Rapid System Development (RSD) Methodologies: Proposing a Selection Framework

Rashmi Jain, Stevens Institute of Technology
Anithashree Chandrasekaran, Stevens Institute of Technology

Abstract: The current global customer trend requires companies across domains to reduce their product development lifecycle. As a result the exploration of methodologies that will support rapid system development has been gaining importance. The primary focus of this article is to provide a framework for comparative analysis of rapid system development methodologies. The purpose of this framework is to help the project managers and systems engineers choose and tailor an appropriate rapid development methodology to suit their development context and environment. Toward this, the framework identifies and defines a set of critical rapid development attributes. The article redefines rapid system development as adopting methodologies, tools, and techniques that can introduce rapidity into the system development processes while optimizing the success factors of development. The success factors are specific to the system under development and they depend on the system, product line, organization, and customers. Some of the common success factors are return-on-investment (ROI), cost of ownership, other performance factors, and customer satisfaction. The article provides a fundamental discussion on the current rapid system development methodologies, metrics, tools, and techniques.

Keywords: Product Development, Research and Development, Systems Development

EMJ Focus Areas: Innovation and New Product Development

Consumer demand, market window, and return on investment are the three terms heard every day in today's globally competitive market. Every company wants satisfied customers, but every consumer and user has a set of specific demands. This has made us realize that one size does not fit all. This realization has exponentially increased the number of features provided with the products/systems developed each year. The traditional approach of system development is no longer consistent with the objectives of a development team focused on responding to and managing market expectations. This inconsistency calls for novel approaches to support quick time-to-market and tailor development processes appropriately in order to address the objectives of a rapid system development team. Some of the emerging rapid system development approaches have been based off specific experiences of development teams. As a result they have come up with methodologies that help achieve rapidity and have demonstrated project success by applying these methodologies; however, an application of a development methodology may sometimes not result in project success or in achieving the desired outcomes. This may be because of the differences in the system environment and supporting strategies that lead to project success. The process of selection and tailoring of rapid development methodologies depends significantly on the context and environment of system development. The selection process requires an understanding of how certain rapid development attributes are addressed and supported in each methodology. Our research focuses on providing a framework that could help in selecting a rapid development methodology appropriate with the strategy of the system development team and the management. This article discusses some popular rapid development methodologies and defines related development attributes. It further describes a framework for a more refined process of rapid development methodology comparison and selection.

The article redefines rapid system development and provides an overview of the current rapid development methodologies. The article then proposes a framework that can be used in the selection process of an appropriate rapid development methodology. The development attributes of the framework are defined. The article also provides an overview on some of the current metrics, tools, and techniques that could be used along with these rapid development methodologies. The article concludes by identifying common patterns that exist between these rapid development methodologies. The article also identifies some areas of future work required to support further tailored application of these methodologies.

Rapid System Development

Time to market and window of opportunity for a product have been shortening exponentially in the present competitive global scenario. The life cycle of product/system development begins from market research of a concept and ends in system deployment and operation. The competition, resources, strategy, and budget create a need to take crosscuts without endangering the cost, schedule, and quality attributes of the system and thereby achieving rapidity in the development of systems. Rapid Systems Development (RSD) is a structured approach with rigid limits on development time frames. The motto of RSD is faster, better, and cheaper. RSD is a way or method that introduces agility and rapidity in the system development process. Our scope of defining rapid or

Refereed research manuscript. Accepted by Associate Editor Keating.

agile development is limited to product/system development and does not include enterprise-level or organization-level rapidity or agility.

Agility can be defined as,

- "…dynamic, context-specific, aggressively change–embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive "storms". It is about succeeding and about winning, about succeeding in emerging competitive arenas, and about winning profits, market share, and customers in the very center of the competitive storms many companies mow fear" (Goldman et al., 1995)
- "…the ability to both create and respond to change in order to profit in a turbulent business environment" (Highsmith, 2002)
- "…the ability to thrive in a time of uncertain, unpredictable and continuous change" (Dove, 1999)
- "…the total integration of business components" (Kidd, 1995).

