THE EMERGING USE OF APPLICATION TEMPLATES

John F. Rockart
J. Debra Hofman

December 1992

CISR WP No. 250
Sloan WP No. 3523-93

Center for Information Systems Research
Sloan School of Management
Massachusetts Institute of Technology

THE EMERGING USE OF APPLICATION TEMPLATES

John F. Rockart and J. Debra Hofman

ABSTRACT

The process by which systems are delivered today has not kept pace with the demands of the business environment. While there have been advances in speeding up the existing process, systems still take too long to build, cost more than expected, often do not meet the business need once they are delivered, and cannot be easily changed to meet continual change in the business. Further, for many organizations, significant resources are still tied up in maintenance rather than in the development of the new systems that are needed. We believe that the use of application templates -- where a "template" is a system that has been built with a CASE tool and reused -- represents a very significant attempt to address these issues. This paper describes our definition of templates, as well as the trends we see in the emerging template market. The majority of use to date has involved one company purchasing a CASE-built system from another company and customizing it to its own needs. The concept of reusing models, however, can also be applied *within* multi-divisional companies and across companies within an industry. While the market for templates is currently in its infancy, we believe that this is a major trend and that templates are the software packages of the future. Further, effective use of a template approach has major implications for the systems development process, and offers opportunities for changes in managing the business.

I. THE PROBLEM

A rapid pace of business change, increasing global competition, time as a competitive differentiator, and a customer focus are all attributes of the current business environment. This external environment has required major change in the way that organizations compete, and in how they are structured, managed, and operated. In the information systems arena, changes are required both in the types of systems that are needed and in the way in which they are delivered.

The process by which systems are delivered, however, has not kept pace. It remains as labor intensive and time consuming today as it was 25 years ago, and is increasingly deficient in meeting the demands of the business environment. While there have been advances in speeding up the existing process (e.g., rapid application development, or RAD, and prototyping), systems still take too long to build, cost more than expected, often do not meet the business need once they are delivered, and cannot be easily changed to meet continual change in the business. For many organizations, significant resources are still tied up in maintenance, rather than in the development of the new systems that are needed. The current trend toward outsourcing can be seen as evidence of a growing dissatisfaction with current internal IS capability (Loh and Venkatraman, 1992).

II. THE ALTERNATIVES

To improve the organization's systems delivery capability, we see four major options facing IS executives: CASE tools, an object-oriented approach, software packages, and templates.[1]

- *CASE Tools:* Computer-Aided Software Engineering (CASE) tools, which help to automate the process of developing systems, first emerged in the mid-

[1] While there are certainly other changes in systems delivery that are being made (e.g., process changes such as RAD), we believe the four noted above are the current *major* alternatives.

1980s. These tools have not proved to be the long awaited "silver bullet" in the development area, for a variety of reasons, both technical and organizational.[2] However, there have been some success stories. And, while the market has been characterized to date by a proliferation of independent tools, it is beginning to coalesce into a smaller number of full life-cycle (I-CASE) tools.[3] In addition, studies of some of the early experimenters have yielded useful insights into better ways to introduce and leverage these tools (Orlikowski, 1991; Orlikowski and Friesen, 1989; and Orlikowski, 1989).

- *Object-Orientation:* A second software development innovation which has gained much attention in the press is the object-oriented approach.[4] While this approach has proven extremely valuable for some types of systems (multi-media, simulation, real time), there are very few business systems to date which have been developed using it. And, while it clearly has the potential to yield significant benefits and improvements in systems delivery, it also requires some radical changes relative to traditional methods of software analysis and design (Fichman and Kemerer, 1991), and is not likely to be quickly adopted by most IS groups (Fichman and Kemerer, 1992).

- *Software packages:* Where CASE and object-orientation are alternatives that seek to improve the development of a system, another alternative is to decide not to build at all. There is a wide variety of application software packages available for purchase. These range from packages that perform specific functions -- companies purchasing these then have to build the bridges among them themselves -- to integrated suites of packages.

- *Templates:* Templates -- CASE-based software packages -- are a relatively new, hybrid alternative. As we note below, they combine the pre-built advantages of packages with the maintainability afforded by CASE tools.

---

[2] Most companies that have adopted CASE tools have realized that the introduction of these tools -- and the methodologies on which they are based -- represent major change for the IS organization, and that effective use of the tools is associated with a significant learning curve (Kemerer, 1991). Also, the benefits have generally been realized not in development productivity, but in quality and maintenance productivity.

