APPLICATION TEMPLATES: 
FASTER, BETTER, AND 
CHEAPER SYSTEMS

J. Debra Hofman
John F. Rockart

July 1994

CISR WP No. 270
Sloan WP No. 3708

©1994 J.D. Hofman, J.F. Rockart

Center for Information Systems Research
Sloan School of Management
Massachusetts Institute of Technology
APPLICATION TEMPLATES: FASTER, BETTER, AND CHEAPER SYSTEMS
J. Debra Hofman and John F. Rockart

Abstract

Organizations today are undergoing massive transformations in the way they are structured, managed, and operated. The ability to develop and change their information systems quickly and often is increasingly important. Two primary approaches to systems development have existed to date: build or buy. Our research over the past two years suggests that a third alternative is emerging which can enable organizations to both develop and change systems faster. This is a "template" approach, which combines most of the best aspects of the other two. Templates are existing systems, built with the aid of "CASE" tools, which are being changed at the design level, and thereby customized for a new organization's use.

This paper discusses the use of the template approach by three companies, as well as the rapidly growing template marketplace. All three organizations cited significant reductions in the time and cost of delivering their systems, as well as improvements in IS-business relationships and the ability to learn new business methods and technologies. At the same time, there are some issues in pursuing a template approach today, including supply, CASE tool acceptance, and competition from other approaches. The template market is currently in its infancy, but is becoming increasingly active as software package vendors, tool vendors, and custom software consultants begin offering templates or announce plans to do so. From what we have seen, the template alternative clearly warrants attention today.
I. A NEED FOR FASTER SYSTEMS DEVELOPMENT

In response to increasingly heated global competition, organizations today are undergoing massive transformations in the way they are structured, managed, and operated. Focusing their attention on their customers, many organizations are redesigning business processes to more effectively deliver products and services. The effective use of information technology to support these processes has, for most, become critical.

Transformation, however, is not a one-time event. Organizations of today and tomorrow must be flexible and nimble enough to redefine themselves easily and often. Unfortunately, an inability to develop or change their information systems quickly enough is proving to be a serious bottleneck to widespread and ongoing change.

Two primary approaches to systems development have existed to date: build or buy. Our research over the past two years suggests that a third alternative is emerging which can enable organizations to both develop and change systems faster. This is a "template" approach, which combines most of the best aspects of the other two. Templates are existing systems, built with the aid of CASE\(^1\) tools, which are being changed at the design level, and thereby customized for a new organization's use.

The use of the template approach by three organizations is discussed in this paper. All of these organizations significantly reduced both the time and cost of delivering their systems. In addition, the template approach has:

- Enabled a major airline to improve its ability to serve frequent flyers by utilizing several new business methods which they learned from the template they purchased:

---

\(^{1}\) Computer Aided Software Engineering (CASE) tools are computer-based tools which help developers build systems: essentially, they are CAD/CAM for systems developers.
- Provided a publishing company with the ability to leverage best practices, applications, knowledge, and expertise across its international divisions; and
- Allowed a midwestern utility to dramatically improve existing relationships between the IS organization and the business, to quickly learn an important new technology that it needed, and to accomplish a merger more quickly and easily than would have been possible in the past.

We turn now to a brief discussion of the build and buy alternatives, followed by a description of the template approach and the three companies utilizing it. We then summarize the pros and cons of this approach and provide a brief outline of the rapidly growing template marketplace.

II. THE TRADITIONAL ALTERNATIVES: BUILD OR BUY?

Until recently, organizations have had to either build or buy new systems. In business terms, the traditional method of building and maintaining a custom system involves a series of fairly well-defined steps. (See Exhibit 1.) Given the need for a new system, developers define the functions the system will be required to perform; analyze detailed user requirements; design the system to satisfy these requirements; develop the system by writing the code; test the system; and, finally, implement it. This "build" cycle essentially takes a series of business tasks defined in English, and refines it in progressively greater levels of detail into a language the computer can understand, i.e., the code. Once the system is in use, changes in the business require changes, or maintenance, to the system.

