Time-to-completion in software package startups

Erran Carmel

Kogod College of Business Administration, The American University
Washington D.C., 20016
ECARMEL@AMERICAN.EDU

Abstract

All businesses operate within an environment of time-based competition. *For product development this means reducing* time-to-completion. *An exploratory study of time-to-completion accelerators was conducted in twelve software package startups. Awareness of time-to-completion was contradictory: deadlines were soft, articulation was poor, and process time recall was uneven. But developers increased effort significantly during crunch periods. Although causality is difficult to verify, the most important accelerator seemed to be the* core development team, *an inner team with cross-functional talents. Core team size ranged from three to six members. Because of its small size and high intra-group communication, the product design and its development methodology were not formalized.*

1 Introduction

We hear much about *time-based competition* these days. Within the area of product development this topic is known as *time-to-completion* (also known as: time-to-market, fast cycle, and rapid development) [1, 2, 8, 11, 13, 17, 19]. Time-to-completion is the simple measure of calendar time from the beginning of the product development cycle to product release. The notion of time-to-completion has received attention in more traditional manufacturing/ product segments: in automobiles, computer hardware, electronics, and others. For example, Chrysler reduced time-to-completion by rolling out the Viper in 36 months instead of the usual 60. Hewlett-Packard reduced its time-to-completion for printers from 54 months to 22. The benefit of rapid development is that market share can potentially *double* because of switching costs and increased customer loyalty.

Software is increasingly becoming a *product* in the full sense of the term. Software can be categorized into two types: custom and package [4]. Custom software development is well-documented in more than two decades of experience and study. Few studies examine software development practices in software *package* firms (exceptions include [12, 15, 18]), an obvious oversight considering the U.S. has 80% of the global software package market [14].

The package sector has been particularly successful and innovative. The rise of software startups that came with the PC revolution of the 1980s is remarkable. These successes are epitomized by Microsoft, which now roughly equals General Motors and International Business Machines in total market capitalization. Yet there has been little examination of what it is that these firms are doing that have made them so successful--particularly in software development. Better data are needed about the key factors that make the sector as successful as it has been, and what will make it successful in the future.

The specific domain of this study is young package firms (hence *startups*). If there is indeed something right in the software development practices of these firms, then these practices may be transferred to other technology sectors.

2 Literature Review

Traditional thinking in Management Information Systems and Software Engineering has focused on the notion of *productivity*. Whereas time-to-completion looks at the simple measure of calendar time from start to finish, productivity is a ratio of effort and measures a different construct-- units of progress, such as

1060-3425/94 $3.00 © 1994 IEEE

Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences, 1994

lines of code, or function points, per day or per programmer.

Perhaps the most important focus in the study of time-to-completion is to identify the critical accelerators: those techniques that help reduce overall time-to-completion. A number of researchers have compiled sets of accelerators [8,11,13,16]. A subset of these is used as a framework for this study:

1 *Development methodology.* The (pre-) defined approach that is used to move through the phases of development from product inception to release. Within software development this is also referred to as a life-cycle model or a process model [7,9].

2 *The development team.* The individual and group characteristics that describe the collection of individuals who make up the development team.

3 *Allocation of resources* for labor and capital assets.

4 *Project Management.* The management of people, milestones, resources, and tasks.

5 *Risk Analysis.* Identifying, assessing, and managing risks.

6 *Incremental innovation.* Making small changes from one product to the next.

Of the six accelerators, the software development literature has focused on the first two. Hence data collection emphasis corresponded to this focus.

This study examines two questions regarding time-to-completion: (1) What is the attitude to the issue of time-to-completion?; and (2) Of the proposed accelerators for reducing time-to-completion, which are utilized? [1]

3 Research Methodology

Twelve software package firms participated in this study. The sample is known as a convenience sample-- all firms are in the same major U.S. metropolitan region. The response rate for interview requests was over 70%.

The median annual revenues for the twelve firms was one million dollars with a high of $12 million. All firms developed products uniquely or predominantly for microcomputer-based platforms (rather than mainframe software). Seven of the 12 firms developed uniquely for IBM-compatible platforms. Eight of the 12 firms develop predominantly in the C or C++ programming languages (see Table 1 for further details of the sample).

