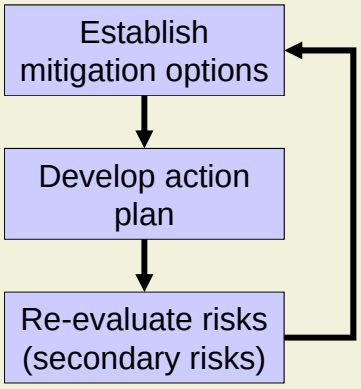
- Prioritize risk according to
  - Severity
  - Timing
  - Time required to mitigate (preliminary estimate)
  - Etc.
- "Top 10" Approach
  - Develop mitigation strategies for top 10 risks
  - Use the top 10 as an agenda item for regular project meetings

Risk Documentation

Risk ID	WBS Number	Risk Event	Owner	Area of Impact (W/B/S)
1	2.04.05	Requirements will grow uncontrollably	PM	B/S

Probable Impact Date	Risk Probability	Risk Impact	Severity	Rank
20.07.2004	High	Very high	Very high	1

Risk Response Planning



- Risk Response Planning is often called **risk mitigation**

Mitigation Strategies

- **Avoid** the path or project to eliminate the risk
- **Ignore / Accept** the risk and its consequences if it occurs
- **Transfer** all or part of the risk to another party
- **Contain** the risk by specific actions to lower the probability and / or impact
- **Establish contingency**: Set funds aside to be used if the risk occurs or when later containment is deemed appropriate

Extended Documentation

Risk ID	Mitigation Strategy	Mitigation Task	Responsible	Status
1	Contain	Use phased approach	PM	in progress

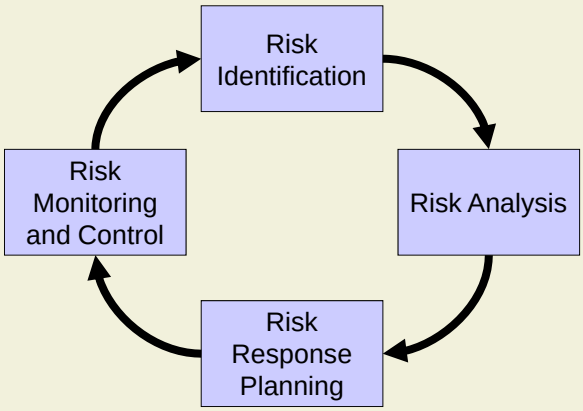
Execution and Post Mitigation

- Mitigation tasks have to be **integrated into project plan** (WBS)
- Execution has to be **closely monitored**
- Risk has to be **re-assessed** to check whether mitigation is successful

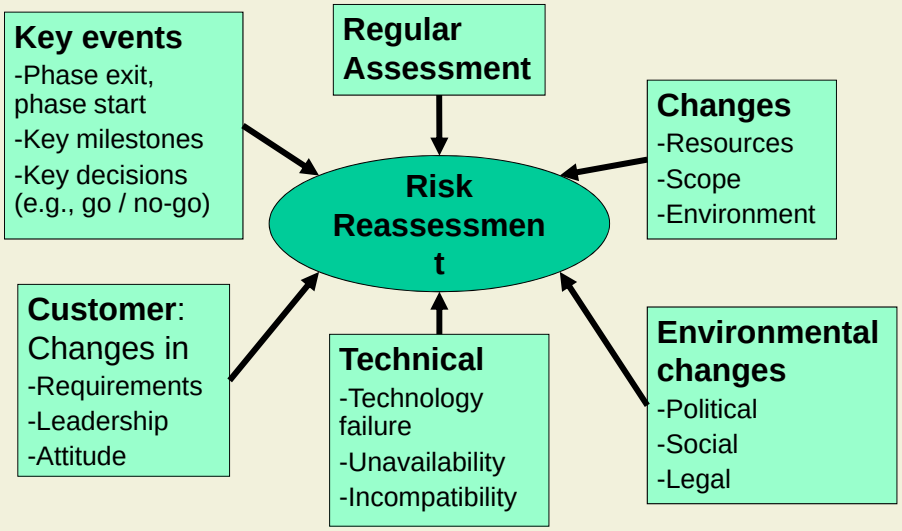
Severity		Impact			
		Very High	High	Medium	Low
Probability	Very High	Very High	Very High	High	High
	High	Very High	High	High	Medium
	Medium	High	High	Medium	Medium
	Low	High	Medium	Medium	Low

