Confirming the Effectiveness of the Requirements Generation Model: An Industry-Based Empirical Study

Markus K. Gröner & James D. Arthur
Department of Computer Science; Virginia Tech; Blacksburg, VA 24061-0106
{groener|arthur}@vt.edu

Abstract

Product quality is directly related to how well that product meets the customer's needs and intents. It is paramount, therefore, to capture customer requirements correctly and succinctly. Unfortunately, most development models tend to avoid, or only vaguely define the process by which requirements are generated. Other models rely on formalistic characterizations that require specialized training to understand. To address such drawbacks we introduce the Requirements Generation Model (RGM) that (a) decomposes the conventional "requirements analysis" phase into sub-phases which focus and refine requirements generation activities, (b) constrains and structures those activities, and (c) incorporates a monitoring methodology to assist in detecting and resolving deviations from process activities defined by the RGM. The results of an industry-based study are also presented and substantiate the effectiveness of the RGM in producing a better set of requirements.

Keywords

Requirements Identification, Requirements Generation, Customer Intent, Requirements Engineering, Software Engineering, Software Methodology

1. Introduction

This paper presents an industry-based study to help assess the effectiveness of the Requirements Generation Model (RGM) [4] in software development efforts. The RGM is designed to minimize the difficult task faced by every requirements engineer, i.e., to capture customer requirements accurately and succinctly. Every software development effort depends on procedures and methods such as those found in the Waterfall Model [6], the Spiral Model [2], and prototyping approaches [7] to name a few. Yet, no matter how much rigor is applied to the process, the resulting product is only as good as the set of requirements by which it is defined. The RGM consists of a framework and a monitoring methodology that constrains and guides a requirements engineer and customer through the requirements definition process. The objective of the RGM is to provide a structure within which requirements are captured meeting customer intent. This paper provides a brief overview of the RGM and presents the findings of an empirical study conducted to determine its effectiveness. A complementary subset of the RGM components was selected for the study. The positive contributions the RGM makes to the requirements definition process is substantiated by the empirical data derived from the study.

The remainder of this paper is structured as follows: Section 2 provides a brief overview of the Requirements Generation Model; Section 3 describes the selection process for two groups in this study; Section 4 outlines how the empirical study was structured; Section 5 presents the data and conjectures supported by that data. Finally, Section 6 provides some concluding remarks.

2. An Overview of the Requirements Generation Model

The Requirements Generation Model (RGM) refines the conventional "requirements analysis" phase by imposing a framework and methodology that structures the requirements elicitation, recording and evaluation processes. Similar to existing models like Participatory Design (PD) [3] and Joint Application Design (JAD) [5], the RGM seeks to reduce the disparity in domain knowledge between the requirements engineer and customer. The RGM differs from those models, however, in that it focuses extensively on the requirements phase, and within that phase, prescribes activities that address a more specific and well-defined set of objectives.

2.1 The Framework

As illustrated in Figure 1, the RGM framework partitions the conventional requirements analysis phase into an initial indoctrination phase, which is then followed by an iterative requirements capturing phase. The objectives of the indoctrination phase are to (a) introduce the customer to the requirements definition process, (b) provide the requirements engineer with an overview of the customer's

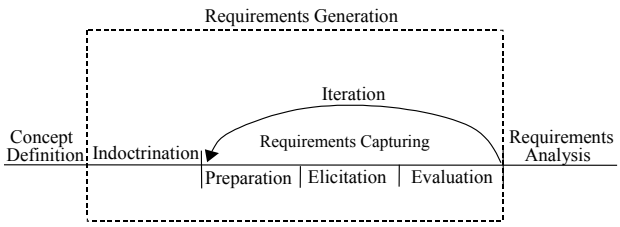

Figure 1: The Requirements Generation Framework

problem domain and needs, and (c) describe the participants' tasks and responsibilities in the requirements definition process. The objectives of the requirements capturing phase are directly related to the three sub-phases that comprise it – preparation, elicitation and evaluation. Respectively, the primary objectives of those sub-phases are to (a) define the scope of the up-coming elicitation meeting and ensure that all participants have completed their pre-meeting assignments, (b) enable the requirements engineer to accurately identify and record software requirements as expressed by the customer, and (c) evaluate the contributions of the preceding elicitation meetings, identify unresolved (or new) issues, and determine if an additional refinement iteration is needed. All three sub-phases are embedded within an iterative framework that, in turn, encourages the progressive identification, refinement and elaboration of individual requirements. Collectively, the three sub-phases and the iterative framework define the requirements capturing process. Through phase-specific guidelines and protocols, and through its iterative framework, this process promotes a structured approach to the interaction format and information exchange between the customer and the requirements engineer.