Four criteria were used for sample selection to achieve the goals of: (1) breadth of product offerings, (2) threshold product complexity, (3) age of firm, and (4) minimal success. First, the firms' products cover a broad spectrum of the software package industry, from statistics through programming environments to artificial intelligence. Second, firms that develop relatively low complexity business applications software (e.g. dental record keeping) were *not* chosen. Several of the firms in the study were recognized in national periodicals for their innovative products. Several of the firms created new niche areas. More than half deal with complex product characteristics such as specialized algorithms, operating system and machine interfaces, real-time requirements, and data communications. Third, the firms were generally "young" (median years from release of first product was three years) so as to focus on the development/process aspects in early stages of the firms' development. Fourth, firms chosen had survived a minimal test of market success-- their software products were being purchased by customers.

Research process: Interviews were conducted with the key development personnel at the firms' offices. Semi-structured interviews were of two to four hours in length resulting in approximately 40 hours of interview transcripts with 18 developers. After the interviews, surveys were distributed to all team members (a total of 48 surveys). Thirty surveys were returned from nine of the firms -- a 63% response rate. All information in this report is carefully sanitized to ensure confidentiality.

[1] There are many interesting related questions which are *not* covered here: the study did not cover issues of product creativity or innovation processes. Nor does this study address whether short time-to-completion leads these software firms to success (e.g., first-to-market advantages): It is assumed that time-to-completion is a success factor. On a different point, there is little separation between organizational level analysis (the entrepreneurial firm) and product-level analysis (the package(s)) since companies in this stage of development are closely coupled with their product.

	Firm #	1	2	3	4	5	6	7	8	9	10	11	12
Application	Business/ office			☑		☑					☑		
	Specialized	☑	☑							☑			
	Systems/ utilities				☑		☑					☑	☑
	Graphics							☑	☑				
Distribution	Mass- market			☑	☑		☑					☑	
	Wide	☑				☑			☑		☑		☑
	Narrow/ high-end		☑					☑		☑			

Table 1: Characteristics of the firms in the study.

Firm	First software product	Subsequent software products
1	ND	24
3	26	in development.
4	5	24
5	13	none
6	36	11,12,13,13
7	ND	6,8,12,2
8	ND	19,10,9
9	48	18
11	24	18,36
12	38	9
Avg.	26.9	14.3

Table 2: Development time in months [ND = No Data].

Development methodology component	Category of implementation	Num. Firms
Specifications. Written concepts, requirements, specifications, and design documents were either nonexistent or ill-defined. When such documents existed they were often described as being brief.	None Informal, brief, a few pages Formal specifications	2 7 3
Formalized and distinct design stage. This stage was not clearly defined, delineated, or documented.	Informal, brief, minimal Formal and distinct design	8 4
Prototyping	Data not available None Little or rarely Prototyping actively used	1 3 1 7
Formalized test stage(s). In some companies, the testing phases were not clearly delineated from the build stage and tests were not formalized. Although gradual release and beta testing are widely practiced, their definitions differ widely.	Informal or non-standard Formalized phases	6 6

Table 3: Development methodology indicators for the firms in the study.

4 Findings

Time-to-completion data are presented in Table 2. Although the data are sparse, time-to-completion generally decreases by a factor of two from first product to subsequent (second+) products.

Time-to-completion was an amorphous concept at these firms. In interviews, only a few firms were able to articulate either a speed strategy, an awareness of time-to-completion as a managerial concept, or an explanation for their time-to-completion. Inspection of Table 2 suggests another inference: poor ability to recall product cycle-time. Several of the figures are multiples of twelve and are suspect as to rounding (some cycle-times were reconciled during interviews).

Deadlines, although widely used, do not seem to be of major importance in reducing time-to-completion (at one firm the interviewee conceded "we always slip"). Strong competitive pressure was a factor in only one firm that competes in a niche with high media coverage. For most firms stronger deadline pressures came from the existing client base which constantly demanded new features.

