A Framework for Integrated CASE

MINDER CHEN, George Mason University
RONALD J. NORMAN, San Diego State University

CASE technology has made significant advances recently, but its potential is limited by integration difficulties. The authors propose an organizational framework to guide integrated CASE development and research.

Today's information-based organizations need comprehensive information systems to support their business. Products and services are often information-intensive and must be brought to market faster if the organization intends to compete globally. The dual trend of distributing the information-systems function into business areas and outsourcing systems-development work has increased the pressure on IS organizations to enhance their credibility. Integrated CASE environments are becoming a key strategic component in IS organizations' attempts to meet these new challenges.

Although CASE has significantly affected the practice of systems development (the article on pp. 12-16 explains the evolution of CASE), its potential is limited by the difficulties involved in integrating tools into a cohesive environment. There is growing interest in the research, development, and deployment of integrated CASE environments, as evidenced by:

- the announcement of comprehensive CASE product strategies by hardware and software vendors (including IBM's AD/Cycle, Digital Equipment Corp.'s Cohesion, and Hewlett-Packard's Soft-Bench product lines),
- the increasing market acceptance of integrated CASE products (including Atherton Technology's Software Backplane, Interactive Development Environments' Software through Pictures, KnowledgeWare's Application Development Workbench, and Texas Instruments' Information Engineering Facility),
- recent efforts in CASE, standards and frameworks (including the American National Standards Institute's Information Resource Dictionary Systems, IEEE's work on tool interconnectivity, and the Electronic Industries Association's in-
There is as yet no coherent strategy for building and using integrated CASE. Technical frameworks exist but don’t put tools in the context of the development organization.

**TECHNICAL FRAMEWORK**

An integrated CASE environment must be as adaptable, flexible, and dynamic as the enterprise, projects, and people it supports. In such an environment, users can coherently mix and match the most suitable tools that support selected methods. They can then plug those tools into the environment and begin working with them.

We have adapted the NIST/ECMA reference model, shown in Figure 1, as a basis for describing the technical aspects of integrated CASE environments. Services defined in the reference model enable three forms of integration:

- **data integration**, which is supported by repository and data-integration services;
- **control integration**, which is supported by process-management and message services; and
- **presentation integration**, which is supported by user-interface services.

This reference model, which describes a wide range of CASE environments and frameworks, can guide standards development and serve as a basis for educating software engineers.

**Data integration.** The ability to share design information is key to integrating tools. According to the IEEE draft standard for tool interconnection (P1175), there are four information-sharing methods:

- **Direct transfer** of design information between two tools is most efficient when real-time integration is required. However, it is very difficult to implement direct transfer when many tools must be integrated.
- **File-based transfer** is the simplest to implement. The CASE Data Interchange Format developed by the FIA is the most mature file-based transfer standard.
- **Communication-based transfer** is appropriate for open systems and distributed environments.
- **Repository-based** transfer supports a tightly coupled, consistent environment and is the cornerstone of several integrated CASE products. A (data) repository provides many basic services, including storage and management of objects/entities and links/relations; version and configuration control; naming services; security; and transaction control.

**Repository requirements.** Data-integration services provide several high-level functions that tightly integrate tools and the repository, including:

- a **metamodel** service, which defines, controls, and maintains a metamodel;
- a **query** service, which supports the retrieval of the metamodel and metadata from the repository;
- a **view** (or subenvironment) service, which lets developers define a subset of objects and operations in the repository and maintains consistency between the environment and this subenvironment; and
- a **data-interchange** service, which sup-

*Figure 1. The NIST/ECMA reference model.*
ports two-way translation between flat-file and repository data.

A repository should also offer pragmatic interfaces, including command- and menu-driven interfaces, and provide application-program interfaces for CASE tool builders.

**Metamodel:** In a CASE framework, a comprehensive metamodel is necessary to achieve a higher degree of data integration. Examples of metamodels include the Information Model of AD/Cycle, the Basic Functional Schema of IRDS, the semantic model of CDF, and the Software Concepts defined in the IEEE P1175.

In ANSI's standard X3.138, IRDS structures are described with an entity-relationship model. Many CASE products support the IRDS standard; others use proprietary object-management systems to manage their repositories.

An enterprise-wide repository makes it easier for systems projects to share information. For example, repository functions like metadata sharing and distribution, merging of actual and trial metadata, version control, downloading and uploading, and concurrency and security controls can facilitate integration across and among large project teams.

