
Tools Overview

Why Do You Need Tools*?

- Software development requires tool support because it ...
 - has to be a highly systematic process due to its complexity,
 - is done by teams which cooperate on subtasks, and
 - reuses generic components and services.

****tool*:** an implement, such as a hammer, saw, or spade, ... used as a means for performing an operation or achieving an end
[Collins: Dictionary of the English Language]

Kinds of Tools

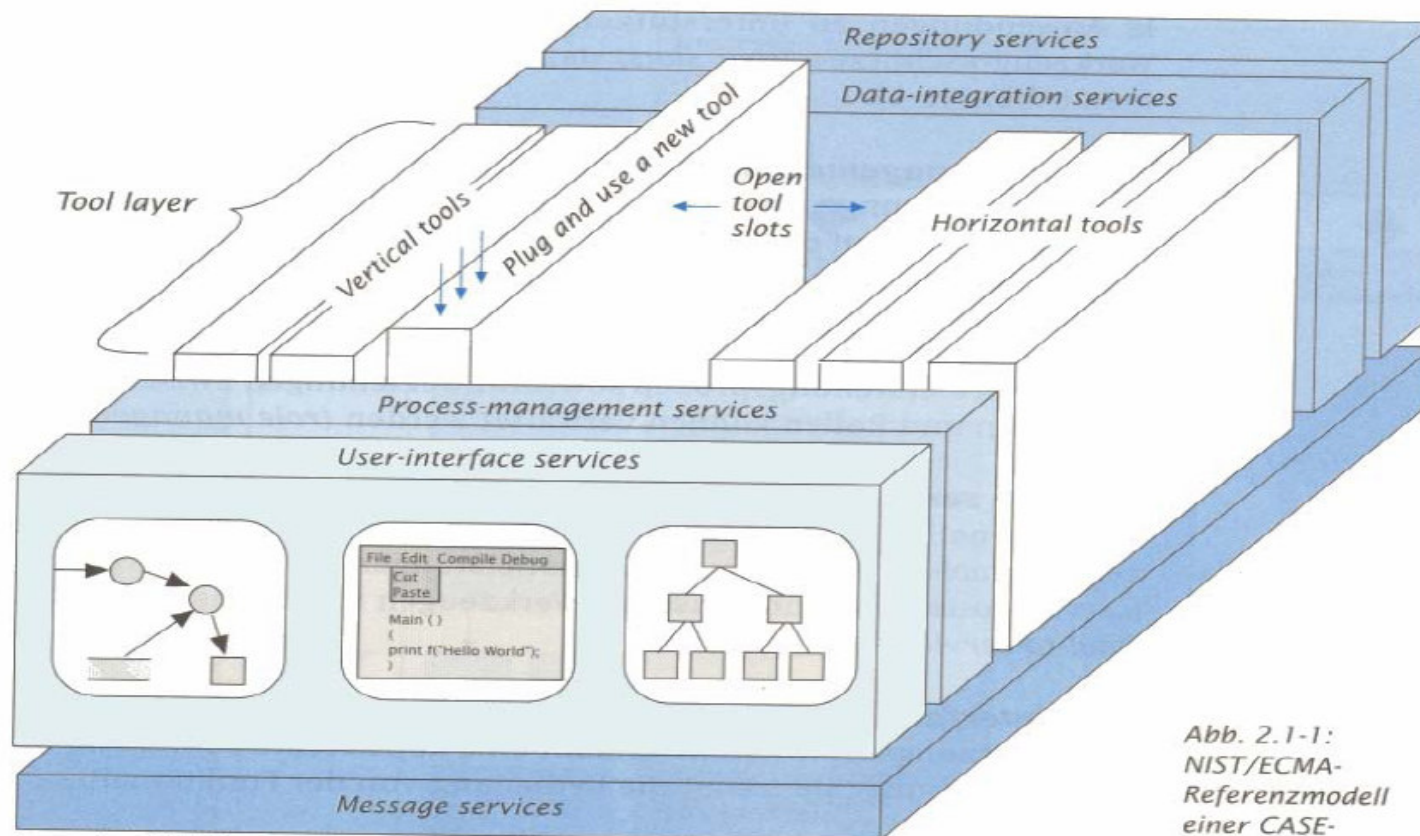
- Development time: CASE tools
 - support software development and project management
 - such tools exist only on the developers' machines
- Run time: "Libraries, toolkits, frameworks, etc."
 - enhance functionality by pre-developed and re-used software
 - such tools (software artifacts) are used at runtime

CASE Tools

- CASE: Computer Aided Software Engineering
- Usually use suite of tools (tool box) to support all aspects of the software development process, e.g.
 - analysis and design diagrams
 - source code creation
 - repository for data management
- It is important that the tools work together!

CASE tools are computerized applications supporting and partially automating software-production activities. [Fugetta]

ECMA Reference model



Legende: Die blauen Teile gehören zur CASE-Plattform
Vertical tools: Werkzeuge, die den gesamten Lebenszyklus begleiten, wie das Konfigurationsmanagement
Horizontal tools: Phasenorientierte Werkzeuge, z.B. SA-Werkzeug

Abb. 2.1-1:
 NIST/ECMA-Referenzmodell einer CASE-Umgebung
 (»Toaster«-Modell)
 /Chen, Norman
 92, S. 19/

Goal of using CASE tools

- ❑ Enhance productivity
- ❑ Increase software quality
- ❑ Simplify project management