There were also indications of the importance of time-to-completion. In a trade-off exercise for success factors given to the development managers at three of these firms[2], all rated development speed as highest or tied with highest in importance. In a sense, there is a contradiction between the little the firms articulated about time-to-completion and the high value they placed on it. One explanation for this was given by one of the respondents who said that the development plan and milestones were "all in their heads" -- in other words, they were not explicit.

Presentation of the remainder of the findings uses the order of the six time-to-completion accelerators:

Time-to-completion accelerator #1: Development Methodology

Eleven of the twelve firms were not fully using an overall software development methodology. There was relatively little effort allocated to formalizing specifications, design, process documentation, or testing procedures (Table 3).

Several indicators of an absence of methodology were found. When asked for a description of the development model, some vague answers were given such as, "We rely heavily on prototyping." One president asserted, "We're not slaves to Yourdon." In several cases, it was difficult, if not impossible, to fit development approaches into a generic design-build-test sequence-- particularly early in the product history.

There are several explanatory factors for the little attention given to a development methodology:

- Many products began with, what has been called, *the accidental life cycle model* in which a custom system for one client is eventually turned into a software package. This haphazard process affected the overall approach toward a development methodology. Nevertheless, as Figure 1 depicts, almost all firms begin a process of formalization (albeit slowly) once they release version 1.0.
- *The entrepreneurial nature of the firms.* The new venture is faced with a stream of critical events that divert attention from the development methodology. Along with a shortage of money, new ventures operate with fairly short time horizons-- they need to take short-cuts. One president, explaining his view of development methodologies, said, "The question about methodology is whether you can run any faster by stopping." Young firms pride themselves on their flexibility. A development methodology, which hints of bureaucracy, is treated with suspicion.
- Three of the firms were strongly committed to *object-oriented development* and commented on their search for, or dissatisfaction with, current object-oriented design and development methodologies.
- The *software architecture* was enigmatic. On one hand it largely did not exist in tangible

2 *The ten-point exercise.* As the interviews progressed, lack of emphasis on time-to-completion became ever more perplexing. A trade-off exercise was added at the end of the interview with the last three firms (which were particularly successful from a financial perspective). They were asked to allocate ten points to four attributes by their relative importance to the success of the firm's products. The four attributes were: development speed, quality, development cost, and innovation. Speed was allocated-- by far-- more points at all three firms (5,5,4); innovation second (1,3,4) and quality, a somewhat distant third (2,1,2).

