View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2002 Proceedings

European Conference on Information Systems (ECIS)

2002

Information Systems Development @ Internet Speed: A New Paradigm in the Making!

Richard Baskerville
Georgia State University, baskerville@acm.org

Jan Pries-Heje
The IT University of Copenhagen, jph@itu.dk

Follow this and additional works at: http://aisel.aisnet.org/ecis2002

Recommended Citation
Baskerville, Richard and Pries-Heje, Jan, "Information Systems Development @ Internet Speed: A New Paradigm in the Making!" (2002). *ECIS 2002 Proceedings*. 68.
http://aisel.aisnet.org/ecis2002/68

This material is brought to you by the European Conference on Information Systems (ECIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2002 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

INFORMATION SYSTEMS DEVELOPMENT @ INTERNET SPEED: A NEW PARADIGM IN THE MAKING!

Richard Baskerville

Georgia State University
E-mail: baskerville@acm.org

Jan Pries-Heje

The IT University of Copenhagen
E-mail: jph@itu.dk

ABSTRACT

Two studies from 2000-2001 – in U.S. and Denmark – have revealed that the nature of IS development has changed with the coming of Internet Speed. It may not be a revolution, but it is definitely and distinctively different. The new kind of methodology implements amethodical emergent systems development as a new package of practices. Systems are continually growing to adapt to emergent organizations where requirements are fluid, architecture and components are key, and maintenance never rises as a concern.

1. INTRODUCTION

The Software world celebrates the Silver Anniversary of the Software Crisis where schedule and budget overruns are typical, often coupled with low quality and functionality. Ten years ago the World Wide Web was invented causing the Internet to grow from an esoteric academic tool to a commonplace personal appliance. Four years ago in 1997 an "e-" was added in front of everything business related, and factors like time-to-market, customer focus and the ability to respond quickly to changing business needs came in focus. Parallel to this development the notion of "Internet Speed" was born, meaning that development cycles for software product for the Internet had tightened to enable fast paced change. As new technologies were penetrating the marketplace at unprecedented speed, software houses were striving to keep up with the speed, releasing new versions of Internet, Web, and e-business applications more and more often.

In January 2000 we began investigating how Internet Speed was influencing software development. We interviewed in 3 companies in Denmark in May and June 2000 , and in 9 companies in the U.S. from July 2000 to January 2001. Data was collected using open-ended interviewing. The interview guide was constructed following a thorough review of the literature. One to three interviewers conducted the interviews and data was collected as field notes. These notes were supplemented with audio/video tape and cognitive maps.

We analyzed the interview data using grounded theory methodology (Strauss & Corbin, 1990). This methodology develops a substantive theory without prior hypotheses. The chosen grounded theory approach is composed of an alternation between three different coding procedures to analyze the collected data: open, axial and selective coding. At the end of the coding we had identified categories, sub-categories and relationships in the data.

2. DANISH STUDY

We interviewed in three Danish companies in the spring of 2000. Two of the companies were new to the authors and the third was a company we had visited over a period of time (starting in 1996) for a longitudinal study. The main facts about the three companies are given below.

Name (Pseudonym)	**Industry and what offered? When Founded, Size?**	**Number of people interviewed and their organizational roles**
Gamma	Software House making custom-tailored Internet products for major international customers. Late 1990s. 50 employees when interviewed.	Four people interviewed: Coordinator (equals VP Development) Project Manager, Software developers
Epsilon	Custom-tailored Internet and Intranet products interfacing with large existing databases. Early 1990s. 40 employees when interviewed.	Six people interviewed: VP Marketing and Customer Contact, VP Development Manager, Project Manager, Software developers
Omega	A general web-based product sold on the market as a standard product for e-commerce. Late 1990s. 12 employees when interviewed.	Two persons interviewed: President & CEO, Chief software architect/developer

In the companies, we noted ten properties of the new methodology. Each of these properties is briefly described below, along with examples of how these properties are manifested in the companies. For further details, see Baskerville and Pries-Heje (2001).