[3] I-CASE, or integrated-CASE, refers to those tools which automate the full systems development life cycle; upper-CASE refers to those tools which help automate systems analysis and design efforts; and lower-CASE generally refers to code generators.

[4] Where a traditional system is composed of programs that define procedures and use data, an object-oriented system is composed of self-contained objects, containing both procedures and data, that send messages to each other.

The software package option in particular is becoming increasingly popular. Software package purchase presents an intuitively appealing option: why build a system from scratch when you can buy a fully working system at lower cost and just change those portions not exactly meeting your needs? In purchasing a package, one is, in effect, "reusing" an entire system. And, reusing previously-developed components -- code, models, or entire systems -- should save time and money and provide improved quality.

Purchasing a package should allow an organization to deliver a system faster and cheaper than building it. However, this is often not the reality. In purchasing the software package, the organization is also purchasing the *business processes* which are embedded in it. These business processes may match those existing in the organization, but often do not. The choice is then to either modify the package to fit the organization's business processes, or modify the business processes to fit the package. Either choice is always more difficult, more expensive, and more time consuming than anticipated; in fact, a total installation cost of ten to twenty times the original purchase cost is not unheard of.

Moreover, the fact that the package is difficult to change does not disappear once it is installed; similar to many internally-built systems, it remains difficult to change on an ongoing basis as well. In today's business environment, the flexibility to change the organization, its business processes, *and the information systems which support those business processes* has become critical to an organization's success.

We believe, after discussions with several organizations as noted below, that application templates offer the benefits of the traditional package approach while mitigating the potential downside. Indeed, as we will discuss later, this alternative is essentially a combination, or hybrid, of the others.

III. TEMPLATES

We are using the term "template" to describe a system that has been built with a CASE tool and reused. That is, a template is a fully working system that is used as a model for another fully working system. It can include any or all of the following: data models, process models, screen models, and code. The organization using a template can reuse all or any portion of the system. It can install and run the entire system as is, or it can customize the models to meet its needs, and simply regenerate the code to get a new system. The fact that a template is built in a CASE tool is key: it allows customization of the models rather than the code. The critical point with a template is that, *it is the design that is reused, not the code.*

In order to illustrate this approach, we turn now to a description of one company that has experimented with it, followed by a brief discussion of a second company example.

Canadian Airlines

Headquartered in Calgary, Canadian Airlines (Canadian) was formed in the mid-1980s through a merger of independent airlines. Slightly smaller than Air Canada, its major competitor, it is the world's 19th largest airline, with approximately 16,000 employees and almost $3 billion in revenues. To support its newly-formed business strategy, Canadian developed an IT strategy which included, among other things, the decision to use the Information Engineering methodology and Texas Instruments' CASE tool, IEF. One of the first systems targeted for re-construction was the frequent flyer system, a highly visible and mission-critical system. Transaction volumes had long surpassed the capabilities of the existing system, which was inflexible and required constant and extensive maintenance. More importantly, the system could not keep up with the speed with which the business changed, since each new frequent flyer promotion required extensive changes to the code.

The first step was a three-month definition of the requirements of the new system, using joint application design (JAD) sessions.[5] Once Canadian knew what they needed, they solicited bids for development of the system. The twelve proposals they received in return ran the full gamut in terms of both cost and time to complete.

Canadian decided to purchase TWA's frequent flyer system, built using the IEF CASE tool. While it was not the lowest cost option, they felt it could offer the best value in terms of demonstrated quality and time to deliver. The purchase price included ten days of on-site support from TWA and ten days of customer support from TI for the tool. What exactly did they receive? They received a handful of floppy diskettes, which contained (in Information Engineering parlance) a BAA and a BSD.[6] They received no binders or documents; the documentation was on the diskettes. While the code was included with the system (it was a fully working system), they used it as a device solely to ensure that they had in fact received the entire system; after the initial run, they never used it again. The on-site support offered by TWA was originally contracted to cover any issues that Canadian might have in understanding the functionality and/or business rules imbedded in the system. Of the ten days of TWA support, Canadian used only one; Canadian was able to easily understand the business functionality of the system through the template.