**Control integration.** Tools must be able to notify each other of events, activate other tools under program control, and share functions.

Control-integration mechanisms include explicit message passing, time- or access-activated triggers, and message servers. To achieve control integration, tools can call message services to provide three types of communication: tool-to-tool, tool-to-service, and service-to-service.

Process-management services can explicitly represent tool-invocation sequences and policies, so users are insulated from the tools' details and are free to deal with the abstraction of tasks and processes. Processes are related to the overall enterprise-wide systems development processes and are considered to be an abstraction of tasks.

Because improved control integration can increase the degree of automation, the environment can support global process management, project-level task management, and tool invocation.

Early implementations of integrated project-support environments focused on control integration to coordinate teams through facilities like electronic mail, configuration management, and context management. (Context management controls what is visible to each tool and each user, simplifying the user's interactions with tools and other users.)

The Portable Common Tool Environment is a tool-interface standard adopted by ECMA to support tool portability. Under PCTE, tools are stored in the repository and executed as processes, making it possible to improve tool execution, composition, communication, and synchronization.

**Presentation integration.** User-interface services let CASE users interact with tools consistently, making new tools much easier to learn. Window-based tools have four levels of presentation integration: the window system, the window manager, the user-interface-development tool kit, and the look-and-feel guidelines.

Most CASE frameworks and environments have adopted generic standards for the user-interface services. For example, Motif is the presentation standard used by most CASE tools in the open-systems world. Individual environments extend these presentation standards with sets of presentation and dialogue design guidelines for displaying and manipulating structured texts, structured graphics, and matrices common across the methods supported.

**Other dimensions.** The NIST/ECMA reference model addresses other dimensions of CASE integration indirectly through a tools layer. Vertical and horizontal tools can be configured and plugged into the tools layer to support vertical and horizontal integration:

- **Vertical** (full life-cycle) integration ensures the completeness and consistency of information generated in various life-cycle phases. Mechanisms that support vertical integration include forward and reverse engineering, configuration and change management, and requirements-tracking tools.
- **Horizontal** (methodological) integration maintains the integrity of design information within each life-cycle phase when many modeling methods are used (such as data, process, event-driven, and object-oriented). Mechanisms that support horizontal integration are a comprehensive repository metamodel, integrity-checking rules, and hypertext-like navigation across multiple modeling perspectives.

**ORGANIZATIONAL FRAMEWORK**

A tool is most effective when it works within an organizational context. The technical framework just described does not consider specific tools' functions. Instead, as Figure 1 shows, tools are plugged into a tools layer, which calls on the framework services to support a particular systems-development function.

The organizational framework shown in Figure 2 seeks to place CASE tools in a development and management context. On the left side of the figure, we group an environment's services and tools into three levels. Components at each level support the corresponding activities on the right side.

On the right side of the figure, the framework divides systems development and management into three activity levels:

- IS infrastructure planning and design is undertaken at the enterprise level.
- Systems project management and decisions are made at the project level.
Software-development processes are carried out at the individual and team level. Higher level activities can control lower level activities; lower level activities can influence higher level activities. An integrated CASE environment must support activities at all levels.

This organizational framework, which complements the technical framework, can guide the development and deployment of integrated CASE environments, direct future research, and help CASE users select and configure tools in an integrated CASE environment.

Planning and design of IS infrastructures. Activities at the highest level determine how development processes should be managed. The major concern at this level is building the enterprise-wide IS infrastructure. Activities at this level include:

- establishing guidelines for the deployment of computer platforms,
- evaluating and adapting methods and process models, and
- formalizing procedures and policies for project management and coordination.

Several organizations have set up development or productivity centers within their IS organizations to perform these functions. The decision to introduce new tools and the evaluation of the integration effort involved are usually made at this level. The introduction of new methods, development standards, and tools is aided by the tool-integration mechanism and the repository services.

Integrating frameworks, which provide an infrastructure for data, tool, and process integration at the enterprise level, should be independent of languages, methods, tools, and platforms so that they can accommodate existing and future tools.

Management of systems projects. The second level includes the management and decision processes that extend across many or all life-cycle phases. These activities include project and process management, impact analysis and change management, documentation, and reuse.

Examples of project-management activities supported by tools are scheduling and tracking, personnel assignment, cost estimation, and coordination.

Process-management tools can invoke appropriate CASE tools and retrieve design objects according to defined process models in a way that enforces development disciplines and policies. Communication tools, such as electronic mail, can also be used to coordinate project teams.