---

2 We recognize that this is a simplification of the "build" or development process, and that there are many different methodologies, tools, and techniques, some of which are mentioned later in this section. The point here is that, while there are clearly many variations, at a higher level there is also some basic commonality. The steps outlined here are performed, either in sequence or iteratively, regardless of methodology.
Define high-level functions 1

Analyze detailed requirements 2

Design system 3

Develop system -- code 4

Test system 5

Implement system 6

Maintain system 7
Until the mid-1980s, when Computer-Aided Software Engineering (CASE) tools were introduced, this entire process was completely manual. Programmers took the specifications from the design stage and wrote the program code line by line. Today, in a major variation of traditional custom development, a number of organizations are utilizing CASE tools to help automate the process. These tools assist developers in producing a computer-based design of the new system, expressed in diagrams and/or text. This design outlines the data relationships, the process flows, the screens, and the programs. The diagrammatic and textual representations are referred to as models. Most CASE tools also provide the ability to then convert the design automatically into code. Minimal, if any, hand-coding is necessary, thus eliminating a major step (step 4 in Exhibit 1). Maintenance is also greatly simplified (step 7 in Exhibit 1). Rather than changing the code to mirror a change in the business, the change can be made at the design (model) level and the code can be automatically regenerated.

Exhibit 2 summarizes the advantages of CASE. However, while the use of CASE tools has steadily increased, it has not increased at the rate originally forecast. Some of the reasons for this slow growth rate include the cost of the CASE software and training, a significant learning curve, and the difficulties associated with making the necessary changes that are required within the IS organization.³

---

<table>
<thead>
<tr>
<th>Tool support provided to designer/maintainer</th>
<th>Trad'l</th>
<th>CASE</th>
<th>Rapid Dev'1</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code automatically generated</td>
<td></td>
<td>X</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Maintain system at design level</td>
<td></td>
<td>X</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Extensive interaction w/user facilitated</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Detailed development of req'ts not necessary</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Start with pre-existing system—provides capability to define scope, reduce time/cost</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>External ideas/knowledge</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

✓ = Rapid Development with CASE support

Exhibit 2
Advantages of CASE, Rapid Development, and Packages
A second major variation on the traditional custom development approach can be referred to as Rapid Development. With this approach, the proposed system is divided into segments and delivered incrementally. Time to delivery for each segment is short, typically 3–6 months. Each segment is a fully functioning system. Rather than moving sequentially through the steps of the development process (Exhibit 1), work is accomplished in an iterative fashion for each segment. With the first iteration, a prototype is built. With each successive iteration, changes are made to the prototype until it becomes the final system. IS and business users work closely together, using the prototype to help specify the detailed requirements for each segment, rather than documenting the detailed requirements for the entire system at the beginning of the project, thus eliminating or reducing a step in the development process (step 2 in Exhibit 1).

CASE or other tools are often used in combination with Rapid Development techniques. The primary advantages of Rapid Development are summarized in Exhibit 2. Many organizations report, as a result of using Rapid Development, a greater level of involvement in the process on the part of business users, greater responsiveness to business needs, and quicker delivery of fully functional applications.

The benefits of building a custom system are fairly obvious: the resulting system should exactly match the specific requirements of the organization. But there are some

---

4 We are using the term “Rapid Development” to refer to a collection of techniques (briefly described here) which have been introduced, under various labels, to improve the systems development process. For further discussion of these techniques, see for example: Arthur, 1992; Hough, 1993; Martin, 1991; Naumann and Jenkins, 1982; and Orman, 1988-89. In addition, there are other variations and improvements on traditional custom development. For example, object-orientation is a relatively recent software process innovation which holds great promise for improving the time, cost, and quality of the application development process. However, object-orientation represents a radical change in the systems development process and there are few business systems to date which have been developed using it. For discussion of the adoption of object-oriented methods, see Cockburn, 1993 and Fichman and Kemerer, 1992 and 1993. Reuse, and code reuse in particular, is an innovation which has a longer history than object-orientation; however, while some individual programmers do reuse code on an ad-hoc basis, it has generally not been institutionalized. The reasons for this have been widely debated, and range from technical to cultural. For discussion of the issues around reuse, see for example: Biggerstaff and Richter, 1987; Caldiera and Basili, 1991; Cusumano, 1991; and Karimi, 1990.
major flaws. First, even with CASE and Rapid Development, time to delivery of the entire system is often too long to meet the business need, especially in today's business environment. Second, the cost of developing a system tailored to the particular organization and the way it operates can be extremely high. Third, the system is often based solely on the organization's internal view of its business requirements, a perspective which can be limited. Fourth, with the exception of those projects using Rapid Development techniques, involvement and input from the key players—those who will be using the system—is too often minimal. Finally, with the exception of those systems built with CASE tools, changing the system once it is in use to reflect changes in the business is notoriously difficult, requiring developers to navigate through the code itself.