Canadian then went to work enhancing and customizing the system to its requirements,[7] with seven IS people and three users. The users were trained in key aspects of the CASE tool and methodology. The system was considered to be extremely large and

---

[5] In a joint application design session, "users and IS developers work together in a structured workshop led by a trained facilitator to complete information systems delivery tasks and activities" (e.g., requirements definition for a new system) (Davidson, 1992).

[6] Business area analysis (BAA) and business system design (BSD) are stages two and three of the seven stages of the information engineering methodology. During the BAA stage, analysts develop a conceptual model of a business area. During the BSD stage, designers develop a high-level design of a business system within a particular business area (Texas Instruments, 1990).

[7] The changes they made were fairly extensive including, among other things, adding bilingual capabilities.

high profile, serving their most valued customers, the frequent flyers. As they explained, information goes "... direct from system to customer without any intervention ... if it's incorrect, it hits the customer directly."

The development team completed the system within the ten months they had promised to management, despite what could have been a major snag in the seventh month. At that point, senior management made a business decision which required major changes in the very structure of the system. The frequent flyer program -- and system -- had, in the past, been separate from the lounge program and system.[8] The customer qualified for each program separately, carried separate cards, and changed privilege levels in each independently of the other. Canadian realized that while this may have made sense from an operational standpoint, it was inconvenient and complicated from the customers' perspective. In month seven of development, the decision was made to go to one card, and therefore to one system. The implications were enormous: all the business rules for qualifying, measuring, and changing privilege levels changed dramatically, and therefore the processes and data in the systems changed as well. According to Canadian, a conservative estimate for how long this change would have taken in a traditional system was six to nine months. They were able to do it in one month, and deliver the new, enhanced system in the ten-month time frame they had promised for the frequent flyer system alone.

Benefits

In buying a template from TWA, Canadian bought two major benefits: 1) a system model, and 2) a robust prototype. The model, which provides both the business rules and the technical design approach, includes the data and process models, and the screens. As Canadian explained it, in the past when you bought a package "you were always buying [a business model], but I don't think you were aware of it; you thought you were buying the code. [When you buy a CASE-based] model from another company, you're very aware that

[8] The lounge program allows members to use Canadian's airport lounges.

what you're buying is their business area analysis." More importantly, in buying another company's business rules, Canadian found better ways of doing business, ways they had not previously considered. In addition to this business information, Canadian acquired technical information as well. Through the template, they bought a system design which was infinitely better than any other they had seen, and that was easier to understand than if it were in a traditional package. For example, in the TWA template system, the rules for frequent flyer promotions were separated from the body of the system. This streamlined design enabled users to implement new frequent flyer promotions themselves, rather than requiring IS to make the changes for them. (According to Canadian, new promotions sometimes take place weekly.)

At the same time, Canadian bought a working prototype. Instead of starting off with the results of a requirements definition in three-ring binders, they started off with a working system which the users could "see ... touch ... and feel..." Seeing a system that actually worked and needed only to be adjusted to Canadian's requirements, it was "not as great a leap of faith," as it had been in the past, for user personnel to believe that the system could be delivered in the time frame promised. Furthermore, because the system was built in a CASE tool and involved only the customization of the business rules and data to be used (with little or no coding involved), the users could then immediately sit down and work with the *business* model and make the necessary changes jointly with the IS team. That is, as Canadian systems people note, this was not a case of "building a prototype first and then building the system ... [this was a case of] developing the system using a prototyping iterative approach." Because the burden of writing code is eliminated, the developer is not averse to continual iteration. And, because the users can see that changing the system is easier and faster than it had been in their past experience, they are more inclined to work with the developers. It is through this iterative process that users and IS can begin to work together to build the trust which is a critical component of the partnership needed to develop and continually change systems in a business environment of continuous change.

It is important to note also that those factors that facilitate customization of the system upon initial implementation also facilitate ongoing customization over time -- or "maintenance." According to Canadian, they have two categories of maintenance: support and enhancements.[9] With the new system, support has been dramatically reduced and enhancements are significantly easier to implement, both because of a streamlined, more modular design and because the system resides in a CASE tool.[10] There is direct business leverage to be gained from the ability to enrich the system. According to Canadian, the *business* units are now leading the way in enhancing the system because their perception is that it can be done in a reasonable time frame, at a reasonable cost and *is therefore worth the investment.*

Placing this in the context of the traditional "build versus buy" decision: buying a template is better than building from scratch, for all the reasons that buying a package is such an attractive option: you *start* with a working system. But it is also better than a traditional package because it can more easily be changed. In essence, *a template is a flexible package.*

