Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1996 Proceedings

Americas Conference on Information Systems (AMCIS)

8-16-1996

Integrated CASE Environments for Information Systems Development -State of the Art and Future Challenges

Robert Winter
Wilhelms University, winter@wi.uni-muenster.de

Integrated CASE Environments for Information Systems Development - State of the Art and Future Challenges

Robert Winter
Wilhelms University
Grevener Str. 91, 48159 Muenster, Germany
winter@wi.uni-muenster.de

Integrated CASE Environments

Process views, data views, functional views, control views, and organisational views of large information systems have to be developed consistently and concurrently. Integrated CASE environments were introduced to support systems development by combining

- graphical support tools for all specification, design, and implementation tasks,
- repository support covering the entire information system life cycle and all systems views,
- powerful phase transition tools, and
- systems development management facilities.

When such environments are based on an appropriate systems engineering methodology, it should be possible to develop large information systems completely, efficiently and consistently.

However, most integrated CASE environments imply a "structured" approach to application development and the utilization of traditional implementation techniques (i.e. "passive" databases and procedural code). But these implications cannot be sustained when non-traditional implementation techniques like active databases, application generators, standardised software packages, object oriented techniques, or internet technologies are used.

In the following, actual limitations of a state-of-the-art integrated CASE environment with regard to these non-traditional techniques are analysed, and requirements for appropriate development support are derived.

Oracle's Designer/2000

Oracle's Designer/2000 toolset for Windows [1] comprises graphical tools for

- business process modeling (*Process Modeller*),
- requirements specification (*Entity-Relationship Diagrammer*, *Dataflow Diagrammer*, *Function Hierarchy Diagrammer*), and
- systems design (*Data Diagrammer*, *Module Structure Diagrammer*, *Module Data Diagrammer*, *Module Logic Diagrammer*).

From requirements specifications, relational schemata and structure charts can be generated automatically (*Database Design Wizard*, *Application Design Wizard*). When systems design is completed, data structures and preliminary 4GL applications can be generated automatically (*Server Generator*, *Forms Generator*, *Reports Generator*, *VB Generator*). If necessary, generated applications can be revised using Oracle's respective development tools.

For storage and retrieval of all development objects, a server repository is used that covers every information systems view (data, functions, event handling, organisation) and the entire IS life cycle (development, maintenance, transition, redesign). By providing locking mechanisms and consolidation / inclusion functionality for all tools, the repository supports concurrent multi-user systems development. If necessary, every repository object can be accessed directly using the *Repository Object Navigator*.

Most important development tasks are performed graphically by editing respective diagrams. In addition to the diagrams, development processes and developed products are documented by a large number of cross-reference reports (*Matrix Diagrammer*) and pre-defined, parametrical *Repository Reports* ranging from property listings of entities, functions, etc. to function point calculations, critical path analyses, and activity-based costing.

Repository objects are assigned to application systems versions. The *Repository Object Navigator* allows to manage (e.g. create/drop, freeze/unfreeze, export/import) application system versions and to control repository access. Repository tables and respective access methods are encapsulated into database packages. Using these packages, third-party tools or individual extensions can be integrated into the CASE environment.

Oracle's requirements specification tools use common Structured Analysis / Structured Design notation. Although inclusion functions allow for some extent of integration, therefore, functional and structural systems aspects have to be specified in separate diagrams. A more integral, but abstract, description of all systems views can be created by the *Process Modeller*. Process diagrams complement information flows, information stores, and functions by events, non-information flows/stores, animation capabilities, and simulation functions (e.g. critical path calculation).

Oracle's systems design tools comprise not only graphical editors for relational schemata and structure charts. For external view design, the *Module Data Diagrammer* allows for the specification of important behavioral elements (e.g. master/detail relationships between tables, lookup tables for attributes). The *Module Logic Diagrammer* allows to assemble procedurally extended SQL code from basic building blocks and, at the same time, creates a conceptual model of the block's control structure.