2.1 Time pressure. Minimizing time-to-market from concept to customer use is an all-consuming activity and achievement of this goal drives almost all other elements of the methodology. This goal is not altogether new in business (Smith & Reinertsen, 1995) nor in software development (Cusumano & Selby, 1995; Iansiti & McCormack, 1997), however, the degree to which it has inflamed widespread systems development methodology has not yet been recognized.

2.2 Vague requirements. The data also contains frequent mention of the need to build software under conditions of ambiguous requirements. The developers and managers believe an inability to pre-define system requirements is the central, defining constraint of Internet time development. Traditionally the heart of systems methodology, Internet time methodology abandons predefinition of the operational goals and strategies in the systems project in favor of starting point in which the goals, and consequently the specific strategies, are permitted to persist in near or full ambiguity.

2.3 Prototyping. This approach is widespread and permeates early and late development work. For Epsilon, prototypes are claimed as part of their core competence: "We live from being technologically in front of our competitors, and from being able to visualise more far-reaching and wide-ranging solutions to our customers than our competitors are able to". Gamma makes 3-4 prototypes within one typical project, some very late in the project.

2.4 Release orientation. Vague requirements continue throughout the projects. One consequence is a "release orientation" in which applications are produced in a series of ever more refined and extensive versions of the product. Each release contains a bug-fixes and new features that satisfies competition demands for significant product and feature changes every few months (Cusumano & Yoffie, 2000, p. 299). It relieves some of the time pressure because a new feature can be postponed to a following release that is never very far away.

2.5 Parallel development. The release orientation demands a fast cycle time that is impossible to meet in a serial process. Serial development assumes that systems must arise from a series of sequentially dependent processes. Parallel development assumes that systems can also arise from a set of simultaneous, mutually interdependent processes. Parallel development and release orientation go hand-in-hand. Products and releases have to be designed and coordinated for parallel development, another aspect common to large-scale Internet software development.

2.6 Fixed architecture. To make parallel development possible, it is also necessary to have some basis on which to divide and coordinate work. All three companies have used a fixed three-tier architecture as this basis. The foundation is a Database with content. In the middle is a Business Logic layer. And at the top is the User Interface.

2.7 Coding your way out. The short time frame introduces a coding focus. As a project manager from Gamma expressed it: "You have to accept that hacks are being made. That you don't have time to think systematically. And that you don't reuse because of the time pressure". Omega simply developed their own programming language to enable the necessary speed of development. "… it allows us to do things fast, incredibly fast.", said the CEO.

2.8 Quality is negotiable. Among the different views of quality we find some that focus on the fulfilment of customer expectations, thus suggesting that quality is the degree of fulfilled expectations (Hunt, 1992). In settings like Internet development, where new functionality is desired fast, customers have seemed prepared to accept crude delivery and limited reliability as tradeoffs. In this way, quality becomes an emergent negotiation between developers and users. Early adopters will trade other quality attributes for early availability, while expecting these other quality attributes to be added later.

2.9 Dependence on good people. Time pressure is the primary reason why Internet software companies focus on recruiting high-quality developers. As one of founders of Epsilon phrases it: "I believe the largest bottleneck we have is to get enough qualified employees". However all types of IT people are in the same level of high demand. For example, traditional analysts are not in the same high-level demand as the programmers who are close to the code.

2.10 Need for new kinds of structure. The data suggest that the older and larger the organization and/or the customers the larger the need for structure. The resources available for systems development drive the demand for structure in Internet-time methodology. There seems to be a need for structure, but the traditional structures seem to fail at Internet speed. When resources grow without structure, quality seems to be driven down, perhaps through ineffective resource use, inefficiency, and poorly coordinated activity. Structure is added almost begrudgingly, and only as little as may be necessary to keep the development activities focused on the goals of fast delivery of desired features.

3. U.S. STUDY

We interviewed in nine U.S. companies in the late fall 2000 and winter 2001. The main facts about the nine companies are given below.

