Chapter 25

Chapter 25 Computer Aided Software Engineering

Learning Objective

... to discuss general issues relating to CASE and CASE technology

Frederick T Sheldon
Assistant Professor of Computer Science
Washington State University

Computer-aided software engineering

Software tool support for software development

Objectives

To discuss general issues relating to CASE and CASE technology

To suggest a classification for CASE systems

To discuss CASE tool integration

To describe the CASE life cycle

Topics covered

CASE classification

Integrated CASE

The CASE life cycle

CASE technology

Production-process support technology

- Tools to support development activities such as specification, design, implementation, etc.

Process management technology

- Tools to support process modeling and management

Meta-CASE technology

- Generators used to produce CASE toolsets

Impact of CASE technology

CASE technology has resulted in significant improvements in quality and productivity

However, the scale of these improvements is less than was initially predicted by early technology developers

- Many software development problems such as management problems are not amenable to automation
- CASE systems are not integrated
- Adopters of CASE technology underestimated the training and process adaptation costs

CASE classification

CASE systems can be classified according to their

- Functionality - what functions do they provide
- Process support - what software process activities do they support
- The breadth of support which they provide

Classification allows tools to be assessed and compared

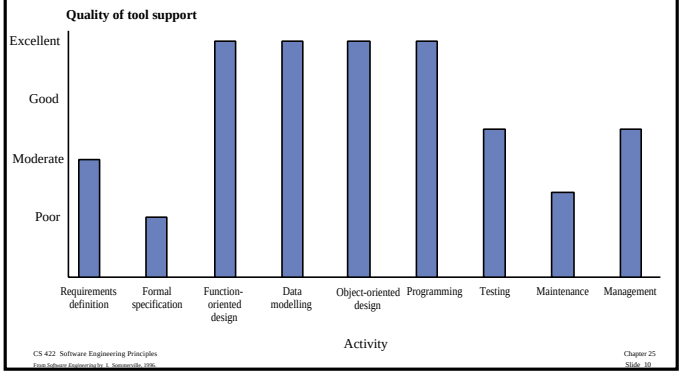
Functional tool classes

Tool type	Examples
Management tools	PERT tools, Estimation tools
Editing tools	Text editors, diagram editors, word processors
Configuration management tools	Version management systems, Change management systems
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

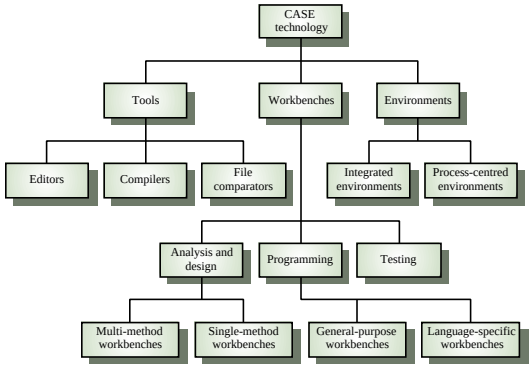
Activity-based tool classification

	Specification	Design	Implementation	Verification and Validation
Test data generation tools				●
Modelling and simulation tools	●			●
Program transformation tools			●	
Interactive debugging systems			●	●
Program analysis tools			●	●
Language-processing tools			●	●
Method support tools	●	●		
User interface management systems		●	●	
Data dictionary tools	●	●		
Diagram editing tools	●	●		
Prototyping tools	●			●
Configuration management tools	●	●	●	●
Document preparation tools	●	●	●	●
Text editing tools	●	●	●	●
Planning and estimation tools	●	●	●	●

Quality of CASE support



Tools, workbenches, environments



Integrated CASE

While individual CASE tools are useful, more leverage is obtained if tools can work together

Specialized tools can be combined to provide wider support for process activities

- Integration of design workbench with a documentation workbench
- Integration of specification, design and programming tools with a configuration management workbench

Levels of integration

Platform integration
- Tools run on the same hardware/software platform

Data integration
- Tools operate using a shared data model

Presentation integration
- Tools offer a common user interface

Control integration
- Tools activate and control the operation of other tools

Process integration
- Tool use is guided by a defined process model

Platform integration

Tools and workbenches run on the same hardware/software platform

UNIX or PC running MS Windows are the most commonly used CASE platforms

Major problems are heterogeneous networks
- Different types of machine on the network
- Different operating systems installed on different machines

