Chapter 1 : Computer Aided Software Engineering Tools (CASE)

Computer-aided software engineering (CASE) is the domain of software tools used to design and implement calendrierdelascience.com tools are similar to and were partly inspired by computer-aided design (CAD) tools used for designing hardware products.

A CASE tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process. Computer-Aided Software Engineering tools are those software which are used in any and all phases of developing an information system, including analysis, design and programming. For example, data dictionaries and diagramming tools aid in the analysis and design phases, while application generators speed up the programming phase. CASE tools provide automated methods for designing and documenting traditional structured programming techniques. The ultimate goal of CASE is to provide a language for describing the overall system that is sufficient to generate all the necessary programs needed. CASE Tools are used to help in the project identification and selection, project initiation and planning, and design phacss, and in the implementation and maintenance phases. LowerCASE tools support database schema generation, program generation, implementation, testing, and configuration management. An automated system development environment that provides numerous tools to create diagrams, forms and reports. It also offers analysis, reporting, and code generation facilities and seamlessly shares and integrates data across and between tools. Computer display and report generators: It makes it easier for the systems analyst to identify data requirements and relationship. The reason for using case may be very straight forward and practical decision such as being easier to use and makes life better. However from a broader perspective, Quality to using case implies how Case tools have improved the quality of software development. Case tool has improved software development in the following; Improve the quality of the system developed. Increase the speed with which systems are designed and developed. Ease and improve the testing process through the use of automated checking. Improve the integration of development activities via common methodologies. Improve the quality and completeness to documentation. Help standardize the development process. Improve the management of the project. Promote reusability of modules and documentation. Improve software portability across environments. And how it helps Software Development. Productivity can be said to be the state or quality of producing something. Or the effectiveness of the productive efforts. Therefore productivity to case can be the achievements gained or the effectiveness of using the CASE technology. Productivity has helped in the development of software in the following ways; Provide new systems with shorter development time. Improve the productivity of the systems development process. Improve the quality of the systems development process. Improve portability of new systems. Improve the management of the systems development process. Also designing the systems model â€" graphically creating a model from graphical user interface, screen design, and databases, to placement of objects on screen Code generation CASE Tool has code generators which enable the automatic generation of program and data base definition code directly from the documents, diagrams, forms, and reports. Documentation CASE Tool has documentation generators to produce technical and user documentation in standard forms. Each phase of the SDLC produces documentation. The types of documentation that flow from one face to the next vary depending upon the organization, methodologies employed and type of system being built. Summary This write up introduces the use of automated tools to support the systems development process. The types of CASE tools were also discussed. Function of CASE tools which include:

DOWNLOAD PDF COMPUTER-AIDED SYSTEMS ENGINEERING (CASE)

Chapter 2 : What is Computer Aided Software Engineering? (CASE) | Analysis and Design | FAQ

Techopedia explains Computer-Aided System Engineering Tool (CASE Tool) The essential idea of CASE tools is that pre-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes.

CollabNet TeamForge helps distributed teams by centralizing assets and tools, creating transparency and enabling collaboration. Organizations moving towards Service-Oriented Architecture must manage their services from development to deployment. Generating reports from Java applications has traditionally been a difficult task with heavy coding effort in development and maintenance. Even where graphical tools are provided, users must learn how to use a new and sometimes cumbersome tool. This download delivers Unitrends Enterprise Backup software, which is designed to ensure IT asset recovery from any location, at any time. This solution is available as a virtual appliance for Microsoft Hyper-V. This IT download provides guaranteed access to the most recent versions of technologies produced by Developer Express for Visual Studio and the. With Genius Project you can. Get a degree view of your project and product pipeline, resources, budgets, earnings, and strategic alignment of all of your projects today! IBM Mashup Center is an enterprise mashup platform enabling the rapid creation, sharing, and discovery of reusable application building blocks widgets, feeds, mashups. You can if you download a free trial of IBM Platform Symphony Developer Edition, software that enables rapid development and test of distributed applications by emulating a production grid -- eliminating the need for a physical grid. Genius Project4Domino is an easy-to-use, Lotus Notes and web-based project management solution. Use the virtual workplace to manage projects and processes, where internal and external people can be involved. Register for our free trial. HP Sprinter software gives your testers rich and powerful tools to run manual software testing. The Internet provides great opportunities for networking and learning â€” and IT is no different. Register now for developerWorks and create your profile for access, not only to articles, webcasts and software downloads, but also a wealth of other IT professionals. View the product demo and access downloads now, so you can discover what these development tools can do for your organization. Rational Team Concert enables Agile development teams to track work items, control source code, create and manage defects, and manage the lifecycle of software projects. Download now and get RTC-Express-C free for up to 10 developers or get your hands dirty in our live sandbox with no download required. JIRA provides issue tracking and project tracking for software development teams to improve code quality and the speed of development. Combining a clean interface for organising issues with customisable workflows, JIRA is the perfect fit for your team. Experience low-maintenance and low-cost authorization controls by downloading your free day trial of SafeNet Authentication Service, a cloud-based solution featuring token and self-service capabilities. Learn how to automate and streamline financial, customer relationship and supply chain processes with Microsoft Dynamics NAV.

