Potential Effect of Creeping User Requirements on Project Management: A Simulation Approach

P. K. Suri[1], Rachna Soni[2], Ashish Jolly[3]

[1]Department of Computer Science & Applications, Kurukshetra University, Kurukshetra (Haryana), India.
[2]Department of Computer Science, DAV Girls College, Yamuna Nagar (Haryana), India.
[3]Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology, Ambala City (Haryana), India.

Abstract
Requirements management has been discussed for at least fifteen years. As a discipline and as a practice, it has become more and more complex. In software projects new requirements are continuously issued and the objective of the requirement management is to elicit, manage and prioritize requirements. In the present study, a specific requirement management process is simulated. The stochastic parameters of the proposed system with specific system boundaries under a given environment have been estimated using event to event simulation. The present study will yield realistic results which are very near to the functioning of the live system. Based on results from simulation, conditions that result in an overload situation are identified. Simulation is also used to find process change proposals.

Keywords:
Requirement volatility, Requirement management, Project management, Risk factors

1. Introduction

Requirements are those externally observable characteristics of a system that a user, buyer, customer, or other stakeholder desires to have present in the system. Requirements management is the set of activities encompassing the collection, control, analysis, filtering, and documentation of a system's requirements. Requirements management consists of three activities:
Requirements Elicitation: The art of understanding the needs of Stakeholder and collecting them in a repository for future analysis.
The needs can be expressed quite abstractly and in terms of a problem, e.g., "I want to reduce my billing error rates by at least 35%." The needs can be expressed quite specifically and in terms of a solution, e.g., "I want there to be a large red button on the operator's console." In all cases, these needs are called features.

Requirements Triage: The art of deciding which features are the appropriate features to include in the product. Rarely is it possible to satisfy every requested feature gathered from every stakeholder during the elicitation activity. Disparate priorities, limited resources, time-to-market demands, and risk intolerance are but a few of the reasons for this. Triage takes into considerations all the painful realities of the marketplace and makes the decision of which features will we build now, which will be built in the next release, and which will be deferred until even later.

Requirements Specification: The art of detailing the exact external behavior of a system that will address the features selected during the triage process. The level of detail of these requirements depends greatly on the situation. For example, if specifying a handheld remote mouse, it might be sufficient to state "The system shall contain three programmable buttons, corresponding to the three buttons on a standard three-button mouse." However, if the device is to be mounted in a holster and controlled by robotic fingers instead of being hand-held, then the statement of requirements for the buttons would need to be considerably more detailed. Seventy-one percent of all software development projects result in complete failure (i.e., premature cancellation or shelf ware upon completion).
Poor requirements management is generally considered one of the major causes for product failure [2, 3]. After all, if we do a poor job of understanding our customers' needs, if we do a poor job of deciding the right features to build, and if we do a poor job of writing down what we think we want out of a system, how can we possibly expect a successful project? All the software development techniques and tools and whatever level of CMM maturity you have achieved will be of no use to you if you are not building the "right" product.
This paper is organized into three main sections, each addressing the techniques, tools, and common wisdom of each of the three aspects of requirements management.

The High Cost of Requirement Errors

There is strong evidence that effective requirements management leads to overall project cost savings. The three primary reasons for this are:

Manuscript received November 5, 2009
Manuscript revised November 20, 2009

1. Requirement errors typically cost well over ten times more to repair than other errors.
2. Requirement errors typically comprise over 40% of all errors in a software project.
3. Small reductions in the number of requirement errors pay big dividends in avoiding rework costs and schedule delays.

Studies performed at GTE, TRW, and IBM measured and assigned costs to errors occurring at various phases of the project life-cycle2. Although these studies were run independently, they all reached roughly the same conclusion: If a unit cost of *one* is assigned to the effort required to detect and repair an error during the coding stage, then the cost to detect and repair an error during the requirements stage is between *five* to *ten* times less. Furthermore, the cost to detect and repair an error during the maintenance stage is *twenty* times more.

Requirements Management

Requirements engineering plays an important role in the software development. As said by [8], a requirement is the condition or capacity that a system that is being developed must satisfy. Therefore, the compliance with requirements determines the success or the failure of a project. The requirements are identified, registered, organized and verified during the project development and that is what it called requirements management, a process that establishes and keeps the agreements firmed between the project team, users and customers related to the changes of requirements in a specific system. The literature states that the problems related with requirements engineering are one of the main reasons for software projects failures. This means that the final product does not have all the requirements gathering from users and customers [12]. Research identified that 70% of the requirements were difficult to identify and 54% were not clear and well organized. Also, it can be identified that [8] requirements are not easy to be described in words. There are different types of requirements in different levels of details. It can be impossible to manage the requirements if they cannot be controlled. Most requirements change during the project time.