Lack of OS standards is a problem

Data integration

Shared files
- Tools communicate through a shared file format

Shared data structures
- Tools communicate through some internal representation of a shared notation

Shared repository
- Tools are integrated around an OMS which includes a public schema describing data entities and relationships

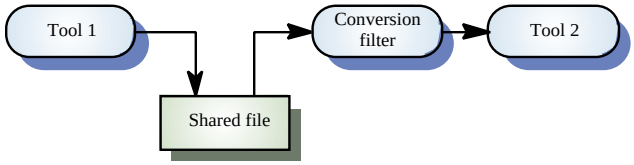
Shared files

Simple and straightforward approach to integration

Most common form of data integration

Requires tools to share a file format or to include translations from one file format to another

Point-to-point tool integration



Shared data integration

Tools are tightly integrated around a shared data structure. All tools are aware of the organization of this structure

Hides the differences between individual tools - user is presented with a seamlessly integrated toolset

However, very difficult to add new tools or extend the system in any way

Language-oriented toolset

Compiler for language translation

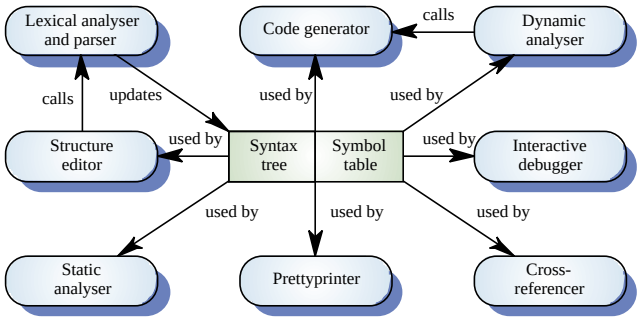
Static and dynamic program analyzers

Structure editing system where the program editor includes knowledge of the program syntax

Prettyprinters and cross-references

All share a common data structure

Integration through shared data



Repository integration

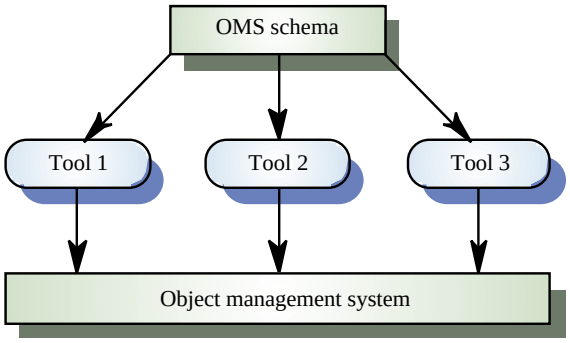
Flexible form of data integration

Tools access data through an object management system whose schema is public. Tools read and write data according to this schema

Disadvantages are:

- Tools have to be specially written for a specific repository to make use of the schema
- Customer must buy the OMS as well as the CASE system

Integration through an OMS



Presentation integration

Window system integration

- Tools use the same underlying window system and present a common interface for window manipulation commands

Command integration

- Tools use the same form of commands for comparable functions. The menus are organized in the same way and similar icons are used

Interaction integration

- The user interacts with graphical entities in the same way. The same direct manipulation operations are used

Presentation guidelines

Presentation integration can be achieved by defining a set of user interface guidelines which all application developers follow

- Easy for window system integration
- Relatively straightforward for command integration. Both the Macintosh and MS Windows have user interface designer's guidelines
- More difficult for interaction integration because of the range of interaction possibilities

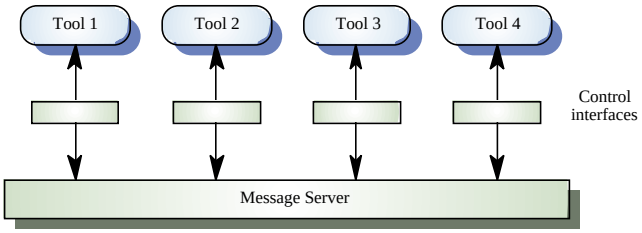
Control integration

Concerned with providing mechanisms for one tool to control the activation of other tools

Tools should be able to start and stop other tools and request particular services provided by other tools

General approach based on message passing has been adopted by a number of tool vendors (Softbench, FUSE, ToolTalk)

