CASE

CASE

- CASE stands for Computer Aided Software Engineering
- It is a Generic Terminology for tools which provide help in different phases of Software Engineering

Scope

- Upper CASE tools: support for the analysis and design
- Lower CASE tools: support for construction (building/coding) and maintenance

Specifically . . .

- CASE denotes any form of automated support for software engineering.
- CASE tool automates some (not all) of the software development activities.

Objectives of CASE

- To increase productivity
- To help produce better quality software at lower cost.
- Help standardization of notations and diagrams
- Help communication between development team members
- Automatically check the quality of the SAD models
- Reduction of time and effort
- Enhance reuse of models or models' components

CASE Environment >

- Although individual CASE tools are useful true power of a tool set can be realized only when all CASE tools are integrated together.
- When such an integrated system is available, it is called CASE environment

CASE Environment >

- In a CASE environment a set of CASE tools are integrated into a common framework or environment.
- If the different CASE tools are not integrated, then the data generated by one tool would have to input to the other tools.
  - Let's see...

CASE Environment

- So.
  - This may also involve format conversions as the tools developed by different vendors are likely to use different formats.
  - This results in additional effort of exporting data from one tool and importing to another. Also, many tools do not allow exporting data and maintain the data in proprietary formats

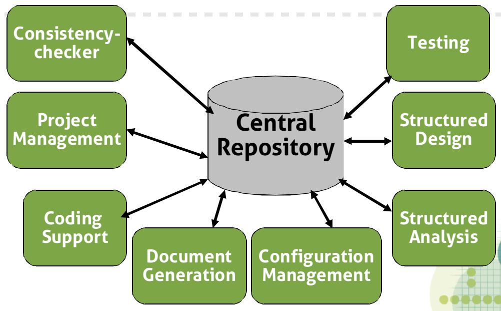
Programming Environment >

- A programming environment is an integrated collection of tools to support only the coding phase of software development.
- The tools commonly integrated in a programming environment are a text editor a compiler, and a debugger.
- >

Programming Environment

- The different tools are integrated to the extent that once the compiler detects an error, the editor takes automatically goes to the statements in error and the error statements are highlighted.
- Examples are Visual Studio, Netbeans, Eclipse, Kdevelop, Anjuta, Monodevelop,Turbo Explorer, Delphi ....so on....

Architecture of CASE environment



Benefits of CASE >

- cost saving through all developmental phases. Approx between 30% to 40%.
- improvements to quality mainly due to the chances of human error is considerably reduced.
- consistent documents Since the important data relating to a software product are maintained in a central repository, redundancy in in the stored data is reduced and therefore chances of inconsistent documentation is reduced to a great extent.

Benefits of CASE >

- CASE tools take out most of the drudgery in a software engi neers work. For example, they need not check meticulously the balancing of the DFDs but can do it effortlessly through the press of a button.

Benefits of CASE >

- cost saving in software maintenance efforts.
- traceability and consistency checks,
- systematic information capture during the various phases of software development as a result of adhering to a CASE environment
- Introduction of a CASE environment has an impact on the style of working of a company, and makes it oriented towards the structured and orderly approach.

Prototyping Support >

- Prototyping CASE tool:
  - often used in graphical user interface (GUI) development,
  - supports creating a GUI using a graphics editor.

Prototyping Support

- The user should be allowed to define:
- data entry forms, menus and controls.
- It should integrate with the data dictionary of a CASE environment.

Structured Analysis and Design

- A CASE tool should:
- support some standard structured analysis and design technique.
- support easy creation of analysis and design diagrams.
- should provide easy navigation through different levels of design and analysis diagrams.

Structured Analysis and Design

- The tool must support completeness and consistency checking.
- The tool should disallow inconsistent operations:
  - but, it is difficult to implement such a feature.

Code Generation >

- As far as code generation is concerned:
  - expectations from a CASE tool is low.
- The CASE tool should support:
  - generation of module skeletons in one or more popular languages.
  - Another reasonable requirement is traceability from source code to design.

Code Generation >

- It should automatically generate header information:
  - copyright messages,
  - brief description of the module,
  - author name and date of creation, etc.

Code Generation >

- The tool should generate data records or structures automatically:
  - using data dictionary definitions.
  - It should generate database tables for relational database management systems.

Code Generation

- The tool should generate code for user interface from the prototype:
  - for X window and MS window based applications.

Testing Support

- Static and dynamic program analysis of programs.
- It should generate test reports in ASCII format:
  - which can be directly imported into the test plan document.

Desirable Features

- The tool should work satisfactorily
  - when many users work simultaneously.
- The tool should support windowing interface:
  - Enable the users to see more than one diagram at a time.
  - Facilitate navigation and switching from one part to the other.

Documentation Support

- The deliverable documents:
  - should be able to incorporate text and diagrams from the central repository.
  - help in producing up-to-date documentation.

Project Management

- It should support collecting, storing, and analyzing information on the software project's progress:
  - such as the estimated task duration,
  - scheduled and actual task start, completion date, dates and results of the reviews, etc.

External Interface

- The tool should allow exchange of information for reusability of design.
  - The information exported by the tool should preferably be in ASCII format.
- The data dictionary should provide
  - a programming interface to access information.

Reverse Engineering Support

- The tool should support:
  - generating structure chart, DFD, and data dictionary from source code.
  - should populate the data dictionary from source code.

Data Dictionary Interface

- Data dictionary interface should provide
  - viewing and updating the data definitions.
  - print facility to obtain hard copy of the viewed screens.
  - analysis reports like cross-referencing, impact analysis, etc.
  - it should support a query language.

Tutorial and Help

- Successful use of CASE tools:
  - depends on the users' capability to effectively use all supported features.
- For the first time users:
  - a computer animated tutorial is very important.

Tutorial and Help

- The tutorial should not be limited to teaching the user interface part only:
  - The tutorial should logically classify and cover all techniques and facilities.
  - The tutorial should be supported by proper documentation and animation.

Current Limitations

- Limitations in flexibility of documentation
- May lead to restriction to the tool's capabilities
- Major danger: completeness and syntactic correctness does NOT mean compliance with requirements
- Costs associated with the use of the tool: purchase + training

Towards Next Gen. CASE Tool >

- An important feature of next generation CASE tools:
  - be able to support any methodology.
- Necessity of a CASE administrator for every organization:
  - who would tailor the CASE environment to a particular methodology.

Towards Next Gen. CASE Tools >

- The user should be allowed to:
- integrate many different tools into one environment.
- It is highly unlikely that any one vendor will be able to deliver a total solution.

Towards Next Gen. CASE Tools >

- A preferred tool would support tune up:
- user would act as a system integrator.
- This is possible only if some data dictionary standard emerges.

Intelligent Diagramming >

- Future CASE tools would
  - aesthetically and automatically lay out the diagrams.

Customization Support

- The user should be allowed to define new types of objects and connections.
- This facility may be used to build some special methodologies.
- Ideally it should be possible to specify the rules of a methodology to a rule engine:
  - for carrying out the necessary consistency checks.

Summary >

- We discussed important features of present day CASE tools:
  - and the emerging trends.
- Use of CASE tools is indispensable for large software projects:
  - where a team of software engineers work together.

Summary

- The trend is now towards:
- distributed workstation-based CASE tools.
- We discussed some desirable features of CASE tools.

Thanks!