Risk Monitoring

- **Implement, track**, and **reassess** mitigation strategies
- **Communicate** risk plan status to stakeholders
- **Update** documents



Triggers for Risk Reassessment



5. Agile Methods

Hermann Lehner

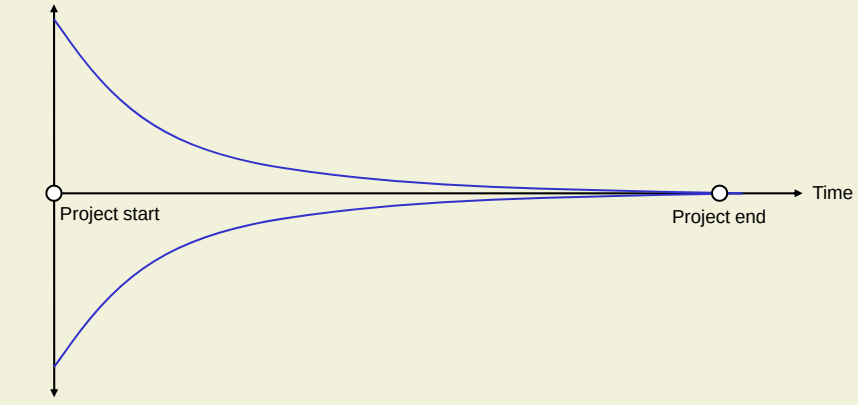
Overview

This introduction to agile planning is divided into five parts:

- Motivation for agile planning
- An agile approach to projects
- An agile approach to planning
- Estimating in agile projects
- The Scrum empirical development process

Agile Project Characteristics

- Many **changes** during the projects
- Wide "**Cone of uncertainty**" in the beginning



Impact on Planning

Planning …

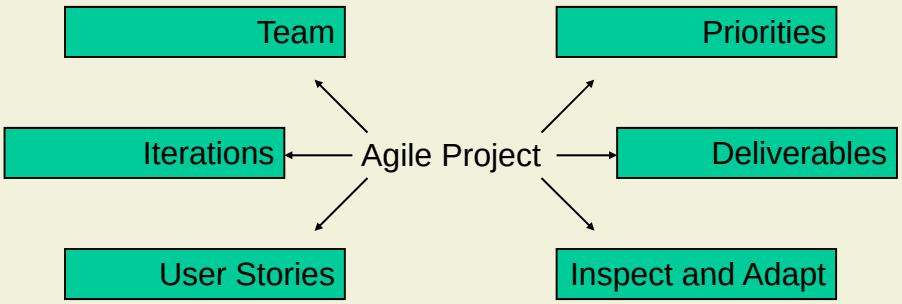
- is focused more on the **process of planning** than on the plan
- encourages **changes**
- results in plans that can be **easily changed**
- is **spred** throughout the project

Why traditional planning fails

- Features are not developed by **priority**
- Planning by **activity** rather than **feature**
  - Activities don't finish early
  - Activities don't have a user value
  - Activities are not independent

- **Uncertainty** is ignored
- **Multitasking** causes further delays

An agile approach to projects



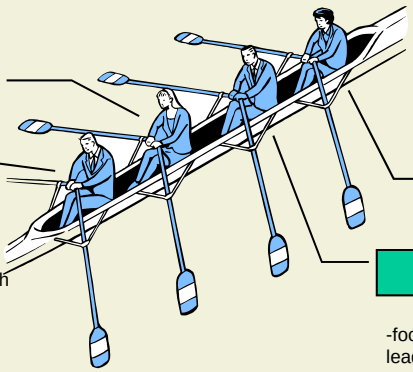
Agile Team – four Roles

-Makes decisions to increase return on investment
-Establishes common vision of the product
-Defines priorities

Product owner

Customer

-funds or buys the product
-sometimes combined with product owner



-develops the product

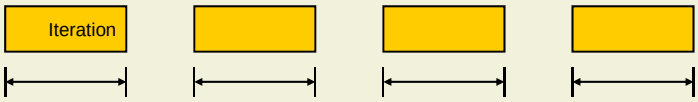
Developer

Project Manager

-focus more on leadership than management

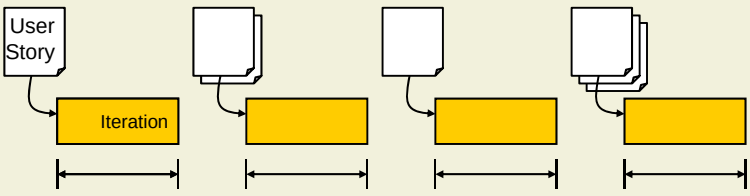
Iterations

- Typical **duration** between **2** and **4 weeks**
- **Timeboxed**: Iteration finishes always on fixed date
- Number of implemented **features** is **variable**