2.2 The Monitoring Methodology

Even within a structured requirements elicitation process problems can still surface, and before being recognized as such, have an adverse impact on that process. Hence, identifying and addressing such problems *as they occur* is an important component of the RGM. The RGM provides this detection/correction component through its *monitoring methodology*. That is, the RGM's monitoring methodology detects deviations in the prescribed elicitation process and suggests appropriate remedies. In more specific terms the monitoring methodology is composed (a) automated and manual procedures that constantly "monitor" the elicitation process for defined problems, and (b) methods that assist in their resolution. An example of one simple procedure is setting a timer to provide an audible "alarm" when a meeting has exceeded its stipulated time duration.

In effect, the framework and methodology of the RGM have been purposefully designed to (a) add structure and control to the requirements generation activities, (b) support the accurate capturing of requirements, and we conjecture, and (c) promote an effective requirements generation process.

3. Selection Process

The selection of participants for this empirical study calls for two groups of comparable size and experience. One group, designated as the "Non-RGM" group, serves as the control group for the study. The Non-RGM defines requirements and implements software using currently accepted practices. The second group, called the "RGM" group, is the experimental group. The RGM group receives training in the use of the RGM and in the application of its activities to the software development process. Both the RGM and Non-RGM groups work in the same industrial setting, supporting financial data processing systems and software. Prior to the study, the development efforts and track records for each group were similar.

The details of the empirical study and a discussion of how we attempted to avoid introducing bias is discussed in the following section.

4. Empirical Study Setup

This section describes the details of the empirical study. We focus on 1) those parts of the RGM that are deployed in the empirical study, 2) the environment in which the empirical study is conducted, 3) characteristics of the projects developed within the empirical study, 4) the people involved in the empirical study, and 5) the approach used to insert the RGM into the development process.

4.1. A Description of the Empirical Study

An empirical study was conducted to assess the benefits of using the RGM within an industrial development environment. In this study, we show that using the RGM instead of the currently existing requirements generation process improves the effectiveness of the requirements generation process, and has an additional positive impact on the remaining development phases. Through this study, we establish three benefits of employing the RGM, although many more are believed to exist:

1. By using the RGM to more effectively capture requirements as stated by the customer, we expect to see less (if any) schedule slippage for major project milestones.
2. By using the RGM we expect fewer adverse impacts in later development phases due to requirements changes.

3. By using the RGM we expect those directly involved in the project development, as well as the customer, to have a higher level of satisfaction with the project.

Changing a development environment is a difficult task at best. We desire, therefore, to implement minimal, but effective changes. Hence, we have selected elements of the RGM that we expect will contribute to the requirements generation process, but require minimal implementation overhead. These elements are:

- *Issues/Participant Notifications* – Participants in upcoming requirements elicitation meetings are notified in advance of issues they are expected to present.
- *Meeting Notifications* – Resolutions and problems are distributed in a timely manner before meetings, as well as meeting place/date/time and alternative ways to attend the meeting are announced.
- *Silent Parking Lot* – Provide a mechanism by which meeting participants can record thoughts for later discussion, and thereby avoid interrupting the current dialogue process.
- *Limited Unstructured Requirements List Reviews* – A thorough review of "just-captured" requirements by the requirements engineer and the customer.
- *Controlled Interrupts* – A method embraced by the RGM and intended to reduce interruptions during a requirements elicitation meeting. This method complements the use of the Silent Park Lot.
- *Avoiding the "Guessing Game"* – This method is used to detect and minimize the introduction of vague requirements stemming from the stakeholder "guessing" at the correct requirement. Responses containing containing terms or phrases like "possibly", "maybe", and "I think so" indicate questionable understanding.
- *Seeking Requirements, Not Solutions* – The RGM is specific in stipulating that the requirements engineer must avoid the introduction of requirements that suggest *a priori* solutions, and thereby, constrain requirements formulation.

4.2. Environment

The study environment is comprised of the physical surroundings, the hardware, and the operating systems being used, as well as the atmosphere established through the organization of the study. This section details those components and discusses the efforts made to avoid the introduction of bias.