Although other types of application systems can also be supported, the toolset is tailored to develop and maintain event-oriented transaction processing applications and reporting applications that utilise a graphical user interface system and a relational database management system.

Support for Active Databases

Traditionally, database management systems are used for data storage, data retrieval, and the rejection of inconsistent data manipulations. In an active database management system, rules are stored that can be executed explicitely by an application, another rule, an user interaction, or that are invoked automatically whenever a certain event is detected. [2] Based on such rules, the traditional, extensional database of stored data is complemented by an intensional database of implied, derivable data. An important aspect of information systems is to provide derived (e.g. aggregate, condensed, simulated) data efficiently and consistently.

Using active databases, it is possible to store derivation rules instead of derivation results. Based on these rules, derived data are generated at runtime rather than having to be stored redundantly in the database (and having to be updated whenever underlying data have been changed). Therefore, as many rules as possible should be implemented. As a consequence, the specification of active rules should be supported by integrated CASE environments.

In traditional systems engineering methodologies, data storage and data processing are specified separately. To avoid dangerous redundancy, either derivation rules or derived data should be specified because rules imply data. The data view's importance for information systems development suggests to specify derived data in the conceptual schema and to bind derivation rules to these structural schema components. In a real-time or non-database environment, in contrast, it would be useful to specify derivation rules in the functional view and to bind derived data to these behavioral specifications.

As a consequence, it should be possible to specify derivation rules using the data view analysis tool. Furthermore, generator capabilities have to be enhanced by creating not only tables and integrity

constraints, but also database triggers and stored procedures. First extensions of CASE repositories and appropriate generators have been proposed to cover inheritance, grouping aggregation, and join derivation. [3] Future research is needed to include more complex derivation variants into conceptual modeling and to generate triggers and procedures not only for dynamic consistency control, but also to implement other behavioral specifications. Since application events and triggered rule execution can be refined consistently, a combination of process modeling and data modeling seems to be more suitable for information system implementation by active databases than the traditional data / dataflow specification approach.

Support for Standardised Software Packages

Standardised software packages (e.g. SAP R/3, Baan IV) implement common business processes by large numbers of pre-defined transactions (e.g. about 22000 in Baan IV) and pre-defined reports. Based on a detailed specification of company processes, these components are assigned to organisational roles and used as building blocks to assemble a set of role-specific applications. Since development costs are shared by a large number of customers, package vendors are able to cover a broad variety of industries, process variants, and organisation schemes, thereby allowing most companies to employ at least a significant portion of the included functionality at comparatively low cost and within short time. For users of standardised packages, costly development of individual software should be limited to mission-critical processes and to processes that are not (or not sufficiently) covered. Since most of the standardised packages come with an integrated database, a communication standard, and reporting tools, individual software should reuse this infrastructure.

Usually, information systems are tailored to a company's specific needs so that individual software has to be developed. On the other hand, most information systems are based on data controlled by standardised packages so that an isolated development is rather pointless. To support the development of individual information systems based on standardised packages, the package's process models, requirements specifications, and design documents must be stored in the CASE repository. This procedure is quite simple as long as the standardised package was developed using the same methodology that the CASE environment is based on. E.g., SAP's R/3 system and the "ARIS toolset" integrated CASE environment are both based on the ARIS architecture. [4] Therefore, it is possible to load R/3 models, specifications, and design documents into the ARIS repository. If this system ("R/3 Analyzer" [5]) would include appropriate generators, it could not only be used for documentation and analysis, but also for customization, adaption and individual extensions of the standardised software package.

In most cases, however, specification and design documents are not disclosed by package vendors, CASE environment methodology is not compatible with the vendor's development methodology, or packages of different vendors have to be combined so that no dedicated repository extension can be used. Further dissemination of standardised packages and increasing demand for individual extensions will create a market for repositories for popular CASE environments. Further research is needed to manage the package integration / module selection process, to allow new packages releases to be integrated with adaptions made to earlier releases, and to handle models, specifications, and documents of very high complexity efficiently and consistently (e.g. by consistent clustering/refinement mechanisms for specifications).