Therefore, it is not difficult to find errors in the requirement specifications, and they can have a large impact in the project costs. An estimative shows that 40% of the requirements generate rework during the project life cycle [12]. It is evident that the earlier a problem is detected and solved (especially during the requirements phase), many other problems are minimized in the following project phases. But in contrast, what it is observed is a short time for the requirements phase in a project, not considering the project type or environment where this phase occurs.

Global Software Development (GSD)

As said by [9], software process is defined by a set of activities, methods, practices and technologies that people and companies use to develop and to keep related software and products. The interest in the software process is based on the following premises: The software quality is strongly dependent on the quality of the process used in its preparation; The software process can be defined, managed, measured and improved. However, it is not a simple task to develop software using a well-defined development process. Such process has become increasingly more complex, whereas the software demands of companies increase according to the strategic importance for its operations. As part of the globalization efforts currently pervading society, software project teams have also become geographically distributed on a worldwide scale. This characterizes Global Software Development (GSD). Tools and technological environments have been developed over the last few years to help in the control and coordination of the development teams working in distributed environments. Many of these tools are focused in supporting procedures of formal communication such as automated document elaboration, processes and other non-interactive communication channels. Moreover, [3], [4], [5] and [10] point out that GSD is one of the biggest business-oriented challenges that the current environment presents under the software development process point of view. Many companies are distributing its software development process in countries such as India, Russia and Brazil. Frequently this process occurs in only one country, particularly in regions with tax incentives or critical mass in some skill or resource areas.

Organizations search for competitive advantages in terms of cost, quality and flexibility in the area of software development [10], looking for productivity increases as well as risk dilution [7]. Many times the search for these competitive advantages forces organizations to search for external solutions in other countries (offshore outsourcing). This epitomizes the traditional problems and the existing challenges in GSD.

Key Benefits of better Requirements Management

Even in light of the above information, some will still argue, why waste time with this unnecessary step; why not proceed directly to implementation? Experience has shown that the benefits of effective requirements management include**:**

Better control of complex projects Lack of knowledge of the intended behavior of the system, as well as requirements creep, are common factors in out-of-control projects. Requirements management provides the development team with a clear understanding of what is to be delivered, when and why. Resources can be allocated based on customer-driven priorities and relative implementation effort. And the impact of changes are better understood and managed.

Improved software quality and customer satisfaction
The fundamental measure of software quality is “does this software do what it is supposed to do?” Higher quality can result only when developers and test personnel have a concise understanding of what must be built and tested.

Reduced project costs and delays Research demonstrates that errors in requirements are the most pervasive and most expensive errors to fix. Decreasing these errors early in the development cycle lowers the total number of errors and cuts project costs and time-to-market.

Improved team communications
Requirements management facilitates early involvement of customers to ensure the application meets their needs. A central repository builds a common understanding between the user community, management, analysts, developers and test personnel of project needs and commitments.

Easing compliance with standards and regulations
Most major standards bodies and regulatory agencies involved with software compliance and process improvement have a keen understanding of the requirements management problem. For example, the Software Engineering Institute’s Capability Maturity Model (CMM) addresses requirements management as one of the *first steps* in improving software quality. DOD, FDA, and ISO 9000 standards and regulations also require companies to demonstrate maturity and control of this process. It is clear that doing a better job of the above will save considerable time and money, not to mention reducing the career challenges that result from unsuccessful or partially successful software implementations.

2. Simulation Model

The simulation accepts a set of input parameters which specify the simulated situation. These input parameters include the number of requirements entering the process each day, the number of available servers (employees) for each phase and average time spent on requirement on each phase.

Model with Poisson Input Constant Service Time
The requirements are modeled to have an exponentially distributed intensity of arrival, i.e. they arrive according to Poisson process.
We assume

(a) S servers (employees)
(b) Each server provides service at the same constant average rate μ.
(c) The average arrival rate is constant; for all n
(d) $\lambda < S\mu$
where
Lambda = arrival rate
L = average number of requirements in the system
L_q = average number of requirements in the queue
L_w = average number of requirements in the nonempty queues
W = average time a requirement spends in the system
W_q = average time a requirement spends in the queue
W_w = average time a requirement spends in the queue if it must wait

$P(n > k) = \left(\frac{\lambda}{\mu}\right)^{k+1}$ = probability of more than k requirements in the system.