Chapter 3 : What is a Computer-Aided System Engineering Tool? - Definition from Techopedia

Computer-aided software engineering (CASE) is the application of computer-assisted tools and methods in software development to ensure a high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers and others to see the project milestones during development.

This involves using software packages to accomplish and automate many of the activities of the information system development including software development or programming. Building Blocks Computer Aided Software Engineering can be a single tool that supports a specific Software Engineering activity to complex environment that encompasses tools, a data of people, hardware, network operating system standards and other components i. But the CASE environment itself needs other building blocks. A set of portability services provides a bridge between CASE tools and their integration framework and the environment architecture. Portability service allows CASE tools and their integration framework to go across different hardware platforms and operating systems without much adaptive maintenance. These new methodologies utilize Rapid Prototyping techniques to develop applications faster, at lower cost and higher quality. Mistakes can be detected and corrected earlier this way. The earlier this can be done, the better because correcting these mistakes gets harder and more expensive when the system is developed further. So a lot of time and money can be saved using Rapid Prototyping. As said above, a new set of tools is necessary. These tools should automate each phase of the life-cycle process and tie the application development more closely to the strategic operations of the business. A lot of different tools have been developed over the years and are being developed right now. There are so many tools, that we could easily get confused. To survey all these CASE tools we divide them in the following categories: These are life-cycle processes, derived from the strategic plans of the enterprise and which provide a repository to create and maintain enterprise models, data models and process models. These are derived from several development methodologies like Gane-Sarson or Jackson. These products at least support data flow, control flow and entity flow, which are the three basic structured software engineering diagramming types. Structured development aids-providing products. These products are providing aids for a structured development of the process. These products are very suitable to be used by the system analysts, because they are helped very much by a structured process, because those can be analysed faster and more accurately. These are products that generate application-code for a specific goal, set by the designer. Most of the products in this area are using a COBOL-generator, which is a tool that generates programming code in a specific language out of specifications set by the system-designer. The heart of a well-designed I-CASE system is a repository, which is used as a knowledge base to store information about the organization, its structure, enterprise model, functions, procedures, data models etc. The meaning represented by diagrams and their detail windows is stored in the repository. The repository steadily accumulates information relating to the planning, analysis, design, construction and maintenance of systems. The repository is the heart of a CASE system. Two types of mechanisms have been used in CASE software to store design information: A dictionary, which contains names and descriptions of data items, processes, etc. A repository, which contains this dictionary information and a complete coded representation of plans, models and designs, with tools for cross-checking, correlation analysis and validation. Before implanting CASE and designing tools, a series of steps should be followed: Conduct a technology-impact study to determine how the basic business of the organization should change to maximize the opportunities presented by rapid technological change Evaluate how the organization should be re-engineered to take advantage of new technology Establish a program for replacing the old systems with the most effective new technology Commit to an overall integrated architecture Select a development methodology Establish a culture of reusability Strive for an environment of open interconnectivity and software portability across the entire enterprise Establish inter corporate network links to most trading partners Determine how to provide all knowledge to workers with a high level of computerized knowledge and processing power Determine the changes in management-structure required to take full advantage of innovative systems, architectures, methodologies and tools When an enterprise takes these actions ahead of its competition, it will gain a major competitive

advantage. An enterprise should also be up-to-date because the rapid advances in computer technology allows the competition to get ahead of you. Some significant trends in the development of new system environments include: New development methodologies, more efficient development life-cycle processes are making it possible to develop applications more rapidly and in closer coordination with end users. Growth of standards, standards are emerging that will govern the future evolution of hardware and software Related Articles:

Chapter 4 : CASE (Computer Aided Software Engineering) Tools

Analysts who adopt the SDLC approach often benefit from productivity tools, called Computer-Aided Software Engineering (CASE) tools, that have been created explicitly to improve their routine work through the use of automated support.