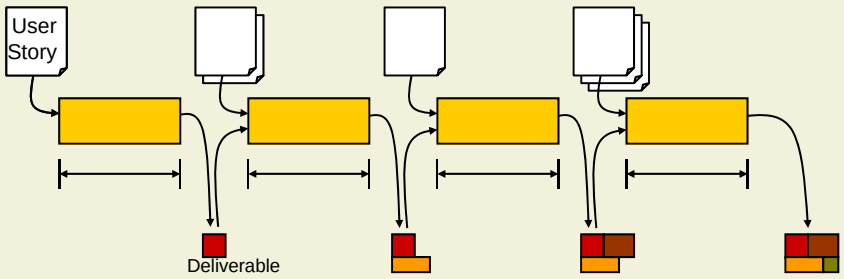
User Stories

- Brief description of functionality as viewed by a user or customer of the system
- Free form, no mandatory syntax, lightweight



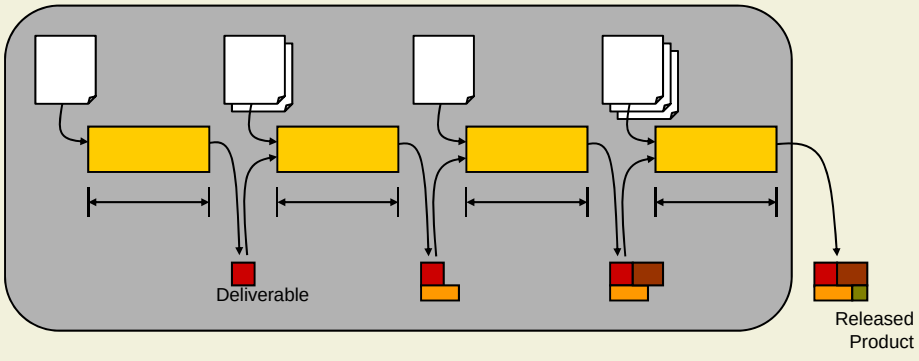
Deliverables

- Each iteration results in a deliverable
- Coded, tested, and **potentially shippable**
- Small addition of functionality



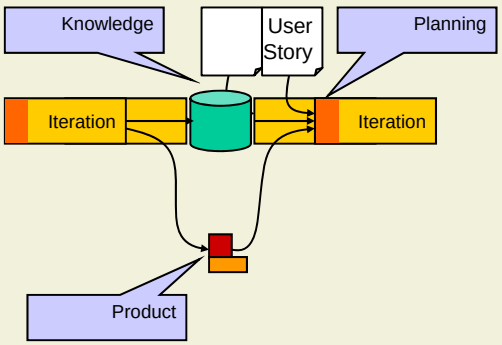
Release

- Consists of iterations that add related functionality
- Usually every 2 to 6 months
- User can see big improvement over last release



Inspect and Adapt

- At start of each iteration, gained knowledge can lead to change of plan
- Changes add more value to a plan



Focusing on Business Priorities

- Features delivered in order specified by product owner

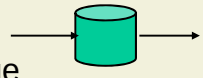
  Decision is influenced by
  - Priority list from customer
  - Team's capabilities
  - Optimal return on investment

- Features need to have **minimized** technical **dependencies**

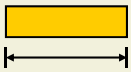
An agile approach to planning

Summary of characteristics of agile projects:

- Project generates **knowledge**. This should be used!
  - Product knowledge
  - Project knowledge



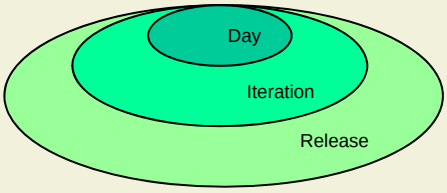
- An agile Project is more a **timed** race than a 10km race