Design information captured by CASE tools and stored in a repository contains many attributes about the process and products. In an integrated CASE environment, measurement tools can generate productivity and quality metrics using this information. Several CASE products collect data and calculate metrics automatically. It is very important to develop a historical project database to calibrate existing metrics and models and derive new models. These metrics can provide feedback for scheduling, cost and staff estimation, and quality control of both the process and the product.

IS personnel can also use measurement results to evaluate tools, methods, and process models and to ensure that ongoing process monitoring and improvement is successful.

Execution of software processes. At the lowest level, integrated CASE should support activities for each life-cycle phase: planning, analysis, conceptual design, technical design, programming, testing, production, and maintenance.

Upper or front-end CASE tools assist upstream activities; lower, or back-end, CASE tools assist downstream activities. These vertical tools can support the elicitation, representation, storage, analysis, and transformation of development information.

Requirements-elicitation tools can make it easier for developers and users to interact as they define system specifications. These tools can elicit and represent information in various formats and store it in local or enterprise-wide repositories. The consistency and completeness of the specifications are checked against a set of integrity rules specific to the method used.

To further assist the mapping of design information, some CASE tools include a transformation function that converts specifications from one life-cycle phase to the next. Automatic two-way transformation between phases is essential to achieve vertical integration. Transformation tools can be used for forward and/or reverse engineering. Reverse-engineering tools are useful when development information is available only in a low-level form (like code), but in the long run they will be less useful because systems developed in integrated CASE environments are main-
tained through high-level specifications.

Different CASE environments require different forms and levels of integration. Science and engineering applications, for example, require more control integration, while business applications tend to require more data integration. CASE users should choose or implement appropriate forms of integration according to the characteristics of their own development infrastructure and practices.

Tomorrow’s complex, integrated applications will be developed using a combination of several enabling technologies (database and knowledge-based systems, object-oriented technology, and hypermedia). Today’s CASE environments support application development in limited domains using one or two specific enabling technologies. Future integrated CASE environments will support a wider range of applications in an open-systems environment.

Current CASE technology still encourages an individual approach to development, although mechanisms like version- and configuration-control and multiuser repositories do support programming in the many. In the future, gains in quality and productivity may come from improved, direct human interaction — both between developers and users and among developers. Future CASE environments will incorporate collaborative tools (groupware) to support cooperative development.

The repositories of today’s CASE environments are used only by IS personnel for systems development, even though the data and process models they contain — information about the organization’s structures, business goals, and processes — are valuable to managers. Future CASE environments will support both systems development and information delivery. Guided by metadata stored in the repository, users and managers will have a context for the application data they access.

The key issue in designing and selecting an integrated CASE environment is how to strike a balance between integration and flexibility. Tighter integration usually means less openness. CASE shells (also called metasystems, CASE tool generators, and metaCASE) are an emerging class of CASE environments that let CASE developers or users customize CASE tools and environments to their new or unique methods. Examples of CASE shells are CADWare’s Foundry, Intersolv’s X1/Customizer, and Systematica’s Virtual Software Factory.

Reuse is one of the most promising approaches to achieving significant productivity gains. Using reusable components not only reduces development cost, but also increases development speed and product quality. The reuse concept should be extended from code to high-level specifications and processes. Object-oriented technology offers effective mechanisms, inheritance and encapsulation, for creating and adapting reusable components. Artificial-intelligence techniques, such as analogical and case-based reasoning, can help identify and select components for reuse.

CASE standards have played a major role in the development of open CASE systems. Integrated CASE environments should be based on these standards. However, many CASE standards overlap and conflict. If we are to realize useful CASE standards, we must harmonize the efforts of formal standards organizations, gain support from vendors and users, and develop CASE standards based on other relevant standards.

Technology transfer and the study of organizational behavior is another fertile, essential research area that is relevant to integrated CASE. Finally, AI techniques may enable integrated CASE environments to incorporate domain-specific knowledge that can help end users develop and maintain their own systems using high-level languages or diagramming tools. Eventually, end users may be able to retrieve or purchase high-level reusable business models, modify them, plug them into an integrated CASE environment, and generate their own applications. What a day that will be!

Acknowledgments
Several important ideas in this article were generated from discussions with Tsiang-Huiang Lee, Edgar H. Sibley, and William Wong. We also thank Jay F. Nunamaker, Jr. and Anthony I. Wasserman for their comments on an early draft of this article.

References

The authors’ biographies appear on p. 16.