Agility itself has its own dimensions such as,

- "The *four principles* of agility are delivering value to the customer, being ready for change, valuing human knowledge and skills and forming virtual partnerships" (Goldman et al., 1995)
- "There are *four core concepts* of agility namely, core competence management, capability for reconfiguration, virtual enterprise, and knowledge driven enterprise" (Yusuf et al., 1999).

Rapid System Development can be defined based on definitions and dimensions similar to agility. One of the key definitions of RSD includes: "the ability to turn an initial system concept into a working system that adds value to business operation in a *short period of time*" (Howard, 2000).

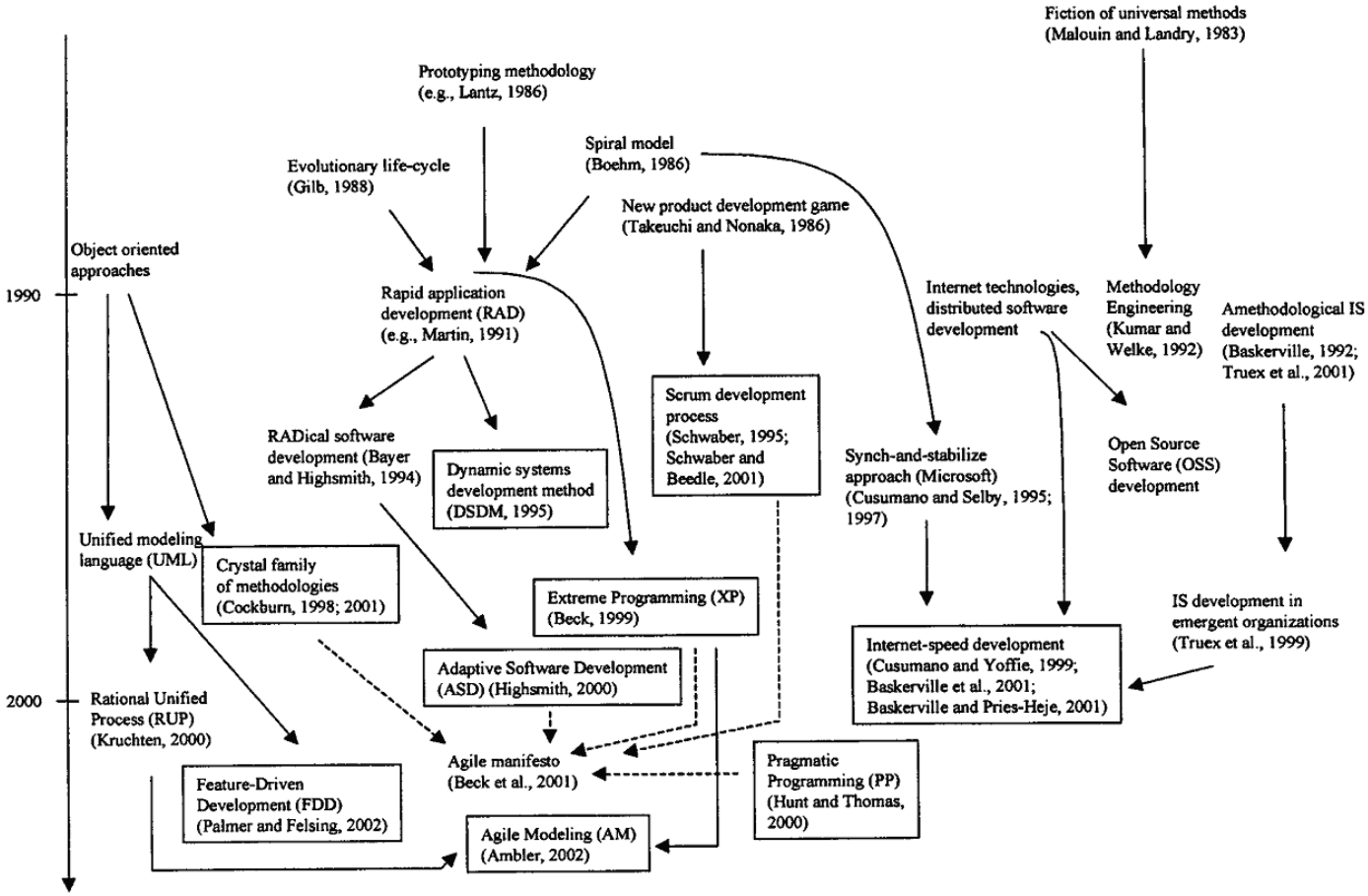
Rapid Application Development (RAD) is a software domain equivalent of RSD. RAD is defined as a systems development framework that offers the possibility of fast, flexible, and "tailored" business information systems. It is the effective and efficient creation of application systems within a full-fledged strategy.