CASE stands for Computer Aided Software Engineering which is software that supports one or more software engineering activities within a software development process, and is gradually becoming popular for the development of software as they are improving in the capabilities and functionality and are proving to be beneficial for the development of quality software. Whenever a new system is installed, the implementation integrates a number of related and different tasks. The process has to be efficiently organized and it is for this very reason that CASE tools are developed. With the help of CASE, the installation process can be automated and coordinated within the developed and adopted system life cycle. CASE tools are the software engineering tools that permit collaborative software development and maintenance. Almost all the phases of the software development life cycle are supported by them such as analysis; design, etc. In general, standard software development methods such as Jackson Structure programming or structured system analysis and design method are also supported by CASE tools. CASE tools may support the following development steps for developing data base application: Creation of data flow and entity models Establishing a relationship between requirements and models Development of top-level design Development of functional and process description Development of test cases. Why CASE tools are developed: CASE tools are designed to enhance and upgrade the computing system adopted and used. It is an important part of various business growth strategies. The CASE tools are developed for the following reasons: Time Saving by reducing coding and testing time. Enrich graphical techniques and data flow. Optimum use of available information. Enhanced analysis and design development. Create and manipulate documentation. Transfer the information between tools efficiently. The speed during the system development increased. Here are the ways where the CASE tools are used: To facilitate single design methodology: CASE tools help the organization to standardize the development process. It also facilitates coordinated development. Integration becomes easy as common methodology is adopted. To improve the speed and quality of system development organizations use CASE tools. CASE tools help in improving the testing process through automated checking and simplified program maintenance. In a traditional software development process, the quality of documentation at various stages depends on the individual. It also ensures the completeness of the documentation. It improves project management activity and to some extent automates various activities involved in project management. Reduce the maintenance cost: Use of CASE tools makes the software easy to maintain and hence reduce the maintenance costs. Automation of various activities of system development and management processes increases productivity of the development team. CASE tools play a major role in the following activities:

Chapter 5 : Using Computer-Aided Software Engineering (CASE) tools - Systems Analysis

CASE stands for Computer Aided Software Engineering which is software that supports one or more software engineering activities within a software development process, and is gradually becoming popular for the development of software as they are improving in the capabilities and functionality and are proving to be beneficial for the development.

Computer-aided software engineering CASE is the domain of software tools used to design and implement applications. CASE tools are similar to and were partly inspired by computer-aided design CAD tools used for designing hardware products. CASE tools are used for developing high-quality, defect-free, and maintainable software. Several papers by Daniel Teichroew fired a whole generation of enthusiasts with the potential of automated systems development. By extending the range of metadata held, the attributes of an application could be held within a dictionary and used at runtime. This "active dictionary" became the precursor to the more modern model-driven engineering capability. However, the active dictionary did not provide a graphical representation of any of the metadata. Under the direction of Albert F. While, at the time of launch, and for several years, the IBM platform did not support networking or a centralized database as did the Convergent Technologies or Burroughs machines, the allure of IBM was strong, and Excelerator came to prominence. CASE tools were at their peak in the early s. The application development tools can be from several sources: IBM has entered into relationships with Bachman Information Systems, Index Technology Corporation, and Knowledgeware wherein selected products from these vendors will be marketed through an IBM complementary marketing program to provide offerings that will help to achieve complete life-cycle coverage. The other trend that led to the evolution of CASE tools was the rise of object-oriented methods and tools. Most of the various tool vendors added some support for object-oriented methods and tools. In addition new products arose that were designed from the bottom up to support the object-oriented approach. Andersen developed its project Eagle as an alternative to Foundation. Several of the thought leaders in object-oriented development each developed their own methodology and CASE tool set: Jacobsen, Rumbaugh, Booch , etc. Eventually, these diverse tool sets and methods were consolidated via standards led by the Object Management Group OMG. Fuggetta classified CASE software different into 3 categories: Workbenches combine two or more tools focused on a specific part of the software life-cycle. Environments combine two or more tools or workbenches and support the complete software life-cycle. Tools CASE tools support specific tasks in the software development life-cycle. They can be divided into the following categories: Business and Analysis modeling. Design and construction phases of the life-cycle. Analyze code and specifications for correctness, performance, etc. Control the check-in and check-out of repository objects and files. Analyze code for complexity, modularity e. Manage project plans, task assignments, scheduling. They support traditional diagrammatic languages such as ER diagrams , Data flow diagram , Structure charts , Decision Trees , Decision tables , etc. Lower CASE Tools support development activities, such as physical design, debugging, construction, testing, component integration, maintenance, and reverse engineering. All other activities span the entire life-cycle and apply equally to upper and lower CASE. It incorporates several development tools: Most commercial CASE products tended to be such workbenches that seamlessly integrated two or more tools. Workbenches also can be classified in the same manner as tools; as focusing on Analysis, Development, Verification, etc. Environments An environment is a collection of CASE tools or workbenches that attempts to support the complete software process. This contrasts with tools that focus on one specific task or a specific part of the life-cycle. CASE environments are classified by Fuggetta as follows: Loosely coupled collections of tools. They typically perform integration via piping or some other basic mechanism to share data and pass control. The strength of easy integration is also one of the drawbacks. These environments are also known as 4GL standing for fourth generation language environments due to the fact that the early environments were designed around specific languages such as Visual Basic. They were the first environments to provide deep integration of multiple tools. Typically these environments were focused on specific types of applications. For example, user-interface driven applications that did standard atomic transactions to a relational database. Examples are Informix 4GL, and Focus. Environments based on a single

