v4.4 (11/11/99) of: ejis99.wpd

Packaged Software: Implications of the Differences from Custom Approaches to Software Development

Steve Sawyer
School of Information Sciences and Technology
The Pennsylvania State University
513 Rider I Building
University Park, PA 16801
(814) 865-4450
(814) 865-5604
sawyer@ist.psu.edu

Suggested Running head: *Implications of Packaged Software*

Steve Sawyer is an associate professor at the Pennsylvania State University's School of Information Sciences and Technology where he conducts social informatics research. His current work focuses on the social processes of software development, systems implementation and related organizational changes. Steve earned his doctorate at Boston University and has also had the privilege of serving on the faculty of Syracuse University's School of Information Studies. To date, he has published in journals such as *Computer Personnel, Communications of the ACM, IBM Systems Journal,* and *Information Technology & People*. With co-authors, Rob Kling, Holly Crawford, Howard Rosenbaum and Suzie Weisband, his first book, *Information Technologies in Human Contexts: Learning from Social and Organizational Informatics*, is due out in 2000.

A slightly revised version of the this paper has been published as:

Sawyer, S. (2000) "Packaged software: implications of the differences from custom approaches to software development," *European Journal of Information Systems*, **9**, 47-58.

Packaged Software: Implications of the Differences from Custom Approaches to Software Development

Abstract

This paper contributes to the literature on software development in two ways. First, by discussing the packaged software domain relative to the more commonly studied custom IS domain. Second, in this paper presents I speculate on the implications of these differences between packaged and custom IS development. Regarding the first issue, while the two domains share many commonalities, the differences are also important to understand. To make this clear I discuss the differences at four levels: industry forces, approaches to software development, work culture and development team efforts. At each level, data from three case studies are used to illustrate the differences between the two domains. To frame the speculation, I contend that the differences between packaged software and the traditional, custom, approach to information systems development will be profound for five stakeholder groups: software development organizations, software development teams, software developers, software consumers and for researchers interested in software development.

Packaged Software: Implications of the Differences from Custom Approaches to Software Development

This paper contributes to the literature on software development in two ways. First, in this paper I summarize some of the differences between packaged and custom IS development. Second, I present a set of speculations, with supporting empirical evidence, regarding some of the implications these differences may entail. My contention is that while these two domains share many commonalities (see Cusumano and Smith, 1997), the differences are such that these should be explicated. In particular I argue that the growth of packaged software will alter how most stakeholders think about software and information systems development. For instance, user involvement, a central belief of custom IS development, is not common in packaged software development (Keil and Carmel, 1995; Carmel and Sawyer, 1998).

Following the discussion of what constitutes packaged software, this paper continues in three parts. In part two I outline the empirical basis for this work. In part three I summarize the contemporary knowledge regarding differences between custom IS and packaged software development. In part four I discuss implications of these differences. Appendix A provides an outline of the case study methods. Appendix B provides a short summary of custom IS development best practices.

Packaged Software

Software development is increasingly being done by specialized companies (software vendors) and sales of software via market mechanisms continues to increase (Leebaert, 1995; Quintas, 1994; Voas, 1998; Price-Waterhouse, 1998). For this paper, packaged software (also known as shrink-wrapped, commercial-off-the-shelf (COTS) and commercial software) means all software sold as a tradable product (purchased from a vendor, distributor or store) for all computer platforms including mainframes, work-stations and microcomputers (Carmel, 1997; Carmel and Sawyer, 1998; Andersson and Nilsson, 1996; Klepper and Hartog, 1992; Grudin, 1991). Typically, packaged software is licensed

for use, not sold.

Custom IS are those made by either an organization's internal IS staff or by direct subcontract to a software house (such as Andersen Consulting or EDS). That is, custom IS are made-to-order systems and are typically built for specific users. This definition of custom IS also includes most government work. For example, US Department of Defense (DoD) software development (the focus of much attention by software engineering researchers) is typically custom (albeit contracted) development. Other examples of custom IS include the legacy systems in many medium and most large organizations.

Packaged software came about as a result of an agreement reached between IBM and the United States Department of Justice in the later 1960's to have IBM unbundle software from hardware (Carmel, 1997). By 1995, packaged software sales represented about 20% of the total software market of $77 billion, a 270% increase in market share since 1986 (Leebaert, 1995). By 1998 the market for packaged software grew to $140 billion of the $200 billion software market. Of this $140 billion, 25% of the sales were for system software, leaving application packages to account for $105 billion (OECD, 1998). Relative to the entire computing industry (the largest industry in the US), on its own the packaged software industry would be the US's fifth largest industry. Certainly many of the largest packaged software firms are well-known (e. g., Microsoft, Oracle, IBM) and their products are also commonly known (e.g., Windows, Oracle, DB2).

The largest growth in packaged software are applications (as different from system software) with large packages (enterprise resource planning or ERP) the fastest growing segment. Still, the annual sales of ERP software were $10 Billion in 1998 (or about seven percent of the total sales of packaged software in that year) [1]. Most large packages, such as ERPs, require extensive post-purchase tailoring and draw heavily on the software services market (-------, forthcoming). Software services include the consulting, contracting, integration, training, and support staffing for hardware and software. This market has also

grown extensively in recent years. Forrester Research (1998) reports that this segment has grown at 16% per annum for more than a decade and accounted for $180 billion in 1997, with more than 50% in support of ERP implementations.

The move to purchasing a software application and then tailoring it to meet specific organizational needs reflects a hybrid response to the "buy-versus-build" decision. Further, both custom IS and packaged software differ from embedded software. Embedded software is typically written "to" the hardware (Gal-Oz and Isaacs, 1998). That is, the hardware and software are intertwined in a way that both must be bundled. Examples of this are many video games (based on proprietary hardware), missiles and "computing appliances." (Badami and Chbat, 1998) Of course, as the hybrid form of ERP software purchase and subsequent tailoring show, the boundaries distinguishing these three forms of software are blurred. For clarity, in this paper, we focus on the differences between the more well-known custom IS domain and the packaged software domain. This excludes the growing and important ERP efforts, which deserve more attention than we can devote in this paper.