Integration through message passing



Tool communication

Tools exchange messages in a known format

Message passing is controlled by a message server

The message server accepts messages from a tool, recognizes the destination and forwards it to that tool (or tools)

System works in a distributed environments

Format of data to be exchanged is encoded in an interface definition language (IDL)

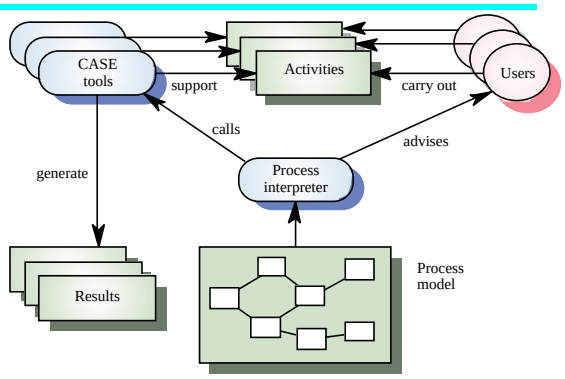
Process integration

The CASE system has embedded knowledge about process activities, their phasing, constraints and the tools used

An explicit model of the process must be defined which is enacted by a process engine

The process should be guided rather than prescribed by the process model

Process integration



Process model creation

Identify process activities.

Identify the deliverables or products of the process.

Define activity coordination and activity dependencies.

Allocate engineers to each activities.

Specify tool support for each activity.

Process models

Software processes are complex and difficult to model

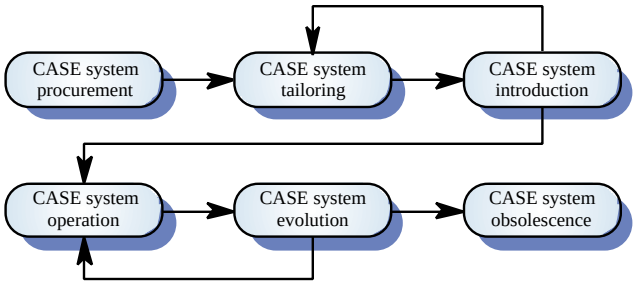
- There is a high process programming cost
- Software engineers dynamically change the process to cope with unexpected circumstances
- It is hard to specify cooperative working in current approaches to process modeling

Process-driven CASE systems are mostly still experimental systems

The CASE life cycle

Procurement
Tailoring
Introduction
Operation
Evolution
Obsolescence

A CASE life cycle model



CASE procurement

Existing company standards and methods

- The environment must support existing practice

Existing and future hardware

- The environment must be compatible with existing hardware. It should run on industry-standard machines

The class of application to be developed

- The environment should support the principal type of application developed by an organization

Security

- The environment should provide appropriate access control facilities

CASE system tailoring

Installation

- Set system dependent hardware and software parameters

Process model definition

- Define the activities that the environment is to support

Tool integration

- Describe what tools are to be part of the environment and how they are to be integrated

Documentation

- Provide appropriate, in-house documentation for using the environment

CASE introduction and operation

May require changes to working practice

- User resistance because of conservatism or a feeling that environments are for managers rather than engineers
- Lack of training. Organizations often don't invest enough in training
- Management resistance. Managers may not see how the environment will reduce project costs

Migrate projects slowly to the CASE system

- New projects should start with the environment after initial pilot projects have demonstrated its advantages
- It is usually impractical to convert existing projects to the CASE system

CASE system evolution

As the system is used, new requirements arise

- Process requirements. Changes in the process model will be identified
- Tool requirements. New tools will become available and will have to be incorporated
- Data requirements. The data organization will evolve

An evolution budget must be available or the environment will become progressively less useful

Forward compatibility must be maintained

CASE system obsolescence

At some stage, an environment will outlive its usefulness and will have to be replaced

Replacing an environment must be planned and should take place over an extended time period

Currently supported projects must be moved to a new environment before their supporting environment is scrapped

Key points

CASE involves providing automated tool support for process activities

CASE technology may be classified by function, process activity supported or by the range of tasks supported

Tools support individual activities, workbenches support sets of related activities, environments support the whole process

There are several levels of CASE integration

Key points

- Data integration can be implemented through shared files, data structures or a repository
- Process integration means that development is guided by an explicit model of the software process
- The CASE life cycle involves procurement, tailoring, introduction, operation, evolution and obsolescence
- CASE is expensive. 5-year cost > $50, 000