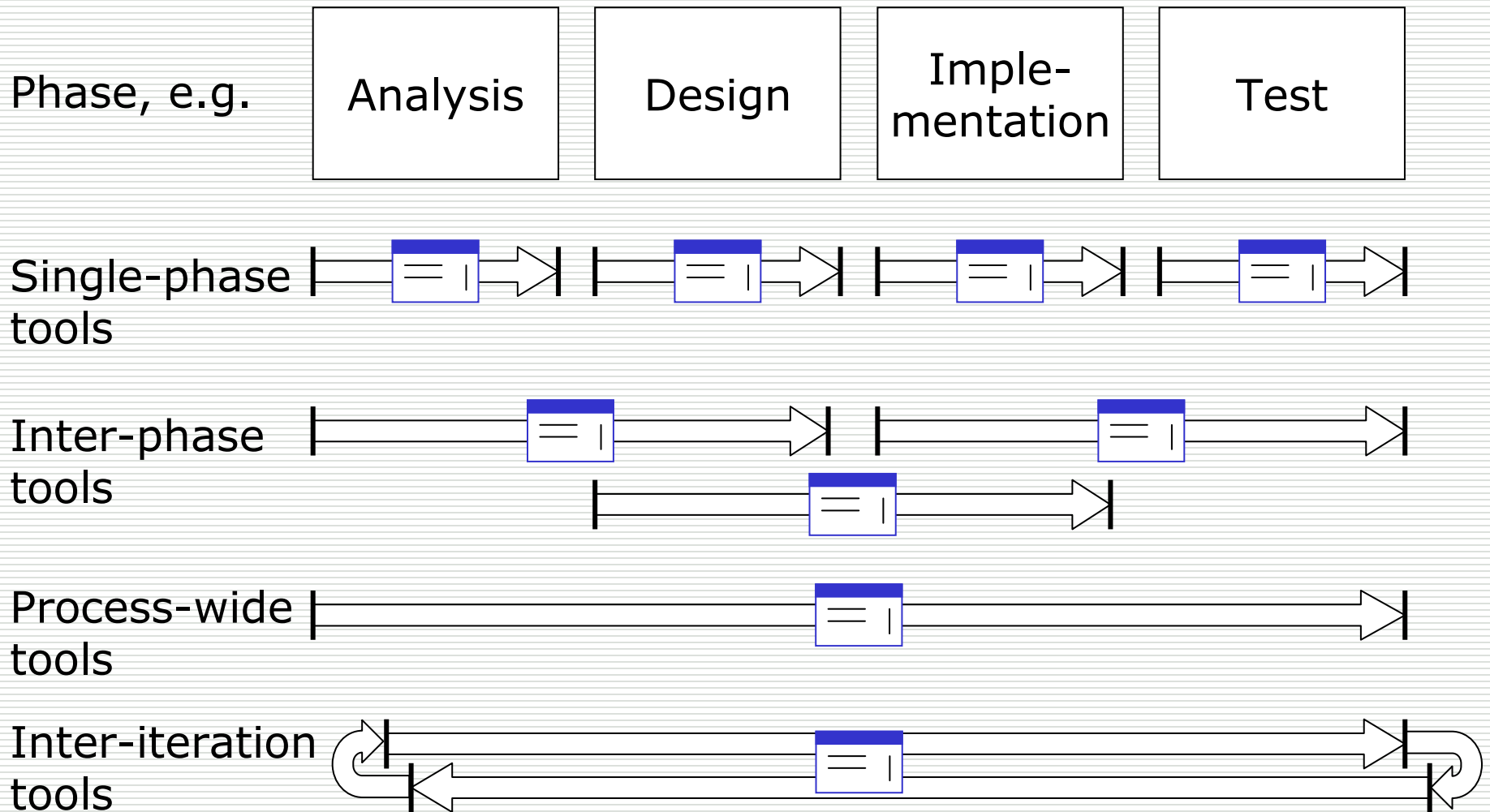
Tools should do the routine work for the developers.

Corollary: Good developers write good software faster with CASE tools. Bad developers just write more bad software in the same time. CASE tools do not automatically create good software!

General Requirements to CASE Tools

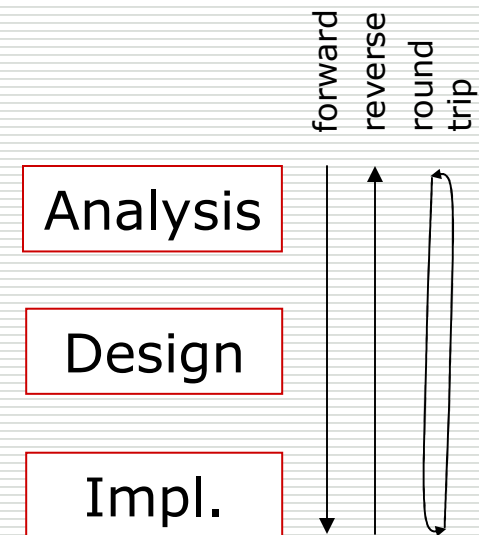
- Support of the software development process and methodology
 - e.g. be able to create/model classes
- Supply basic functionality, do routine tasks automatically
 - e.g. be able to support editing of code in a particular (programming) language, supply refactoring methods
- Features to enhance efficiency
 - e.g. generate parts of models and code automatically
- Features to enhance quality
 - e.g. support of design patterns
- Intuitive use
- Integration with other tools
 - e.g. code editor works with code repository