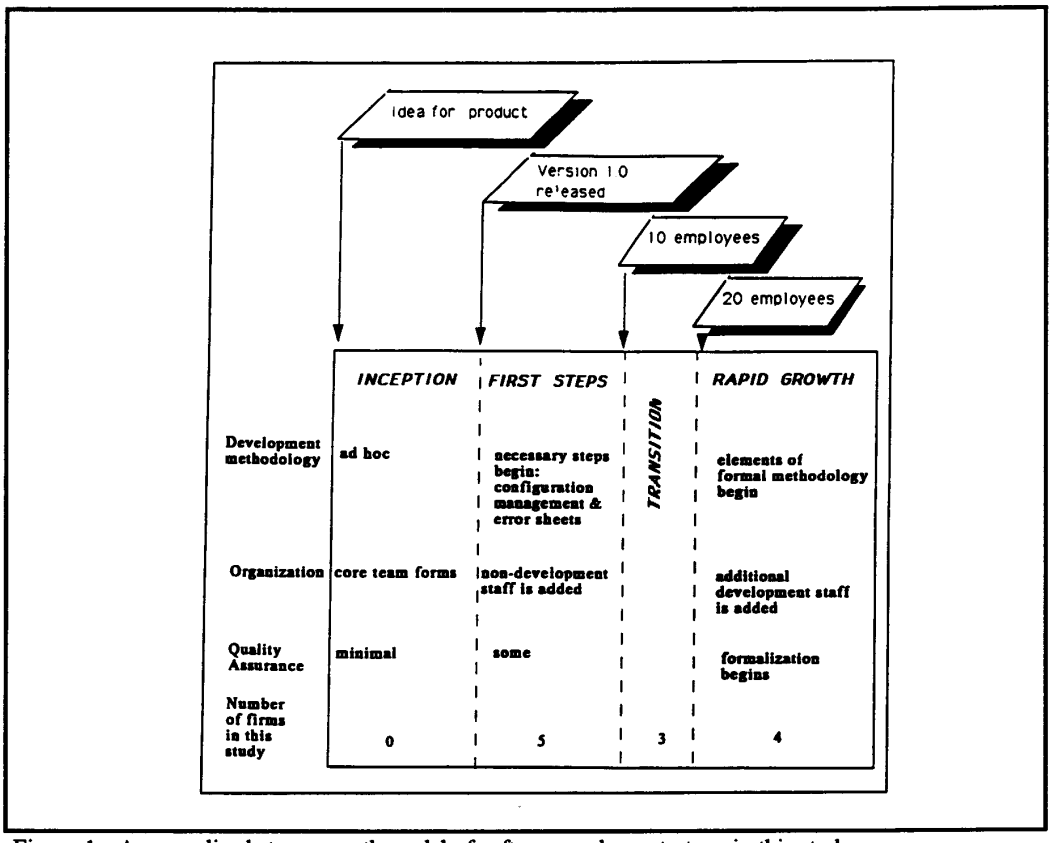

Figure 1: A generalized stage growth model of software package startups in this study.

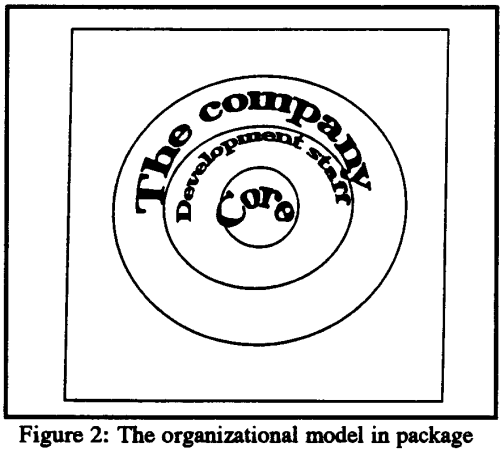

Figure 2: The organizational model in package startups

Firm #	Years since release of first product	Total number of employees... company -wide	in the develop. staff	in the core team
5	<1	6	3	3
12	2	8	3	3
8	2	8	4	3
3	2	5	3	3
9	3	13	7	4
2	3	20	6	6
1	4	10	6	3
4	5	45	5	4
6	6	70	45	4
11	6	57	35	6
7	7	50	11	4
10	9	20	11	5
	Avg	26	11.6	4

Table 4: Size of team relative to age.

form (i.e. on paper). On the other hand, as suggested by the description of the core team (see below), it was well understood among the members of the small, tightly-knit, core team without resorting to formalization.

While some methodological attributes were weak others were more promising:

- Although not always formalized, most firms made use of *prototyping*, proof-of-concepts, mock-ups, and demos to customers in order to test functionality. Prototyping may decrease time-to-completion while increasing the level of quality.
- In a sense these firms implicitly used *overlapping phases* [7, 17] as their development approach. The "theory" of overlapping phases prescribes product development in which engineering, manufacturing, and marketing act in parallel and in cooperation to complete the product. In many of the small firms in this study all three functions were represented by members of the core team. Data in Table 3 indicate that many firms make no use of traditional phased project planning, known as the waterfall method, in which phases are clearly delineated. However, some firms were not practicing overlapping phases, but rather, chaotic, poorly-planned development. Data were not sufficient to differentiate between these types of firms.

Time-to-completion accelerator #2: Development Team

If there is one factor that may best explain time-to-completion performance in software package startups -- it is the collection of people and group characteristics which comprise the core development team (referred to from heron as the *core team*). The small core teams are homogenous, highly motivated, possessing in-depth experience sets. The teams have a history of working together in a loose organizational structure.