OUR EMPIRICAL APPROACH

In summarizing and speculating on issues regarding the special characteristics of packaged software development we draw on data from three sources. The first source is the contemporary academic literature. The second source of data are articles drawn from the trade press serials such as *ComputerWorld*, *CIO*, *Datamation* and *Software.* Both sets of literature are used to establish the characteristics of packaged software development and the differences from custom IS software development. To provide empirical support for the contentions we make, for our third source we draw on data from three case studies. Two of the case studies focus on packaged software teams (Team A and Team B) and the third case study is of a custom information systems development (ISD) effort (Team C).

These three case studies provide two simple contrasts (Yin, 1984). The contrast of Teams A and B from C provide a comparison of the issues that differentiate packaged software development efforts from custom ISD. The comparison of Teams A and B allow for some contrast in the approaches taken by packaged software developers.

Two case studies are drawn from one fast-growing portion of the packaged software market: natural language processing (NLP) products. The third case study is of a custom IS NLP-system. effort An NLP product is one that takes commands from a user presented in typical grammar, converts this query into machine format and uses this resulting translation. Thus, NLP products are often front-end systems for text-based data searches. The NLP market represents one of the fastest growing and most knowledge intensive of the various growth areas of packaged software (Wilks, 1996) The market for NLP software is spurred by increases in processor power; the growth of interest in natural language interfaces (due primarily to the explosion of the world wide web's popularity) and the increased sophistication of the available NLP technologies (Wilks, 1996). Contrasting NLP development efforts allows for a comparison of like efforts, reducing the potential of confounds due to product differences.

All three case study sites agreed to participate in this research, though they choose to remain anonymous. We collected data on Team A over a period of twelve months. Data for Team B were collected over a period of 30 months. Data for Team C were collected across four months. In Appendix A we outline the data collection and analysis approaches used to build these cases studies. Summaries of the three cases complete this part.

Team "A" Team A built, and now both maintains and extends, a natural language-based product for retrieval of text from large text data bases. Team A represents the original product developers of a privately held company (called Company A in this paper) which now employees about 90 people who develop and support three products. The company has been in existence for seven years and selling

products for four. They relied on both extensive venture capital and government grants for survival in the years prior to the release of their first product. Thus, the original product came about in response to three forces: the interests of the funders, the interests of key development team members and the needs of the various key members to connect their individual system/modules together into one cohesive product.

Recently, Team A delivered the third release of their first flagship product (which we will call Magic). The product takes a (typed) natural language query and uses linguistic analysis to expand the query to run against an indexed corpus of free-text files (articles). This product is now a dominant market force in the business intelligence market. All of the original members of Team A continue to work for Company A. At present, the company's founder, a member of the original team, has a more managerially-oriented role. However, she remains an active contributor to the product design. Team A has six of the original seven members and now numbers nine people (including technical sales and support).

Team "B" Team B was also building a natural language-based product (which we will call Puff). The product was recently canceled and the team disbanded. Puff was being designed to take user questions (typed in normal language) and turn them into structured queries (SQL-like) for accessing a range of common database systems (such as Oracle, SyBase, Access and DB2). The product was to be an add-on component to several larger products. Team B existed as a small group inside one product group of a major commercial software development company that has been in existence for several decades (called Company B in this paper). Company C is highly regarded for its excellent development processes and has won both international quality awards and had extensive coverage of its development practices.

Team B drew financial support from discretionary funds provided by one of Company B's senior software product managers. Team B's members had been rewarded with release time from their stated

responsibilities to other products because of their history of superior performance and extensive personal credibility within the organization. Team B also relied on several external-to-the larger-organization consultants for key aspects of the natural language processing functionality. At the time of project cancellation – two years past inception – the original four members of the team had three specialized consultants and two contract programmers working with them at various levels of effort. Further, two of the original four members were also consultants that had been hired on full-time. At the time of product cancellation, Team B had a working demonstration product (a "demo") but was at least several months from a working product (or "alpha").

Team C Team C was typically comprised of five employees (and one part-time consultant) of a large software and hardware manufacturer. In August of 1994 they were asked to develop a forms-based system that could handle (typed) natural language queries about internal technical support (which we call Kelly). Essentially, this system was to automate the intake and redirection of email queries to the company's internal technical support staff. The project to build this system was budgeted for six month's duration in the company's 1992 development plan but was not funded until 1994. In 1998 the project team delivered and the system is currently in use.

Team C's members worked for company C's internal software development staff (30 developers in a group of about 80 IS staff) and were charged with supporting the production staff's efforts to make software product for sale to a range of customers. That is, the production staff of Company C were also software developers. The members of team C, however, were in support role, assisting the production staff with internal systems. Company C is well known for several industry-dominating large software application products.

Team C's membership changed steadily over the fifty months of work. For instance the team leader changing four times. Further, no member of the team (save for the consultant) was on the project for

more than thirty five months. The team's membership, team member's roles, and the project goals were in nearly constant flux during the entire project. The IS leadership team also changed once during the project. Despite the team membership turn over, the project followed the prescribed systems development method and adhered to the project and cost management methods used by the internal IS group. This attention to process control earned an internal process excellence award in the third year of the project. Team C disbanded after Kelly was operational. The application is supported on an as-needed basis by a contract programmer working for the internal IS group of Company C.

DIFFERENCES BETWEEN PACKAGED AND CUSTOM IS DEVELOPMENT

In this part we summarize some of the observed differences between packaged and custom IS development. In doing this, we again acknowledge that there are many similarities between packaged and custom software development. However, in this section we focus on specific differentiating characteristics of packaged development relative to custom software development. Like other studies, this comparison of the two domains is developed at four levels: industry forces, software development approaches, cultural milieu, and development team efforts (Carmel and Sawyer, 1998; Curtis, Krasner and Iscoe, 1988). For each level, the differences between the two domains are outlined and supported with data from the three case studies. A summary of this comparison is presented in Table 1.