Scope of Tool Support

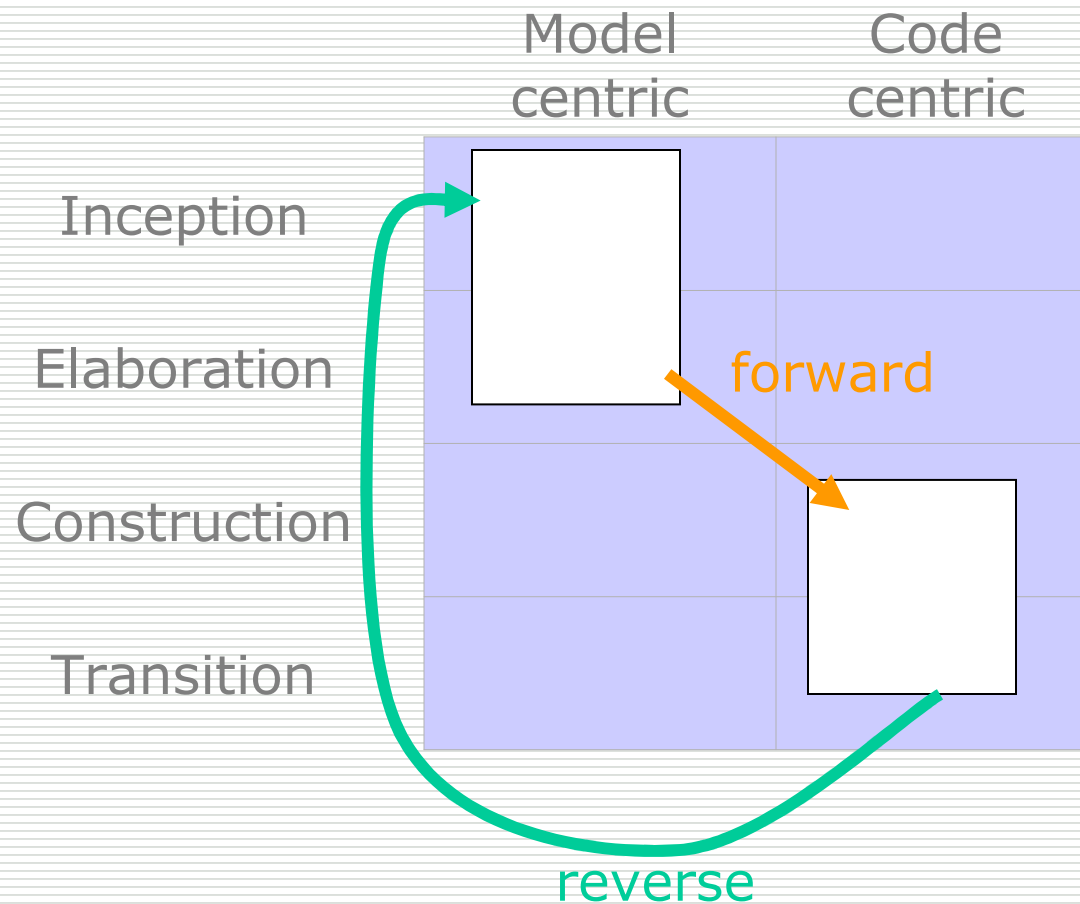


Software Development Methodologies

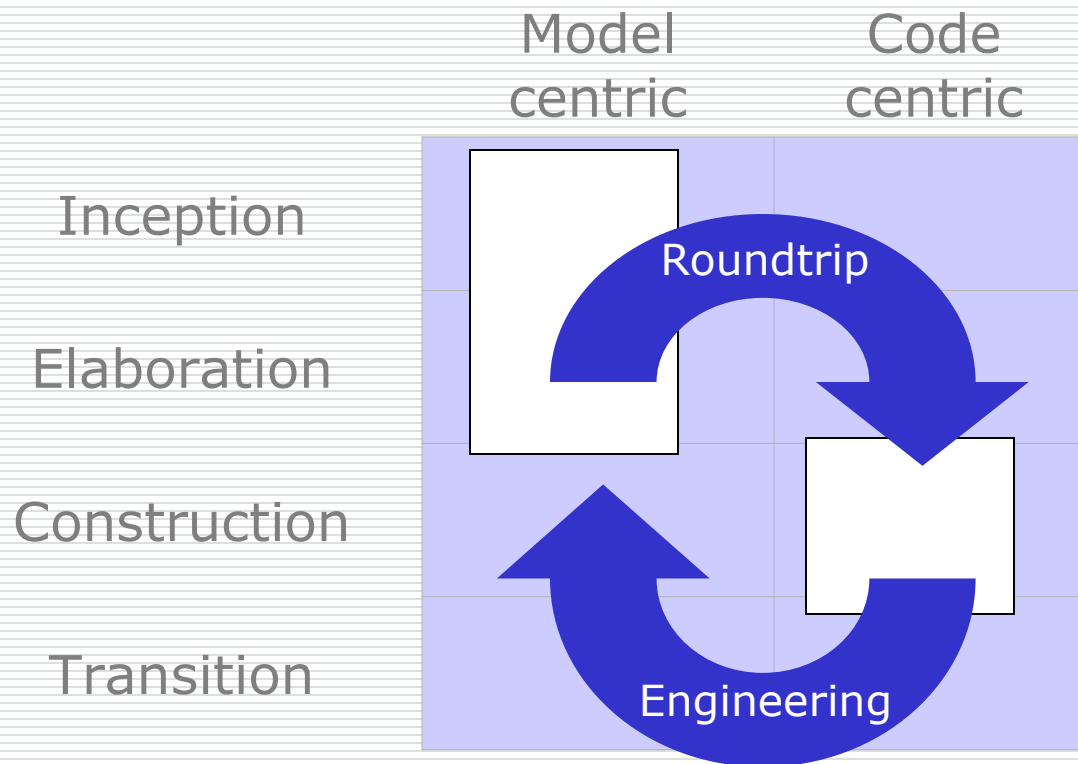
- Forward engineering
 - analysis-design-implementation
- Reverse engineering
 - given the implementation create the specifications
- Roundtrip engineering
 - start anywhere, end anywhere (or nowhere, never)
 - usually a combination of forward and reverse engineering
- Re-engineering
 - Restructure and rebuild (partially) an existing system to fit new requirements