4.2.1. The Physical Environment.

The empirical study is performed in a large financial institution. The department within which the study takes place is comprised of five separate groups. Four of these groups work within the same building but are physically separated. The fifth group works in a separate building. The physical environment is one of cubicles for each employee, with either a single PC running MS Windows 95, or two PCs, one with MS Windows and a second with Unix. Developers use either MS Visual Basic or C/C++ on the Unix operating system.

4.2.2. Avoiding Bias

One concern when conducting a study is the introduction of bias – bias compromises otherwise valid results. A first step in avoiding the introduction of bias is to set up an "experimental" group and a "control" group and *minimize their interaction*. For example, those elements to be studied, i.e. changes to the normal environment, are introduced through the experimental group, while the control group continues to function the same as before.

Our empirical study employs an experimental group and a control group. The experimental group (referred to as the RGM group) uses those previously described components of the RGM. The control group (referred to as the Non-RGM group) continues to employ the currently established development process. For both groups we use several techniques to minimize the introduction of bias:

- Both groups work in the same department, but work in physically separate locations.
- Both groups work on the same type of applications but which have no overlapping components. Each group has a completely different set of customers.
- Only the experimental (RGM) group is trained in the RGM process. No one else is aware of this special training.
- Necessary artifacts are given only to the RGM group.
- We monitor both groups throughout the empirical study.
- Neither group is aware that they are part of a study. The data to be collected is a standard component of both group's submission requirement for projects.

The six items noted above serve to minimize the introduction of bias into the empirical study.

4.3. Project Characteristics

This section presents the characteristics of the projects used in the empirical study. Those characteristics include the elements that determine selection, such as project size, application type, and development and target environments. Also included are the roles of those involved in the projects.

4.3.1. Project Selection

The following set of characteristics were used in selecting the projects for the groups in the study:

- Project Time Length – must be estimated at no more than six months to completion.
- Project Size – each group must have two large projects and two small projects. Project size in the given development environment is determined by estimated project cost. A project of greater than $250K is large, and below $250K small. This cost includes the purchase of hardware as well as employee time. Size is not an indicator for the expected project length, as small projects may be developed in parallel with less than 100% human resource dedicated to a single project at a time. In this environment, it is common for small and large projects to start and finish at the same time due to this factor.
- Project Complexity – all projects must be afinancial processing application with database access. The large projects were critical and essential to the daily operations of the business. The small projects were system functionality enhancements.
- Personnel – between the two groups corresponding participants must have equal qualifications, experience, and length of time within their groups.

Additional project characteristics and division between the two development groups are as follows:

RGM (Experimental) Group:

- Two large projects (Projects 1 & 2).
- Two small projects (Projects 4 & 5).
- All projects are financial applications with simple data manipulation (calculations), database look-ups, and database stores.
- All projects are to be developed in either MS Visual Basic or C++ on Windows 95 or Unix based operating systems, respectively.
- All projects are deployed on either Windows 95 or Unix based operating system machines.
- *Roles:*
  - Large projects each have 1 project manager, 1 systems analyst, and 2 developers, as well as a single customer.
  - Small projects each have 1 project manager, 1 systems analyst, and 1 developer, as well as a single customer.

Non-RGM (Control) Group:

- Two large size projects initially. One project is canceled early during requirements definition. (Project 3 is one that remained).
- Two small size projects (Projects 6 & 7).
- All projects are financial applications with simple data manipulation (calculations), database look-ups and database stores.
- All projects are developed in either MS Visual Basic or C++ on Windows 95 or Unix based operating systems respectively.
- All projects are deployed on either Windows 95 or Unix based operating system machines.
- *Roles*:
  - Large project has 1 project manager, 1 systems analyst, and 2 developers, as well as a single customer.
  - Small projects each have 1 project manager, 1 systems analyst, and 1 developer, as well as a single customer.

For each group, we selected two large projects and two small projects using the criteria stipulated above, any other similarities or differences were coincidental. The selection process for the RGM projects, however, did include the exception that project start dates must be after the anticipated completion date for the training of the RGM group.

4.3.2. Personnel

In this section we provide an overview of the people who are involved in the study projects. It is important for the study that all have comparable skill sets, education and roles.