Increasingly frustrated with the custom development approach for the above reasons, more and more companies are turning to the second "traditional" alternative and are buying software packages. It seems like a good idea. After all, why build when you can buy a fully working system at lower cost and just change those portions which do not exactly meet your needs? Purchasing a package provides an external perspective and new ideas about the business process, and should allow an organization to deliver a system faster and cheaper than building it. (Exhibit 2.) However, this is often not the reality. Most companies find that they need to modify the package—often quite extensively—to fit their organization's needs. Modifying a package generally means extensive changes or additions to the package code, an often long and arduous task. Our research suggests that, by the time the package is actually installed and running, most corporations realize that it has been more difficult, more expensive, and more time consuming than anticipated. In fact, total installation costs of ten times the original purchase cost are not unheard of.

---

5 While there are some packages which can be modified somewhat through the use of tables rather than changes to the code, changing the tables remains a very involved and lengthy process. The alternative to modifying the package to fit the organization is to modify the organization's business process workflow to fit the package. However, this too can be extremely difficult and costly.
III. A Third Alternative

A third major alternative which has emerged recently is the template approach, which essentially combines favorable aspects of both “buy” and “build.” A company using a template approach takes a system which has been built with a CASE tool, and customizes the system to meet its requirements by making changes at the design level. Depending on the particular CASE tool the company is using, the code can then be automatically generated from the changed design. *What such a company is doing is using an existing system as a “template” for their new system.*

What is a template? Any system built with a CASE tool has the potential to be a *template.* But what actually makes it a template is the way in which it is used. That is, a CASE-built system becomes a template when a new company or division customizes it at the design level to meet a new set of requirements. The key characteristics of a template are: 1) it is an existing system, 2) it has been built with a CASE tool, and 3) changes are being made at the design level.

In contrast to the traditional “build” approach, regardless of whether CASE and/or Rapid Development are used, the template approach starts with a pre-existing system, or portion of a system. In this sense, it is similar to the “buy” approach; an organization which buys a package is also starting with a pre-existing system which it then typically modifies. However, modifying a traditional package involves understanding and changing the code itself, since most packages do not contain automated and easily-displayed design models. In contrast, with the template approach, modifications can be made at the design level, and they are facilitated by the CASE tool. *With a template, it is the design which is customized, rather than the code.*

To understand this in a more general context, consider as an analogy the development of an automobile. A manufacturer could design and build the automobile
completely "from scratch"; in software, this would be analogous to building a custom system. A second option would be to buy an existing automobile, and physically alter it to the desired shape and form; this might include replacing certain parts, possibly adding some chrome, and/or restructuring the fenders to get a different curvature. This option is essentially similar to buying a software package and changing it to meet the organization’s needs. A third option for the automobile manufacturer is the following scenario: take a computer-based (CAD) representation of another car—one either made internally or by another manufacturer—which has some of the desired features, look, and feel; alter the representation, changing the curvature, and electronically adding some features; then, "push a button" to automatically generate the code to operate the numerically controlled tools which will build the new automobile. This third option is comparable to an application template.

The three company examples outlined below illustrate three different approaches to using templates. While each company's story is unique, they all use some form of a template. They all cite significant cost and time savings, more effective partnership between IS and business users, more targeted systems that better address business needs, and more flexibility to allow business change. In effect, these companies believe that templates combine many of the benefits of both the custom and package alternatives in a single solution.

IV. THREE TEMPLATES CASES

In the course of studying this trend, we have spoken with and visited both companies who are using templates and vendors who sell them. The three companies discussed below provide a sample of template users. At each site, interviews were conducted with two to six members of the implementation team, representing senior IS executives and IS developers.
A. 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. As part of its newly-formed business and IT strategies, Canadian decided to re-engineer its frequent flyer system, a highly visible and mission-critical system used to service the airline’s most valued customers, its frequent flyers. The original system had been designed to handle 100,000 customers, but was now required to handle more than one million customers. As such, transaction volumes had long exceeded the capabilities of the existing system, which was inflexible and required constant and extensive maintenance. 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.

Canadian solicited bids for development of a new system, receiving twelve proposals which varied widely in terms of both cost and time to complete. They decided to purchase TWA’s frequent flyer system, built using Texas Instruments’ IEF CASE tool. The template consisted of a handful of floppy diskettes that contained a fully working system. Canadian received no binders or documents; the documentation was on the diskettes. While the code was included with the system, Canadian used it as a device solely to ensure that they had in fact received the entire system; after the initial “check” run, they never used it again.