Tools for Software Development



Tools for Software Development

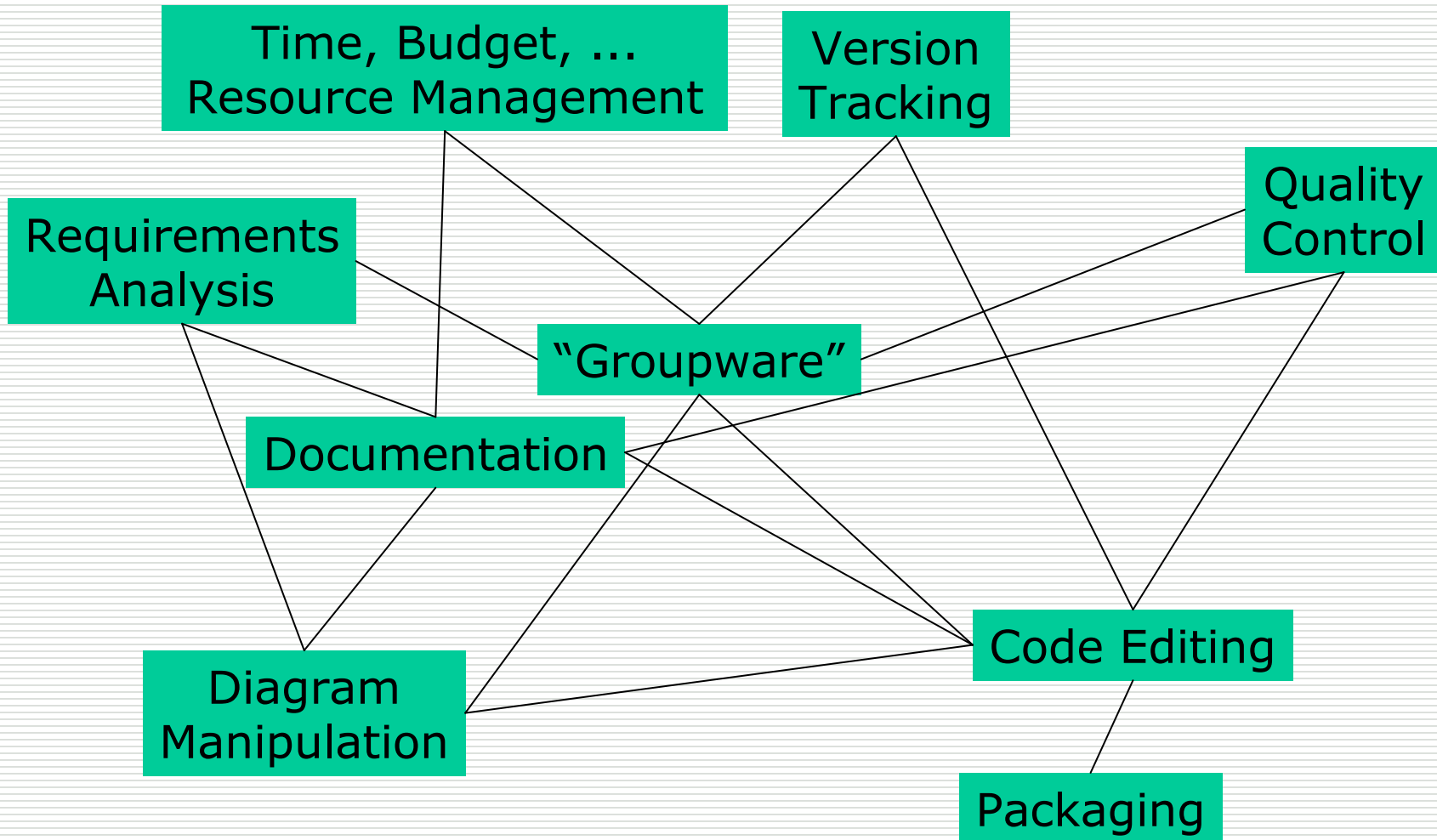


Roundtrip engineering requires special tool support!