< insert Table 1 near here>

Industry Forces

Relative to custom IS, time pressures (not cost) dominate the packaged software development industry. For example, most packaged firms are either very poor (i.e., garage shops) or very rich (i.e., Microsoft). This means that financing is either done with sweat equity or supported by huge cash reserves. Further, it is common (especially in places like California's Silicon Valley) for start-up

software companies to garner support from either venture capital or state-supported incubator money (Mall, 1998). The pressure to create a return on this investment leads to intense attention at bringing both new and innovative products to market. In industry terms this means "breaking new ground" and "hitting the ship date." (Zachary, 1994).

For those packaged firms who are someplace in the middle, life is harsh. Either these companies fail or they are bought up by larger companies. Due to both its economic impact and extensive venture capital investment, the packaged software industry is heavily scrutinized by Wall Street. For example, the Wall Street Journal assigns six reporters to cover Microsoft (and only three to IBM).

The packaged software industry products also have different success measures than do custom IS systems. Packaged software products seek favorable product reviews in trade publications and the degree of "mind share" – the awareness of a product in the minds of the target population – as key factors delineating success. Finally, due to the rewards of being a first mover in this field, the packaged software industry has two ways to achieve success: developing a large installed base and/or creating new markets (Brynjolfsson, 1994; Andersson and Nilsson, 1996). The former suggests that products begin to develop product trajectories which constrain design choices but can draw on current users for upgrades (Quintas, 1994). New market opportunities evoke the concepts of the "killer app:" the application that leads people to buy a computer solely so they can use that application. The web browser is one such example. Custom IS systems seek user satisfaction and integration (such as ROI payback) as recognition of value to the larger corporate mission.

The companies in the case studies reflect these points. Company A used external venture capital, Company B provided Team B with resources as needed in the form of internal seed money. Further, both Magic and Puff, the two products being produced by the teams in the case studies, were designed to open new markets. Team A beat their rivals to the market and have established a dominant position. Team B

was beaten to the market by a competitor and Puff was canceled several weeks later. Magic has received excellent product reviews. Puff was kept a secret and no market development was attempted.

Company C followed a typical pattern for custom IS. The Kelly project was delayed until funding could be designated by the IS steering committee. Thus, Team C 's work was funded by annual appropriations in the internal IS budget. Decisions on budget allocation were made by an ISD steering committee which met yearly. User satisfaction with the new systems was assessed (via phone survey) several months after implementation.

Software Development Approaches

Several factors underscore the differences between packaged software and custom IS development approaches. For example, in packaged software firms, developers hold line positions so their needs are central to the performance of the organization. In effect, they are the company's production mechanism as they generate revenue. This is not the case for custom IS developers who typically are part of corporate staff and serve supporting roles. A second factor is the distant relationship that packaged software developers have with their user population. This separation means that intermediaries – such as help desk personnel and consultants – link users to developers (Maiden and Ncube, 1998; Keil and Carmel, 1995; Grudin, 1991). This also means that implementation of packaged software is typically done by third parties, not the developers (------, forthcoming). Thus, while user involvement is still a part of packaged software development, the user's concerns and issues are filtered through intermediaries.

Packaged software developers also tend to have a product (not process) view of development (Carmel, 1995; Carmel and Becker, 1995; Cusumano and Smith, 1997). A product focus means that the dominant goal of the software development effort is to ship a product and all other activities are secondary[2]. This focus underpins the entire discussion of Zachary's (1994) recounting of Microsoft's NT

operating system. This product focus is also reflected in their belief that hacking (the attention to coding as a defining art) is good: whatever it takes to get the product out is more important than how it is done (Zachary, 1994). This belief is derived from commonly accepted wisdom that all products are driven by a person and the process that evolves is based around particular people (or even a person), not around specified roles (Zachary, 1994; Carmel and Sawyer, 1998). The best hackers are seen as visionaries and it is their vision that drives the product (Cringely, 1992; Zachary, 1994). This product focus also implies that these products have distinct trajectories. That is, the software evolves through a planned set of releases (even if the exact form and content of each release are far less carefully planned). Most custom IS developers believe in the importance of process – even if many follow a more ad-hoc approach to developing their software (Humphrey, 1989).

In the personalized, and product-focused environment of packaged software, design and development are intertwined. The resulting process is highly iterative, flexible, and constantly evolving. Project control is typically maintained by a small core team that rely as much on contention as they do on consensus (Carmel and Becker, 1995; Zachary, 1998). This approach underscores Microsoft's "Synch and Stabilize" method that Cusumano and Selby (1995; 1997) have documented.

The development efforts of Team A and Team B reflect both separation from the user and a product-oriented view. For example, no users were involved in the development of either product, though Team A did have annual demonstrations to let their funders observe progress. Presently, Team A invites existing users to preview (Beta test) upcoming releases. Team A can also clearly delineate several functions that are planned for future releases. However, when they began to search for funding, they were unable to explain their development processes to interested venture capital firms and struggled for resources. They still cannot draw an organization chart (or identify key managerial controls in the development effort) and do not clearly delineate either roles or steps to follow in creating new releases.

A second example of the product focus in packaged development is Team B's approach. At the beginning of the effort, Team B's members settled on a product architecture (how all the pieces would fit together) and worked as individuals, coming together every few months to assemble and test. The inability (or lack of desire) of one key member to complete their element led to others trying to do that piece very late in the product's development. Team B could not recover from the failure of this one person to complete their portion of the product.

Conversely, Team C's efforts reflect some of the best practices in custom IS development. Team C followed their internal development methodology very carefully. The process called for user involvement at key stages, including some use of joint-application design (JAD) techniques to bring developers and users together. The process methodology required extensive interim documentation. This process orientation allowed Company C to move people on and off the Kelly project team without bringing the effort to a complete stop.

The Work Culture

For this paper culture refers to the collection of patterns of social behavior and resulting artifacts that represent the ideas, values and shared norms of a distinct social unit (Schein,1992, p.9). One type of culture is a work culture. A work culture reflects the social behaviors, artifacts and shared norms of people who's social unit is defined by sharing similar work (such as farmers or packaged software programmers). Certainly, the work culture of packaged software development also inherits attributes from the many larger social units in which they are embedded. For example, an IS development team at a large insurance company may be closely aligned with the business culture of the insurance industry. Another example is that Microsoft employees often emulate many of the characteristics of their leader (Cusumano and Selby, 1995; Zachary, 1994).