Name (Pseudonym)	**Industry and what offered? When Founded, Size?**	**Number of people interviewed and their organizational roles**
Calliope	Offers forecasting tools for energy and communications industry. Mid 1990s. 20 employees when interviewed	3 people interviewed: VP Operations, Project Manager, Software Developer
Clio	Low-price health care and utilities. for groups of customers. Late 1990s. 35 employees when interviewed.	Six people interviewed: President & CEO, VP Technology Operations, Director of Marketing Research, Chief Information Officer, two developers
Erato	Offers to help Brick & Mortar companies getting online. Late 1990s. 55 employees when interviewed.	Four people interviewed: Director, Chief Financial Officer, Chief Operations Officer, and developer
Euterpe	Film and Television Industry. Offers high-tech tools online. Mid 1990s. 80 employees when interviewed.	Four people interviewed: Project managers, marketing specialists, senior web developers
Melpomene	Carries out personnel administration. for other companies online. Mid 1990s. More than 100 employees when interviewed.	Seven people interviewed: Project managers, architects, user interface designers, web developers

Polyhymnia	Offers online services for transport and tourist industry. Early 1990s. More than 1000 employees when interviewed.	Six people interviewed: Senior managers,Project managers, QA manager, lead developers, web developer
Terpsichore	Offers industrial insurance online. First half of 20th Century. More than 10000 employees when interviewed.	Three people interviewed: Human Resources Manager, Internet site manager and Internet site developer
Thalia	Online service for transport and logistics industry. First half of 20th Century. More than 100000 employees when interviewed.	Six people interviewed: CIO, Senior manager, project managers, architects, senior developers, web developers
Urania	Business-to-business communication. Last half of 20th Century. The part we looked at was founded in the 1980s. More than 100000 employees when interviewed	Six people interviewed: Senior manager, Project managers, quality assurance manager, QA specialist, Web developers

In analyzing the interviews using Grounded Theory we identified three major categories of observations *causing* a change, and three major categories of categories *resulting* from the changing causes. Furthermore one of the resulting categories – New Software Process – was easily sub-categorized. Each of the categories is described below, along with examples of how the subject companies manifested these properties.

3.1 Cause: Desperate Rush-To-Market. The desperate rush to market began when technology for Internet software products suddenly became available such as the free World Wide Web browser. This browser effectively created a standardized client architecture as a simple framework for networked applications. There were three factors that intensified this rush. First, deployment of Internet software products occurs in a matter of seconds. Given the standard client, all development focuses on the server. There are no complex distribution channels. Server-based application software is updated, and the user access to the new or revised software product is nearly instantaneous. Second, the Internet, the Web and the browser created an international market with a breadth and scale of nearly inestimable potential. Third, venture capitalists, particularly in the US, were prepared to pour huge amounts of money into small companies with high Internet market potential as new possibilities opened for creating and licensing valuable new forms of intellectual property. For example, patenting client patterns such as Amazon's one-click purchase. As a result, there is a sense of extreme urgency in Internet software development: Opportunities seem like obvious gaps that someone else will grab unless the market is captured first and held fast against all competitors to follow. "Time-to-market - I hear that constantly. Bigger, faster, better. Everything is very rush, rush, rush," they told us at Polyhymnia.

3.2 Cause: Different Kind of Software Market Environment. The second factor causing the evolution of this new kind of software development process is the environment into which these software products are being placed. The particular collection of characteristics of this environment represents a large and uniquely different marketplace for software products. This environment is notable for a number of aspects, but attention importantly centers on flexibility and constraints. The market is flexible in terms of requirements and quality. For example, the software in this marketplace rarely deals with mission-critical or life-critical applications such as those in medical, defense, aerospace or process control applications. Demands for quality are not as stringent, especially in the early stages of a product's introduction. Further, as successive versions of products often have very short life spans, some of the lapses in quality in one release can be quickly fixed in a successive release. Requirements are negotiable from release-to-release in a market-defined process where pragmatics is allowed to intervene to limit the scope of features in each release. Quality factors, such as scalability and maintainability can often be postponed for later releases. Constraints in this new market environment are different. A major constraint is the incredibly short time frame for software development imposed by the rush-to-market. There are also narrow technical constraints on the

software architecture imposed by the standardized browser clients, the Internet speeds and protocols, and the architecture of existing legacy systems or large-scale backend systems.