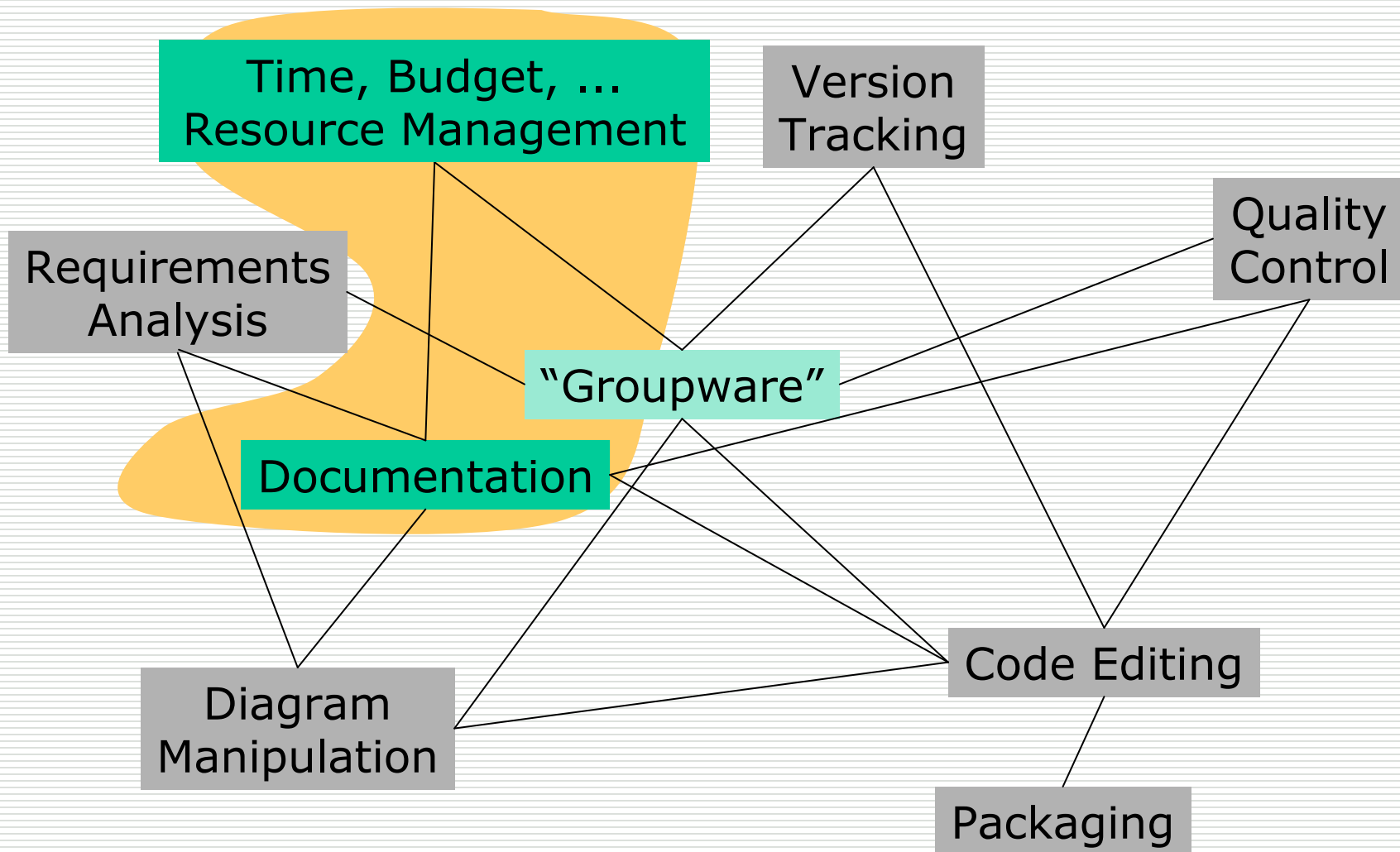
Impact of Methodology on Tool Selection

- Dependencies between the used software development process and tools
 - Different views on each phase through specialized tools?
 - Interoperability of tools for adjacent phases?
 - Tools used in the whole process?
 - Do you need roundtrips?
- Structure of the team
 - Fixed responsibilities?
 - Producer/consumer chains?
 - Everybody does everything?
(e.g. eXtreme Programming)

Software Development Activities



Project Management Tools



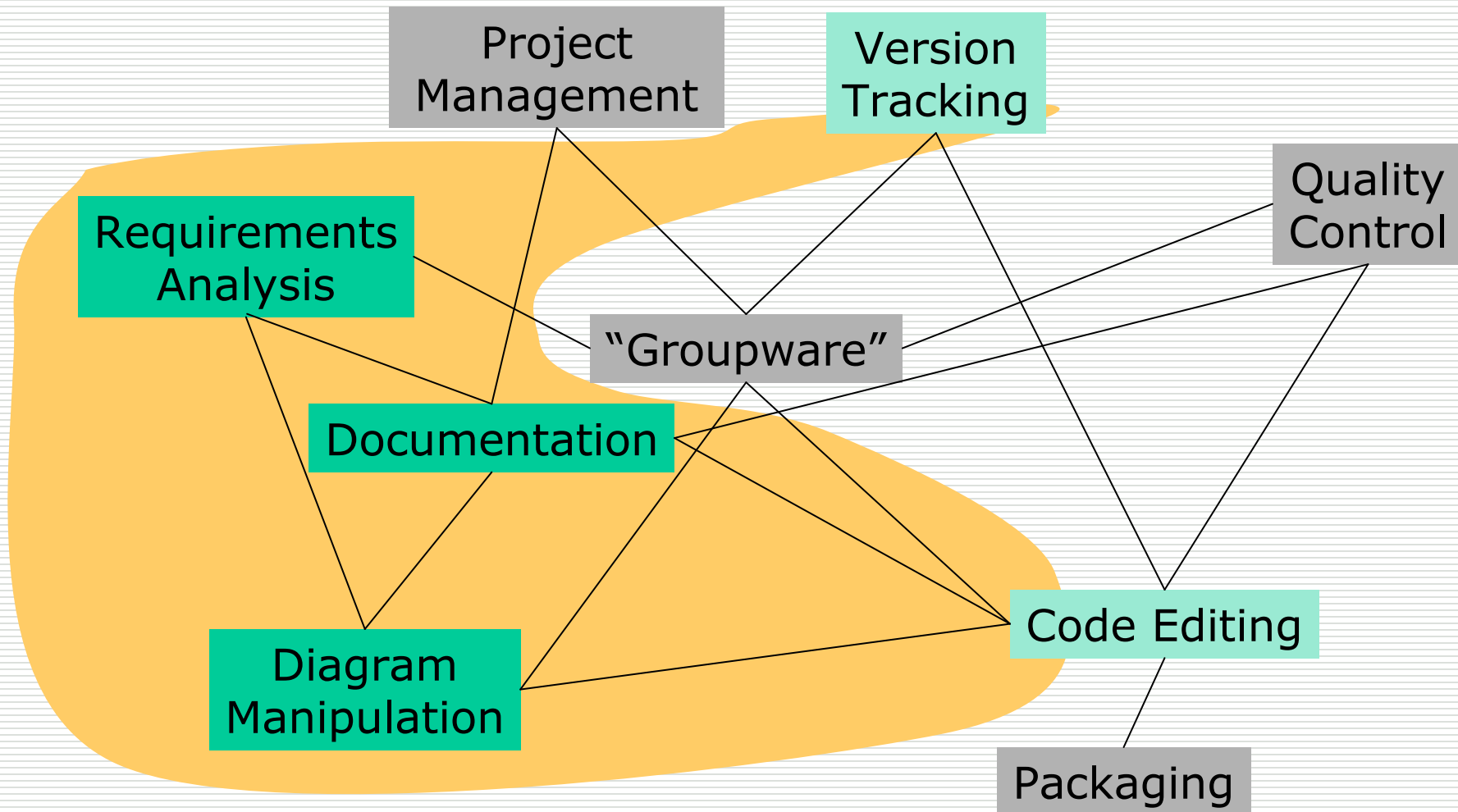
Project Management Tools (2)

- Tasks and their dependencies
 - scheduling
 - milestone definition
 - critical path analysis
- Resource planning
 - time
 - people
 - money
 - machinery and software
- Monitoring
 - timely reaching of milestones
 - controlling cost

