CASE Integration

An integration framework for a CASE environment consists of the basic infrastructure of architectural and functional capabilities that allow independent tools to be integrated in a common environment. The common user interface gives the user a common "look and feel" for each tool, offering presentation integration. The integration agent provides functions that enable tools to share data, and to communicate with one another. The object management system represents the logical or physical repository for the above function. An integration framework can help implement an Integrated Project Support Environment (IPSE).

An IPSE is desirable because it covers all phases of development and support and acts as an overall backdrop for consolidating the activities of software development. Though fast and inexpensive, users and vendors must be aware of the limits and capabilities of current CASE tool integration frameworks and IPSEs so their expectations are realistic and their proposed solutions are achievable.

The most common tool integration issues are: which system development phases and activities can be covered, which tool choices the IPSE provides, where the data resides, the type of user interfaces the IPSE provides, how the tools communicate, how adaptable the approach is to new technology, and who does the integration and supports it. Technologies proposed for CASE tool integration include the repository, the link manager, and the message switch. Each tries to provide a public tool interface that addresses some or all of the following: data interchange, management, and relationship management; tool execution, composition, and communication; security or access control; distribution in a network; the user interface; and licensing.

There are technical, political, and economic issues that surround CASE tool integration frameworks. While tool integration may not yet be fully mature, it offers an important enabling technology for the 1990s.

Source: "Perspectives on CASE tool integration," by Nicholas Wybolt, *Software Engineering Notes*, July 1991, pp. 56-60.

CASE User's Bill of Rights

Joseph R. Schofield, Jr.

Prescription for presenters, encouraging them to concentrate on the core issues of CASE and its advancement

The accelerated pace of changes in the CASE industry places increasing importance on the information exchanged at CASE and "user" conferences. Increasing pressure to deliver CASE-based systems, and expand the deployment of CASE within organizations, adds to the necessity of obtaining valuable insights at professional gatherings. Given the stated changes and time constraints, CASE conference attendees deserve a substantial return on their investment. The following CASE user's "Bill of Rights" offers a list of early warning signs for conference attendees that may help identify presentations where lesser value is likely to be received. The list also serves as a prescription for presenters that will encourage them to concentrate on the core issues of CASE and its advancement.

1. ***You have the right to avoid statements of direction that are replaced, updated, or revised monthly.*** The strategic nature of directional statements precludes constant alteration. Other ramifications are less obvious. First, ongoing changes undermine the confidence that the CASE customer seeks from a CASE supplier. With whom is there an alliance today? What's the latest hearsay concerning tomorrow's partners? What will the impact be of the upcoming takeover or merger or attempt? Second, planning in a volatile environment requires additional thought as the numerous likely paths to implementation unfold, which forces additional contingency planning upon the CASE user community. A preferred approach would be to provide strategic vision with real products from existing companies, and associated delivery dates.

2. ***You have the right to avoid suppliers who proclaim their existence in the CASE industry for greater than five years.*** Given the emergence of CASE and its early descriptions from Carma McClure and others in the late 1980s, supplier credibility is greatly reduced by attempts to substantiate their pioneering of the CASE industry (as they sometimes claim) for the past 10 to 20 years. While admittedly part of their product lines are BC (Before CASE), the tools were neither conceived nor developed with the key CASE components in mind. A preferred approach would be to admit to the particular niche of the software engineering process supported, without speculating as to how the product makes "whole" current development techniques.

A quick review of key CASE components (or core competencies) includes: a single encyclopedia that 'reposites' the rules to enforce a rigorous methodology that encompasses the entire life cycle (with due emphasis on complete and correct code generation), integrated as a single product. This definition preempts import and export capabilities to other products (to supplement those life cycle phases either not supported well or unsupported). Import and export features indicate an emphasis on interfacing rather than integration. Further, import and export features endorse a reliance on terminology such as upper and lower CASE, front- and back-end CASE (all four imply an incomplete life cycle approach), and I-CASE (which is a superfluous designation).

3. ***You have the right to avoid announcements considering future releases that should have been delivered years ago.*** Active observers in CASE advancements are often disappointed in the promotion of "missing pieces products." Since such capabilities have long been delivered by other tools suppliers, it's difficult to determine why suppliers would strongly promote enhancements to their own product lines when similar components are *common* among the competition. Exuberant enthusiasm for overdue components raises doubts as to whether such suppliers are cognizant of CASE market needs. A preferred approach would be to *quietly* announce the common, while reserving the jubilant proclamations for newly acquired and unique strengths in the areas

of methodology, completeness, and consistency techniques.