$p(T > t) = e^{-\mu(1-\lambda/\mu)t}$ = probability the time in the system is greater than t

With these assumptions (a),(b),(c) and (d),we have

$$p_n = \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n p_0 \qquad n = 0,1,\ldots\ldots,s-1$$

$$p_n = \frac{1}{s!s^{n-s}}\left(\frac{\lambda}{\mu}\right)^n p_0$$

$n \geq s$

$p(n \geq s) =$ probability an arrival has to wait for service
= probability of at least s requirements in the system

$$= \sum_{n=s}^{\infty} p_n$$

$$= \frac{(\lambda/\mu)^s p_0}{s!(1-\lambda/\mu s)}$$

$$p_0 = \frac{1}{\left\{\left[\sum_{n=0}^{s-1} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n\right] + \frac{1}{s!(1-\lambda/\mu s)}\left(\frac{\lambda}{\mu}\right)^s\right\}}$$

$$L_q = \frac{\left(\frac{\lambda}{\mu}\right)^{s+1} p_0}{s.s!(1-\lambda/\mu s)^2}$$

$$L = L_q + \frac{\lambda}{\mu}$$

$$W = \frac{L}{\lambda}$$

$$W_q = \frac{L_q}{\lambda}$$

$$P(T>t) = e^{-\mu t}\left\{1 + \frac{(\lambda/\mu)^s P_0\left[1-e^{-\mu t\left(s-1-\lambda/\mu\right)}\right]}{s!(1-\lambda/\mu s)(s-1-\lambda/\mu)}\right\}$$

3. Results and Discussion

Results of the simulation model stated above, (where arrival rate is generated through Poisson distribution) are as given below:

Case 1: mu=6, S=3 and T=10 days

LAMBDA	5.00000	10.00000	15.00000
L	.85553	2.04137	6.01124
LQ	.02220	.37470	3.51124
W	.17111	.20414	.40075
WQ	.00444	.03747	.23408
PT	.14141	.19459	.46198
PN	.36011	.28777	.11236
PZERO	.43213	.17266	.04494
PS	.05771	.29976	.70225

Table 1

For higher values of Lambda, queueing system not valid because Lambda < s*mu.

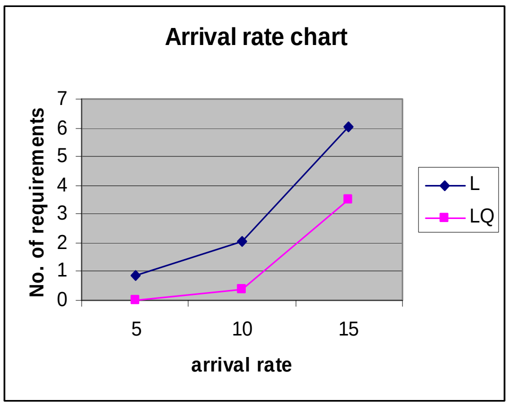

Fig 1: Arrival rate chart

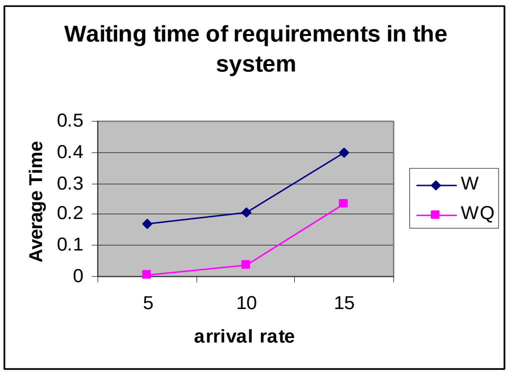

Fig 2: Waiting time of requirements in the system

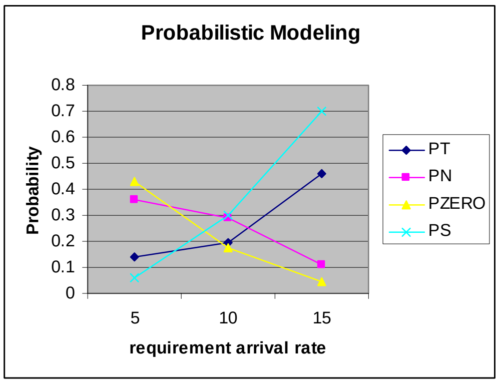

Fig 3: Probability Graph

From the table it is observed that average no of requirements waiting in the queue increases with increase in the arrival rate of the requirements, so the average time a requirement spends in the system(w) and queue (both) increasing and the probability that no requirement is in the queue is decreasing. The probability of at least s requirements in the system is increasing. Case 2 of the study deals with the problem of high waiting time in the queue.