Midwest Savings and Loan

A second company, Midwest Savings and Loan (not its real name), found many of the same benefits from templates that Canadian did. Midwest is a $1 billion savings and loan with approximately sixty branches and an IS staff of forty, half of whom are applications developers. Management's primary goal in purchasing CASE tools was to reduce its maintenance backlog. After trying a few tools, they settled on Texas Instruments' IEF CASE tool. Following the completion of a requirements analysis, they purchased TI's

---

[9] Support includes: 1) "fixing it when it breaks;" 2) changing the system in order to implement new frequent flyer promotions; and 3) changing the system to reflect regulatory changes.

[10] CA has allocated 1.5 maintenance personnel to this application; this compares to 7 individuals on a system comparable in size, but residing in a different technology.

General Ledger template for their first IEF-based application. As they describe it, this turned out to be an effective introduction to and training on the tool itself. More significantly, they delivered the system in three months.

In addition to providing systems that one can "see and touch" (prototypes), Midwest believes that templates serve to mitigate what they see as "the downside of CASE." That is, using the Information Engineering methodology and CASE tool results in more time at the front end in analysis and design than was typical in the past. In Midwest's view, one of the problems with longer analysis and design phases is that IS has nothing to show the users during those phases. With the template, they had something to show immediately. Further, the users could try it out themselves and "tweak" it.[11] Like Canadian, Midwest found business features in the template which they had not previously considered, and saw this as a benefit.

These are only two company examples, intended here to clarify the concept of templates. Other companies are pursuing this delivery strategy as well, and we continue to look at these and other tool-based templates. Vendors currently in the template market include Andersen Consulting, Synon Corporation, Omnicase, and others.

IV. INTERNAL MODELS -- A TEMPLATE PRECURSOR

To date, the emphasis has been on the purchase of *externally* developed templates or packages. We believe that templates can also be used *within* multi-divisional companies. To illustrate this concept, we turn to MultiCo. MultiCo (not its real name) is a Fortune 100, multi-national industrial firm with approximately thirty business divisions. By the mid-1980s, Division A had twelve manufacturing plants, each with its own systems. Realizing that they could reduce costs if the plants shared inventory, management decided to shift

[11] Of course, as with other prototypes, they had to explain to some users why they had to wait three months rather than being able to have the system immediately!

toward a data-oriented approach and begin to build common systems across the various plants. They built a central data dictionary and defined common data definitions, instituting a policy that all new development should use the common data dictionary. Convinced of the usefulness of data-oriented approaches, the IS group in Division A initiated a strategic data planning effort that produced both data and process models.

Approximately one year later, Division B decided to embark on the development of a strategic systems plan. After considering a number of different options, including the use of consultants to build the plan, they decided to look at the strategic plan developed by Division A. After much consideration, Division B decided that instead of "reinventing the wheel," they would use the data and process models developed by Division A as the basis for their own plan. Division B proceeded to analyze Division A's models, looking for commonalities. Of the 112 business processes defined by Division A's model, Division B added 2, deleted 2, modified[12] 60, and accepted 50 without modification. That is, 110 of the 112 business processes were viewed as generic enough to be used by Division B. A similar analysis was performed on the data entities included in Division A's model, with similar results. In Division B's view, the Division A model provided a 95% fit with its own model, and significantly reduced its own modeling efforts.

MultiCo's corporate data management group has since become the custodians of an increasingly generic process/data model. At this point in time, the generic model is a document that contains the following: data models; process models; information flows; and maps of existing applications on top of the process model. Other divisions within the company have used the generic model, or portions of it, as a starting point or "strawman" for their own efforts. In this way, the generic model has been continually updated, expanded, and utilized for the common good of many divisions.

---

[12] In "modifying," they were generalizing the business process, rather than making it more specific. A simple example of a generalization might be defining the entity "enterprise," rather than defining two entities, "customer" and "vendor."

One division, for example, decided to rethink its purchase order process. A review yielded a current purchase order process which had 40 tasks, took 10 days, and cost $240 per purchase order. They then compared this to the generic model which showed 15 tasks. Simplifying their purchase order process with no addition of automation allowed them to reduce the time to 6 days per purchase order. Further analysis showed that, with the use of some automation, they could reduce the processing time to one day at a cost of $30 per purchase order.