We define Rapid System Development as,

- …adopting methodologies, tools, and techniques that can introduce rapidity into the system development processes while optimizing the success factors of development. The success factors are specific to the system under development and they depend on the system, product line, organization, and customers. Some of the common success factors are return-on-investment (ROI), cost of ownership, other performance factors, and customer satisfaction. The word *system* in here represents product, service, and system.

Rapid system development methodologies have been around for more than two decades. In the last 5-10 years there has been tremendous support and popularity for RSD process by the stakeholders of system development realizing its need and importance. There has been an increase in the number of RSD methodologies and models introduced in the last 5 years addressing today's market challenges. This trend of rapidity in system development can be observed in Exhibit 1. This figure

Exhibit 1. The History of Rapid Development (Abrahamsson, 2003)



analysis of the methodologies of rapid development. This article discusses the existing RSD methodologies and models, and provides a comparison chart that could help in the decision process for selecting and tailoring an RSD methodology for a system development project.

Rapid Development Methodologies

Rapid development relies on principles such as useful product delivery, reliability, collaboration, technical excellence, simplicity, and adaptability, but there is no one way to achieve all this. "Agile is the way of thinking, not a particular practice" (Highsmith, 2002). The drivers of agility are quality, speed to market, widening customer choice and expectation, competitive priorities of responsiveness, new product introduction, delivery, flexibility, concern for the environment, and international competitiveness (Goldman et al., 1995). There are several rapid development methodologies that have developed in response to these drivers. These methodologies have been built upon some existing rapid development principles and core concepts. Our use of the term *methodology* in this context includes "a body of practices, procedures, and rules used by those who work in a particular field or specialty" (McConnell, 1996).

Methodology is a disciplined way of approaching a system. There are various methodologies for rapid development now being followed. One methodology cannot fit all types of projects. Depending on the time, size, cost, and technology, one can decide which practices and lifecycle models are applicable and most suited to the system under development. The generic models that layout a plan for system development are called lifecycle models. A lifecycle model is a perspective model of what should happen between the first glimmer and last breath (McConnell, 1996). The main function of a lifecycle model is to establish the order in which a project specifies, prototypes, designs, implements, reviews, tests, and performs its other activities. Each system development methodology starts with planning for a system development lifecycle model. The rapidity/agility in system development is achieved by reducing the time of one or more phases of system development such as design, development, manufacturing, and systems integration or compressing and/or collapsing an entire phase. In other words, how easy and flexible it is for the development methodology to change its order of activities in response to market demands or other pressures of rapidity will determine the agility/rapidity of the methodology.

Our review of the current rapid development methodologies can be classified under four generic categories based on how the life cycle is defined in each methodology. The four categories are Modified Waterfall, Evolutionary Development, Design to Cost, and Design to Tools.

Modified Waterfall

For decades Waterfall model has been the traditional development lifecycle model. The idea of sequencing the stages of development, a stage is triggered by the completion of the previous stage, has always allured the stakeholders of product, system, service development. But the driving forces of rapid made them rethink the tradition. As a result iteration, feedback, and concurrency were introduced in the waterfall model. A lifecycle model with traditional waterfall lifecycle stages fall under the generic category of Modified Waterfall. There are various instantiations of Modified Waterfall based on the system and its environment. The popular instantiations follow.

Modified waterfall with overlapping phases (Sashimi Waterfall). Waterfall with overlapping phases has concurrency of the different development phases. This is ideal for a project that gathers more insights as we move on further into the development cycle.

Modified waterfall with subprojects (component based development). In waterfall with subprojects system level, detailed design begins before completing the architectural design and development, and testing starts before detailed design is completed. Development effort is broken down into subprojects. This is analogous to component-based development.