often object-oriented language such as the Symbolics Lisp Genera environment or VisualWorks Smalltalk from Parcplace. In these environments all the operating system resources were objects in the object-oriented language. This provides powerful debugging and graphical opportunities but the code developed is mostly limited to the specific language. For this reason, these environments were mostly a niche within CASE. A common core idea for these environments was the model-view-controller user interface that facilitated keeping multiple presentations of the same design consistent with the underlying model. The MVC architecture was adopted by the other types of CASE environments as well as many of the applications that were built with them. These environments attempt to cover the complete life-cycle from analysis to maintenance and provide an integrated database repository for storing all artifacts of the software process. The integrated software repository was the defining feature for these kinds of tools. They provided multiple different design models as well as support for code in heterogenous languages. One of the main goals for these types of environments was "round trip engineering": These environments were also typically associated with a particular methodology for software development. This is the most ambitious type of integration. These environments attempt to not just formally specify the analysis and design objects of the software process but the actual process itself and to use that formal process to control and guide software projects. These environments were by definition tied to some methodology since the software process itself is part of the environment and can control many aspects of tool invocation. In practice, the distinction between workbenches and environments was flexible. Visual Basic for example was a programming workbench but was also considered a 4GL environment by many. The features that distinguished workbenches from environments were deep integration via a shared repository or common language and some kind of methodology integrated and process-centered environments or domain 4GL specificity. Organizations usually have to tailor and adopt methodologies and tools to their specific requirements. Doing so may require significant effort to integrate both divergent technologies as well as divergent methods. For example, before the adoption of the UML standard the diagram conventions and methods for designing object-oriented models were vastly different among followers of Jacobsen, Booch, and Rumbaugh. The proponents of CASE technologyâ€”especially vendors marketing expensive tool setsâ€”often hype expectations that the new approach will be a silver bullet that solves all problems. In reality no such technology can do that and if organizations approach CASE with unrealistic expectations they will inevitably be disappointed. As with any new technology, CASE requires time to train people in how to use the tools and to get up to speed with them. CASE projects can fail if practitioners are not given adequate time for training or if the first project attempted with the new technology is itself highly mission critical and fraught with risk. CASE provides significant new capabilities to utilize new types of tools in innovative ways. Without the proper process guidance and controls these new capabilities can cause significant new problems as well.

Chapter 6 : Computer Aided Software Engineering (CASE) | Salamtura's Blog

The Computer-Aided Systems Engineering and Analysis (CASE/A) modeling package was cre- ated as a generalized system engineering program. CASE/A is a very versatile and useful analytical tool.

It means, development and maintenance of software projects with help of various automated software tools. CASE tools are used by software project managers, analysts and engineers to develop software system. Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development. Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary. CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools. Case Tools Types Now we briefly go through various CASE tools Diagram tools These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts. Process Modeling Tools Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer Project Management Tools These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. Documentation Tools Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project. Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. Analysis Tools These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. Design Tools These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. Configuration Management tools deal with â€” Version and revision management Baseline configuration management Change control management CASE tools help in this by automatic tracking, version management and release management. Change Control Tools These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization. Programming Tools These tools consist of programming environments like IDE Integrated Development Environment , in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse. Prototyping Tools Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product. Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder. Web Development Tools These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. Quality Assurance Tools Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of

quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. Maintenance Tools Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC.