3.3 Cause: Lack of Experience Developing Software Under These Conditions. There are too few knowledgeable and experienced developers who can meet the speed and market challenges above. This shortage of experienced professionals has two effects: First, making the marketplace for developers tight and expensive, and second, creating development organizations that lack sufficient experience and expertise. Though many developers have lot of experience in traditional software development environments, they lack sufficient expertise and experience with the Internet environment. One manager from Melpomene lamented that "lots of people [in our organization] came from more corporate environments where it took forever to get things out of the door," but much of their prior experience may turn out to be a hindrance rather than a benefit in the new environment.

3.4 Result: A New Software Process. The software process for Internet software development has to be different. The differences include nine distinct characteristics. Although there may not be any single characteristic that is particularly unique to this new software process, the collection of characteristics is unique and remarkably common to Internet software development processes. We now examine these characteristics in more detail.

3.4.1 Parallel Development. The high speed release cycles compress development into a time frame where only overlapping, parallel development can meet the demands. Releases may be totally developed in parallel, or staged onto the market such that design, development, and quality assurance are all taking place simultaneously, but sequentially on different releases. Sometimes, coding begins even before the requirements have been fully understood. Development proceeds in anticipation of the features that will be required in the final version of the product. However, it is quite possible for a feature that has almost been fully implemented to be pushed to a later release due to changing customer demands. "In these projects, even during the requirements phase, our teams will begin coding expecting what will be in the final requirements and in various releases", they told us at Urania.

3.4.2 Release Orientation. "People have a perception of Internet Speed. They expect it. So we've had to scope our delivery or deliver a smaller set of features. Thereby releasing more often", told a Manager from Euterpe. Clio said: "Development cycles last from 2 to 15 days… timing is important. Usually product 'launches' happen every 15 days". Requirements are kept fluid through constant monitoring and prioritization of the features that will be included in the product in successive releases. Features that cannot be completed in time can slip from one release to the next. Similarly, an unexpectedly important new feature can be slipped in rather late in the process when market conditions require it. The fast cycle time softens the penalty from slipping a feature. Though during early stages of a product, releases are made in short cycles, as the product matures, release cycles get longer. Only then it is possible to accommodate customer requirements for robustness and stability. The need to address such quality factors appears to be correlated with the maturity of the product and the size of the customer base. When the product is more mature and has attracted a large client base, the need for quality factors such as security, stability and robustness seem to overtake the need for speed.

3.4.3 Tool Dependence. Many Internet software development organizations make heavy use of development tools and environments that speed up the design and coding process. The infrastructure and tools provided by the new technologies offer much of the functionality that used to be custom built in traditional software development. Urania estimated that "fifty percent of development is already taken care of by tools we use such as iplanet or, websphere. The APIs to these tools gives a lot of functionality". Further, these new tools also help create a system that is well modularized and architected, even in the absence of a formal process to achieve these desirable qualities in the system. For example, the separation of concerns about the user interface, business logic and data management that is enforced by these tools indirectly imposes an architecture even though it was not consciously

planned. Or as they said at Urania: "Separation of responsibility in the code given by these environments is very helpful for people who do not have such experience."

3.4.4 Customer involvement. In many Internet software development projects "requirements are fuzzy", as they told us at Polyhymnia. In such situations having close access to customers helps in the "prioritization of features based on customer's demands". Thus intimately involving customers to cope with evolving and unstable requirements is typical. Customers are often co-located with the development team, and participate closely in all phases of development. Most projects rely on such involvement rather than a formalized requirements management process. Focus groups are used to ascertain and prioritize requirements when there are several stakeholders representing different 'user communities'. Close involvement also implies that customers receive immediate feedback on the costs and schedule implications of any changes in requirements. Also, chunks of requirements can be scheduled to be included in different releases by constant consultation with customers.

3.4.5 Prototyping. The fastest way to settle requirements specifications seems to be by creating a prioritized list of features. Instead of using formal requirements documents, most projects use prototyping as a way to communicate with their customers to validate and refine requirements. Customers describe the basic functionality for new or changed features and these are quickly prototyped for demonstration and experimentation. Prototyping is used to communicate with customers and obtain quick feedback. "We are supposed to have a full [requirements and design documents] but a lot of programmers use the prototype and go back and forth to check, or go back and ask: what was this supposed to do", they told us at Melpomene. "We [usually] implemented the [customer] suggestions before the next meeting" a Urania manager told. To a certain extent, the production software itself can be a form of an operational prototype, a refinement of the code created for experimentation with features. The rush to market encourages a tendency to deploy the prototypes rather than wait for robust implementations of the desired functionality. Further, the ability to quickly replace the current version of the product with newer versions is also another factor justifying this practice.

