CSE3019: Software Quality and Reliability

L13: Details of CASE Tools and Their Effect on Software Quality

Dr. Subrata Tikadar
SCOPE, VIT-AP University

Recap

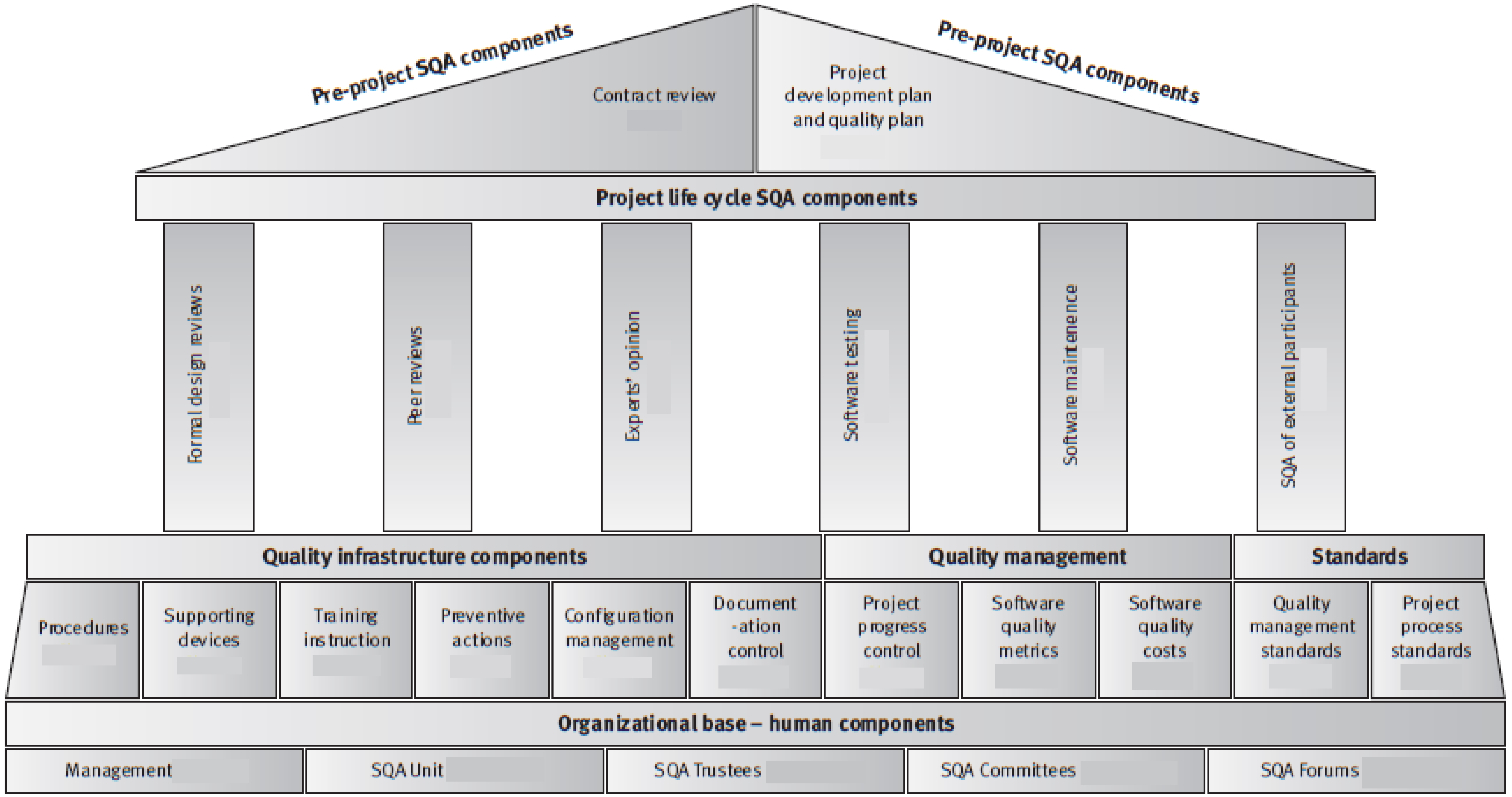


Image Source: Daniel Galin, “Software Quality Assurance: From Theory to Implementation”, Pearson Education, 2004.