The work cultures of packaged software developers can be distinguished by several traits. Firstly,

they are tangibly different from custom IS shops in the way people dress, behave, and approach their work. This, in part, reflects (and reinforces) the different industrial forces and software development approaches of the two domains.

The packaged software development work culture is entrepreneurially-oriented. Partly this is due to the low entry and exit barriers in this industry (where it is possible for nearly anyone with a PC to build software). This entrepreneurial orientation is also reflected in the extensive work ethic of packaged software developers. (Carmel and Bird, 1997). In the US – where the majority of packaged software is presently developed – the work culture is individualistically oriented: legends of hackers and "code cowboys" personify and mythologize the modern-day equivalent of "rugged individualism" (Carmel, 1997).

Both Team A and Team B reflect the packaged software development work culture. Team A's members are all vested in Company A's success through both bonuses and lucrative contracts. Further, the company structure is focused on the needs of the developers (for example, allowing for extensive personalization of work environments, work locations, and work schedules). Team B operated as a loose confederation, each person acting semi-autonomously. When Puff was canceled, every member of Team B left Company B to find work in other companies. Most were re-employed within 48 hours of Puff's cancellation, all of them were working at a different company within a month. The members of Team C reflected the more traditional nature of internal IS. They had specific regulations about what "business casual" dress meant. They had additional requirements regarding (more formal) attire when interacting with customers. No member of Team C received a significant bonus during this period of their work, though the current members were treated to ($100) gift certificates after the project was cited for process excellence.

Software Development Teams Efforts

At the level of the software development team, differences between custom IS and packaged software development domains manifest themselves in the structure and conduct of the software developers (Carmel and Bird, 1997). For example, relative to custom IS, packaged software development teams are smaller and are typically co-located. They also tend to have stable teams where members remain committed to the product over several version/releases. That is, they act as teams in the sense that they work together towards defined goals over prolonged periods. Typically custom IS development efforts are characterized by team membership turnover, division of labor by both phase and function, and disbanding following the completion of the first release (Cusumano and Smith, 1997). Thus, these are more like ad-hoc work groups, not teams (Goodman, Ravlin and Argote, 1989). Packaged software development team members are also motivated by financial rewards that custom IS developers do not have. For example, stock options and performance bonuses create 17 millionaires per week at Microsoft. Lucrative incomes are possible for members of small firms when they can produce a useful product for niche markets.

Both Team A and Team B reflect many of these characteristics. For example, the members of Team A receive bonuses that exceed their salary. And, until recently, all members of the team were co-located. Team B's members were rarely co-located (and this may have contributed to their product's delayed completion). Further, most of the consultants had lucrative contracts with large delivery bonuses tied to their specific modules, not to the overall product (and this may have also contributed to Puff's eventual cancellation). Team C's personnel were co-located. That is, all members of the IS department were located on the same floor in a 6000 square foot facility. This space was divided into cubicles and project rooms. Given the steady turnover of the team, the members did not relocate to be physically next to each other, though the longest distance between cubicles was about 250 feet.

IMPLICATIONS OF THE DIFFERENCES

In this part we speculate about potential implications of the differences between the two development domains. The previous part's summary of differences between packaged and custom IS development suggest that, despite many commonalities, the two domains have enough differences to be considered distinct domains. These differences suggest that, as packaged software continues to proliferate, there may be profound changes in the way stakeholder groups – such as software development organizations, software development teams, software developers, software consumers and researchers interested in software development – view software.

Speculating on the effects of packaged software development is appropriate since this is a means of exploring the implications of the differences and assessing the relevant conceptual issues. And, both of these are important aspects of theory building (Weick, 1995; Vaughn, 1992). The rest of this part is divided into sections to speculate on effects of packaged software development relative to the five stakeholder groups: software development organizations, software development teams, software developers, software consumers and researchers interested in software development.

Software Development Organizations

Like Quintas (1994), we believe that the emergence of the packaged software development organization as a distinct entity is changing how software is made. Since these organizations survive on their ability to produce software, software developers are their main production engine. This implies that organizational success will be tied, in large part, to the talent of their developers and development managers. Thus, a key factor of their success is software development talent. The nearly insatiable need for talent leads to incredible competition for the very best developers (Maglitta, 1997; Price Waterhouse,

1998). A common occurrence in places like California's Silicon Valley is for larger companies to buy smaller companies for their people, not their products. Another sign of this need is the relocation of firms to Silicon Valley (such as what both SAP and Baan have done recently) to be closer to talented developers. Said one developer from Team B, "I can go across the street tomorrow and get a job that pays 15% more." After Puff was canceled, he did.

This focus on hiring "the best <coding> athletes" goes beyond the degree boundaries of computer science (i.e., Ullman, 1995). Instead, content or domain expertise is more highly valued and often more important to the ultimate success of the product. For example, Team A had no computer scientists. Team B's only computer scientists were three of its consultants. However, seven of Team A's members hold doctorates (in linguistics and computer engineering) and all but one member of Team B had a doctorate (in either linguistics, philosophy, computer science or mathematics). Many of the members of both teams are acknowledged as leaders in their respective fields. Team B was unable to overcome the lack of performance of one such key player since there was no one else qualified to develop and code the specialized algorithms that were needed in that particular module. The educational backgrounds of Team C's members included business, computer science and vocational/trade schools. One of the attractions of working on the Kelly project was the opportunity to receive training in the C++ programming language.

A second implication is that packaged software development organizations must have strong technical leadership. For example, at Microsoft, most development managers code (Zachary, 1994). This is also seen in the cases. All the members of team A coded and all but one person on Team B coded (and this person served primarily as a boundary spanner between the team and the rest of Company B). This suggests that packaged software product management requires both sophisticated technical and managerial competence (Zachary, 1994; 1998). While Team C's project leaders were all ex-programmers, none were capable of doing (or did) any coding during their time as manager. The issues

raised by the last project leader were all schedule and budget-oriented.