3.4.6 Criticality of Architecture. A well-planned architecture seems to help smooth the rapid development process that is never quite stable. It enables each release to be developed with some similarity, and the largest possible reuse of components. A three-layer architecture is common: (1) Database layer, an interface to the underlying data or to a legacy "back-end" systems, (2) Business logic layer, the detailed processing code, (3) User interface layer, the web-based "front-end" that is delivered through the browser. Some organizations have paid a lot of attention towards developing a common architecture and standardizing it across applications. They view this as a worthwhile investment that will pay huge dividends in developing scalable and maintainable systems. A well developed architecture can help separate concerns and help employ optimal solutions in different phases of development.

3.4.7 Components Based Development and Reuse. The Internet speed development can be achieved often only if the software can be assembled with as many reusable components as possible, rather than crafted from scratch. "Internet speed needs reuse. We need to take components or assets and know how to put them together", a Thalia developer said, and a Manager from the same company continued: "The strategy is to acquire, integrate, and assemble components with wrappers – to get things done quickly …". The reuse of components at all levels of the architecture (business logic, interfaces and back-end infrastructure) is very prevalent. The software development process in many cases is just a process of achieving interoperability and integration among components developed elsewhere or purchased off the shelf.

3.4.8 Maintenance ignored. One of the consequences of short life span of Internet software is that often maintenance of the software is not given serious consideration. "Products are not documented. No design document, no requirements specification. The person who did it is gone. It takes much longer time. Often we can start from scratch. It leads to a throw away mentality.", as they said at Polyhymnia. When the software is retired quickly and replaced with newer versions developed from

scratch, this may not be a serious issue. However, it is not often the situation. Many components of Internet software are developed by different groups in the organization and need to be integrated and maintained to work together within an evolving system. In such cases, the absence of concern for understandability and maintainability of this software causes serious problems.

3.4.9 Tailored methodology. The processes and methods used in Internet software development vary considerably depending on the composition of the project team and the nature of the product. Some organizations have developed an overall framework within which individual projects are allowed to tailor their methodologies. "We have an overall methodology. But we have to tailor processes for individual teams", they said at Urania. With the intense demands of speed, many organizations use just "enough process to be effective", Euterpe added. Often the tendency is to skip phases or tasks that may impede the ability to deliver the software on time, though this may be done at the risk of producing lower quality software.

3.5 Result: A Changed Culture. Internet software development organizations have a distinct culture that appreciates less structure, smaller team sizes and diverse team compositions. In some senses, there is more emphasis on individuality. Individual developers relish the feeling that they make a difference in shaping the product and the organization's strategy. They are not just replaceable programmers in a sea of developers. They value being sharp and discovering clever tricks like development process shortcuts. Yet at the same time, there seems to be a tight bond among Internet software developers, a sense of belonging with others who share the same values. It every case, it was important to know that the organization prized their contributions to the software development efforts highly. Overall, this culture represents a change in the worldview of these developers. These Internet software developers seem more critical, willing to challenge dogma, and more ready to invent new ways of developing software than studying old styles. It encourages people to "take a risk, make a mistake. But, be smart not to make the mistake twice", as they said at Thalia, "We are not 9 to 5 people down here. We are more dynamic … There is lot more excitement and enthusiasm here."

Several key factors drive a need for really smart, energetic people who can invent what they need on the fly and learn from their other more experienced colleagues with different expertise. These factors include the people orientation in the software process, the disregard for knowledge of tried-and-true methodology, the relative newness of the architecture and tools, and the ineffectiveness of traditional thinking, Erato has found one way to do this: "Team up experienced people with inexperienced people – like in eXtreme Programming."