Chapter 7 : CASE / Computer Aided Software Engineering Tools

Short for Computer Aided Software Engineering, a category of softwarethat provides a development environment for programming teams. CASE systems offer tools to automate, manage and simplify the development process.

For example, when establishing the functional requirements of a proposed application, prototyping tools can be used to develop graphic models of application screens to assist end users to visualize how an application will look after development. Subsequently, system designers can use automated design tools to transform the prototyped functional requirements into detailed design documents. Programmers can then use automated code generators to convert the design documents into code. Automated tools can be used collectively, as mentioned, or individually. For example, prototyping tools could be used to define application requirements that get passed to design technicians who convert the requirements into detailed designs in a traditional manner using flowcharts and narrative documents, without the assistance of automated design software. Automated tools can also facilitate the coordination of software development activities through the use of data warehouses or repositories. Repositories provide a means to store and access information relating to a project, such as project plans, functional requirements, design documents, program libraries, test banks, etc. Organizations generally implement automated development tools to increase productivity, decrease costs, enhance project controls, and increase product quality. However, only by managing the various risks associated with automated technologies will organizations ensure they develop systems with appropriate functionality, security, integrity, and reliability. Common CASE risks and associated controls include: Inadequate Standardization - Linking CASE tools from different vendors design tool from Company X, programming tool from Company Y may be difficult if the products do not use standardized code structures and data classifications. File formats can be converted, but usually not economically. Controls include using tools from the same vendor, or using tools based on standard protocols and insisting on demonstrated compatibility. Additionally, if organizations obtain tools for only a portion of the development process, they should consider acquiring them from a vendor that has a full line of products to ensure future compatibility if they add more tools. Implementing CASE strategies usually involves high start-up costs. Generally, management must be willing to accept a long-term payback period. Controls include requiring senior managers to define their purpose and strategies for implementing CASE technologies. Quick Implementation - Implementing CASE technologies can involve a significant change from traditional development environments. Typically, organizations should not use CASE tools the first time on critical projects or projects with short deadlines because of the lengthy training process. Additionally, organizations should consider using the tools on smaller, less complex projects and gradually implementing the tools to allow more training time. Weak Repository Controls - Failure to adequately control access to CASE repositories may result in security breaches or damage to the work documents, system designs, or code modules stored in the repository. Controls include protecting the repositories with appropriate access, version, and backup controls.

Chapter 8 : Computer-aided software engineering - Wikipedia

CASE is an acronym for computer Aided Engineering. This involves using software packages to accomplish and automate many of the activities of the information system development including software development or programming.