Case 2: Lambda=15, T=10 days and mu=6

LAMBDA	15.00	15.00	15.00	15.00	15.00
MU	6.00	6.00	6.00	6.00000	6.00
N	1.00	1.00	1.00	1.00000	1.00
S	3.00	5.00	7.00	9.00000	11.00
T	.33330	.33330	.33330	.33330	.33330
L	6.01	2.63037	2.50	2.50046	2.50002
LQ	3.511	.13037	.00858	.00046	.00002
W	.40075	.17536	.16724	.16670	.16667
WQ	.23408	.00869	.00057	.00003	.00000
PT	.46198	.14654	.13596	.13539	.13536
PN	.11236	.20025	.20495	.20520	.20521
PZERO	.04494	.08010	.08198	.08208	.08208
PS	.70225	.13037	.01544	.00119	.00006

Table 2

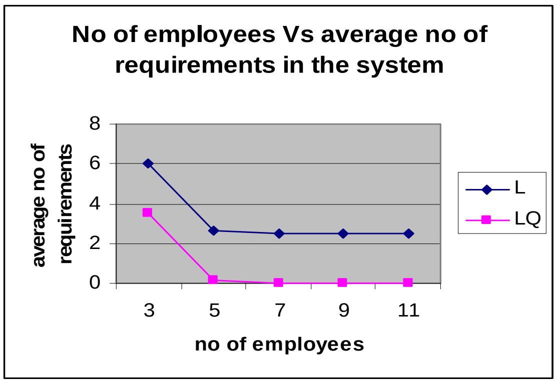

Fig 4: No of employees Vs average no of requirements in the system

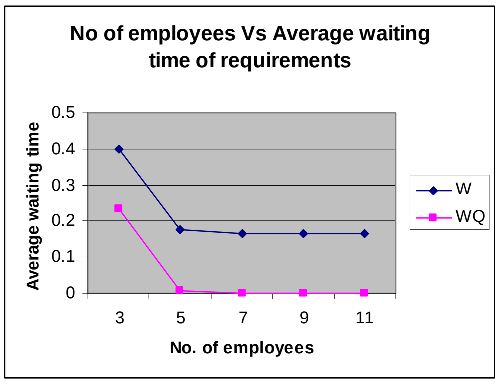

Fig 5: No of employees Vs average waiting time of the system

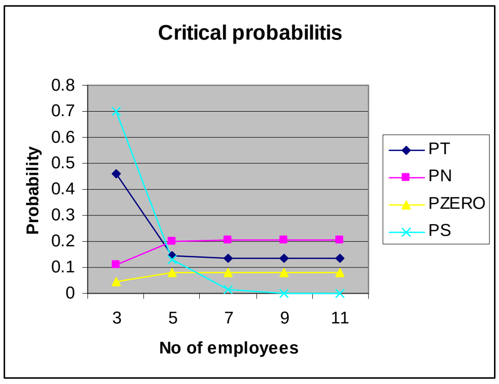

Fig 6: Probability Chart

The above graph reveals that by adding two employees in the system the requirement creep can be stabilized. It will reduce the average time a requirement spends in the system and queue.

4. Conclusion

In the above work a simulation technique has been used to know the potential effect of creeping user requirements on the project. The substantial progress has been achieved in the areas of requirement elicitation, analysis and specification. The present simulator will be an asset in IT industry in order to optimize the software development process and to release the software product in an estimated scheduled time. The future research lies in the holistic view and system vide solutions to the entire management process.

References

[1] 1] The Merlin-project web site. (2005-07-12) URL: http://virtual.vtt.fi/merlin/

[2] Jones, C., “Patterns of Software Systems Failure and Success”, International Thomson Press, 1996.

[3] Parviainen P., Vierimaa M., Tihinen M., Kääriäinen J., Takalo J. “Industrial Inventory Summary (Merlin-project deliverable D1.1.3.)”, pp. 37, 2004.

[4] McConnell, S., *Rapid Development*, Microsoft Press, 1996, p. 86.

[5] Guidelines for Requirements Management United States Department of Energy, Software Quality Assurance Subcommittee, pp. 31, 2000.

[6] Bosworth M., “Solution Selling”, New York, McGraw Hill, 1995.