Documentation Tools

- Word processors
 - write textual descriptions of system model
 - manage glossaries of used terms
- Graphics tools
 - create screenshots, mockups, ... for communication with future users
 - often, model centric tools are used here
- Report generators
 - extract documentation from source code

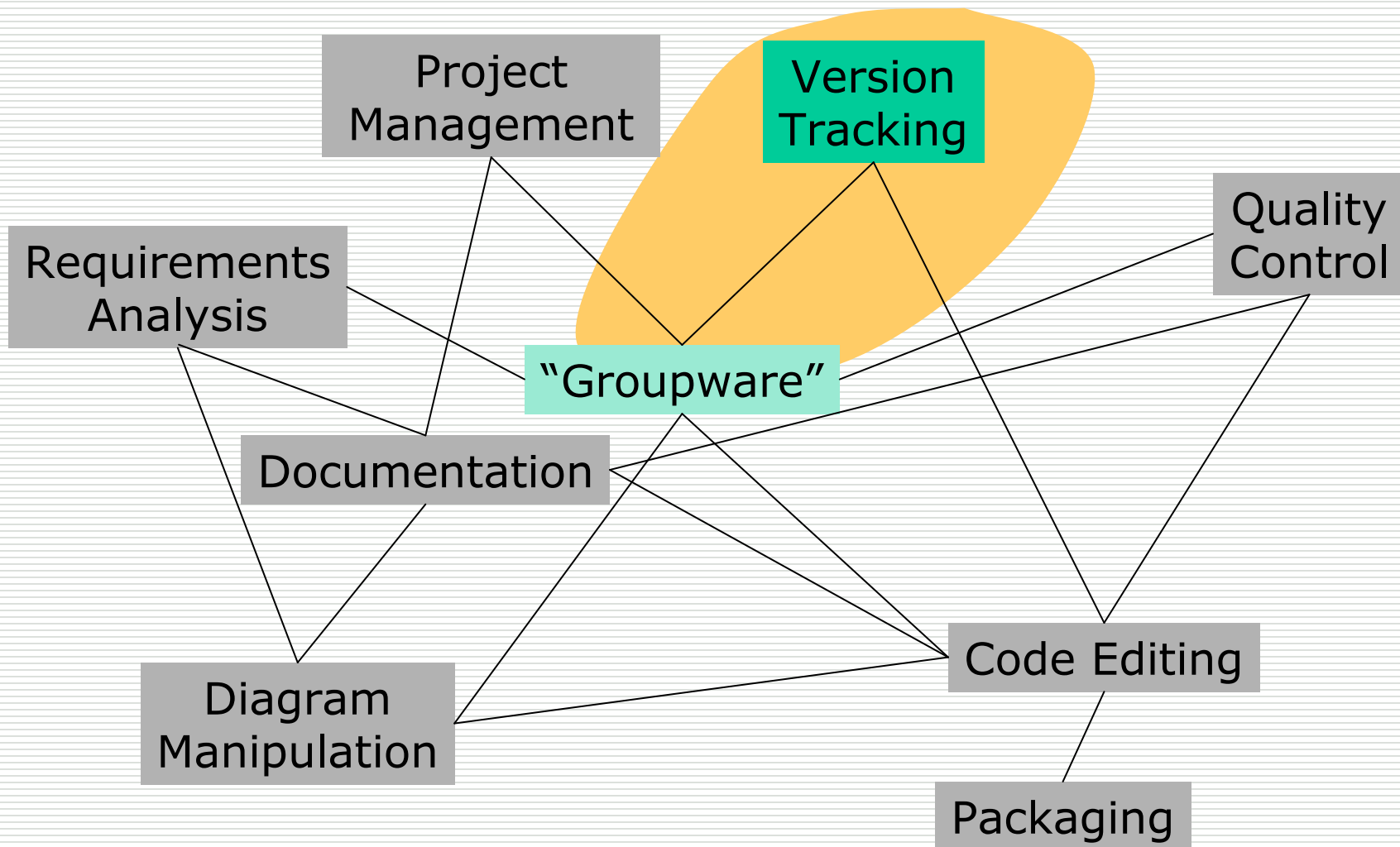
Model Centric Tool



Model Centric Tool

- Based on a visual language; most often UML
 - create and develop diagrams
 - diagrams visualize results of different phases
- Model verification
 - offer limited support for model checking
 - usually weak semantics
- Transition to implementation phase
 - generate partial code (signatures, interfaces, ...) based on modeling
 - often, additional information is added (target language, constructs, ...) in the form of stereotypes
 - sometimes, for roundtrip engineering, model centric tools create diagrams based on given code