There are certainly differences between what MultiCo is doing with its generic model and what Canadian and Midwest have done in purchasing templates. First, MultiCo's current goals in using the generic model are to help rethink business processes, and determine what systems will be needed to support those business processes. While it is true that the components of the generic model can eventually be used to define particular systems, the immediate goal in using the model is not to necessarily build a system. In contrast, the immediate goals for both Canadian and Midwest were to deliver particular systems. A second difference is the storage medium: MultiCo's generic model is on paper; it is not built in a CASE tool.

However, while the goals and technology may be different, the underlying concept is much the same. All three companies are, in effect, *reusing models*. Canadian and Midwest are taking a set of process and data models developed externally by another company for a particular function and applying it to that same function within their own companies. At MultiCo, a set of process and data models developed internally in one part of the company is being used in other parts of the company. The generic model allows one business area to leverage the work performed in other business areas, offering in essence a "template" which can be modified. According to MultiCo, this approach saves time, increases the quality of the end product and provides a common frame of reference and language.

There is another common theme underlying the two, seemingly dissimilar, approaches discussed here: in order to effectively reuse system components, whether they are externally or internally developed, it is essential to understand the similarities and differences between a target business process and the existing process model. In using its generic model for a particular process in a new division, MultiCo must determine where the areas of commonality lie between that process and those documented in the generic model, and what parts are unique. The same holds true for Canadian and Midwest. In both of these cases, there was some core functionality which could be used unchanged, representing those parts of the process that were similar, and other parts of their process which were unique and which therefore required changes in the system. The tendency in most companies is for each business area to believe that its business processes are unique; this tendency arises from multiple factors including the culture and age of a given company, as well as independent and autonomous business groups. In fact, there is more commonality than is typically understood or admitted.

As noted earlier, MultiCo's generic model is not built using a CASE tool. Building the model using a full life-cycle CASE tool would clearly offer several important benefits. In effect, each division would have the two major benefits offered by a template, as discussed above: a business model and a working prototype. The tool would facilitate the customization of the generic model, as well as the delivery of the systems defined by it. We are currently tracking a number of organizations who might be taking this approach.

V. PACKAGES OF THE FUTURE -- THE NEW BREED

Current Market

The market for templates is currently in its infancy, in terms of the types of companies that are buying and selling them, the types of templates being offered, and the environments (hardware, operating software, and CASE tools) in which they can be run. To date, the types of companies that have been involved in this market include:

- *CASE tool vendors:* Some of the CASE tool vendors are beginning to enter this market in a "broker" capacity. For example, Texas Instruments will offer for sale a template built by one of its customers to other customers;

- *Software package vendors:* Some of the software package vendors are beginning to offer templates, although this appears to be limited at this time to the smaller vendors and/or those vendors who service a specific niche of the market;

- *Industry consortiums:* In certain industries (for example, airline, electric, and retail) consortiums are emerging, in which templates will be built and exchanged among a group of companies with similar needs and interests;

- *Companies who sell their CASE-based systems to other companies:* There is some direct interaction between companies, as between TWA and Canadian; and

- *Template vendors:* New companies are entering the market specifically to sell templates, as well as to offer other template services and cross-CASE tool bridges.

The types of templates which are currently offered vary across a wide range of business applications as well as technology models. Business applications include, for example, financial services, manufacturing and distribution, hospital systems, utilities, and general financial applications such as general ledger, accounts receivable, accounts payable, etc. There are also templates available specifically for technical functions, e.g., data access and screen and report templates.

In terms of environments in which these templates will operate: a number of suppliers offer templates for the AS400 market with the remainder emphasizing IBM mainframe as well as some VAX and PC environments. Finally, there are a number of CASE tools within which templates are currently offered, both I-CASE as well as lower case.[13]

---

[13] For a discussion of current vendors and their templates, see for example: *CASE Strategies* (1992) and *Yankee Watch* (1992).

Market Perspectives

As noted earlier, there are a number of current and potential players in this market, each with a different perspective. This section briefly examines these varying perspectives.[14]

Company buyers -- From the perspective of a company buyer, it is clear that templates should be a critical component of a systems delivery strategy. Combining the advantages of a traditional package and that of a prototype, a template:

- allows a user to see a fully working system on day one;
- includes models of both the business and the technical design with ideas and an approach which may not have been previously considered;
- serves as a base from which to start;
- offers relative flexibility and ease of expansion and enhancement;
- can be less expensive than development; and
- improves time to market.