The in-depth interviews serendipitously led to identification of this inner team (see Figure 2). It was validated in all of the twelve startup firms.

The core team is comprised of the architects and principal programmers developing the software product. It is the software startup equivalent of a cross-functional team. The core team sets the tone and makes decisions for the products and the firm as a whole. Its members have internalized the overall product design, its use, and its long-term destiny. The core team is a cohesive unit and communication links among members are dense. Individual core members are generally involved in development for the entire duration of the project.

Core team size. All firms had relatively small core team sizes-- with a median of four (see Table 4). Small core teams reduce communication and coordination time. The key developers in these firms recognized that the small core team was important to the development process as the quotes further below suggest. The core team size grew only slightly with (1) the time the firm was in existence (measured as the number of years since release of the first product), and (2) the total number of employees (Pearson correlation: core size and total size $=.48$ (n.s.); core size and firm age $=.53$ ($p<.10$)).

The data also suggest that for young firms (i.e., the first four firms in Table 4) the core team is essentially the only important organizational boundary. As the firm grows, the second layer, that of the development staff, begins to form and expand (depicted in Figure 1).

Team composition. The individuals in the dozen core teams were fairly uniform in age (most in their 20s and 30s), gender (all were male), and country of origin (only one non-American), but varied considerably in education. These were meritocratic work units where it was common to find a developer without a college degree working side by side with a Ph.D. Motivation levels for achievement are very high, "We've got the best [programmers] on the planet... [they want to excel] ... not for their wives or for the money, but so that their peers would say --*WOW*--." Motivation may also be driven by monetary inducements. Of respondents in the core development team 87% had either equity or stock options. Additionally, six of eight teams responding considered their wages to be at or above market rates.

President of firm #11: "*There are six of us who know the intimate details of what [the product] does.* We're the keepers of the code. Six people-- all of whom have extremely high bandwidth of communications. There are at most 100 pages of design... and no one knows where that is anymore-- its in peoples' brains."
President of firm #5: "I attribute our success to one key factor-- communication-- [the 3 core team members were] in one room-- *like one brain*-- no phone calls, no interruptions."
Director of Development at firm #10: [One of the factors for achieving speed of development is] "*The teams are very small.* The largest team is the [database] module which has four... Its my experience ... that teams greater than 5-6 tend to thrash a lot because of communication problems and that the communication aspects of the team start to seriously impact the development schedule. We already see that with [the database] module with 3-4. There is a definite impact. [John] spends more and more of his time managing those people and keeping them focused and directed and spending less of his time actually doing coding."
President at firm #12 "*All we need is at most 4-5 top developers* even if we get to be a $10 million company. We couldn't survive with anymore."

Experience of core individuals. The individuals in the core teams possess two strong experience sets: technical/ computational experience and application-specific knowledge. Curtis, et al. [6] point out that one of the key problems in software development is not technical expertise but the lack of application-specific knowledge (which is largely gained from work experience).

- Technical/computational capabilities were clearly evident: 85% of the degrees were computer related. Most firms had at least some developers who were "hackers" within the core teams. Core team members had been coding for thirteen years on average.
- Application-specific knowledge is generally derived from work experience. At these firms, with a very narrow product line (sometimes just one product on one platform), a core member will have developed deep application knowledge within a few years. The team members had been with the firm for, on average, 4.5 years and inside the core team for 3.8 years. These numbers reflect only part of the experience set of each individual in the specific product domain. There is anecdotal evidence for deeper experience in application-specific domain. For example, in one firm each of the three most senior members of the core team spent several years, prior to starting up their own firm, in a larger company which was in the same application area as their current line of products.

Duration of co-work. Individuals who have a history of working *together* are more productive. The above data measuring experience also measured team history. The average core team member had been with the firm for over four years and many team members had worked together with their peers before work in the present company. However, there was some variation on duration of co-work: several teams went through periods of turnover (specific data were not collected).

Team structure. The team structures are amorphous. None of the classic structures such as "egoless team" (purely democratic) and "chief programmer" were descriptive of any of the core teams. However, there still was considerable variation. In some teams there was a strong player, who set the tone in terms of his technical or visionary skills. In other teams, there was no dominant leader.