While sharing a mission-critical application with a competitor could be perceived as a disadvantage in some industries, this does not seem to be the case in the airline industry. Since the time of this case study, Canadian has resold the template to Lufthansa. The IS executive at Canadian explained that 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 the “competition will eventually catch up anyway.”
Canadian customized the system to its requirements, with a team of seven IS and three business people. The changes they made to the system were fairly extensive including the addition of bilingual capabilities, changes in the way upgrades were handled, and changes in point and award accumulation. Business users were trained in key aspects of the CASE tool and methodology. Changes were made to the design models, and the code was regenerated by the tool. The team was able to easily understand the business functionality of the system from the design models in the template, requiring minimal support from TWA.

The development team completed the system within the ten months they had promised to management, despite a major business change in the seventh month which required extensive modification of the system. At that point, senior management made a business decision that 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. 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. The decision was made to go to one card, and therefore to one system. The implications were enormous: all the 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 development project was six to nine months. They were able to do it in one month, and to deliver the new, enhanced system in the ten-month time frame they had promised for the frequent flyer system alone.

7 The lounge program allows members to use Canadian’s airport lounges.
Canadian: Benefits

Based on their original estimate that a custom solution would require approximately eighteen months, Canadian believes they saved eight months with the template approach. At the same time, they believe that using the template approach offered additional, though less quantifiable, benefits.

The design models which came with the system contained both business as well as technical information. In buying another company's approach to the business, Canadian found better ways of operating a frequent flyer program, ways they had not previously considered. While a package would have contained this information as well, it would have been buried in the system code. As Canadian explained, in buying a package in the past, "you were always buying [this information], but I don't think you were aware of it. You thought you were buying the code. [When you buy a CASE-based] design from another company, you're very aware that what you're buying is their business analysis." In addition to this business information, Canadian acquired technical expertise as well. Through the template, they bought a system design that was easier to understand than if it were in a traditional package, and that was better than any other they had seen. 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 business 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 could use the template as 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 business users could "see ... touch ... and feel." Given 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 business personnel to believe that the system could be delivered in the time frame
promised. More important, because the system was built with a CASE tool and changes could be made at the design level rather than to the code, business users could make the necessary changes jointly with the IS team. As Canadian developers 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 was eliminated, the developers were not averse to continual iteration. And, because the users could see that changing the system was easier and faster than it had been in their past experience, they were more inclined to work with the developers.

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.” Canadian has two categories of maintenance: support and enhancements.8 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.9 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.

B. John Wiley & Sons, Inc.

While Canadian purchased a template externally, our second company developed its own template internally. Wiley is a mid-sized publisher of technical and scientific books headquartered in the Northeast, with worldwide subsidiaries in the U.K., Canada, Australia, and Singapore. Recognizing the leverage which could be gained from a global

---

8 Support and enhancements include: (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.

9 Canadian has allocated 1.5 maintenance personnel to this application; this compares to 7 individuals on a system comparable in size and complexity, but residing in a different technology.
delivery mechanism, senior management decided in 1988 to adopt a common hardware and software platform. They chose IBM's AS400 for the hardware. To help them with systems development, they chose a CASE tool from Synon, Inc., a U.S.-based firm specializing in CASE tools for the AS400.

Realizing that their existing systems had become obsolete and could not adequately handle the increasing complexity of a multi-national firm, management decided to replace, in phases, the entire portfolio of existing systems. For the first phase, they chose to replace the systems which support what they refer to as their core business process: order processing, distribution, warehousing, publishing support, and fulfillment activities. Their first step was the development of a worldwide data model, designed to reflect the needs of multiple country constituencies. As they explained it, while there is some variation among countries, "a book is a book is a book."

With the data model as the base, they developed the system first in Singapore, their smallest office. The project, including learning the Synon CASE tool, was completed in six months with seven developers involved. The implementation team then took the Singapore system and transplanted it as a template first to Australia, and then to the U.K. They are currently implementing it in the U.S., with Canada as the next step. The system has been tailored to each site. In customizing the on-line portion of the system, all changes are applied at the design level in the CASE tool, and the code is regenerated through the tool. Depending on the type of change, business users see the results either...

---

10 There are three major categories of business activity in publishing. The first is book project management, which includes the activity starting with signing an author to write the book, through the point at which a book is in the warehouse. The second major category is the "core process" which, as described, covers all activity from the warehouse out to the customer, and the third is sales and marketing. The core process was chosen as the first candidate for replacement systems because it was considered the most stable of the three, and the requirements could therefore be more accurately defined. Wiley has since begun replacing the systems which support book project management and is using the template process to help redesign the business process.

11 While the Singapore system was easier in the sense that it was smaller, it was also more complex functionally than some of the other sites (multi-currency, etc.).
immediately or the next day. When they are finished in Canada, they plan to turn around and go back the other way, applying some of the beneficial changes that have been made further down the line to those countries in which the system has already been installed.