Support for Generation of Event-Oriented Applications

For some classes of applications, it has been shown that a general flow of control exists and, as a consequence, that it is possible to transfer runtime control nearly completely to a standardised trigger system. At development time, only relevant events and respective reactions have to be specified declaratively. At runtime, the trigger system detects event occurence, releases respective reactions, and coordinates their execution. Based on this concept, commercial application generators (e.g. Oracle Forms) allow interactive, database-oriented applications to be specified declaratively and to be controlled by standardised software. Application development is then reduced to declarative specifications, and application semantics can be entirely stored in the CASE repository. Based on such a repository, reporting applications can perform automatical documentation, application elements can be easily retrieved and

reused, and maintenance is reduced to the real application kernel, i.e. to application elements that cannot be transferred to database management system, user interface management system, or trigger system. [6]

Since Structured Analysis / Structured Design assumes the flow of control to be specified procedurally during systems design, most integrated CASE environments do not support event-oriented, declarative application specification. Although retaining structure charts as a high level model of module structure, Designer/2000 replaces textual / graphical minispec editors by specification of external views and database procedures. When *Module Data Diagrammer* and *Module Logic Diagrammer* are used for module design, important behavioral application elements like master/detail-connections between field groups, lookup functions, and window/page assignments of fields can be edited graphically and are stored declaratively in the repository. However, the specification of object/event connection cannot be supported by appropriate (state transition) charts, and important application elements (e.g. window layout, intra-application navigation) cannot be specified graphically.

Future research is needed to examine how multi-stage process modeling and appropriate decomposition principles can be used to create state transition diagrams from which event-oriented applications can be generated. Since the acceptance of application generators in commercial application development suffers from a complete lack of standardisation, at least a basic set of event types and general application structure has to be agreed upon for selected classes of applications (e.g. OLTP, OLAP, reporting). Since, due to its declarative nature, event-oriented application specification is a perfect case for CASE, significant progress in information systems engineering productivity may be expected in this area.

Emerging Implementation Techniques and Further Support Requirements

Being non-traditional implementation techniques for information systems, active databases, standardised software packages, and application generators have been discussed because they have already gained significant importance in commercial application development. For some other concepts, even more drastic consequences on CASE environments as well as greater commercial importance may be expected: If, based on some general standard, procedurally extended relational databases develop into object oriented databases, the boundaries between data view and functional view in requirements specification has to be redefined, and the migration to novel tools in systems design is inevitable. Finally, internet applications will have an even higher impact on systems engineering methodology and CASE support after security, privacy, standardisation, and performance problems will have been cut back to an extent that allows to justify large development projects. E.g., Java applications are assembled by some browser at some net site using application blocks retrieved from distributed, replicated libraries. To develop safe, complex applications for an execution environment that is virtually unspecified at development time, methodologies are needed that replace the traditional, structured approach of separate views by integral, flexible environments. Process modelling and repository-based, integrated CASE environments already offer some of these features.

Selected References

[1] Oracle Corp., Designer/2000 Product Overview, Release 1, Part No. A32496-2 (1995)

[2] McCarthy, D.R. / U. Dayal: The Architecture of an Active Data Base Management System, ACM Sigmod Record, 18 (1989), 2, 215-224

[3] Winter, R.: Formalised Conceptual Models as a Foundation for Information Systems Development**,** in: P. Loucopoulos, ed., Entity-Relationship Approach - ER'94 (Springer 1994), 437-455

[4] Scheer, A.-W.: Architecture of Integrated Information Systems (Springer 1992)

[5] SAP AG, SAP R/3 Analyzer (1993)

[6] Winter, R.: Application Development in a CIM Environment, in: Olling, G.J. / F. Kimura, eds., Human Aspects in Computer Integrated Manufacturing (North-Holland 1992), 465-473