3.6 Result: Quality is Negotiable. Many quality factors are not as critical in Internet speed development as they may have been in traditional software development projects. Quality is definitely lower, a function of implicit negotiations in the market between software competitors and customers. The intense time pressure of Internet speed development creates a setting where something must give way. Creative people, operating with a different set of constraints, are willing to rethink the meaning of quality. Values traditionally appreciated by the majority of software developers – such as product scalability and easy maintenance – become much less rational ideals. New developers are willing to abandon throwaway code rushed to production. Customers and users seem to appreciate immediate functionality and are willing to defer a certain amount of reliability and performance. Developers are willing to rebuild badly designed or coded features later when their deferment runs out. "It is different working at Internet speed. Compressed cycles means that quality suffers. With speed we are sending less quality out the door.", they told us at Polyhymnia. Or as they said at Urania: "with e-speed sometimes we can not do the QA as well as other projects."

4. DISCUSSION

The following compares the factors found in the Danish and US studies:

Danish Study	**U.S. Study**	**DK-US Comparison**
2.1 Time Pressure	3.1 Desperate Rush-To-Market	Two notions of the same thing
	3.2 Different Kind of Software Market Environment	The differences in flexibility and constraints compared to traditional software were more explicitly expressed in U.S. Study
2.2 Vague Requirements		The Danish Companies seems much more willing to live with vague requirements than were the U.S. companies
2.3 Prototyping *and* 2.4 Release Orientation *and* 2.5 Parallel Development	3.4.5 Prototyping *and* 3.4.2 Release Orientation *and* 3.4.1 Parallel Development	These 3 solutions mainly to market-rush and time pressure were commonly found in both U.S. and Denmark
	3.4.4 Customer Involvement	If there was a difference in the use of prototypes in Denmark and U.S it was that prototypes more often were used as a vehicle for customer involvement in U.S.
2.6 Fixed Architecture	3.4.6 Criticality of Architecture	Another frequently used solution to cut down development time
2.7 Coding your way out		Coding as the last escape was mentioned in Danish Study, and rarely in U.S. study.
2.8 Quality is negotiable	3.6 Result: Quality is Negotiable	Same thing found in U.S. and Denmark: Quality takes second place to Time
2.9 Dependence on Good People	3.3 Lack of Experience *and* 3.5 Result: A Changed Culture	All the Danish companies and a few of the U.S. mentioned good people as a major thing. In U.S. it was coupled to a widespread lack of experience among the employees, and it became part of the changed culture
	3.4.3 Tool Dependence *and* 3.4.7 Components based development and reuse *and* 3.4.8 Tailored methodology	These three solutions were only found in U.S. There was widespread trust among the companies that these three would provide much needed structure
2.10 Need for new kind of structure		One of the Danish companies had tried both a company specific Methodology and SW-CMM without success

Most of the factors, such as prototyping, rush-to-market, parallel development and release orientation, are not individually new (at least in theory). However, the discovery of a frequent pattern of usage

that involves a different mix of such factors in practice amounts to a significant shift in the way software systems are being developed. The rise of these common sets of elements amounts to a new paradigm of best practice. For decades, theoretical advances in methodology have failed to supplant structured, waterfall approaches to development. But this new paradigm stands in stark contrast to the decades of a practical fascination with the structured tradition; and the new paradigm appears to have grown dominant in at least some sectors of development practice.

4.1 Paradigmatic Differences

There are five general differences in this new set of factors revealed in this study that are paradigmatic. First, analysis must be done differently because the requirements are fluid and ambiguous. Second, the keystone role of architectural design is different because good architecture is required as an enabling basis for survival of the system. This survival implies scalability and maintainability. Third, detailed design is different because of its basis on components and tool suites. Fourth, the rush to coding and implementation has been enveloped by the paradigm as a substitute for unambiguous requirements. Fifth, the attitude toward maintenance is different, because in some of these settings, maintenance is generally ignored.

Collectively, these differences appear in the convergence of the studies above. This convergence appears as a paradigmatic "package" of practices driven by these differences, and from the evidence above, appears to be proliferating as practice internationally.