In each implementation of the template, IS and business personnel work together to confirm the scope and verify the functionality of the new system, and to identify any additional data or processes which might be needed. Development of the new system is then accomplished by changing and adding to the template. Small segments of the system are presented to the business units for verification as they are completed, with IS and business users sitting together and making changes.

The U.S. implementation of the core template is currently underway and provides some interesting insights. There was initial resistance to the concept on the part of both the business and technical (IS) communities in the U.S. As Wiley explained, the reaction from the business side was "... [we] don't want anything to do with it ... they're much smaller than us. It'll never handle our volume." A trip to the U.K. to see the system was organized for eight representatives of the various user departments, from the manager level down. After a week-long review of the system in which they could use the screens and see the ease with which modifications (which previously required programmer code changes) could be made, they decided to go with the template. The system was demonstrated to them by their business peers, not by IS. As Wiley described it, "the main thing with this system is that we're trying to get IS out of it. It is a user system. They're tailoring the screens for their own requirements. It's their system." One business manager who initially resisted the template system has become so convinced of its superiority, he is now marketing it to his business peers in another site.

\[12\] While the data is relatively similar across sites, business processes do vary. For example, payment cycles vary by country. A normal accounts receivable cycle in the U.S. might be 30 or 90 days; in the Far East, it might be 210 days before an initial payment is expected.
Wiley: Benefits

Wiley personnel believe that this approach to systems delivery provides them with significant leverage, allowing them to share—or "reuse"—best practices (both business and IS), applications, knowledge, and expertise. This has some major benefits. First, this approach has allowed them to accomplish two goals simultaneously which had previously been in conflict: they can aggregate data at the firm level (from the common database), while tailoring the business process and system to local needs. Second, smaller sites get greater functionality than they would be able to afford on their own.

Third, they believe they have been able to reduce both time (of development and maintenance) and cost. Comparing the use of the template to custom development, Wiley estimates savings of approximately 30%, assuming moderate modification. They also estimate that purchase and customization of an external package would cost significantly more than customization of their internal template solution. Moreover, they note that they are able to avoid the difficulties of integrating an external package with their internal systems. And, IS has found that "a template provides a useful, and relatively easy, way to learn a CASE tool."

Wiley also believes that their internal template approach has significantly improved the quantity and quality of IS-business interaction. The ability to interact and work with the proposed system allows a deeper level of understanding for business users than would be the case with a textual description. Delivering a constant stream of system segments reinforces this improved confidence. The fact that IS and business users sit together in front of the screen to make changes and the fact that those changes happen with an immediacy previously unseen provides two important benefits: (1) it contributes

---

13 They must still spend some time understanding, reconfirming, retesting, and re-documenting each function. It should be noted, however, that the effort involved in understanding each function is significantly reduced by working at the design level, since it is easier to understand a design model than it is to understand someone else's code.
to the improved level of trust and interaction; and, (2) it enables the business to develop a sense of ownership which is crucial to the success of an implementation. And, while business users are learning more about what it takes to deliver a system, IS is becoming more business literate as well. For IS, this ability to better understand the business is a major benefit of the template approach. As Wiley IS people describe it, “It’s the only way they’ll ever get any faith in us really—if we can prove to them that we understand what they are talking about.”

Finally, starting off with a defined scope was seen as a major benefit. That is, using the template as the starting point essentially provides a clearly defined boundary for the functionality of the proposed system. At the same time, however, they did not feel that this pre-defined boundary represented a constraint, since the system is relatively flexible and easy to change.

C. Western Resources

The product of a merger of several gas and electric utilities in the mid-1980s, Western Resources in Topeka, Kansas is the fifth largest electric and gas utility in the U.S., serving approximately 1.5 million customers. In 1986, the new Chief Information Officer (CIO) decided that the company needed a new, flexible customer processing system which would take them into the next decade. For a utility, the customer processing system is critical, encompassing billing, credit and collection, meter history, transformer history, and general customer service information. Indeed, the monthly billing envelope is considered the primary “communication link” between the company and its customers.

Western examined a range of potential solutions, including off-the-shelf package vendors, custom software vendors, and other utilities. They decided to purchase Andersen Consulting’s Customer/1 DesignWare product, which was the design for a customer processing system that Andersen had developed with another utility.
Specifically, the product consisted of twelve books that contained design models as well as some prototyped screens. In essence, what Western bought was a system design, the CASE tool with which it was built, and consultants who had industry expertise.