[7] Lormans M., vanDijk H., van Deursen A., Nöcker E., deZeeuw A. “Managing Evolving Requirements in an Outsourcing Context: An Industrial Experience Report” in Principles of Software Evolution, 7th International Workshop on (IWPSE’04), pp. 149-158, 2004

[8] Gause, D., and G. Weinberg, “Exploring Requirements”, New York, NY, Dorset House, 1989.

[9] Kotonya G., & Sommerville I. “Requirements Engineering: Process and Techniques”, John Wiley & Sons, pp. 282, 1998.

[10] Goguen, J., and M. Jirotka, “Requirements Engineering”, Boston, MA, Academic Press, 1994.

[11] Parviainen P., Hulkko H., Kääriäinen J., Takalo J., Tihinen M. “Requirements engineering, Inventory of technologies”, VTT Publications 508, pp. 106, 2003.

[12] Leffingwell, D., and D. Widrig, “Managing Software Requirements”, New York, Addison Wesley, 2000.

[13] Spurr K., et al., “Computer Support for Cooperative Work”, New York, John Wiley, 1994.

[14] Bosworth, M., “Solution Selling”, New York, McGraw Hill, 1995.

[15] Rinne J., Kohti hajautettua ohjelmistokehitystä. Pro gradu – tutkielma. Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, Tampere, pp. 54, 2001.

[16] Paasivaara M. & Lassenius C., “Collaboration Practices in Global Interorganizational Software Develop-ment Projects”, in Software Process Improvement and Practice, 8, pp. 183-199, 2004.

[17] Ohrwall Rönnbäck A, “Interorganizational IT Support for Collaborative Product Development”, Dissertation from the International Graduate School of Management and Industrial Engineering, IMIE. No. 59, Doctoral Dissertation, pp. 83, 2002.

[18] Teppola S., Takalo J., Kääriäinen J. Tool chain. (Merlin-project deliverable D1.3.3.), pp. 18, 2005.

[19] Davis, A., “Software Requirements”, Englewood Cliffs, NJ: Prentice Hall, 1993.

[20] Hoffmann M., Kühn N., Weber M., Bittner M. “Requirements for Requirements Management Tools”, in 12th IEEE International Requirements Engineering Conference (RE´04) pp. 301-308, 2004.

[21] Davis, A., et al., “Identifying and Measuring Quality in Software Requirements Specification,” IEEE International Software Metrics Symposium, Los Alamitos, CA, IEEE Computer Society Press, pp. 164-175, 1997.

[22] Jäälinoja J. “Requirements implementation in embedded software development”, University of Oulu, Department of Information Processing Science, Master’s Thesis, pp. 85, 2004.

[23] Cleland-Huang J., Zemont G., and Lukasik W. “A Heterogeneous Solution for Improving the Return on Investment of Requirements Traceability” in 12th IEEE International Requirements Engineering Conference (RE’04) pp. 230 – 239, 2004.

Dr. P.K. Suri received his Ph.D degree from Faculty of Engineering, Kurukshetra University, Kurukshetra, India and master’s degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science & Applications, Kurukshetra University, Kurukshetra - 136119 (Haryana), India since Oct. 1993. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised eleven Ph.D.’s in Computer Science and thirteen students are working under his supervision. He has more than 125 publications in International / National Journals and Conferences. He is recipient of 'THE GEORGE OOMAN MEMORIAL PRIZE' for the year 1991-92 and a RESEARCH AWARD –“The Certificate of Merit – 2000” for the paper entitled ESMD – An Expert System for Medical Diagnosis from INSTITUTION OF ENGINEERS, INDIA. His teaching and research activities include Simulation and Modeling, SQA, Software Reliability, Software testing & Software Engineering processes, Temporal Databases, Ad hoc Networks, Grid Computing and Biomechanics.

Dr Rachna Soni is Associate Professor and Head, Deptt of Computer Science & Applications, did her M.Phil from IIT, Roorkee and Ph.D from Kururukshetra University, Kurukshetra. She has more than eighteen years teaching experience in the institution of repute. She has published eight papers in International/national journal /conferences. Her area of interest includes Software Risk Management, Project Management, Requirement Engineering, Component based Software Engineering, Simulation etc.

Ashish Jolly received his MCA degree from University of Madras, Chennai in the year 1999. Currently he is pursuing Ph.D in Computer Science from Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India. He is working as a Asstt Professor and Head in the Department of Computer Science & Applications, Shri Atmanand Jain Institute of Management & Technology (affiliated to Kurukshetra University, Kurukshetra), Ambala City, Haryana, India. He has more than nine years of teaching experience. He has published four research papers in referred International journals of repute. His research area includes Simulation, Software Engineering and Software Project Management.