Software Development Teams

At the team level, one implication of the differences between custom and packaged software development teams are the differences in social dynamics (Dube, 1998; Krishnan, 1998; Zachary, 1998). For example, the interactions among team members reflects contention, not consensus (Zachary, 1998; 1994). This differs from current best practices in custom IS that focus on developing intra-team consensus (Robey Farrow and Franz, 1989) This disparity was observed in Team B, where members were often at odds with each other. Both contentious email and public displays of anger during meetings were common. And, the angry outbursts among Team B's members did not lead to extended discussion of the issues and this may have crippled their performance. Team C's approach explicitly advocated consensus. Conflict negotiation and issues surfacing approaches were embedded in the project methodology and Company C had an active total quality management initiative that reinforced this consensus orientation.

Team A's members remain outwardly collegial and their disagreements are muted. Debates regarding technical directions, product features, and even work assignments tended to simmer for long periods. Team A's feuding, which is both genteel and time-consuming, is an important element of their success since it means that all aspects of the design are constantly open for debate. Perhaps because the PhD-trained members of Team A are used to academic debate they are able to thrive in this form of interaction. However, this was not the case for Team B. The more critical factor may have been the reward structure differences between Team A and Team B. For instance, the lack of agreement (and, thus, progress) on Team B did not prevent the consultants from being (well) paid. Conversely, Team A's members were rewarded (with large bonuses) after their product was shipped.

A second implication is that different stakes are involved for rewarding packaged software developers. Most packaged software developers buy in to a project because of either loyalty toward the leader of that team or towards the product (Zachary, 1994). This task/product orientation is central to the concept of "hot groups" discussed by Lipman-Blumen and Leavitt (1999). For example, Team A's leader is a charismatic pioneer and the members see themselves as partners with her. In contrast, Team B's existence came about because of the individual member's desires to build such a product and were focused by one (charismatic) person. For Team B, the extensive success of the individual developers (most were wealthy and some were millionaires) made it difficult to motivate them as a team. For Team A, many of the developers have a work-is-play myopia – they recreate by computing and work with computing – and remark that they are saddened by the loss of friends who do not share this type of intensity. This leads to an over-immersion and the potential of developer burnout (Zachary, 1994).

The dynamics of Team C reflected a less intense, more transient, orientation toward working together. The attention to careful documentation was seen as a collegial act. Said one developer "we're helping those that come on board to more easily get up to speed." Workers were amicable and project status meetings were subdued. Work days were predictable and overtime was both discouraged and rare.

Software Developers

We speculate that the implication for software developers due to the importance of programming talent is that, contrary to current discussions regarding the skill needs of IT professionals, technical ability alone may carry a career (Heckman, 1998; Sawyer, Eschenfelder, Diekema, and McClure, 1998). Factors such as the individualistic culture, the product orientation, and centrality of programming skill to product success suggest that technical prowess can suffice. This is not the case for custom IS, where business understanding, teamwork skills and client interactions are increasingly dominant (Sawyer, et al,

1998; Heckman, 1998). Thus, even as many computer-oriented curriculum are focusing on 'soft skills,' the greatest rewards are being paid to those few technical superstars (Maglitta, 1997). The shortage of truly superior programmers means that these people get to live with leading edge technology and will have no problems finding interesting work. Since these people are scarce, the market prices them high. Further, given the entrepreneurial nature of packaged software development, a failed start-up effort is not considered bad, merely an accepted risk that provides valuable experience.

The members of both Team A and Team B reflect this talent pressure. All are paid very well and given great latitude. When Team B's project was canceled, Company B's senior leadership was looking to put their talent to work on new products. However, disgusted with the decision, every member of Team B left the company for other start-ups. The two contract programmers (both having earned PhD's in computer science from a top-rated US university) started their own company. The members of both teams also show the social wear – none of Team A's members were married (several were divorced) and only three of Team B's members were (two consultants and the technical lead). Few had social relationships external to the team. When the consultant to Team B did not deliver, subsequent discussions centered on him being "washed up" and "done as a programmer." The expectation of Company B's technical leadership is that their best people would be highly productive for less than ten years. Company A has made no plans to accommodate developer burnout. Conversely, most of Team C's members had strong extra-curricular activities. Pictures of families adorned their walls and many spoke of their involvement in local community theater and church organizations.

Software Consumers

By software consumers we mean the organizations who are purchasing software instead of making the software internally. This reliance on the market means that the consumer can better control purchase

price (and the purchase price for software may be dropping). Further, the market provides for an increasing array of choices. A discussion of the individual-level implications of a market-purchase for users of this software is beyond the scope of this paper.

One implication is the increased pressure on purchasing organizations to integrate the products which they purchase – *caveat emptor*. Since so many products are being developed by so many vendors, and standards are either emerging/evolving along with the products or do not yet exist, each new product demands extensive integration into the existing product mix. Software product vendors are not able to provide the integration testing at the level that, say, IBM did when they provided the entire computing solution to a consumer. For example, Company A encourages customers to set up their system on a stand-alone server to reduce the potential of conflicts (and they price service differently for shared-server installations).

A second implication of vendor-made products for software consumers is "version thrashing" with respect to both implementing and maintaining the product. That is, software vendors make their money in two ways: selling to new customers and upgrading existing customers to new versions of the software. Both means suggest attention by the vendor to developing and maintaining a large, loyal, installed base (Brynjolfsson, 1994). To encourage a captive installed base of users, upgrade pressure is done through both pricing and support. For example, Company A matches all competitor's purchase deals. They also control upgrades by discontinuing service on any version more than two releases behind the current release. Customers who fall behind have to (re-) purchase Magic without the benefit of the steep discounts provided to current customers who are upgrading.