With help from Andersen, Western customized the design and completed the system. Many of the changes that Western made have become part of the Customer/1 DesignWare now sold by Andersen. The system was completed in three years, with 104 team members. Top management played a significant and visible role in the implementation in the form of an upper management steering committee comprised of the CIO, CFO, and COO. This steering committee set a clear mandate from the beginning of the project: there would be no modification of the design “without good reason.”

There was also major user involvement throughout the development of the system. The original evaluation team consisted of five people from the business units and two people from IS, and the development team included twenty-one full time business people. Business users learned the CASE tool and participated in portions of the development such as redesigning the screens where necessary. In addition to some initial prototypes, the development team then presented key functional segments of the working system to the business units as they were completed. In these presentations, IS and business people would work together with the screens, making additional changes where necessary.

The ability to quickly modify the system was tested four months after its implementation, as Western merged with another gas and electric utility. Some of the changes were major, reflecting different rate structures, payment plans, and billing practices. They were able to accomplish the merger in 2½ months, including the system modification and data conversion.
Western: Benefits

Western estimates that it saved approximately 12-18 months and $20 million by using DesignWare over a custom solution. They also believe that they have achieved their target of paying for the system in less than three years through head count and other cost reductions.

However, while these time and cost savings are important, Western believes it gained other benefits that are even more significant. The system design provided both business and technical information. On the technical side, the Customer/1 DesignWare helped the IS people learn a new database technology (DB2) and the new CASE tool. As one of the key developers at Western explained, the IS people knew nothing about DB2. If they had been designing from scratch, they would have based the new design on the old system and would not have effectively used the DB2 product; in effect, they would probably have “tried to DB2-ize the old system.” The DesignWare gave them something to start with and learn from, and they could fill in any gaps by talking to the original designers. On the business side, Andersen’s knowledge of “best practice” in customer processing from other utilities was built into the design.

The DesignWare contained business process information that provided new ideas, rather than simply automating the current business process. At the same time, however, the design helped define the scope of the project, and keep it within predefined boundaries. As the CIO explained, “... if we started from scratch ... we would have undoubtedly spent far more time on the design ... the scope would have gotten out of control ... if you don’t start out with a fence surrounding the project, it is everything in the world to everyone. Every single little requirement creeps in.”

Western also believes that it gained some leverage in certain technical aspects of the system. While the CASE tool did not automatically generate code from the design models, it did come with pre-coded “program shells” for some of the technical functions.
common to all programs (e.g., validation edit routines). The shells provided two benefits. First, initial development was faster because some of the work had already been done and the programmers could focus on the business rather than technical functions of the programs. Second, the program shells enforced a level of standardization which, in turn, has made maintenance easier.\(^1\)

Finally, Western emphasized some unexpected, but significant, benefits arising from an increase in business involvement. As described, IS and the business worked together on the development of the system, with business users designing new screens and modifying existing ones using the CASE tool. They also helped to design and implement the training and system tests. An important outcome of this greater involvement was that these users began to better understand the process of systems development, while IS gained an appreciation of what goes on in the field on a daily basis. Moreover, both groups were learning at the same time, focusing together on operational screens rather than on each other's shortcomings, as is too often the case in a traditional systems development effort.

V. DISCUSSION

There are certainly differences among the three company examples. Canadian and Western took a system developed by another company and applied it within their own companies. At Wiley, a system developed internally is being applied to multiple locations within the company. The companies themselves are in different industries, and the systems vary in functionality, size, and complexity. Moreover, the systems were built with different kinds of CASE tools. Canadian's template was built with an integrated,\(^{14}\)

---

\(^{14}\) One difference between this company and the other two examples is in the area of maintenance. In the Canadian and Wiley examples, changes were made to design models rather than the code during both development and maintenance. At Western, changes were made to design models during development; however, once the code had been generated (i.e., during maintenance) subsequent changes were applied to the code itself.
full lifecycle CASE tool, a type of tool which provides automated support for all phases of the systems development lifecycle, from analysis through coding. Wiley’s template was built with a lower-CASE tool, a tool which provides automated support of the design and coding phases. In both cases, the tool would automatically generate code from design-level models. Western’s template, on the other hand, was built with an upper-CASE tool. This type of tool supports the analysis and design phases and therefore helped produce design-level models, but did not automatically generate code from those models.

At the same time, however, there are some basic and important similarities in what these companies are doing, how they are doing it, and in the benefits they note. The benefits are significant. All three companies noted that: (1) they were able to deliver their respective systems faster and at lower cost than had been possible in the past; (2) that these systems were more maintainable, and therefore could better accommodate future change; and (3) that the systems were closer to what the business wanted.