Modified waterfall with risk reduction. In the waterfall with risk reduction methodology the risk-reduction spiral is put at the top of the waterfall to address the requirements risk. The requirements analysis and architectural design are addressed as part of the risk reduction phase. The risk-reduction spiral can also be used to address other risks associated with system development. If there is a high-risk functionality identified in a system architecture that is critical to the system, as there may be other functional components dependent on it, then the risk-reduction cycle will be applied in completing the development of such a critical functionality before attending to the remaining functionalities. Once this risk-reduction spiral is completed, the rest of the development will follow a waterfall model methodology.

Spiral. The spiral model is a risk-oriented lifecycle model that breaks the project into modules. Each module addresses one or more risks until all the major risks are covered. After all the risks have been addressed, the spiral model terminates as a waterfall lifecycle model would. To start with, explore the risks on the small scale in the middle of the spine and make plans to handle it, and then commit to it and move to the next iteration. Successive iteration moves the project to a larger scale (McConnell, 1996). It is complicated but can be used with other life cycle models. This approach provides good management with check points at the end of the iteration.

Evolutionary Development

In evolutionary development, life cycle of development is phased and iterative. In each phase (cycle of development) the concentration is on building a feature of the total system capability or functionality. Each phase of development includes build-test-integrate-validate-control and feedback cycle. Stakeholder participation and feedback in each evolution or cycle is the key to success in evolutionary development. An architectural roadmap of the development is planned early and revised upon stakeholder feedback in each phase. Some of the common evolutionary development methodologies follow.

Evolutionary prototyping. In evolutionary prototyping the system concepts are developed as we move through the project. We begin with developing the most visible aspects of the system, demonstrate that to the customer, and then continue to develop the prototype based on the feedback. The point at which the customer agrees with designer that the prototype is good enough, any remaining works are completed on the prototype and are released as the final product. Evolutionary prototyping is very useful when the requirements are rapidly changing or there is no clear understanding of the end product. It is also useful when the designers are not sure of the architectures or the tools. It produces

visible signs of progress when there is a strong demand for speedier development. The main disadvantage is that the duration of the project is difficult to calculate at the beginning.

Staged delivery. In the staged-delivery model the system is shown to the customer in successively refined stages. Unlike prototyping, what exactly is going to be built is not known before starting to build. The system is delivered in successive stages through the project and not in one release like prototyping. It is also called incremental implementation. The advantage is that it allows putting useful functionality into the hands of the customers earlier rather than delivering the complete product at the end of the project. The main disadvantage is it needs very careful planning both from the management and technical side.

Evolutionary delivery. Evolutionary delivery is a type of staged delivery where a product is developed, showed to the customer, and refined based on the customer's feedback. It straddles the ground between the evolutionary prototyping and the staged delivery. Initial emphasis is given to the core of the system which consists of lower level system functions that are unlikely to be changed by customer feedback.

Feature Driven Development. Feature-driven development (FDD) (Coad et al., 2000; Datar et al., 1997) is a process-oriented development method for developing business critical systems. The FDD approach focuses on the design and building phases like the staged delivery approach. This approach follows an iterative development and also adopts some of the industry best practices to handle the challenges posed by each feature that is being developed. The specific blend of these ingredients makes the FDD processes unique for each case. It emphasizes quality aspects throughout the process and includes frequent and tangible deliveries, along with accurate monitoring of the progress of the project. FDD's overall model involves forming the modeling team, conducting a domain walkthrough, studying documents, developing sub-team models, developing a team model, logging alternatives, and inspecting log action items. Plan, track, and report is the key to FDD (Highsmith, 2002).

Design to schedule. The design to schedule lifecycle model is similar to the staged delivery model, but the final stage may not be possible if the deadline is reached. The stage at which the deadline is met becomes the final stage – hence the stages are prioritized. The features are prioritized and implemented in the early stages. The primary disadvantage is the wastage of time in designing and architecting systems that are not going to be shipped due to end of the stages when the deadline is reached (McConnell, 1996).