- *Project Managers:* There are a total of four project managers – two in each group. All project managers have a minimum of ten years of experience in managing projects for various organizations, and all have been within the same department for a minimum of one and a half years. Three of the project managers have an MBA degree, and one has an MIS degree.
- *Systems Analysts*: Two systems analysts (one in each group) are part of the empirical study. Both analysts have a minimum of three years of work experience as both developers and analysts. Each has a Masters Degree in Computer Science.
- *Developers:* Each group in the empirical study has four developers working on the projects. Each developer has a minimum of two years of work experience programming in both MS Visual Basic as well as Unix C/C++. All developers have a Bachelors Degree in Computer Science.

4.4. Empirical Study Process

To maximize the probability of a successful study and to support the collection of valid data, several issues had to be resolved before the study is conducted. This section focuses on those particular issues.

4.4.1. Training

The RGM group had to be trained in the selected activities and methods of the RGM to be used during the study. Only project managers and systems analysts are trained because they alone have direct contact with the customer and will be using the RGM. Training of the group includes an overview of the RGM, as well as a discussion and rationale for using the selected RGM methods and artifacts. To help avoid the introduction of bias or other behaviors that might impact the validity of the study, we did not reveal to the group that a study was being conducted.

4.4.2. Oversight

To ensure adherence to the RGM process and to assist the group in prescribed activities thereof, we attended several of the earlier requirements elicitation meetings. Assistance was required only a few times, and those were primarily related to *Controlled Interrupts* and the use of the *Silent Parking Lots*. Following the requirements elicitation process we met with the project managers and systems analysts to provide them feedback as to their adherence to the RGM, and to receive their thoughts on how we might further enhanced the RGM.

5. Data Collection, Presentation, and Interpretation

The data described in this section was collected from a company-wide status-reporting database. In addition to reporting phase initiation and completion dates, project managers are required to update their project status when major milestones are reached (such as finishing the requirements phase and moving into design), and when project status change. Project status colors are established to indicate how the project is progressing – those colors, and what they imply, are as follows:

- *Green* – project is proceeding on schedule and within budge. Overall the project is encountering no current risks.
- *Yellow* – project has run into a possible risk that could impact either schedule or budget. The project needs to be closely managed with an emphasis on minimizing perceived risk.
- *Red* – project has encountered a setback that adversely impacts either schedule or budget. At this time the setback needs to be evaluated and necessary project changes negotiated with the customers to resolve the red status.

During the study we also collected observational data during meetings with, and among, the customers, project managers and analysts. This data was used to make adjustments to the RGM and helped explain certain unexpected findings.

Together, the subjective and objective data collected through the status reporting database serve to confirm three conjectures concerning the beneficial impacts of using the RGM. We examine each conjecture and related data items in the following subsections.

5.1. Conjecture 1

> *Projects using the RGM to capture customer intent should experience less slippage in project phase completion dates.*

Stated in other terms, we expect that projects developed under the auspice of the RGM will meet their projected phase completion dates more often than similar projects which do not employ the RGM. To confirm this conjecture, we need to know the original date a phase is supposed to be completed, and the actual date the phase is completed.

Although the RGM has its highest impact on the requirements generation phase, its benefits are also realized in succeeding development phases. For the purpose of this study, therefore, we include an examination of the requirements definition, design, and implementation phases.

The data presented in Tables 1 and 2 represent phase-by-phase slippage days for the RGM and Non-RGM projects respectively. Slippage days are computed as the difference between the originally specified ending date for each phase and the actual ending date for that phase. (Adjustments were made to account for the compound impact of slippage on originally specified ending dates.) A negative number indicates the project ended earlier than expected. The projected duration is the number of days originally projected. The projected duration allows one to place the slippage days in perspective.

Table 1: RGM Group Projected Duration And Slippage Days

Project 1 *(large)*	**Projected Duration**	**Slippage Days**
Definition	31	0
Design	16	7
Implementation	54	-7
Total	101	0
Project 2 *(large)*	**Projected Duration**	**Slippage Days**
Definition	31	0
Design	246	-9
Implementation	2	9
Total	79	0

Project 4 *(small)*	**Projected Duration**	**Slippage Days**
Definition	25	0
Design	59	-3
Implementation	5	15
Total	89	12
Project 5 *(small)*		**Slippage Days**
Definition	26	0
Design	52	-1
Implementation	14	25
Total	92	24

Table 2: Non-RGM Group Slippage Days

Project 3 *(large)*	**Projected Duration**	**Slippage Days**
Definition	27	27
Design	0	215
Implementation	44	-8
Total	71	234
Project 6 *(small)*	**Projected Duration**	**Slippage Days**
Definition	19	9
Design	30	-2
Implementation	151	-31
Total	200	-24
Project 7 *(small)*	**Projected Duration**	**Slippage Days**
Definition	15	52
Design	14	1
Implementation	18	5
Total	47	58

5.1.1. Analyzing Slippage Days

To confirm (or refute) our first conjecture, we examine the data presented in Tables 1 and 2 from two different perspectives.