In all three cases, an existing system is serving as a template for a new system, by reusing the design models. These companies are using the design models of the system to understand and learn what is in the existing system; and, the necessary changes are being made to these design models, rather than to the code itself, in all its overwhelming detail. The work is thus being done at a higher level of abstraction than is typical in a traditional systems development effort. In effect, what they are doing could be described as model-based development.\(^{15}\)

There are also some key underlying similarities in the way these companies used the templates. These companies combined the use of a template with the Rapid Development techniques described in Section II. Rather than starting the project with a

\(^{15}\) For discussion of the advantages of working with models, see for example: Curtis, Kellner and Over, 1992; Hess, 1990; Rockart, 1970; and Zachman, 1987.
detailed list of requirements, they began with a set of high-level functional requirements. The template, which satisfied these high-level requirements, provided the basic structure of the new system. IS and business personnel then jointly carved out the details of the new system, segment by segment, through an iterative process of working directly with the screens and changing the template's design-level models. The template, in effect, served as a prototype.

The companies all noted certain characteristics of the template process. In all three of our company examples, the template served as a learning vehicle for both IS and business personnel. For IS, the template provided a useful introduction to the new tools and systems approaches they needed: the CASE tool, some technologies (DB2), and a model technical design. On the business side, the template provided new ideas and business processes. However, templates provide more than simple knowledge transfer; after all, external ideas, knowledge, and expertise can be transferred to an organization through traditional types of packages as well. More importantly, templates provide the capability for both IS and business users to jointly interact with the new system—to understand and make changes easily. It is this joint, hands-on interaction which primarily distinguishes templates from packages. And, it is this interaction which facilitates learning.

In addition to learning more about their own functions, IS and business users learned about each other's as well. Thus, the template served also as a communication vehicle. In effect, it provided a forum in which the two groups could communicate with each other, understand each other's jobs better, and begin to build a partnership and a sense of mutual responsibility for the delivery of the system. Using the screens to visualize and articulate the requirements of the new system, rather than a blank sheet of paper, helped not only to improve the IS-business relationship, but also to improve the system itself.
In all three cases, a sense of user ownership, not simply involvement, was articulated. The ability to interact with, communicate through, and make changes directly to the system helped to foster this. (Exhibit 3 summarizes the advantages of templates).

The template process described here, then, is a process characterized by improved learning, improved IS-business interaction, and user ownership. There are, however, some other elements of this process which offer important benefits as well.

- By starting a project with only the basic requirements rather than with voluminous detail, these companies were more inclined to reuse a system or portion of a system which already existed, rather than automatically taking the “full custom development” route. Several of our interviewees noted that starting with a detailed list of requirements gathered from everyone involved will almost always result in a custom developed system or a heavily modified package, since no existing system could precisely match the several thousand requirements that are typically gathered with this approach.

- Omitting the detailed requirements phase through use of the template helped to contain the scope—and therefore time, cost, and expectations—of the project. One question which arises is, is it possible that starting with a pre-defined scope might be a constraint, potentially limiting creative new ideas and solutions? Our template users acknowledge this, but believe it is more than offset by the availability of creative thought from outside the company which is embedded in the template.

- The incremental nature of the delivery of the system—the fact that business personnel see results more quickly—also helped improve the credibility of IS and the level of trust.

- For a company pursuing an “internal templates” strategy there can be additional benefits: the ability to leverage best practices across the organization, and to share applications, knowledge, and expertise.

Some of the benefits noted are possible with the purchase of an off-the-shelf software package. Others are possible with a custom solution, particularly one which uses CASE tools and Rapid Development techniques. The template solution described here, however, contains the most favorable aspects of both the buy and build alternatives, and thus offers many of the benefits of both.
<table>
<thead>
<tr>
<th></th>
<th>&lt;--------Build--------&gt;</th>
<th>&lt;---Buy--&gt;</th>
<th>Package</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trad'l</td>
<td>CASE</td>
<td>Rapid Dev't</td>
<td></td>
</tr>
<tr>
<td>Tool support provided to designer/maintainer</td>
<td>X</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code automatically generated</td>
<td>X</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintain system at design level</td>
<td>X</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensive interaction w/user facilitated</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed development of req'ts not necessary</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start with pre-existing system—provides capability to define scope, reduce time/cost</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External ideas/knowledge</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adapt system at design level (upon initial development)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

✔️ = Rapid Development with CASE support
VI. ISSUES AND IMPLICATIONS

While the template approach clearly can offer some significant benefits, it is also not a "silver bullet." Potential issues include:

- **Supply:** There is a limited supply of templates available on the market today, although it appears to be growing rapidly. Of course, this issue applies only to those companies which are seeking to purchase templates; it does not prevent a company from pursuing an internal template delivery strategy.