The 4CC framework. The four cycle of control framework (Rautiainen, 2002) is a rapid development process model for software development in small scale industries. The four cycles of control combines the business and process management. The four cycles of controls are: 1) strategic release management that provides the interface between business management and product development, 2) release project management that handles the development of individual product versions, 3) iterative management that deals with the incremental development of product functionality with in release project, and 4) mini-milestones that are used for daily and weekly task scheduling and monitoring to get an idea of the status of system during development. This framework can be used to assess the current state of the project as well as serve as a blue print for improving or reengineering the product development management.

Design-to-Tools

Design to tools is used in time sensitive environments because tools have become more flexible and powerful. The idea behind this model is that a capability or functionality is included into the product only if the development tool supports it. This model can be combined with other flexible lifecycle models to get the best results. The disadvantage of this approach is that it is easy to lose a lot of control over the product. It is not possible to implement all the features needed in the way it's needed due to the limitation of the tools (McConnell, 1996).

Design-to-Cost (Value Engineering)

Design-to-cost is a popular technique for controlling costs. Design-to-cost is a method of controlling cost by establishing cost goals at specified levels of a work breakdown structure and then requiring the project to make trade-offs that will ensure that the system built will meet those cost goals (Brennan et al., 1992; Dean, 1990). This approach is very similar to the concept of Value Engineering. Value engineering (Howard, 2000) aims to eliminate all work in a project that does not add value to a product. Value engineering is an iterative process, the aim being to arrive at the simplest and easiest-to-implement set of requirements and system designs. It helps in keeping things simple and easy to maintain.

Review of Current Rapid System Development Methodologies

Exhibit 2 provides a comparison of the different rapid development methodologies. It summarizes our findings of this research. It reviews and compares eight different RSDI methodologies against ten attributes that are relevant and significant to Rapid System Development. Each attribute is rated on a three-level scale from 'Attribute Absent' to 'Attribute Present' against each of the eight methodologies. These attributes are:

- **Adaptability –** the ability of the methodology to be flexible with the changing scope of the system functionality in order to meet the goal of the project
- **Cost Scoping** – the ability of the methodology to define and address the scope of the project in terms of a given cost
- **Time Sensitivity** – the ability of the methodology to respond to changes in the schedule of the project
- **Evolutionary** – the ability of the methodology to address the development process from an evolving perspective in terms of staged delivery, build frequency, releases etc.
- **Technology Tradeoffs –** the ability of the methodology to optimize on technology choices to meet the goal of the project
- **Return-on-Investment (RoI) –** the ability of the methodology to analyze and address the preferred Return of Investment (ROI) on the system
- **Testing and Integration** – the ability of the methodology to address the testing and integration of the system in a comprehensive manner
- **Requirements** – the ability of the methodology to progressively lock down the requirements
- **Risk Reduction –** the ability of the methodology to address and contain different kinds of risks
- **Iterative Development –** the ability of the methodology to involve iterative and concurrent development process.

Exhibit 2. RSD Methodologies Review Matrix

Methodologies	Classification	Adaptability	Cost Scoping	Time Sensitivity	Evolutionary	Technology Tradeoffs	ROI	Testing & Integration	Requirements	Risk Reduction	Iterative Development
Modified Waterfall (sub systems with overlapping phases)	Modified Waterfall	⊗	○	⊗	○	⊗	⊗	●	●	○	⊗
Modified Waterfall (Risk reduction)	Modified Waterfall	⊗	⊗	○	⊗	⊗	⊗	●	⊗	●	●
Evolutionary Prototyping	Evolutionary Development	●	○	○	●	○	⊗	●	●	⊗	●
Staged delivery / Incremental Implementation	Evolutionary Development	●	⊗	●	●	⊗	●	●	⊗	●	●
Feature-Driven development	Evolutionary Development	●	○	○	●	○	⊗	●	●	⊗	●
Design to schedule	Evolutionary Development	○	●	●	●	⊗	○	⊗	○	⊗	○
Design to tools	Design to tools	○	⊗	⊗	○	○	○	●	○	⊗	○
Design to cost	Design to cost	○	●	●	⊗	●	●	⊗	⊗	●	○

● Attribute Present ○ Attribute Absent ⊗ May be

On comparing the eight RSD methodologies in terms of their ability to support the ten different development attributes, we found that the Staged Delivery or Incremental Implementation stands out as the leading RSD methodology to be pursued for achieving rapid system development. It emerged with seven of the ten attributes in the "Present" and remaining three in the "May be Present" category.