- Result of Project is unknown and **unknowable**

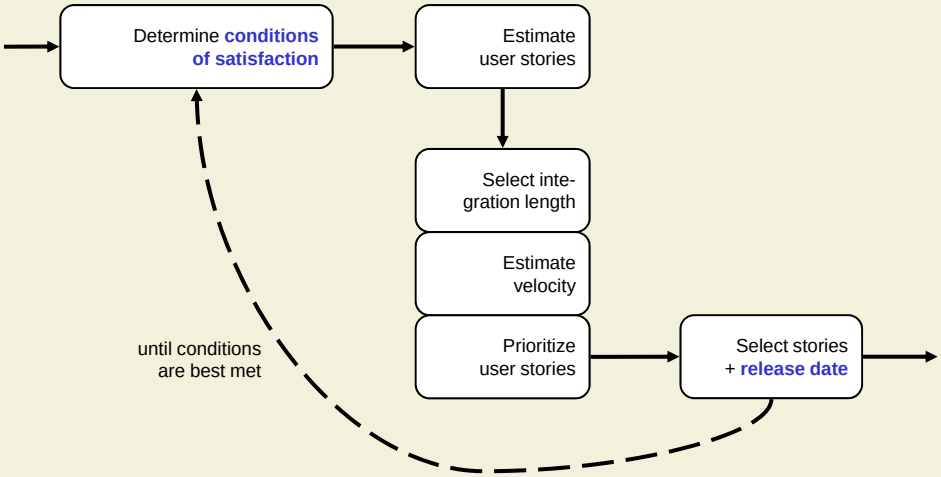
Levels of Planning

- Planning is looking ahead
  … but you can't look at things behind the **horizon**!
- Different horizons

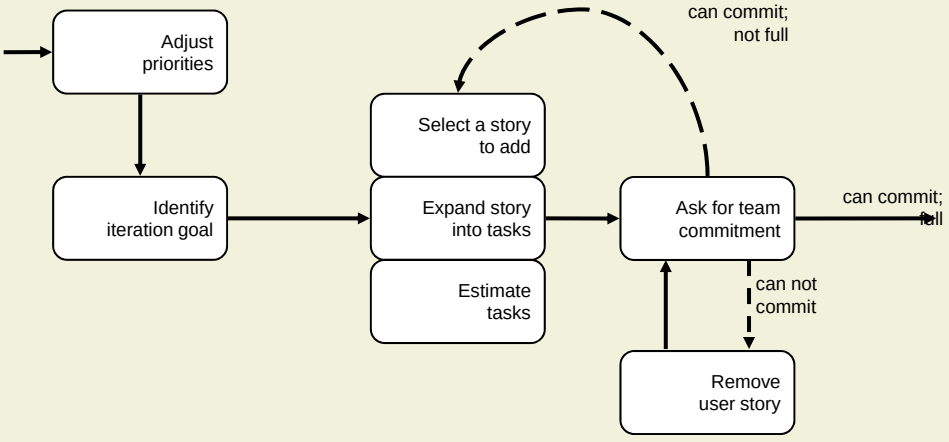


- Take time to look further ahead once you reached the horizon

Release Planning



Iteration Planning



Daily Planning

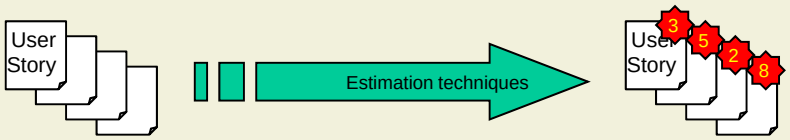
- Stand-up meeting
- Coordinate activities of tasks
- Synchronize daily efforts
- Horizon is end of day (don't plan any further!)

Estimating in agile projects

- Estimation techniques do not change
  - Expert Opinion
  - Analogy (Top Down)
  - Decomposition (Bottom Up)
  - Delphi method
- Estimation units
  - Story points ←
  - Ideal days
- Deriving duration (velocity)
- Estimation scale

Story Points

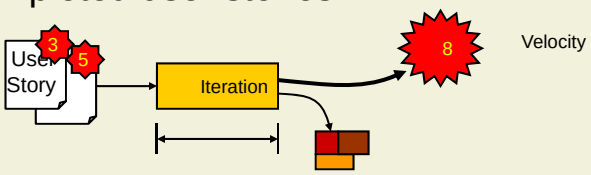
- Assign 1-10 points to all user stories



- Start with medium sized stories (assign 5 points)
- Estimating story points can be easier/faster than estimating ideal days
- Humans can only compare things within one order of magnitude

Deriving Duration (velocity)

- After first iteration, sum up story points for completed user stories



- Take that value as **velocity** of the team
- **Correct** velocity value after every iteration