A third implication of the increasing use of packaged software by organizations is the changing locus of the strategic control of IT assets (Davenport, 1996; Quintas, 1994). This is particularly acute for those organizations implementing vendor-provided enterprise resource planning (ERP) packages such as SAP's

R/3 and Peoplesoft. First, the detailed analysis and customization efforts needed to fit these systems into the host organization are done after the purchase of the software (----------, forthcoming). This suggests that the user/developer interactions are shifting from collaboration on a development project to intermediated through focus groups, user groups, product defect reports (developed from customer call ins) and trade-show/beta-version use. Second, the consumer organization is committing to having their computing infrastructure follow the upgrade path that is dictated by market forces (Davenport, 1996). A third aspect of this changing locus of control is the increasing reliance on consulting and integration firms to provide the requisite IT knowledge base to implement and sustain these ERP (Davenport, 1996).

Researchers

In this section we speculate about some of the implications for those who conduct research on software development. In making the comparison between custom IS and packaged software development approaches, we believe that the evidence is compelling enough to suggest questioning cherished beliefs regarding at least four elements of contemporary software development wisdom. The first is the importance and the form of software development processes. Packaged development approaches are much more product oriented and much more personalized than are custom IS efforts (Carmel and Becker, 1995; Dube, 1998; Cusumano and Selby; 1995; 1997; Cusumano and Smith, 1997; Sawyer and Guinan, 1998). Simply, what is the proper balance between a product and a process focus (e.g., McConnell, 1993; MacGuire, 1995)?

The second area where this comparison of domains challenges accepted norms is the role of users. In custom IS, they are increasingly seen as critical to successful implementation (Markus, 1983; Kling and Iacono, 1984; Kieback, Lichter, Schneider-Hufschmidt and Zullighoven, 1992). However, users are not integral to contemporary packaged software development efforts (Grudin, 1991; Keil and Carmel, 1995;

Kjaer and Madsen, 1997). Two pertinent areas that arise from this observation are, first, how are user's needs reflected in packaged software products? Specifically, what are the forms and mechanisms being developed by packaged software developers to gather and represent user needs? And, second, in what ways are third party participants (such as integrators and consultants) affecting/intermediating the user/developer relationship?

Third, the differences between packaged software and custom IS development provide opportunities for comparative research on the evolution and use of methods, techniques and tools for software development. These types of comparisons have not been done, though there is a growing body of literature focused on domain engineering and/or software engineering component methods (where a range of standard sub-component approaches can be united to customize the development process) (Brinkkemper, Saeki, and Harmsen,1999).

Fourth, the consensus-premised nature of custom IS teamwork does not seem to be central to the "hot group" orientation of packaged software development (Zachary, 1998; Lipman-Blumen and Leavitt, 1999). This leads to a paradox regarding the importance between individual effort and teamwork (Yourdon, 1993; 1996; Sawyer and Guinan, 1998; Sawyer, Farber and Spiller, 1997). Perhaps the social mechanisms of multi-programmer development efforts are fundamentally different in the two domains (e.g., Zachary, 1998)?

Perhaps the similarities between packaged software and custom IS development serve to mask the important differences? Our discussion of the characteristics of packaged software development highlights these differences at the industrial level, the approaches to developing software, the work culture, and the efforts of the development teams. Building on this, we have speculated on the implications of these differences for five stakeholder groups: software development organizations, software development teams, software developers, software consumers, and researchers. What emerges is a picture of two

different approaches to software development.

References

----------, (forthcoming) "Information Systems Development: A Market-Oriented Perspective, *Communications of the ACM*.

Andersson R. and Nilsson A (1996) The standard application package market – an industry in transition?. In *Advancing Your Business: People and Information Systems in Concert* (Lundeberg M and Sundgren B, Eds.). EFI: Stockholm School of Economics, Sweden.

Badami V and Chbat N (1998) Home appliances get smart. *IEEE Spectrum* **X(y)**, 36-43.

Barley, S. (1990) "Images of Imaging: Notes on Doing Longitudinal Field Work", *Organization Science,* Vol. 1 No. 3, pp. 220-247.

Boehm, B. (1981) *Software Engineering Economics*, Englewood Cliffs, NJ: Prentice-Hall.

Bogdan, R. , and S. Biklen. (1982) *Qualitative Research for Education*, Allyn and Bacon, Inc, Boston.

Brinkkemper, S., Saeki, M. and Harmsen, M (1999) "Meta-Modelling Based Assembly Techniques for Situational Method Engineering." *Information Systems* **24(3)**: 209-228.

Brooks, F. (1974) "The Mythical Man-Month", *Datamation,* pp. 44-52.

Brynjolfsson E (1994) The productivity paradox of information technology. *Communications of The ACM* **36(12)**, 67-77.

Carmel E (1997) American hegemony in packaged software trade and the "culture of software. *The Information Society* **13(1)**, 125-142.

Carmel E (1995) Cycle-time in packaged software firms. *Journal of Product Innovation Management* **12(2)**, 110-123.

Carmel E and Becker S (1995) A process model for packaged software development. *IEEE Transactions on Engineering Management,* **41(5)**, 50-61

Carmel E and Bird B (1997) Small is beautiful: a study of packaged software development teams. *Journal of High Technology Management Research,* **8(1)**, 129-148.

Carmel E and Sawyer S (1998) Packaged software development teams: What makes them different? *Information Technology & People,* **11(1)**, 7-19.

Cringely R (1992) *Accidental Empires.* Reading, Mass: Addison-Wesley.

Curtis, B., H. Krasner, and N. Iscoe. (1988) "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM,* Vol. 31, No. 11, pp. 1268-1287 , .

Cusumano M and Selby R (1997) How microsoft builds software. *Communications of the ACM,* **40(6)**, 53-61.

Cusumano M and Selby R (1995) *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. New York: Free Press/Simon & Schuster.

Cusumano M and Smith S (1997) Beyond the waterfall: Software development at microsoft," in *Competing in the Age of Digital Convergence* (Yoffie D, Ed.), 371-411. Boston: Harvard Business School Press.

Davenport T (1996) Holistic management of mega-package change: The case of SAP. Working paper CB1039, Center for Business Innovation, Ernst and Young LLP, Boston MA, USA.