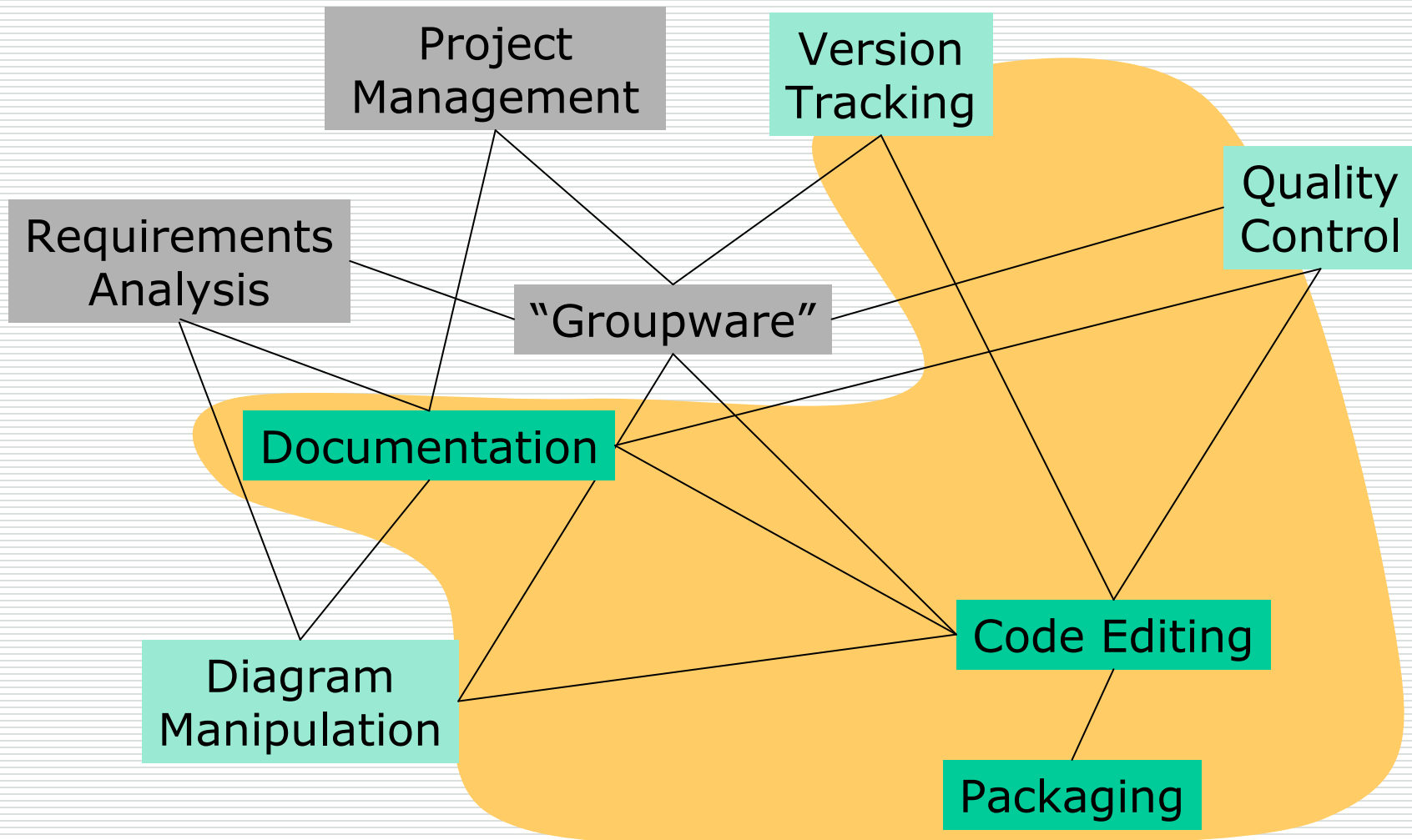
Repository System



Repository System

- Team development
 - distribution of code, documentation, etc.
 - access control
 - concurrency:
conflict management, e.g., differencing, merging,
conflict resolution, change propagation
- Integrity of artifacts
 - consistency of code, documentation, ...
- Management of variants
 - revision management
 - system versions, e.g., development branches for
delivery and development versions

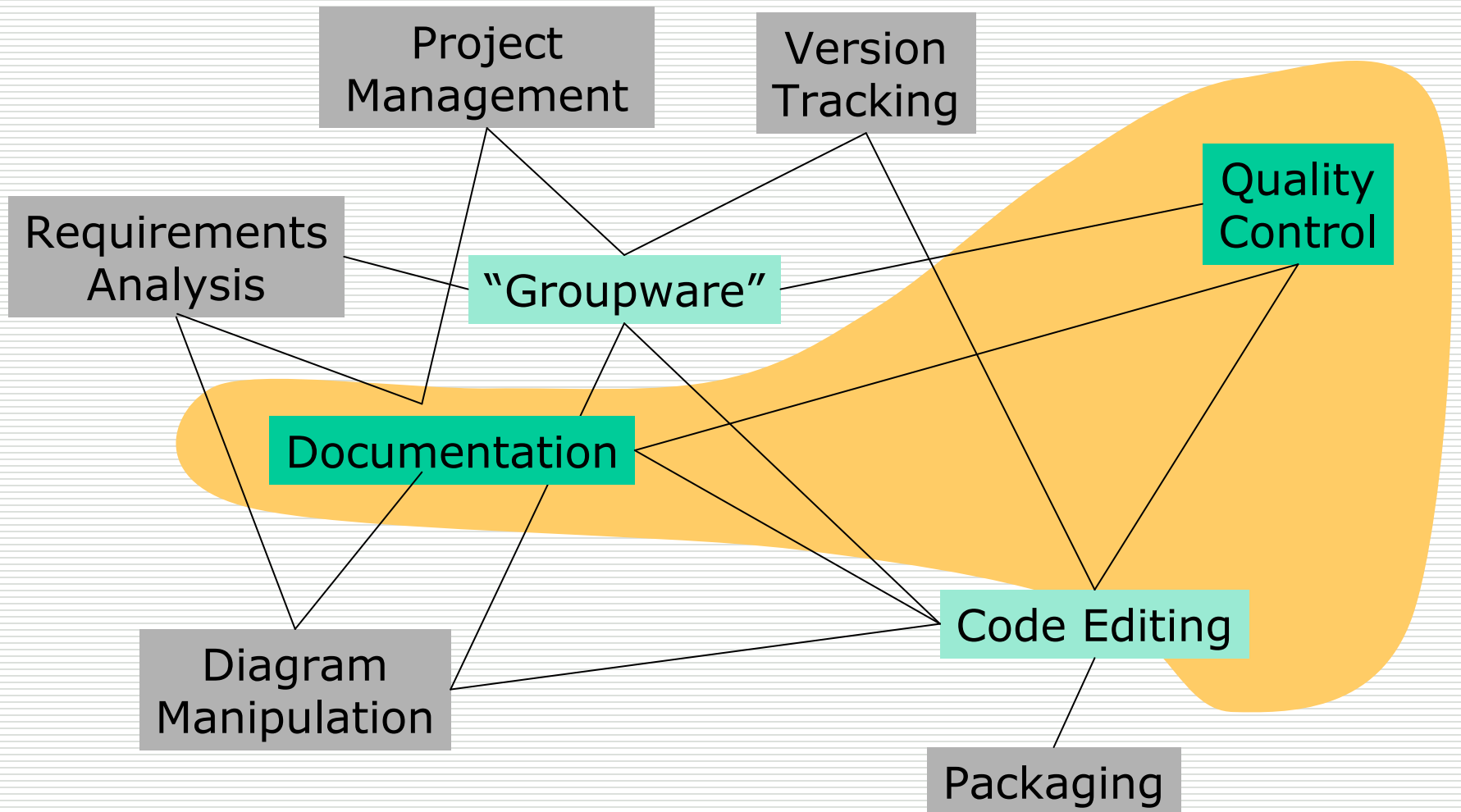
Code Centric Tool: Integrated Development Environment (IDE)