4. ***You have the right to avoid proprietary disclosures concerning multi-operating system environments intended to execute on a single vendor's platforms.*** Too much attention is focused on attempts to provide similar "look 'n feel" across multiple operating systems. These attempts trigger two concerns. First, why the need for so many operating systems from a single vendor? The support required to support the multitude of operating systems is necessarily a cost passed on to the customer. Second, why not focus on the more serious issue of cross-platform (i.e., cross-*vendor* platform) CASE environments? A preferred approach would be for hardware suppliers to collaborate with CASE tool suppliers toward truly open systems that would then facilitate common integration across both tools and platforms.

Why the need for so many operating systems from a single vendor?

5. ***You have the right to avoid the introduction of more CASE products that do not address the entire life cycle.*** What tools would survive in the CASE market if all the front-end, back-end, upper- and lower-CASE tools were removed from the market? The correct answer is: "None of the above—almost." Only the true contenders of CASE would remain. The pretenders would need to reference their products as what they are—incomplete offerings. The front- and back-end, upper- and lower-CASE labels intentionally fragment the CASE market. Unfortunately, I-CASE proponents have had to distinguish their products from the prevailing clutter when 'integration' within a product set was an original aspiration of CASE environments. A preferred approach would be to exercise patience and deliver life cycle-based, integrated productivity environments as part of Release 1.0, and avoid a disservice to the industry and to customers by releasing anything less.

6. ***You have the right to avoid proponents that confuse reverse engineering with their current re-engineering products.*** Recognizing the need for CASE products that assist with the estimated 60 to 80 percent of software budgets dedicated to current product support is not novel. Neither is the notion that reverse engineering, by definition, must revert the software product to an earlier life cycle stage. Preferably, mature reverse engineering would restore software to its initial requirements level—a predesign level. Equally important, and far less understood, is the need to reverse engineer both data and processes. The deliverance of a complete toolset for software engineering draws discomfort from the supplier community. Until delivered, however, maximum application of CASE tools is not achievable. A preferred approach would be to recognize the limited value of partial reverse engineering, and concentrate on process models. Further, maximize organizational CASE value by addressing new product development, and eventually current product engineering. In effect, this suggestion positions CASE organizations for reverse engineering of products currently being developed, without expanding the "backlog" of "to be reverse engineered" products.

7. ***You have the right to avoid an approach which encourages a "roll-your-own methodology."*** A dilemma exists for a number of CASE suppliers today. The dilemma surfaces from an inconsistency in marketing strategies emanating from equally unclear product representations. The most blatant example of inconsistencies sounds something like the following: "As strategic partners (in the sale of a multitude of products, which are in varying degrees of integration, and not quite complete), our strategy is to offer a non-methodological approach to software engineering. This approach offers our client base the maximum flexibility in defining their own competitive strategies and platforms." When this drivel is challenged and the need for a methodology emphasized, the response typically becomes: "Of course you realize that our original product contained the name of what we believe to be a preferred methodology, and through some of our other partnerships we still support the methodological rigor you need to build quality systems." This second response is the legitimate response for serious software engineers; but hearing it as a contingency plan somehow undermines the integrity of the supplier and the product. A preferred approach would be to recognize that the methodology is the heart of a CASE tool; a primary source of productivity and quality improvement opportunities.

An even more irritating dilemma surrounds methodology adherence. This second dilemma is the provision that allows the purchaser to *build* a methodology (which sometimes translates into implementation of the existing methodology in use). Application of this flexibility by potential CASE engineers will not facilitate any transformation leading to true process improvement. In effect, this approach automates that which does not work, and accelerates defects. The current environment does not work because the rules for completeness and consistency cannot be added to a methodology—the rules are the *essence* of the methodology. Too often, software engineers attempt to test quality into the final product—well beyond the requirements and design phases where specifications need to first be validated. Several authors have documented the economy of defect detection early in the life cycle. A preferred approach would be to avoid any CASE supplier who entrusts the purchaser with finishing the methodology. Few software engineers are knowledgeable enough, in all life cycle phase aspects, to construct their own methodology.

8. ***You have the right to avoid messages that describe the CASE payoff in two to four years.*** Sometimes this ploy is cleverly disguised as, "CASE will pay for itself in the maintenance phase." Review the ongoing turmoil in the CASE industry. Conceivably, a number of suppliers making these promises will not exist in two to four years. Investors in CASE technology deserve noticeable and measurable process improvements on their initial project. Despite the reported "steep learning curve" using robust CASE products, productivity gains are obtainable. On the contrary, if enough CASE suppliers continue to devalue productivity in first generation CASE products, success stories will become increasingly more difficult to substantiate, and equally difficult to motivate. A preferred approach is to expect the best, including introductory benefits that have an *immediate* impact on productivity and quality.