Estimation scale

- Don’t overdo it! (Again: don’t ignore uncertainty)
- Reasonable scales are
  - 1, 2, 3, 5, and 8
  - 1, 2, 4, and 8
- Vague user stories are often described as ‘**Epic**’ or ‘**Theme**’ (not in one order of magnitude)
- One can extend scale for them
  - 13, 20, 40, 100
- Split them up as soon as knowledge allows it

6. SCRUM

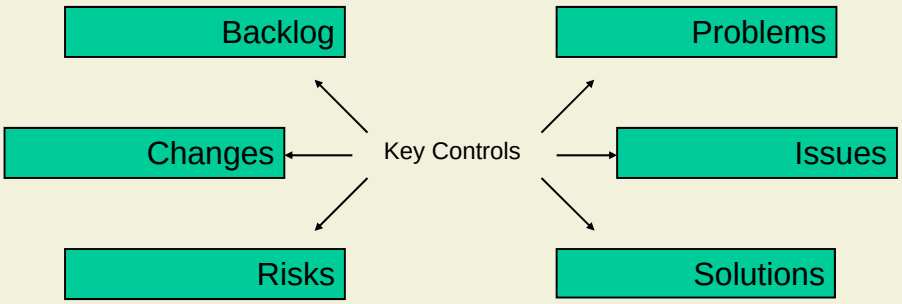
Hermann Lehner

The SCRUM Development Process

What is Scrum?

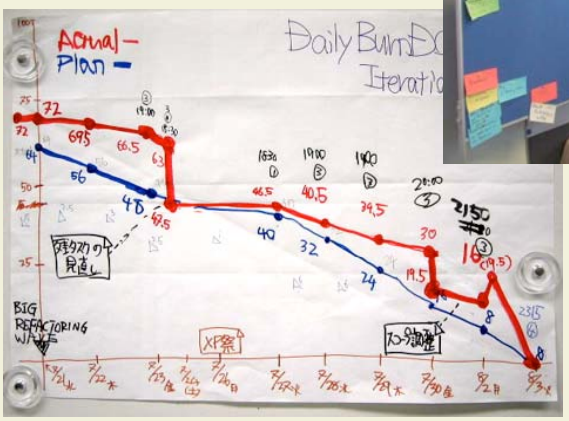
- **Agile** development process
- **Establishes**, **maintains** and **monitors** key control parameters.
- Use **measurements** to maximize **flexibility** while maintaining control.

Key Control Parameters



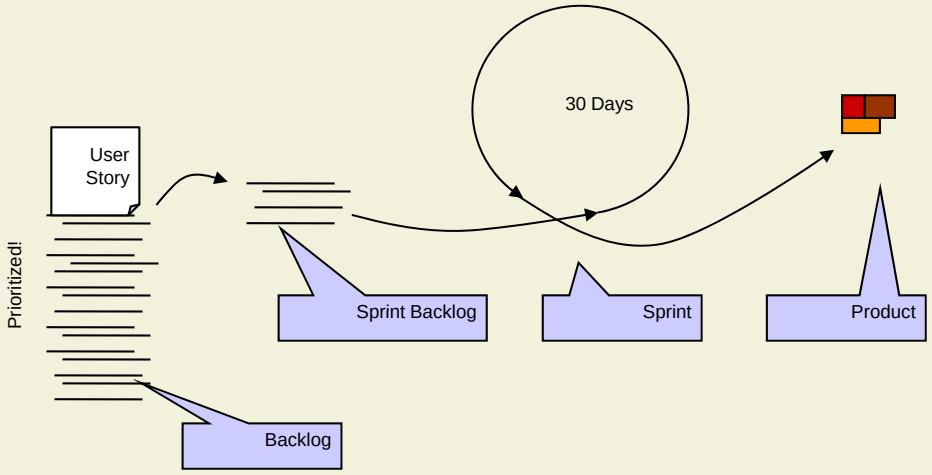
Measurements

Burnd Down Chart



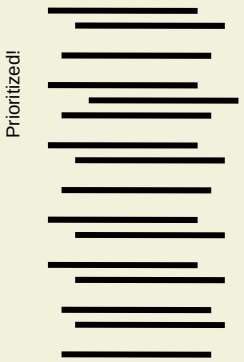
Meetings

The Scrum Process



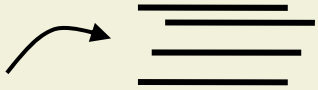
Product Backlog

- **Prioritized** list of features
- Free form
  - User Stories
  - Scenarios
- Measurement for progress
- **Changes** over time
- No commitment



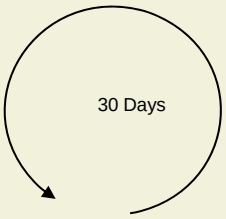
Sprint Backlog

- Work to **do** in one sprint
- Team estimates effort



Sprint

- Get all sprint backlog items into a shippable piece of product
- Takes 30 days
- Team organizes itself
- Results presented at the end
- Influences priorization of backlog
- On the edge of chaos



Daily Scrum

- 15 min meeting
- Standing (shorter)
- Tool for measurement!