Closely following was the Design to Cost methodology which was found to be covering five of the attributes with "Attribute Present" category and three with "May be Present" category.

Much to our expectation, overall the broad category of Evolutionary Development methodologies emerged as the major source of rapidity with all popular methodologies included such as Feature Driven Development, etc.

Rapid Development Metrics

Measurement is a practice that has both short term motivational benefits as well as long term cost, quality, and schedule benefits. It overcomes the problem arising due to poor estimates, poor scheduling, and poor progress visibility. Measurement activity should be managed by a separate objective group but with a high level of commitment.

The demonstration of rapidity or agility of development is ridden with the challenges of coming up with credible categories of metrics. The ten different development attributes used for comparing the RSD methodologies in Figure 2 have not been researched enough to define their units of measurement (metrics); however, it will be of tremendous value to define these attributes in terms of how to measure them. This will enable the project managers' to keep track of the level of rapidity in their development process as well as know what aspects are weak in supporting rapidity. The current literature on the measurement of rapid development processes has focused on measurement techniques based on time and volume statistics and on development flexibility. In this section we are summarizing some measurement techniques that may be relevant to rapid development without going into the details of their application. These are:

- Flexibility Index (Thomke and Reinertsen, 1998)
- Probabilistic Model (Messica and Mehrez, 2002)
- Hazard Function Model (Datar et al., 1997)
- Correlation Model (Callahan and Brian Moretton, 2001)
- Effectiveness and Efficiency Measure (Goldense, 1997)

Rapid Development Tools and Techniques

We have discussed the need and importance of RSD, what aspect of development need to be addressed to achieve RSD, and to what extent these need to be emphasized to meet the targeted levels of rapidity. The next question that arises is, "How can it be achieved?" There are tools and techniques that help to reduce the duration of the development phases, thereby increasing the rapidity of the development cycle. A prototype can be built using automated tools like CAD. These tools help in easy capture and traceability of requirements. In software development there are rapid development languages that help in quick and fast development. Object-oriented languages is an important example of rapid development software language

The tools and techniques of RSD act like catalysts to produce the desired agility in a given environment with the right order and proportion of improved system development process and organizational process. Most of the rapid development tools and techniques tend to focus on either specific stages of development or on specific aspects of the development environment. In the latter category the focus is commonly around teaming, leadership, management skills, and feedback-review sessions. Some of the popular agile or rapid techniques that are currently being applied in software development along with the existing RSD methodologies are: timebox development, joint application development (JAD), adaptive software development, agile modeling, crystal family, dynamic systems development method (DSDM), extreme programming (XP), internet-speed development, and scrum.

Conclusions

An analysis of the above discussed rapid development methodologies reveals certain important common patterns. These patterns form the basis for achieving rapidity in the system development process. They are:

- RSD&I methodologies are based on life-cycle models
- they focus on looking at "rapidity" from an organizational perspective
- these methodologies progressively lock down requirements
- they focus on rapidity through prioritizing based on "value"
- they use overlapping/concurrent development phases
- Iteration is emphasized
  - "scoping" drives development with stakeholders' involvement

Most of the literature on RSD from a development perspective has recently evolved from the information technology (IT) field. Though there are a few rapid development methods and techniques in other fields, they are either specific to a company or to a particular product line, thus limiting their application as a case study example. Generic frameworks have to be developed for rapid system development in non-IT sectors. Also there is very limited work on metrics of "rapid" and "agile". More focus is needed to develop metrics to measure rapidity, and on metrics to confirm that as a result of reduced development effort the system functionality is not compromised and that the system meets its intended use and remains within its bounds.

References

Pekka Abrahamsson, Juhani Warsta, Mikko T. Siponen, Jussi Ronkainen, "New Directions on Agile Methods: A Comparative Analysis," *Proceedings of the 25th International Conference on Software Engineering* (May 2003), pp. 244-254.

James R. Brennan, Jerrell T. Stracener, "Designing to Cost Effectiveness: Enhancing Quality," *Proceedings of the Reliability and Maintainability Symposium* (January 1992), pp. 44-52.