First, we view the data as a simple phase-by-phase comparison of slippage for both the large and small projects. To help simplify the process and to help remove any potential bias that could be introduced by looking at the relative and absolute magnitude of the numbers, we use two indicators, "<" and ">", to capture "slippage." That is, "<" means that the RGM group had less slippage than the Non-RGM group for that particular phase on a same size project. Conversely, ">"indicates less slippage for the Non-RGM group. Using the above slippage indicators, Table 3 depicts a cross comparison of the large projects for the experimental (RGM) and control (Non_RGM) groups. Similarly, Table 5 provides the cross-comparison for the small projects.

Our second perspective is based on a magnitude indicator. The magnitude indicator is intended to reflect the *collective* extent to which the slippage days differ between the RGM and Non-RGM development efforts. The magnitude indicator is computed as a function of the number of slippage indicators ("<" and ">") given in Tables 3 and 5. That is, for a given set of cross-comparisons if all of the slippage indicators within a phase are "<" then we set the magnitude indicator for that phase to "++", indicating a significant advantage for the RGM development effort. If more than half but not all slippage indicators within a phase are "<", then the magnitude indicator for that phase is set to "+", indicating a simple advantage for the RGM group. If the number of slippage indicators is evenly split among the cross-comparisons for a given phase, then we set the magnitude indicator for that phase to a "0", indicating that no advantage exists for either the RGM or Non-RGM development effort. Similarly, for the ">" slippage indicators we compute phase-wise magnitude indicators of "--", "-", and "0" denoting the extent of the advantage for the Non-RGM development effort. Tables 4 and 6 provide the magnitude computations for the large and small project cross-comparisons, respectively.

We now provide and analysis of project slippage using slippage days (Tables 1 and 2), slippage indicators (Tables 3 and 5) and magnitude indicators (Tables 4 and 6).

Table 3: Large Project Slippage Comparison

	RGM Project 1	**Comp.**	**Non-RGM Project 3**	**Diff.**
Definition	0	<	27	-27
Design	7	<	215	-208
Implementation	-7	>	-8	1
Total Days	0	<	234	-234
% Slippage	0	<	330%	-330%
	RGM Project 2	**Comp.**	**Non-RGM Project 3**	**Diff.**
Definition	0	<	27	-27
Design	-9	<	215	-224
Implementation	9	>	-8	17
Total Days	0	<	234	-234
% Slippage	0%	<	330%	-330%

Table 4: Large Projects Magnitude Indicators

	Comp. Dir.	Magn.	Total Diff.	Avg. Diff.
Definition	2 < and 0 >	++	-54	-27
Design	2 < and 0 >	++	-432	-216
Implementation	0 < and 2 >	--	18	9
Total	2 < and 0 >	++	-468	-234
% Slippage	2 < and 0 >	++	-659%	-330%
Total Magnitude	8 < and 2 >	++		

5.1.2. A Comparison Among the Large Projects

Table 3 provides a cross-comparison of slippage days for the large RGM and Non-RGM projects. The comparisons show that both RGM group projects (1 & 2) performed better than the Non-RGM group project (3) in the requirements definition and design phases. The Non-RGM group, however, faired better in its implementation phase for Project 3 by finishing eight days ahead of schedule. Nonetheless, we also note that Project 3 completed its implementation phase only one day ahead of the RGM Project 1, which completed its implementation seven days ahead of schedule.

Looking at the total number of slippage days, as well as the percent of slippage, the two RGM group projects (1 & 2) finished the combined definition, design, and implementation phases on the time. The Non-RGM project did not fair as well. Due to the large setback in the design phase, the project came in 234 days late, or 330% slippage over the initially planned schedule.

Examining the magnitude indicators shown in Table 4, we also observe that the large RGM projects fair much better than the Non-RGM counterpart – the one exception is in the implementation phase. The overall (or "total") magnitude indicator also significantly favors the RGM development effort.