Outline

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

Outline

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

What is a CASE tool?

CASE tools - Definition

Computer-Aided Software Engineering (CASE) tools are computerized software development tools that support the developer when performing one or more phases of the software life cycle and/or support software maintenance.

- ***Classic* CASE tools** – well-established computerized software development support tools – example: interactive debuggers, compilers, and project progress control systems.
- ***Real* CASE tools** – new tools that support the developer for a succession of several development phases of project development – example: software package for automatic code generation, automatic repository generation, etc..
  - Upper CASE tools – support analysis and design phase
  - Lower CASE tools – support coding phase
  - Integrated CASE tools – combination of upper and lower case tools – support all the phases

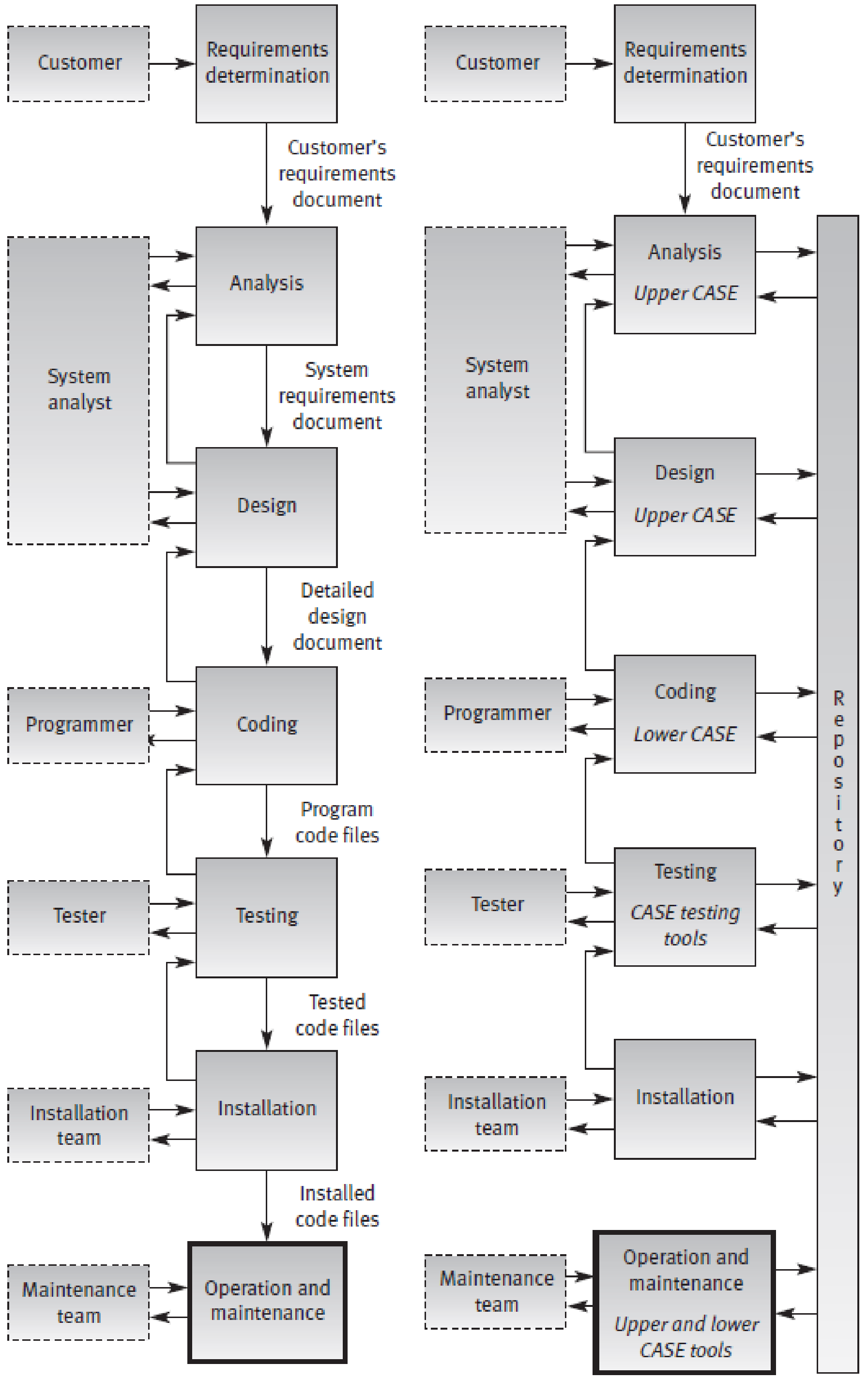
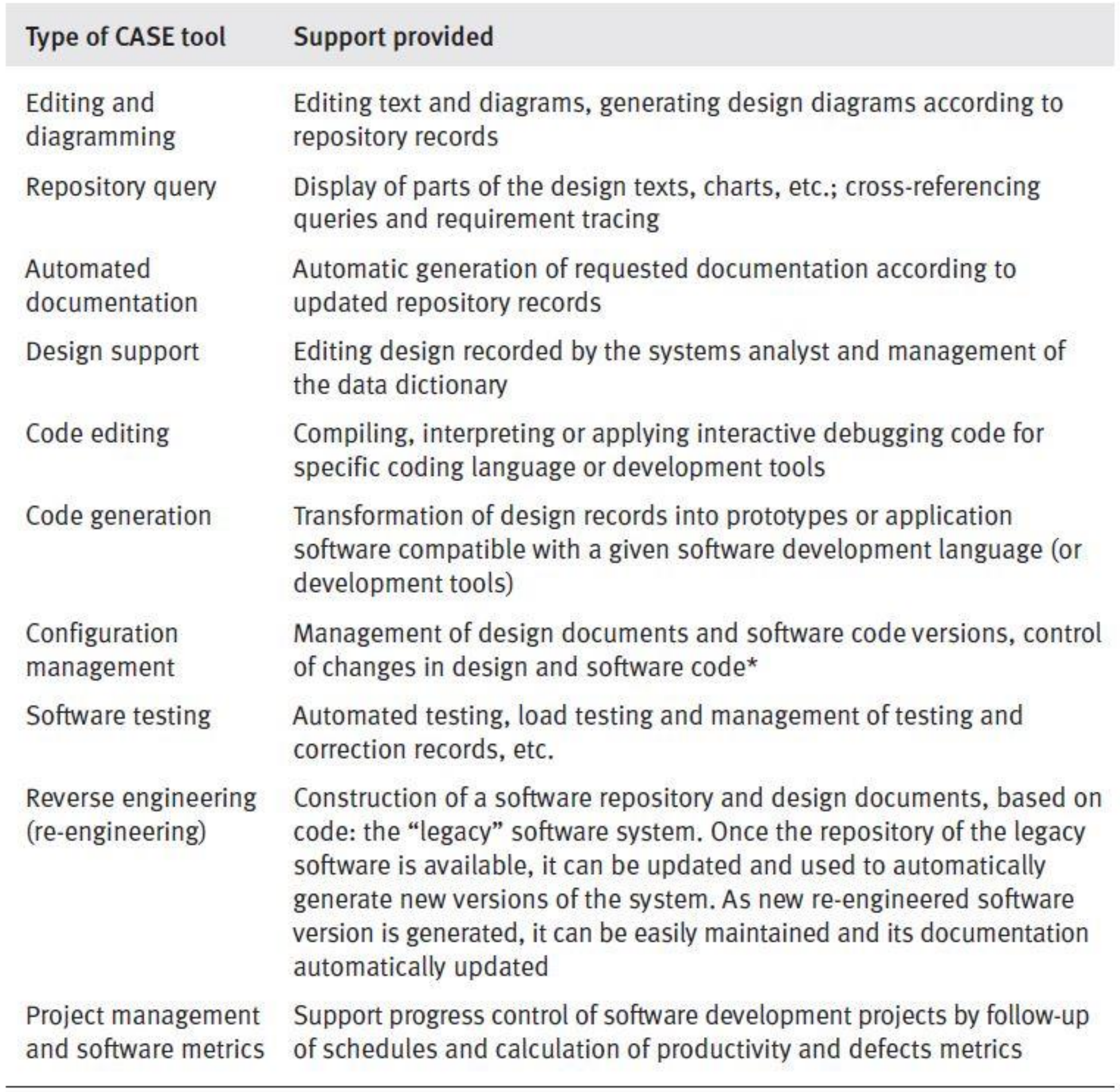


Image Source: Daniel Galin, "Software Quality Assurance: From Theory to Implementation", Pearson Education, 2004.