Dube L (1998) The difficult journey from a sequential to a team-based development process: The story of software co. *Information Technology & People* **11(1)**, 37-58.

Forrester Research (1998), *Sizing Commerce Software*. Boston, MA.

Gal-Oz S and Isaacs M (1998) Automate the big bottleneck in embedded system design. *IEEE Spectrum* **X(y)**, 62-67.

Goodman, P., E. Ravlin, and L. Argote. (1986) "Current Thinking About Groups: Setting the Stage for

New Ideas", in P. Goodman and Associates (eds.), *Designing Effective Work Groups,* Jossey-Bass, San Fransisco, .
Grudin J (1991) Interactive systems: Bridging the gap between developers and users. *IEEE Computer* **24 (5)**, 59-69.
Heckman R (1998) Planning to solve the 'skills problem' in the virtual information management organization. *International Journal of Information Management* **18(1)**, 3-16.
Humphrey W (1989) *Managing the Software Process*. Reading, MA: Addison-Wesley.
Jackson, Bruce. (1987) *Field Work*, University of Illinois Press, Urbana, IL.
Keil M and Carmel E (1995) Customer-developer links in software development. *Communications of the ACM* **38(5)**, 33-44.
Kieback A Lichter H Schneider-Hufschmidt M and Zullighoven H (1992) Prototyping in industrial software projects: Experiences and assessment. *Information Technology & People* **6(2-3**), 109-143.
Kjaer A and Madsen K (1997) Customer-vendor co-operation. *Information Technology & People* **10(3)**, 205-223.
Klepper R and Hartog C (1992) Trends in use and management of application package software. *Information Resources Management Journal* **5(4)**, 33-37.
Kling R and Iacono S (1984) The control of information systems developments after implementation. *Communications of the ACM* **27(12)**, 1218-1226.
Krishnan M (1998) The role of team-factors in packaged software quality. *Information Technology & People* **11(1**), 20-36.
Leebaert D (1995) News from the frontiers. In *The future of software* (Leebaert D., Ed.), 1-28. Cambridge, MA: MIT Press.
Lipman-Blumen, J., and H. Leavitt. (1999) *Hot Groups*, Oxford University Press, New York.
Maiden, N. and Ncube C (1998) Acquiring COTS software selection requirements. *IEEE Software* **15(2)**, 46-56.
Maglitta J (1997) Super programmers. *Computerworld*, 11/24/97, 95-96.
Mall E (1998) A good way to start? *Your Company* **2**, 64-65.
Markus M (1983) Power, politics, and MIS implementation. *Communications of the ACM* **26(6)**, 430-444.
McConnell S (1993) *Code complete*. Redmond, Washington: Microsoft Press.
McGuire S (1994) *Debugging the development process: Practical Strategies for Staying Focused, Hitting Ship Dates, and Building Solid Teams*. Redmond: Microsoft Press.
Merlyn, V., and J. Parkinson. (1994) *Development Effectiveness*, Wiley, New York.
Miles M and Huberman M (1994) *Qualitative data analysis, second edition*. Sage: Thousand Oaks CA.
OECD, (1998), *The Software Sector: A Statistical Profile for Selected OECD Countries*, Committee for Information, Computer and Communications Policy, Directorate for Science, Technology and Industry, Organization for Economic Co-operation and Development, Report DSTI/ICCP/ AH(97)4/REV1 at http://www.oecd.org/dsti/sti/it/infosoc/stats/software.htm (Sep. 20, 1999).
Price Waterhouse, (1998), *1998 Software Business Practice Survey*, Boston.
Quintas P (1994) The commodification of software. *Information Technology & People* **7(4)**, 1-22.
Robey, D., D. Farrow, and C. Franz. (1989) "Group Process and Conflict in Systems Development", *Management Science,* Vol. 35 No. 10, pp. 1172-1191.
Sawyer S Eschenfelder K Diekema A and McClure C (1998) Corporate it skills needs: The case of bigco. *Computer Personnel*.
Sawyer S Farber J and Spillers R (1997) Supporting the social processes of software development teams.

Information Technology & People **(10)1**, 46-62.
Sawyer S and Guinan P (1998) Software development: Processes and performance. *IBM Systems Journal* **37(4),** 552-569.
Schaff, W. (1998), ERP: Fact Vs. Fiction, *Information Week*. http://www.informationweek.com/704/iufin.htm
Schein E (1992) *Organizational culture and leadership, 2cnd Edition*. San Francisco: Jossey-Bass.
Seidler, John. (1974) "On Using Informants: A Technique for Collecting Quantitative Data and Controlling Measurement Error in Organization Analysis", *American Sociological Review,* Vol. 39 No. 12, pp. 816-831.
Ullman, E. (1997) *Close to the Machine: Technophilia and its Discontents*, New York: City Lights.
Vaughan, D. 1992. Theory Elaboration: the Heuristics of Case Analysis. in *What is A Case: Exploring the Foundations of Social Inquiry.* Ragin, C. & Becker, H. (Eds.). Cambridge, MA: Cambridge University Press.
Voas J (1998) COTS software: The economical choice? *IEEE Software* **15(2**), 16-19.
Weick K (1995) What theory is not, theorizing is. *Administrative Science Quarterly* **40**, 384-395.
Wilks Y (1996) Natural language processing: Introduction. *Communications of the ACM* **39(1)**, 60-63.
Yin R (1984) *Case study methodology* London: Sage Publications.
Yourdon E (1996) *Rise and resurrection of the American Programmer*, Prentice Hall.
Yourdon E (1993) *Decline and fall of the american programmer.* New York: Prentice Hall.
Zachary G (1998) Armed truce: Software in the age of teams. *Information Technology & People* **11(1)**, 59-66.
Zachary G (1994) *Showstopper: The breakneck race to create windows-nt and the next generation at microsoft.* New York: The Free Press.