History[edit] The Information System Design and Optimization System ISDOS project, started in at the University of Michigan , initiated a great deal of interest in the whole concept of using computer systems to help analysts in the very difficult process of analysing requirements and developing systems. Several papers by Daniel Teichroew fired a whole generation of enthusiasts with the potential of automated systems development. By extending the range of metadata held, the attributes of an application could be held within a dictionary and used at runtime. This "active dictionary" became the precursor to the more modern model-driven engineering capability. However, the active dictionary did not provide a graphical representation of any of the metadata. Under the direction of Albert F. While, at the time of launch, and for several years, the IBM platform did not support networking or a centralized database as did the Convergent Technologies or Burroughs machines, the allure of IBM was strong, and Excelerator came to prominence. CASE tools were at their peak in the early s. The application development tools can be from several sources: IBM has entered into relationships with Bachman Information Systems , Index Technology Corporation, and Knowledgeware wherein selected products from these vendors will be marketed through an IBM complementary marketing program to provide offerings that will help to achieve complete life-cycle coverage. The other trend that led to the evolution of CASE tools was the rise of object-oriented methods and tools. Most of the various tool vendors added some support for object-oriented methods and tools. In addition new products arose that were designed from the bottom up to support the object-oriented approach. Andersen developed its project Eagle as an alternative to Foundation. Several of the thought leaders in object-oriented development each developed their own methodology and CASE tool set: Jacobsen, Rumbaugh, Booch , etc. Eventually, these diverse tool sets and methods were consolidated via standards led by the Object Management Group OMG. Fuggetta classified CASE software different into 3 categories: Workbenches combine two or more tools focused on a specific part of the software life-cycle. Environments combine two or more tools or workbenches and support the complete software life-cycle. Tools[edit] CASE tools support specific tasks in the software development life-cycle. They can be divided into the following categories: Business and Analysis modeling. Design and construction phases of the life-cycle. Analyze code and specifications for correctness, performance, etc. Control the check-in and check-out of repository objects and files. Analyze code for complexity, modularity e. Manage project plans, task assignments, scheduling. They support traditional diagrammatic languages such as ER diagrams , Data flow diagram , Structure charts , Decision Trees , Decision tables , etc. Lower CASE Tools support development activities, such as physical design, debugging, construction, testing, component integration, maintenance, and reverse engineering. All other activities span the entire life-cycle and apply equally to upper and lower CASE. It incorporates several development tools: Most commercial CASE products tended to be such workbenches that seamlessly integrated two or more tools. Workbenches also can be classified in the same manner as tools; as focusing on Analysis, Development, Verification, etc. Environments[edit] An environment is a collection of CASE tools or workbenches that attempts to support the complete software process. This contrasts with tools that focus on one specific task or a specific part of the life-cycle. CASE environments are classified by Fuggetta as follows: Loosely coupled collections of tools. They typically perform integration via piping or some other basic mechanism to share data and pass control. The strength of easy integration is also one of the drawbacks. These environments are also known as 4GL standing for fourth generation language environments due to the fact that the early environments were designed around specific languages such as Visual Basic. They were the first environments to provide deep integration of multiple tools. Typically these environments were focused on specific types of applications. For example, user-interface driven applications that did standard atomic transactions to a relational database. Examples are Informix 4GL, and Focus. Environments based on a single often object-oriented language such as the Symbolics Lisp Genera environment or VisualWorks Smalltalk from Parcplace. In these environments

all the operating system resources were objects in the object-oriented language. This provides powerful debugging and graphical opportunities but the code developed is mostly limited to the specific language. For this reason, these environments were mostly a niche within CASE. A common core idea for these environments was the model-view-controller user interface that facilitated keeping multiple presentations of the same design consistent with the underlying model. The MVC architecture was adopted by the other types of CASE environments as well as many of the applications that were built with them. These environments attempt to cover the complete life-cycle from analysis to maintenance and provide an integrated database repository for storing all artifacts of the software process. The integrated software repository was the defining feature for these kinds of tools. They provided multiple different design models as well as support for code in heterogenous languages. One of the main goals for these types of environments was "round trip engineering": These environments were also typically associated with a particular methodology for software development. This is the most ambitious type of integration. These environments attempt to not just formally specify the analysis and design objects of the software process but the actual process itself and to use that formal process to control and guide software projects. These environments were by definition tied to some methodology since the software process itself is part of the environment and can control many aspects of tool invocation. In practice, the distinction between workbenches and environments was flexible. Visual Basic for example was a programming workbench but was also considered a 4GL environment by many. The features that distinguished workbenches from environments were deep integration via a shared repository or common language and some kind of methodology integrated and process-centered environments or domain 4GL specificity. Organizations usually have to tailor and adopt methodologies and tools to their specific requirements. Doing so may require significant effort to integrate both divergent technologies as well as divergent methods. For example, before the adoption of the UML standard the diagram conventions and methods for designing object-oriented models were vastly different among followers of Jacobsen, Booch, and Rumbaugh. The proponents of CASE technologyâ€”especially vendors marketing expensive tool setsâ€”often hype expectations that the new approach will be a silver bullet that solves all problems. In reality no such technology can do that and if organizations approach CASE with unrealistic expectations they will inevitably be disappointed. As with any new technology, CASE requires time to train people in how to use the tools and to get up to speed with them. CASE projects can fail if practitioners are not given adequate time for training or if the first project attempted with the new technology is itself highly mission critical and fraught with risk. CASE provides significant new capabilities to utilize new types of tools in innovative ways. Without the proper process guidance and controls these new capabilities can cause significant new problems as well.

Chapter 9 : FFIEC IT Examination Handbook InfoBase - Computer-Aided Software Engineering

Computer-aided software engineering (CASE) is the use of software tools to assist in the development and maintenance of software. Tools used to assist in this way are known as CASE Tools.