Type of CASE tool	Support provided
Editing and diagramming	Editing text and diagrams, generating design diagrams according to repository records
Repository query	Display of parts of the design texts, charts, etc.; cross-referencing queries and requirement tracing
Automated documentation	Automatic generation of requested documentation according to updated repository records
Design support	Editing design recorded by the systems analyst and management of the data dictionary
Code editing	Compiling, interpreting or applying interactive debugging code for specific coding language or development tools
Code generation	Transformation of design records into prototypes or application software compatible with a given software development language (or development tools)
Configuration management	Management of design documents and software code versions, control of changes in design and software code*
Software testing	Automated testing, load testing and management of testing and correction records, etc.
Reverse engineering (re-engineering)	Construction of a software repository and design documents, based on code: the "legacy" software system. Once the repository of the legacy software is available, it can be updated and used to automatically generate new versions of the system. As new re-engineered software version is generated, it can be easily maintained and its documentation automatically updated
Project management and software metrics	Support progress control of software development projects by follow-up of schedules and calculation of productivity and defects metrics

Source: Daniel Galin, "Software Quality Assurance: From Theory to Implementation", Pearson Education, 2004.

Outline

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

The contribution of CASE tools to software product quality

Cause of Software Error	Extent and manner of contribution to quality	
	Classic CASE tools	**Real CASE tools**
1. Faulty requirements definition		**Almost no contribution** Computerized examination of requirements consistency or correctness is rarely possible.
2. Client-developer communication failure		**Almost no contribution** In most cases, computerized identification of communication failures is impossible. Communication failures can be located or prevented only when a change or other information is found to be inconsistent with repository information.
3. Deliberate deviations from software requirements		**High Contribution** Based on information stored in the repository, deviations from recorded requirements are identified as inconsistent and labeled as errors. Such deviations can also be identified by repository-based requirements tracing tools and cross-referenced query tools.

The contribution of CASE tools to software product quality

Cause of Software Error	Extent and manner of contribution to quality	
	Classic CASE tools	**Real CASE tools**
4. Logical design errors		**High contribution** 1. Re-engineering enables automated generation of the design of legacy systems and their recording in a repository. 2. Use of the repository is expected to identify design omissions, changes and additions inconsistent with repository records.
5. Coding errors	**Very high contribution** Application of compilers, interpreters and interactive debuggers.	**Very high contribution** Application of lower CASE tools for automated code generation achieves full consistency with the design recorded in the repository. In addition, as coding is automatic, no coding errors are expected .
6. Non-compliance with coding and documentation instruction	**Limited contribution** Use of text editors and code auditing supports the standardization of structure and style of texts and code and facilitates identification of non-compliances	**Very high Contribution** Application of lower CASE tools for automated code generation assures full compliance with documentation and coding instructions.

The contribution of CASE tools to software product quality

Cause of Software Error	Extent and manner of contribution to quality	
	Classic CASE tools	**Real CASE tools**
7. Shortcoming in the testing process	**High contribution** Automated testing tools perform full regression and automated load testing. Computerized management of testing and correction reduces error by improvement follow-up.	**High contribution** Application of lower CASE but especially of integrated CASE tools prevent coding errors and reduces design errors. Application of repository tool (cross-referenced queries and performance consistency checks) to corrections and changes during the development process prevent most software errors.
8. Procedural errors	**High contribution** Control of versions, revisions, and software installation by means of software configuration tools.	**Limited contribution** Use of updated and full documentation is expected to prevent many of the maintenance errors cause by incomplete and/or inaccurate documentation, especially if the design has been revised several times.
9. Documentation errors	**Limited contribution** Application of text editors only.	**High Contribution** Use of repository automatically generates full and updated documentation prior to each correction or change.