4.2 An Emergent, Amethodical Paradigm

The package of practices may indeed be an amethodical exemplar responding to emergent systems. Emergent organizations endure continual change, a state in which these organizations are constantly seeking stability, while never achieving it. Emergence demands radically different kind of IS development: not as a series of projects each having a clear beginning and end, but rather as continuous redevelopment of the entire organizational portfolio of systems (Truex, Baskerville, & Klein, 1999). Amethodical development implies management and orchestration of systems development without a predefined sequence, control, rationality, or expectations for universality of method. An amethodical development activity is unique and unpredictable for each information systems requirement (Truex, Baskerville, & Travis, 2000).

The package clearly reflects the needs of emergent systems because it responds to changing and ambiguous requirements. Features like the release orientation and parallel development mean the systems development does not begin or end, but moves rapidly from release to release in a form of continuous redevelopment. The package embodies amethodical development because it does not respond to a defined process or notation, but rather stands as a set of common constraints and imperatives. The package is a methodology, but not in the sense of a process of steps or stages. It is a methodology because it guides developers as they innovate unique techniques in the face of a familiar set of demands and constraints.

4.3 Focus on evolving systems

The package of factors represented in the two studies lead to a focus on evolutionary systems that provide support for emergent organizations. Systems development is converging with maintenance such that it becomes difficult to distinguish these activities. It becomes difficult to determine whether development is continuing across the life of an operational system, or that the development is really a continuing maintenance activity amounting to continuous redevelopment. Clearly though, systems development projects never reach completion, only evolution that exists throughout the system lifespan.

5. CONCLUSION

The need for amethodical and emergent development practices has been building for some time (cf. Baskerville, Travis, & Truex, 1992; Truex & Klein, 1991). However, the instantiation of these practices in a widespread way can now be recognized in the empirical study above. The set of factors analyzed in the study above reveals a systems development activity that is not like other methodologies. It exists chiefly as a set of imperatives or constraints, rather than specified processes or notation. Systems are not built in a single project that completes with a delivery, but rather are continually "growing" (or being grown) to adapt to the emergent organization. These systems are nurtured and evolved in a new development culture, rapidly, to meet the evolving demands of markets and customers. The nurturing addresses requirements that are fluid and ambiguous, depends on architectural design, components and tool suites, involves a rush to coding and implementation in which maintenance never rises as an concern.

REFERENCES

Baskerville, R., & Pries-Heje, J. (2001). Racing the E-Bomb: How the Internet Is Redefining Information Systems Development Methodology. In B. FitzGerald & N. Russo & J. DeGross (Eds.), *Realigning Research and Practice in Is Development: The Social and Organisational Perspective* (pp. 49-68). New York: Kluwer.

Baskerville, R., Travis, J., & Truex, D. (1992). Systems without Method. In K. Kendall & K. Lyytinen & J. DeGross (Eds.), *The Impact of Computer Supported Technologies on Information Systems Development* (pp. 241-270). Amsterdam: North-Holland.

Cusumano, M., & Yoffie, D. (2000). *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft*. New York: Touchstone.

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

Hunt, V. D. (1992). *Quality in America: How to Implement a Competitive Quality Program*. Homewood Il.: Business One Erwin.

Iansiti, M., & McCormack, A. (1997). Developing Products on Internet Time. *Harvard Business Review, 75*(5), 108-117.

Smith, P. G., & Reinertsen, D. G. (1995). *Developing Products in Half the Time 2nd Edition*. New York: Van Nostrand Reinhold.

Strauss, A., & Corbin, J. (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Newbury Park, Calf: Sage.

Truex, D., Baskerville, R., & Travis, J. (2000). Amethodical Systems Development: The Deferred Meaning of Systems Development Methods. *Accounting, Management and Information Technology, 10*, 53-79.

Truex, D., & Klein, H. (1991). A Rejection of Structure as a Basis for Information Systems Development. In R. Stamper & R. Lee & P. Kerola & K. Lyytinen (Eds.), *Collaborative Work, Social Communications and Information Systems* (pp. 213-236). Amsterdam: North-Holland.

Truex, D. P., Baskerville, R., & Klein, H. K. (1999). Growing Systems in an Emergent Organization. *Communications of The ACM, 42*(8), 117-123.