Table 1: Summary Table of differences between packaged software and custom IS development

	Packaged Software	**Information Systems**
INDUSTRY		
	Time to market pressures	Cost pressures
	Success measure: profit, market share, mind share	Success measures: satisfaction, user acceptance, ROI
SOFTWARE DEVELOPMENT		
	Line positions	Staff positions
	User is distant and less involved	User is close and more involved
	Process is immature	Process is more mature
	Somewhat integrated design and development	Separated design and development
	Design control via coordination	Design control via consensus-building
CULTURAL MILIEU		
	Entrepreneurial	Bureaucratic
	Individualistic	Less individualistic
TEAMS		
	Less likely to have matrix/ project structure. More likely to be self-managed	Matrix managed and project focused
	Involved in entire development cycle	People assigned to multiple projects
	More cohesive, motivated, jelled	Work together as needed
	Opportunities for large financial rewards	Salary-based
	Likelier to be small, collocated	Grow larger over time and tend to disperse
	Share a vision of their product(s)	Rely on formal specifications/ documents

Table 2: Summary of Speculations regarding the effects of packaged software on development

	Speculations
Software Development Organizations	
	Software development talent is highest priority
	Need for strong development management/leadership
Software Development Teams	
	Differing social dynamics (more contention)
	Different set of team rewards & motivators
Software Developers	
	IT-skill needs dominate (all others are far less important)
	Entrepreneurial orientation
Software Consumers	
	Rely on market for software (implies *caveat emptor* re integration of multiple vendor's products)
	System upgrades and version thrashing
	Changing locus of IT control (from internal to external)
Software Researchers	
	Challenge beliefs re: development process/effort
	Challenge beliefs re: role of user in development effort
	Opportunities for contrast and comparison re: development

Appendix A: Case Study Data Collection and Analysis

The purpose of this paper is to speculate on implications of the differences between packaged and custom IS development. To support this speculation, we draw on data from three case studies to provide empirical examples (Yin, 1984). In this way we theorize about the differences (Weick, 1995; Vaughn, 1992) to encourage both additional debate and more careful study. In the following two sections we outline our case study data collection and analysis approaches.

Case Study Data Collection

In all three cases data were gathered using multiple sources including interviews, surveys, observation, and archival records. These are traditionally considered the tools of fieldwork (Jackson, 1988). For all three cases the researcher's role was as a participant observer: known to all members of the organization as a researcher (Barley, 1990). The data collection approach used in each of the three cases differed. The three approaches are outlined in the following paragraphs.

Data about Team A's efforts were gathered in a series of several interviews with key informants from both Company A and Team A (Seidler, 1974; Schein, 1992). Interviews with these key informants were repeated from three to five times (depending on the informant) over a period of 30 months. Informants also provided archival information (memos, email, and product documents) both as requested and voluntarily. That is, some of the informants would contact the researcher to offer additional evidence. The researcher's access was made possible by personal (student) contacts with members of Team A. Since many Team A's members were also trained researchers, they were supportive and receptive of the researchers' presence. This data set includes 20 formal interviews, field notes, and archival documents.

Data collection for Team B was done in a series of week-long visits (approximately once each five weeks) for 18 months. However, the first six months were mostly spent getting the members of Team B to accept the researchers' presence. The study was funded by Company B as their senior managers were trying to understand how to make a large organization more entrepreneurial. Early on members of Team B were concerned that the researcher was reporting on activities to management. Through the course of the first six months this apprehension faded and the team B members accepted the researcher as a "chronicler of their work" and saw it as a sign that what they were doing was central to Company B's future. This data set consists of 50 formal interviews, several hundred pages of field notes, and many archival documents.

Data about Team C were gathered through a series of interviews (both in-person and phone-based) following the disbanding of Team C. Access to Team C's last team leader was provided by a member of Team B. Subsequent interviews were done via snowball sampling: each interviewee provided names of other ex-members of Team C. This data set consists of seven formal interviews, each lasting from 40 to 60 minutes, and some archival records provided by the last team leader.

Case Study Data Analysis

All interview notes were either transcribed from tape or, if the interview was not taped, from rough notes. Summaries of archival records were developed to serve as pointers into the material. Field notes were constructed following the guidance of Bogdan and Bicklin (1982) and had two components: a log of each

observation and some interim reflection on that log. Data from these sources were analyzed using explanatory matrices (Miles and Huberman, 1994). In using these explanatory matrices, the *a priori* issues we raise in section three form one axis, sources of data form the other, and pointers to supporting data fill the intersecting cells. A different explanatory matrix was developed for each of the three case studies and for both the comparison and speculation aspects of this paper. Thus, there are six explanatory matrices developed to support the discussions in this paper.

Appendix B: A Brief Summary of Best Practices in Custom Information Systems Development

This summary is not intended to be exhaustive, merely evocative, of current standards of best practice in custom IS development. To develop this summary we made three assumptions. First, that there is a several year lag between research findings relative to software development practices being developed. This leads to the second assumption: that leading texts and highly visible books codify current best practices. Third, we also believe that best practices may be known, but not followed. We organize the best practices using the same structure as developed for our analysis of custom IS relative to packaged software development.

Table B1: Summary Table of custom IS development best practices

	Custom Information Systems Development Best Practices
INDUSTRY	
	Cost pressures dominate (Boehm, 1981, p.17)
	Success measures are user satisfaction and user acceptance (Merlyn and Parkinson, 1994, p. 43)
SOFTWARE DEVELOPMENT	
	Process-orientation with user involvement (Humphrey, 1989)
	Design control via consensus-building (Brooks, 1974)
CULTURAL MILIEU	
	Bureaucratic and group oriented (Yourdon, 1996, p.266)
TEAMS	
	Project-based with specialized roles, requirements-driven (Brooks, 1974; Yourdon, 1996; Meryln and Parkinson, 1994)

End Notes

1. This figure is developed from two separate sources. The number is reported in an Information Week (Schaff, 1998) article citing a Forrester Research report which is unavailable from Forrester research. Secondly, 1998 software sales data from the top nine ERP vendors (SAP, Peoplesoft, Oracle, Baan, J.D. Edwards, Lawson, IBM, Computer Associates, and Manugistics) were summed and rounded to the nearest billion.

2. One perspective on the packaged software development approach is Eric Raymond's article "The Cathedral and the Bazaar" found on line at http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html.