Code Centric Tool

- Code creation and modification
 - interactive editing
 - manual editing (code aware: syntax highlighting etc.)
 - code amendments (code completion etc.)
 - refactoring (renaming, shifting methods etc.)
 - limited semantic checks (before actual compiler run)
 - code generators, e.g., from DDLs, IDLs, MDA
- Packaging
 - bundling related classes for distribution
- General support
 - code management (consistency checks)
 - repository integration
 - tool configuration (compiler paths, external libraries, ...)

Test Tools



Test Tools

- Code level tests
 - automatic testing:
 - test cases defined by programmer, e.g., by assertions, pre-, post-, and side conditions, ...
 - automatic tests for technical aspects; mainly for functional requirements
 - manual tools that are used by developers, e.g., debuggers
- System level tests
 - derived from use cases
 - test system functionality
 - test functional as well as non-functional requirements

Test Tools (2)

- Bugtracker
 - records issues reported by developers or users
 - keeps track of state of the issue (assigned support team member, classification, documentation, resolution)
- Quality Control
 - (automatic) integration tests before delivery
 - of whole system
 - of system plus bugfixes
 - manual tests (“beta tests”)
 - compare ISO 900x standards on quality assurance

Phase Matrix of Development Tools

	A	D	I	T
Wordprocessor	x	x		
Model centric	x	x	x	
IDEs			x	x
Build tools			x	x
(Unit-)testing tools			x	x
Version control	(x)	x	x	x
Issue tracking		(x)	x	x
Groupware	x	x	x	x

tools

forward process

Role Matrix of Development Tools

	Analyst	Designer	Programmer	QA Engineer	Support	Project Mgr	Deployer	Admin	User
<i>r = read access</i> <i>w = write access</i>									
Wordprocessor	w	w				w			
Project Management Tool	r	r	r	r		w	r	r	
Model centric	w	w	r		(r)				
IDEs			w		w				
Build tools			w		w				
(Unit-)testing tools			w	r	r				
Version control	w	w	w	w	w				
Issue tracking		(r)		w	r	w		(w)	w
Groupware	w	w	w	w	w	w	w	w	

Runtime Tools

- Generic, reusable software artifacts to be used for / included in a software product, e.g.,
 - Libraries
 - Frameworks
 - Components

There will be an extra lecture on runtime tools.

Toolkits

- A toolkit is a set of related and reusable classes designed to provide useful, general-purpose functionality. They are the object-oriented equivalent of subroutine libraries.
 - usage is usually determined by API
 - control-flow has to be managed by the programmer
 - toolkits emphasize code reuse.
 - offer isolated functionality only, for example,
 - persistence
 - mathematical functions
- Examples:
 - C++ I/O stream library
 - Java collection classes

Frameworks

- A framework is a set of cooperating classes that make up a reusable design for a specific class of software. They ...
 - offer *prefabricated components* as building blocks combined by design patterns,
 - capture architectural and implementation artifacts that are invariant and
 - defer the variant parts to application-specific logic,
 - manage the control-flow.
- Example of frameworks:
 - HotDraw / JHotDraw (Drawing Framework in Smalltalk / Java)
 - Java Swing GUI framework

Components

- The idea behind software components is to
 - *reuse* (pre-fabricated, matured) software (used in many products)
 - stop the NIH (Not Invented Here) syndrome (buy vs. make)
 - add structure to design and implementation (design for reuse)
- The driving force behind component architectures:
 - programming as an engineering discipline; (analysis and design have already been understood as important tasks for building successful software systems.)

Tool-o-mat



- Wordprocessors
- Model centric: [Together](#), [Rational Rose](#), [Poseidon UML](#)
- IDEs: [Eclipse](#), [JBuilder](#), [IntelliJ](#), Netbeans
- Build tools: [Ant](#), [make](#), [jam](#)
- Testing: [JUnit](#), [Jakarta Cactus](#), [HttpUnit](#), [jWebUnit](#)
- Profiling: [JMeter](#), [Extensible Java Profiler](#)
- Repositories: [CVS](#), [Perforce](#), [subversion](#), [Source Safe](#)
- Bugtracking: [bugzilla](#), [sourceforge](#)
- Groupware: [Email](#), [News](#), [Lotus Notes](#), [sourceforge](#)
- Tools for application and [SW knowledge management](#)
- Tools for (computer-based-) training: [Ecesis plugins](#) (for Eclipse)

[available at/through STS](#); [freely available](#)

References

- Random choice of software:
 - Eclipse (www.eclipse.org)
 - Together (www.borland.com/together)
 - Poseidon (www.gentleware.de)
 - CVS (www.cvshome.org)
 - Perforce (www.perforce.com)
- Books:
 - H. Balzert. **Lehrbuch der Software-Technik**. Springer Verlag
 - Ghezzi et al. **Fundamentals of Software Engineering**. Prentise Hall International
 - E. Gamma et al. **Design Patterns, Elements of Object-Oriented Software**. Addison-Wesley 1995
- Articles:
 - A. Fugetta. **A Classification of CASE Technology**. IEEE Computer 26(12):25-3, Dec 1993