Time-to-completion accelerator #3: Resource allocation for labor and capital assets

Software startups have one major flexible resource: *labor*. Firms of this size rarely add labor to a project, but as the development cycle nears its end, developers devote extra time and effort. In effect they are adding resources which, to a large extent, have no cost. Only one developer (out of thirty respondents) worked 40 hours/week in "normal" periods. During "crunch" periods 87% of these developers worked more than 56 hours a week and 47% worked more than 71 hours/week.

Capital assets: The startups invested relatively little capital in software tools (and none used CASE). The data on tools spending

indicate a very large variance-- two orders of magnitude: half the firms spent less than one thousand dollars/ per year/ per developer on outside tools. Some firms were resourceful in finding software tools: they used utilities from freeware and shareware sources, from customers, and as trials from vendors. Some firms invested in in-house developed utilities and testing programs. Three firms spent more on in-house developed software than on outside software (by their own accounts). Low spending on tools can be explained by a distrust in others' software that is endemic to many hackers and engineers.

Time-to-completion accelerator #4: Project Management

Six companies responded to the long survey and gave indications of their relative usage of Project Management (PM) tools, techniques, timelines, scheduling, etc. The three smaller firms did virtually nothing in this area. The larger of these six perform PM functions most of the time, although only one uses a PM software tool all the time.

Meetings. Most key developers attended 1-2 weekly meetings. The organic culture and small size serve to minimize meetings. Coordination took place through less formal channels.

Time-to-completion accelerator #5: Risk Analysis

There were almost no indications of any systematic approach to risk analysis (although risk items were not queried directly). The discussion here is based on one operationalization of risk: formulating trade-offs at the *end* of the development process in order to reduce time-to-completion. The trade-off at that point is between "quality" and "functionality." While some firms were aware of the trade-off and had made decisions based on it, others had no policy or history regarding such a trade-off. Thus, even an elementary aspect of risk analysis was absent at firms where this trade-off strategy was never articulated.

Time-to-completion accelerator #6: Incremental Innovation

Most of the firms adhered to the prescription of incremental innovation. They built product families in which products were closely related or derived from one another. They introduced new versions at fairly regular intervals with manageable updates. Of the one dozen firms, six firms' first product represented a major innovation within their niche or established a new niche. Only one firm waited until the second product to introduce a major product innovation.

Another factor affecting time-to-completion: Quality

Testing software is an enormously complex activity and is costly in calendar time. Testing was compromised in many of these firms. Quality assurance, in the broader sense, was largely absent. This is not surprising given the weak methodological underpinnings discussed above. Quality (the absence of bugs) frequently lost in the intentioned or unconscious trade-offs between "units of quality" and "units of time." For a further discussion of quality findings see [3, 5].

5 Summary

Of the six accelerators analyzed in this study one presents itself as having the largest impact on time-to-completion, and seems to be critical for the overall success of the firm. That accelerator is the *team*, particularly the *core team*. Most of the team characteristics that are proposed in the literature as leading to reduction in time-to-completion were present in the startup teams in this study (cross-functional, small size, selective staffing, motivation, full-time involvement). The small core team was recognized as an asset by the individuals within the software startups. The core teams were homogenous, highly motivated, possessing in-depth experience sets. The teams had a history of working together in a loose organizational structure. Two other accelerators seem to play a significant role in time-to-completion reduction: use of extra effort during crunch times and the building of similar products (incremental innovation).

The presence of several other time-to-completion accelerators appears to be weak in software startups: they did not fully use development methodologies, they made little use of software tools for increasing

productivity, there was a weak sense of risk analysis, and project management indicators were weak.

The overall awareness of time-to-completion seems ambivalent and contradictory. On one hand, deadlines were soft, there was little articulation of development speed strategy, and there was poor recall of calendar times. Yet the high degree of effort that the developers expanded during crunch times suggests that development speed was important.

6 Research implications

Implications are presented with caution since the purpose of this study was exploratory and the sample in the study was small.