John Callahan, Brian Moretton, "Reducing Software Product Development Time," *International Journal of Project Management*, *19*:1 (January 2001), pp. 59-70.

Peter Coad, Eric LeFebvre, Jeff De Luca, *Java Modeling In Color With UML: Enterprise Components and Process*, Prentice Hall (2000).

Srikant Datar, Clark Jordan, Sunder Kekre, Surendra Rajiv, Kannan Srinivasan, "New Product Development Structures and Time-to-Market," *Management Science*, *43*:4 (April, 1997), pp. 452-464.

Edwin B. Dean, *The Design-To-Cost Manifold,* NASA Langley Research Center (1990).

Rick Dove, "Knowledge Management: Response Ability and the Agile Enterprise," *Journal of Knowledge Management*, *3*:1 (January, 1999), pp. 18-35.

Bradford L. Goldense, "Motivators and Metrics for Product Development," *Professional Program Proceedings of Electronics Industries Forum of New England* (May 1997), pp. 67-83.

Steven L. Goldman, Roger N. Nagel, Kenneth Preiss, *Agile Competitors and Virtual Organizations – Strategies for Enriching the Customer Needs*, Van Nostrand Reinhold (1995).

Jim Highsmith, *Agile Software Development Ecosystems*, Addison Wesley (2002).

Alan Howard, "Rapid Application Development: Rough and Dirty or Value for Money Engineering," *Communications of the ACM*, *45*:10 (October 2000), pp. 27-29.

Paul T. Kidd, *Agile Manufacturing, Forging New Frontiers*, Addison-Wesley (1995).

Steve McConnell, *Rapid development - Taming Wild Software Schedules,* Microsoft (1996).

Avi Messica, Abraham Mehrez, *Time-to-Market, Window of Opportunity, and Salvageability of a New Product Development*, Wiley InterScience (2002).

Stephen R. Palmer, John M. Felsing, *A Practical Guide to Feature-Driven Development*, Prentice Hall (2002).

Kristian Rautiainen, Casper Lassenius, Reijo Sulonen, "4CC: A Framework for Managing Software Product Development," *Engineering Management Journal*, *14*:2 (June 2002), pp. 27-32.

Stefan Thomke, Donald Reinertsen, "Agile Product Development: Managing Development Flexibility in Uncertain Environments," *California Management Review*, *41*:1 (Fall 1998), pp. 8-30.

Yahaya Y. Yusuf, Mansoor Sarhadi, Angappa Gunasekaren, "Agile Manufacturing, the Driver, Concepts, and Attributes," *International Journal of Production Economics*, *62*:1-2 (May 1999), pp. 33-43.

About the Authors

Rashmi Jain is Associate Professor of Systems Engineering at Stevens Institute of Technology. Dr. Jain has over 15 years of experience of working on Information Technology (IT) systems. Prior to joining Stevens she was with Accenture (formerly known as Andersen Consulting). Over the course of her career she has been involved in leading the implementation of large and complex systems engineering and integration projects. She has done invited lectures internationally at Keio University, and Shibaura Institute of Technology, Japan, Overseas Chinese Institute of Technology (OCIT), Taiwan, Indian Institute of Technology – Delhi, etc. She is a visiting associate professor for System Architecture and Integration at Keio University. Her teaching and research interests include systems integration, systems architecture and design, business process reengineering, and rapid systems engineering. Dr. Jain has authored several papers on these topics. She holds Ph.D. and MS degrees in Technology Management from Stevens Institute of Technology.

Anithashree Chandrasekaran is a doctoral candidate in the School of Systems and Enterprise at Stevens Institute of Technology. She is also a Technology Risk Analyst at SIG. Her research interests include rapid systems development and its processes, development process reengineering, risk management and modeling, SI, and system design and architecture. She is currently working on developing a dynamic risk assessment model for rapid system development process. She obtained her BE in Electrical and Electronics Engineering from P.S.G. College of Technology, India. She obtained her MS in Systems Engineering from Stevens Institute of Technology. She is the founding president of Stevens INCOSE student chapter.

Contact: Rashmi Jain, PhD, Associate Professor of Systems Engineering, Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030; phone: 201-216-8047; rashmi.jain@stevens.edu