Avoid any CASE supplier who entrusts the purchaser with finishing the methodology.

9. ***You have the right to avoid unprepared presenters.*** These presentations usually begin with one of two excuses. First, their presentation was not completed far enough in advance for their handouts to be

published with the proceedings. Inevitably, these are the same folks who also did not make enough copies for attendees. Nevertheless, the interested parties' business card, left on the table, will suffice as a request for materials which will be mailed promptly upon return to the office. Be cautious that the lack of planning is not disguised as "current information." Buyer beware!

Second, because the presented materials are so current, the audience will not have most of the presented materials in the pre-published proceedings. In marketing, this technique is referred to as "bait 'n switch," and is often illegal. The underlying fraud remains the same as for the first excuse. Find another session if the presenter is incapable of incorporating up-to-the-minute announcements with a well-prepared topical discussion. A preferred approach would be to measure the extent of change and avoid presenters who have a history of altering greater than 20 percent of their materials. A disorganized presenter does not represent the espoused techniques, strategies, or product very well.

10. ***Lastly, you have the right to avoid and be most careful of the illusion that the usage of the phrase "open architecture" implies "accessibility" rather than "standardization."*** CASE supports structure. CASE requires rigor. Software engineering supports structure. Software engineering requires rigor. CASE is software engineering for the 1990s. Its value rests upon a foundation of standards; not loosely defined concepts. A preferred approach would be to require tools, not just strategies; techniques, not just concepts; standards, not just open architectures.

The preceding text has described many essentials of CASE, why they are important, and why their quest is worthwhile. These descriptions prescribe a suggested platform for meaningful CASE presentations for future conferences, symposiums, and seminars. In addition, specific recommendations are offered to improve the content and representation of CASE products and approaches. Presenters take note—your audience has high expectations for the technology and how it's promoted.

Joseph R. Schofield, Jr. has actively worked in a CASE environment for five years, teaches graduate classes in analysis and design at the College of Santa Fe, has made numerous presentations at industry conferences, and has authored several articles in information systems trade journals.

SOFTWARE REFLECTIONS

Lying to Management

Robert L. Glass

Ethics may occasionally be ignored for the sake of getting the job done

I've taught a lot of seminars over the years.

My favorite part of a seminar is the interaction with my attendees. I love interaction, and I encourage it. I've even been known to toss out an "outrageous bias" or two, just to stir the attendees up a little and get them to put words to an opposing view. In this software field of ours, less than half a century old, we are still exploring what the correct answers are to a lot of important questions. And only if people are willing and able to express their beliefs and opinions will we be able to sort out conflicting views and begin to identify Truth.

So it was a special pleasure for me when one group of seminar attendees actually took over the seminar and steered it in some new directions! I had come armed with several days' worth of lecture notes, as usual, but I was only part way into the first day when I realized that the group was ready and able to charge off somewhere else!

So off we went. It started when I set up a workshop exercise, gave the attendees a problem definition, and asked them to explore what the design process was for the problem in question.

The problem definition contained an intentionally impossible schedule. What I thought would happen was that the attendees would solve the problem, ignoring the schedule constraint.

I was very wrong. Almost to a person, the attendees did whatever they did *within* the impossible schedule I had established for them.

That fact was worth exploring in the seminar, I thought. And as we discussed what had happened, the discussion took some more unexpected turns.

First, why had the attendees stayed with the schedule? Because, they said, that's the way it is in software development these days. Schedule conformance is the most important goal to strive for, and management grades you on how well you do. Regardless of what it means to the quality of the final product.

From there, the conversation took an even more surprising turn. We started talking about "lying to management." And I don't know whether this particular group is typical or not, but what I learned at that mid-afternoon, unplanned bull session scared my socks off.

Here are some quotes:

- A middle manager said, "I have to lie 30 to 50 percent of the time in order to get my work done." He went on: "I had to check my ethics at the door when I went to work here."
- A consultant said, "I make wildly optimistic promises to get management off my back."
- A group leader said, "Lying gets me resources I wouldn't otherwise get. I invent fictional projects and attach key people to them in order to hold onto them."
- A high-level manager said, "If it's easy to get caught you don't lie." He went on, "I'm an honest person, but my honesty has gotten me in trouble sometimes because people don't want to hear the truth." And then, "Managers who don't tolerate failure I especially lie to."
- A consultant said, "At the company where I'm working, the fixer is the hero. A lot of things end up needing fixing there."

"I had to check my ethics at the door when I went to work here."

The consensus, in this rapid-fire conversation, seemed to be that nearly everybody lied to management. Most of the lies, someone pointed out, were "white lies" rather than black ones. When pressed for a distinction, he said that black lies "were those where one said 'A' when 'B' was demonstrably