- Three Questions
  - What did you do since the last meeting?
  - What will you do until the next meeting?
  - What problems do you have?

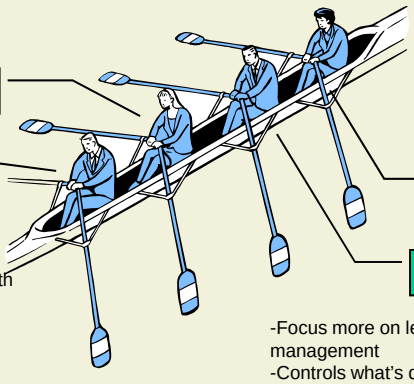
Roles

-Makes decisions to increase return on investment
-Defines backlog item priorities

Product owner

Customer

-Funds or buys the product
-Sometimes combined with product owner



-Develops the product

Scrum Team

Scrum Master

-Focus more on leadership than management
-Controls what's don
-Bad guy

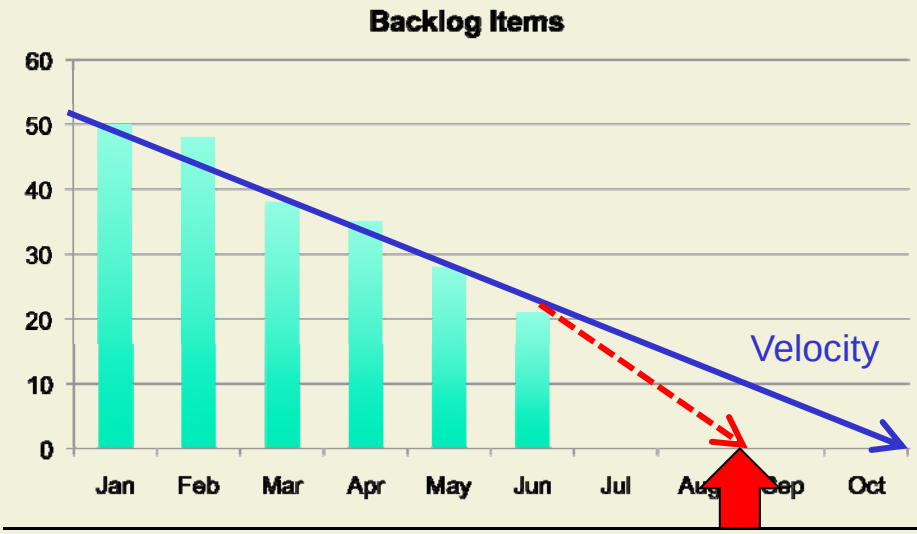
Transparency

- Daily Scrums
  transparency in team
  peer pressure
- Sprint presentations
  transparency towards customer/owner

“Done”

- A very important word in Scrum
- **Done = Analyzed, designed, coded, tested, committed, documented**
- Done = Removed from the backlog

Burndown Chart



Scrum Philosophy

Transparency
Timeboxed
On the edge of chaos
Getting things done (DONE)
Burndown chart

7. Tools

Spreadsheet

- Applications
  - Estimation
  - Risk management
  - Planning and controlling for small projects (task lists)
  - Reporting
  - Documentation of assumptions, open issues, change requests, identified risks, etc.
- Benefits: computations, sorting, filtering, visualization
- Popular tools: MS Excel, OpenOffice Calc

Project Management Tool

- Applications
  - Scheduling
  - Resource allocation
  - Controlling
  - Reporting
- Benefits: Elaborate computations, elaborate visualizations, support for various methods (e.g., Earned Value)
- Drawbacks: Complexity
- Popular tools: MS Project