Outline

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

The contribution of CASE tools to software maintenance quality

- Corrective maintenance
- Adaptive maintenance
- Functional improvement maintenance

The contribution of CASE tools to software maintenance quality

- Corrective maintenance
  - CASE-generated full and updated documentation of the software enables easier and more reliable identification of the cause for software failure.
  - Cross-referenced queries enable better identification of anticipated effects of any proposed correction.
  - Correction by means of lower CASE or integrated CASE tools provides automated coding, with no expected coding errors as well as automated documentation of corrections.

The contribution of CASE tools to software maintenance quality

- Adaptive maintenance
  - Full and updated documentation of the software by CASE tools enables thorough examination of possible software package adaptations for new users and applications

The contribution of CASE tools to software maintenance quality

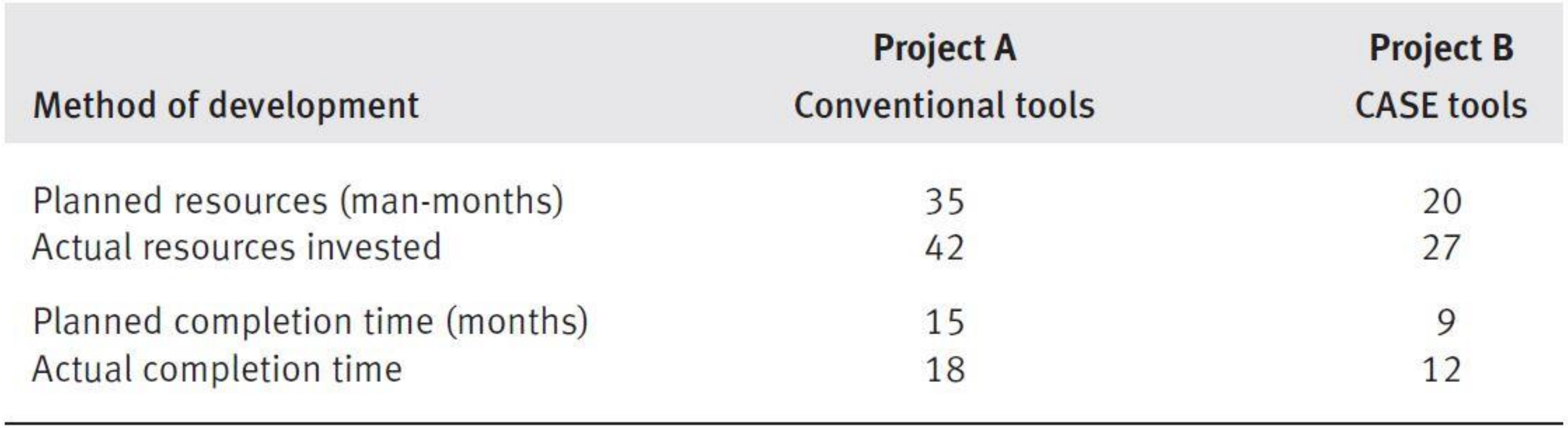
- Functional improvement maintenance
  - Use of the repository enables designers to assure consistency of new applications and improvements with existing software systems.
  - Cross-referenced repository queries enable better planning of changes and additions.
  - Changes and additions carried out by means of lower CASE or integrated CASE tools enable automated coding, with no expected coding errors as well as automated documentation of the changes and additions

Outline

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

The contribution of CASE tools to improvec project management

- Let us analyze this for the following case study



Method of development	Project A Conventional tools	Project B CASE tools
Planned resources (man-months)	35	20
Actual resources invested	42	27
Planned completion time (months)	15	9
Actual completion time	18	12

1. *The advanced CASE method was much more economical than the conventional method.*
2. *The quality of management in both projects was similar, with resources and schedule estimated at below the required levels.*

Summary

- What is a CASE tool?
- The contribution of CASE tools to software product quality
- The contribution of CASE tools to software maintenance quality
- The contribution of CASE tools to improved project management

Reference

- Daniel Galin, “Software Quality Assurance: From Theory to Implementation”, Pearson Education, 2004.
  - Ch-13: Sec 13.1-13.4
  - Self reading: Ch-12 & Ch-13

Next

- Software Quality Infrastructure Components

Thank You