In short, templates offer the ability to significantly improve productivity and quality of "development" and "maintenance."

At the same time, there are some issues to consider. A template is a system built in a CASE tool. In order to take full advantage of this fact, a company that purchases a template and wants to customize it should be making changes to the models, not to the code itself. This means that the company must have -- and know how to use -- not only the template but also the underlying CASE tool. For some companies, such as Midwest, this can be a very useful means by which to introduce the tool. For other companies it may not be as useful, particularly if a company has already chosen another tool.[15] In any case, the

---

[14] In addition to the two companies who have bought templates (one of whom has also sold its template), we spoke with four software package vendors, two CASE tool vendors, and one custom software vendor.

[15] That is, the goal is not to have multiple CASE tools which are redundant in functionality, or tools which do not overlap in functionality but which also are not connected to each other. This would only increase complexity. This will become less of an issue when and if cross-CASE bridges become available.

decision to purchase a template implies a decision regarding CASE which should be made in the context of the specific business and development environment. This can have major implications (Rockart and Hofman, 1992).

Some of these implications have to do with CASE tools themselves and are not new. For example, training costs, both in terms of time and money, can be high for these tools. In addition, the use of CASE tools can require major changes in the IS organization in terms of skills, roles, and responsibilities, user involvement, management and measurement processes, etc.[16]

Software package vendors -- Some of the issues faced by software package vendors considering the template market might include:

- *Intellectual property rights:* As discussed earlier, both the business rules and the technical model are more obvious in a template than in a traditional software package. For some software package vendors, the technical model is considered proprietary product information with which they are reluctant to part. For other software package vendors, however, this is not perceived to be a barrier to entry. They believe that those competitors who truly want to understand the technical design are able to do so with a traditional package as well.

- *Maintenance:* Similar to traditional packages, customers will have to "recustomize" their systems whenever they acquire a new release of the template system, but this should be easier than was the case with traditional packages. A practical approach might be for the vendor to design the template system in such a way that there are two segments: a core that cannot be changed by customers, and a second portion that can be changed.

- *Other:* Another issue from the perspective of the software package vendors is the choice of CASE tool(s) in which to build their systems. Vendors who sell packages for specific

[16] For discussion of the change associated with the introduction of CASE tools, see for example: Orlikowski (1991), Orlikowski and Friesen (1989), Orlikowski (1989), Rockart and Hofman (1992), and Chen and Norman (1992).

industries can target the CASE tool which is most in use in that industry, if possible. For horizontal applications, the decision is not as clear cut.

According to one of the software package vendors with whom we spoke, the reasons to get into the template market are greater than the reasons not to. In producing a traditional package, the package vendor first produces a beta version of a new system, which is tested by a select group of customers. These customers suggest changes, providing feedback considered critical to the ability of a vendor to provide a good product. The vendor evaluates the suggested changes, and incorporates *only* those perceived to be useful to a wide customer base. In contrast to a traditional package, a template allows the vendor to satisfy a greater portion of customer needs. The vendor can still incorporate into the product that feedback which is useful to the wider customer base, but can also provide each customer with the ability to more easily add functionality unique to its own business operation. That is, the use of templates allows the vendor to keep more customers happy and, in the words of this package vendor, "Happy customers buy more software!"

Custom software vendors -- For most of the large custom software vendors, the current products are primarily custom built systems -- i.e., systems built from scratch -- as well as some package installation. A market in which there is a significant demand for templates represents a major shift in focus for these vendors.

CASE tool vendors -- From the perspective of the CASE tool vendors, the analogy to the early PC market is striking. It is clearly very important to them to have as many templates as possible available for their CASE tool. It is not clear, however, that developing these templates is in their best interests. First, building application software is not their core business; and second, some of their customers are package vendors, with whom they are reluctant to compete. Alternatively, they can foster and encourage the development of templates by other entities. The major issue for the CASE tool vendors is

the time it will take to build strategic alliances and encourage the availability of templates on the market for their CASE tools.