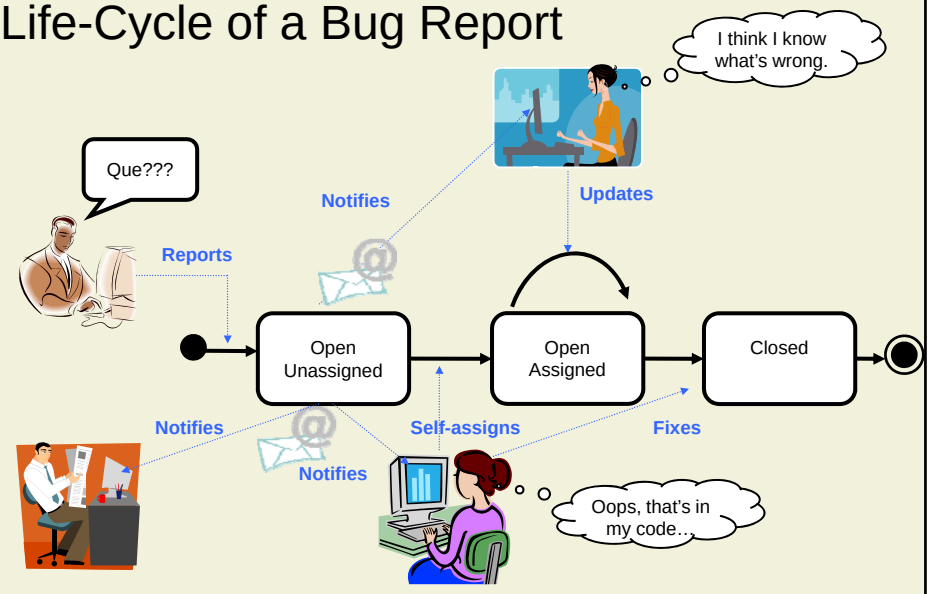
Collaboration Tool

- Applications
  - Coordination of (possibly distributed) teams
  - Integration of configuration management, bug database, etc.
  - Information dissemination via web
- Benefits:Transparency (tasks, bugs, etc.), Support for agile methods
- Drawbacks: Complexity
- Popular tools: Rational Team Concert (Jazz), Trac

Bug Tracking

- Applications
  - Manage list of all known bugs
  - Maintain history for each bug
  - Manage collaboration
- Benefits: Transparency, statistics
- Widely-used products: Bugzilla, Mantis, GNATS, Flyspray

Life-Cycle of a Bug Report



Bug Report Attributes

- Date: report, updates, close
- Status: open, closed, deleted, assigned
- Short summary
- Detailed description
- Severity and priority
- Platform and version number
- Category: GUI, installation, certain module
- Reproduction: description or test-case

Request ID	Summary	Open Date	Priority	Assigned To	Submitted By
1457565	Startup Fails	2006-03-23 23:49	5	datallah	be_brg_peh

Bug Report Guidelines

- One issue per report
- Make sure it is not reported yet
- How to write useful bug report
  - Summary: short but quickly and uniquely identifies bug
  - Be specific: product, version, component, platform, OS include list of third party software used
  - Reproducible: provide least amount of steps

Configuration Management

- Applications
  - Manage source code and other resources
  - Manage versions
  - Manage collaboration
- Benefits:
  - Permit concurrent development
  - Store whole history
  - Back-up
  - Good anchor for quality assurance activities
- Popular tools: cvs, subversion (svn)

Automatic Code Inspection

- Applications
  - Enforce coding conventions
  - Check for common errors (beyond compiler and IDE)
- Benefits:
  - Reduces effort for manual code reviews
  - Large set of pre-defined rules
  - Rules can be adapted and extended
  - Can be integrated in IDE
- Popular tools: CheckStyle, PMD, FindBugs

Profiler

- Applications
  - Gather performance data for application
  - Identify bottlenecks
- Benefits:
  - Prepare code optimization
  - Optimize only those parts that are actually problematic
- Popular tools: Integrated in IDE (NetBeans, Visual Studio)

Conclusion

Feedback

- What did you like about the course?
- What should be improved?
- Did the course meet your expectations?

Appendix

A. Appendix

A.1 Definitions

A.2 References

Project Management

- Definition of Project Management (PM):
  Project Management is the application of knowledge, skills, tools, and techniques to project activities to meet project requirements.

What is an IT-Project?