Implications for software package startups. The founders of software package startups need to take proactive steps from the time of the firm's inception to create successful core teams. One talented developer-entrepreneur may not be enough to keep a product viable, if that entrepreneur fails to build a core team around him/her. Investors and venture capitalists considering financial involvement, and corporations seeking alliances with a software package startup, need to look for a well-formed, experienced core development team--and not just a set of product ideas and features.

Are there technology transfer opportunities from this research?

Yes, for traditional Information Systems (IS) development environments. Some elements of the core team concept can be carried over to larger organizations and larger development units. The traditional project-oriented units need to be changed to create small core teams organized by application function. Such teams will foster the cohesive vision while nurturing the necessary application-specific skills.

Indeterminate, for international technology transfer to software startups in Europe and Japan. Can such foreign startups emulate these success factors to strengthen their competitive abilities? The success of U.S. firms seems to revolve around the team. Much of what was discussed in relation to the team is culturally-driven and may be difficult to replicate outside of North America [10]. Further research is required on this question.

■■■■■

This study was supported by research grants from the Center of Innovation Management Studies (CIMS) at Lehigh University and The American University. It is condensed from a March, 1993 research summary report. I am indebted to Dr. Barbara Bird and Dr. Shirley Becker who worked closely with me throughout.

References

[1] Bower, J.L. Fast cycle capability for competitive power. *Harvard Business Review*, 66: 110-118, November-December 1988.

[2] Brown, W.B. and Karagozoglu, N. Leading the way to faster new product development. *Academy of Management Executive,* 7:1:36-37, 1993.

[3] Carmel, E., Becker, S.A., and Bird, B.J. "Go full speed ahead in coding and we'll clean after you later": Do young software product development firms practice Quality Assurance? *Proceedings of 2nd International conference on software quality*, Research Triangle Park, NC., October, 1992.

[4] Carmel, E. A discussion of special characteristics of software package development life cycle models. *Software Engineering Notes* April, 1993.

[5] Carmel, E. How quality fits into package development. *IEEE Software*, September, 1993.

[6] Curtis, B., Krasner, H. and Iscoe, N. A field study of the software design process for large systems. *Communications of the ACM,* 31:11:1268-1287, 1988.

[7] DeGrace, P. and Stahl, L.H. *Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms.* New York: Yourdon Press, 1990.

[8] Gold, B. Approaches to accelerating product and process development. *Journal of Product Innovation Management,* 4:81-88, 1987.

[9] Grahm, D.R. Incremental development: review of non monolithic life-cycle development models. *Information and Software Technology*, 31:1, 1989.

[10] Greenbaum, J. and Johnston, M. Euro-entrepreneurs in a bind. *Upside*, 5:10:65-82, October, 1993.

[11] Griffin, A. Metrics for measuring product development cycle time. *Journal of Product Innovation Management*, 10:112-125, 1993.

[12] Grudin, J. Interactive systems: bridging the gap between designers and users. *IEEE Computer*, April, 1991.

[13] Millson, M.R., Raj, S.P. and Wilemon, D. A survey of major approaches for accelerating new product development. *Journal of Product Innovation Management*, 9: 53-69, 1992.

[14] National Trade Data Bank. *U.S. Department of Commerce Documents*, International Trade Administration, U.S. Industrial Outlook, Chapter 27, CD-ROM, 1992.

[15] Price Waterhouse *Software industry business practice survey*. Boston, MA., 1992.

[16] Smith, P.G. and Reinersten, D.G. *Developing products in half the time*. New York: Van Nostrand Reinhold, 1991.

[17] Takeuchi, H. and Nanaka, I. The new product development game. *Harvard Business Review*, January-February, 1986.

[18] Teach, R.D., Schwartz, R.G., and Tarpley, F.A. Differences in managerial work between entrepreneurial and professional managers in the software industry. In *Frontiers of Entrepreneurial Research, Center for Entrepreneurial studies*, 1988.

[19] Vesey, J.T. Speed to market distinguishes the new competitors, *Research Technology Management*, November-December, 1991.