- **CASE tools:** These tools have not been as widely embraced by the IS community as was originally envisioned. A company which does not use them may be reluctant to consider templates; on the other hand, some companies have noted that the template facilitated learning a CASE tool and provided a very effective means by which to introduce the tool into the organization. Those companies which have chosen the CASE tool route and have already settled on a particular vendor's tools are faced with a different issue: currently, they are limited to templates built with that tool. However, this will become less of an issue as cross-CASE bridges are introduced, which will allow interaction between the different CASE tools.

- **Other approaches:** There is competition for the template approach, most notably the object-oriented approach, which is a very different systems development paradigm than any other used to date. However, while object-oriented is very promising, it has not developed to the point at which it can be easily used by the vast majority of corporations for their standard business systems (Fichman and Kemerer, 1993, 1992). Packages, also, are still a viable alternative at this point in time for some systems, particularly those packages which are table-driven and which require relatively few code-level modifications.

Using a template (and a CASE tool) effectively requires changes in skills, roles, responsibilities, and mindset, changes which must be managed.\(^\text{16}\) For a company pursuing an "internal templates" delivery strategy, the skills required to design a system which can be reused from one site to another must be learned. At Wiley, for example, the initial worldwide data model had to be designed in a way that would reflect the

---

\(^{16}\) For discussion of the changes associated with the introduction of CASE tools, see for example: Chen and Norman, 1992; Friesen and Orlikowski, 1989; Orlikowski, 1993; Rockart and Hofman, 1992.
needs of multiple countries. This requires different problem-solving skills than those used in traditional system design. Using templates also requires a shift from the "not invented here" mindset which seems to be typical, even today, of some IS development groups. Some business users, too, tend to believe that their business process is unique and fundamentally different from others, and that, therefore, a previously developed system cannot be used without major modifications. In fact, there is more commonality than is typically understood or admitted.

Finally, the type of change described here requires leadership and active involvement on the part of management in order to be successful. For example, at Western, senior management set forth a clear mandate of "no change in the template without good reason." At Wiley, it was the senior management team who mandated the use of common hardware and software as a key component of their business strategy, thus driving the use of templates.

VII. Future Directions

It is clear that dramatic change from the status quo in information systems is necessary. For years, organizations have been redeveloping essentially similar functionality from business unit to business unit. The economics of this scenario—the huge expenses incurred in taking this approach—are no longer acceptable in many companies.

The template market is currently in its infancy, but is becoming increasingly active. Over the past year, supply has increased steadily as software package vendors, tool vendors, and custom software consultants have begun offering templates or have announced plans to do so. Players in this market to date have included:

17 See, for example: Ricciuti, 1993; CASE Strategies, 1992; and InformationWeek, 1993.
• **CASE tool vendors:** Some CASE tool vendors operate in a “broker” capacity. For example, Texas Instruments offers templates which have either been built internally or by its customers, and KnowledgeWare is investigating similar possibilities for its CASE tool. Oracle Corporation, a $2 billion CASE tool, database and consulting firm, recently announced its entry into the template market and is offering industry-specific pre-developed models, or templates, and consulting expertise. Synon recently entered the market with a set of financial templates.

• **Software package vendors:** Currently, most of the activity in the template market by package vendors seems to be among those vendors who offer packages for the mid-range market, specifically AS400 hardware and the Synon CASE tool. Cantoc, a Toronto firm, and American Software of Atlanta are two such companies.

• **Software consultants:** One of the leading software consultants, Andersen Consulting, offers what it refers to as “DesignWare,” or templates for specific industries.

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

In addition to the above, there is some direct interaction between companies, as between TWA and Canadian. New companies, such as MetaSolv Software in Dallas, are also entering the market to sell templates for a variety of CASE tools, as well as 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.
Clearly, templates are a promising trend in the software industry. The essential concept behind them—working with design-level models which represent the system rather than working with the system code itself—is one which intuitively makes sense, and for which there is successful precedent in other industries. It may well be that, in the software industry, templates are a first step on the road to a radically new IS process in which systems can be easily assembled from multiple predefined components, replacing the notion of system "development" altogether. From what we have seen, the template alternative is clearly one which warrants attention today.
REFERENCES


"Consulting the Oracle," InformationWeek, August 2, 1993, pp. 32-33.


Riccuti, M. "Build Custom Apps At Packaged Prices," *Datamation*, June 1, 1993, pp. 71-72.


Nov 17, 1938

Lib-26-67