- Definition:
  An IT-project is a project to create a product or service, of which the usage of information technology is the decisive characteristic

- Examples
  - The development of a software application is an IT-project (IT-based product)
  - The development of a car is not an IT-project, although information technology is involved substantially

Project Success

- Definition:
  A project is successful if the specified results are delivered in the required quality and within the predetermined time and resource limits.
- Computer scientists tend to focus on scope and quality only
  - The development of a technically perfect application is not a success if the cost exceeds the price clients are willing to pay
  - Excellent project results often are worthless if they come too late (temporary market windows, external deadlines)

Progressive Elaboration

Characteristics of a unique product or service must be progressively elaborated

Continuing steadily by increments

Worked out with care and detail

- During the project, characteristics are defined in more detail as the project team develops a better and more complete understanding of the product

Constraints

- Definition:
  Constraints are factors that limit the project team's options
- A single project may contain cost, time, human resource, technical, and other constraints
- Examples
  - External deadlines (e.g., Y2K, Euro)
  - Fixed upper limits for budget
  - Dependencies on other projects, etc.

Baseline

- Definition:
  The originally approved plan plus or minus approved changes.

- Baselines are used to compare the actual performance and forecasts of the project with the original plan

Deliverables

- Definition:
  Any measurable, tangible, verifiable outcome, result, or item that must be produced to complete a project or part of a project

- Examples
  - An object-oriented design, described by a UML diagram
  - A project schedule as MS Project file
  - A user guide for a new application
  - Software, delivered as compiled binary

Work Breakdown Structure (WBS)

- Definition:
  A deliverable-oriented, hierarchical grouping of project elements that organizes and defines the total work scope of the project. Each descending level represents an increasingly detailed definition of the project

Stakeholders

- Definition
  Individuals and organizations that are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or project completion; they may also exert influence over the project and its results
- Key stakeholders
  - Project manager
  - Customer
  - Performing organization
  - Project team members
  - Sponsor

Float

- Definition:
  The amount of time that an activity may be delayed from its early start without delaying the project finish date
- Float = LF – EF = LS – ES
- Interpretation
  - Float > 0: Time is available
  - Float = 0: Situation is critical
  - Float < 0: Project is behind
- Sometimes called *Total Float*, *Slack*, or *Total Slack*

Critical Path

- Definition:
  The series of activities that determines the duration of the project (the longest path through the network)
- Sum of float on critical path is zero (or negative)
- Critical path is important
  - To shorten project duration
  - To focus progress control
  - To identify schedule risks
- There can be several critical paths in a project

Milestones

- Definition:
  A significant event in the project, usually completion of a major deliverable

- Milestones have no effort or duration
- Milestones do not have resources

- Example: Painting completed

Risk

- Definition:
  An uncertain event or condition that, if it occurs, has a positive or negative effect on a project objective

- Risks have three components
  - A possible future event (uncertainty)
  - Probability of the occurrence of that event (likelihood)
  - Impact of that event (consequence)

Risk Management

- Definition:
  Systematic process of identifying, analyzing, and responding to project risk. It includes minimizing the consequences of adverse events to project objectives.

A. Appendix

A.1 Definitions

A.2 References

Literature: IT-Projects

- Zehnder, Carl August: Informatik-Projektentwicklung, vdf Hochschulverlag, 2003
- Jenny, Bruno: Projektmanagement in der Wirtschaftsinformatik, vdf Hochschulverlag, 2000
- Steinweg, Carl: Projektkompass Softwareentwicklung, Vieweg, 2002
- Frühauf, Karol: Software-Projektmanagement und -Qualitätssicherung, vdf Hochschulverlag, 2002
- Gaulke, Markus: Risikomanagement in IT-Projekten, Oldenbourg Verlag 2002

Literature: General Project Management

- A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Project Management Institute, 2000
- Mulcahy, Rita: PMP Exam Prep (4th Edition), RMC Publishing, 2002
- Fleming, Quentin, Koppelman, Joel: Earned Value Project Management, Project Management Institute, 2000
- Mike Cohn: Agile Estimating and Planning, Robert C. Martin Series, Prentice Hall, 2006
- PMI – Project Management Institute, www.pmi.org

Literature: Agile Methods and SCRUM

- Mike Cohn, *"Agile Estimating and Planning"*, Robert C. Martin Series, Prentice Hall, 2006
  ISBN 0-13-147941-5
- Control Chaos
  http://www.controlchaos.com
- Google Tech Talk by Ken Schwaber
  http://www.youtube.com/watch?v=IyNPeTn8fpo

Contact Information

- Peter Müller
  - Email: peter.mueller@inf.ethz.ch
  - Web page: www.pm.inf.ethz.ch
  - Course page: www.pm.inf.ethz.ch/teaching/as2009/IPM/index.html
    - Login: project
    - Password: management