Companies who sell their CASE-based systems to other companies -- The benefits for these sellers are relatively straightforward: they can recover some, if not all or more, of the costs of building the system. Selling the template indirectly through a third-party mitigates the potential cost of sales (packaging, support,[17] etc.). The issues faced by these companies are similar to that of the package vendors: they may be reluctant to give away business rules which they consider to be a source of competitive advantage. Clearly, this applies to some systems more than others; for example, depending on the company, this may be more of an issue for a marketing system than for an accounting system. However, even for "strategic" systems, this may not be as much of an issue as it appears to be at first glance. In the view of Canadian, for example, the true advantage lies not in the system itself but in the way in which it is used. Moreover, in Canadian's view, the advantages of recovering some of the cost of the system far outweigh the disadvantages, since their "competition will eventually catch up anyway."

Implications and Future Market

As discussed above, adopting a template approach implies some significant changes for the systems delivery process. Both Canadian and Midwest cited increased user involvement, ability to spend more time on the business process and less time on the technical aspects, and improved time to market. The template approach illustrated in the MultiCo example -- i.e., using templates as "internal packages" -- also has major implications for the systems development process. The ability to reuse models and/or systems across divisions within an organization is something which must be built into the design of those models; it requires a different analysis and design process than that used by the majority of companies today. A system which has been designed for reuse is a modular, streamlined,

[17] Although it should be noted that in the examples provided above, the support needed was minimal.

well-engineered system which leverages the commonalities across its composite functions.[18] Further, the use of such a system for multiple instances -- that is, the reuse of the system -- requires a significant shift in the mindset and reward system of many developers today.

Templates represent a major transformation in the software application market, blurring the boundaries among package vendors, custom development vendors, and CASE tool vendors. It is unclear exactly what form this transformation will take. However, two things are clear: the software business will change dramatically, and the template market will pave the way for new companies to emerge on the scene.

Returning for a moment to the multiple alternatives discussed in Section II -- CASE, object-orientation, packages, and templates. In effect, we are seeing these alternatives merge into one: templates, as they currently exist, are a hybrid of packages and CASE tools. Over the next 5-10 years, we will have template-based packages relying on CASE tools with object-oriented components. Templates, it may be suggested, are the packages of the future.

---

[18] For a discussion of this, see, for example: Hess (1990).

REFERENCES

CASE Strategies newsletter (4:1), Arlington, MA, January 1992.

Chen, M. and Norman, R.J. "Integrated Computer-Aided Software Engineering (CASE): Adoption, Implementations, and Impacts" *IEEE*, 1992, 0073-1129, January 1992, pp. 362-373.

Davidson, E.J. "An Exploratory Study of Joint Application Design in Information Systems Delivery," MIT Sloan School of Management unpublished manuscript, October 1992, p. 2.

Fichman, R.G. and Kemerer, C.F. "Adoption of Software Engineering Process Innovations: The Case of Object-Orientation," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 242, Cambridge, MA, June 1992.

Fichman, R.G. and Kemerer, C.F. "Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 230, Cambridge, MA, June 1991.

Friesen, M.E. and Orlikowski, W.J. "Assimilating CASE Tools in Organizations: An Empirical Study of the Process and Context of CASE Tools," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 199, Cambridge, MA, October 1989.

Hess, Milton S. "Information Systems Design in Industrial Practice," in *Concise Encyclopedia of Information Processing in Systems and Organizations*, ed. A.P. Sage, Pergamon Press, May 1990, pp. 1-12.

Information Engineering Facility Methodology Overview, TI Part Number 2739900-8024, Texas Instruments, January 1990, p. 3.

Kemerer, C.F. "Learning Curve Models for Integrated CASE Tool Management," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 231, Cambridge, MA, November 1991.

Loh, L. and Venkatraman, N. "Diffusion of Information Technology Outsourcing: Influence Sources and The Kodak Effect," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 245, Cambridge, MA, October 1992.

Orlikowski, W.J. "Division Among the Ranks: The Social Implications of CASE Tools for System Developers," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 194, Cambridge, MA, July 1989.

Orlikowski, W.J. "Radical and Incremental Innovations in Systems Development: An Empirical Investigation of CASE Tools," MIT Sloan School of Management, Center for Information Systems Research, Working Paper No. 221, Cambridge, MA, April 1991.

Rockart, J.F. and Hofman, J.D. "Systems Delivery: Evolving New Strategies," *Sloan Management Review* (33:4), September 1992, pp. 21-31.

"The CASE is Not Closed," *Yankee Watch Management Strategies: Outsourcing* white paper (2:7), The Yankee Group, Boston, June 1992.