Therefore, based on the data provided in Tables 1 – 4, we are confident in stating that the larger projects developed within the RGM framework tend to meet their projected phase completion dates more often than their counterpart projects which did not employ the RGM.

Anecdotal note: We noticed on several occasions that the Non-RGM group developers started implementation well before the requirements or design documentation was completed. Often, implementation can proceed based on "knowledge" about the overall project's goals – and therefore independent of requirement and design details. This might provide an insight as to why they finished their implementation phase ahead of schedule. We also noticed that the "base" code from similar applications was later modified to meet requirements.

Table 5: Small Project Slippage Comparison

	RGM Project 4	Comp.	Non-RGM Project 6	Diff.
Definition	0	<	9	-9
Design	-3	<	-2	-1
Implementation	15	>	-31	46
Total Days	12	>	-24	36
% Slippage	13%	>	-12%	25%

	RGM Project 4	Comp.	Non-RGM Project 7	Diff.
Definition	0	<	52	-52
Design	-3	<	1	-4
Implementation	15	>	5	10
Total Days	12	<	58	-46
% Slippage	13%	<	123%	-110%

	RGM Project 5	Comp.	Non-RGM Project 6	Diff.
Definition	0	<	9	-9
Design	-1	>	-2	1
Implementation	25	>	-31	56
Total Days	24	>	-24	48
% Slippage	26%	>	-12%	38%

	RGM Project 5	Comp.	Non-RGM Project 7	Diff.
Definition	0	<	52	-52
Design	-1	<	1	-2
Implementation	25	>	5	20
Total Days	24	<	58	-34
% Slippage	26%	<	123%	-97%

Table 6: Small Projects Magnitude Indicators

	Comp. Dir.	Magn.	Total Diff.	Avg. Diff.
Definition	4 < and 0 >	++	-122	-30.5
Design	3 < and 1 >	+	-6	-1.5
Implementation	0 < and 4 >	--	132	33
Total	2 < and 2 >	0	4	1
% Slippage	2 < and 2 >	0	-144%	-36%
Total Magnitude	11 < and 9 >	+		

5.1.3. A Comparison Among Small Projects

Table 5 provides a comparison among the smaller projects in the study. More specifically, it depicts four cross-comparisons using two RGM group projects (4 & 5) and the two Non-RGM group projects (6 & 7). The data

provided in Tables 1 and 5 confirms that the RGM group experienced no slippage during the requirements definition phases and actually ended the design phases for their projects ahead of schedule. The RGM group did experience some slippage in both projects during implementation – twelve and twenty-four days of total slippage (13% and 26% slippage) for Projects 4 and 5, respectively.

Yet, when compared to the Non-RGM group the RGM group still fairs better. The Non-RGM group experienced slippage for both projects during their definition phases (as compared to no slippage for the RGM projects). In comparing the design phase, the RGM projects faired better in 3 out of 4 of the cross comparisons, loosing out only in a comparison between RGM Project 5 and Non-RGM Project 6 – and even in this comparison, both projects finished design ahead of schedule. Table 5 does not show any definite advantage either way when comparing slippage during implementation and overall total – they appear to split evenly. A pertinent observation, however, is that the Non-RGM Project 6 finished almost one month early. As we stated in the previous anecdotal note, we impute such success to a tendency for Non-RGM developers to start implementation before definition and design are complete.

Because the magnitude indicators reflect a more collective view of project "success", an examination of the data presented in Table 6 reveals a clearer picture of how the small RGM projects compared to their Non-RGM counterparts. Relative to project slippage, the magnitude indicators show that the RGM projects performed significantly better during the requirements definition phases than did the Non-RGM projects. The RGM group also performed better during the design phase (although not as impressive when compared to the requirements definition phase). During the implementation phases the Non-RGM group did significantly better than the RGM group. From an overall perspective, however, the magnitude indicator favors the RGM projects. On a final note, when we compare slippage percentage, the RGM projects experienced 36% less slippage than did the Non-RGM projects.

Once again, and similar to the conclusions drawn for large projects, there does appear to be a beneficial contribution in using the RGM for small project development efforts. Hence, we assert that the data presented in Tables 1 – 6, and the comparisons provided in the previous sections confirm our first conjecture, that is,

> *Projects using the RGM to capture customer intent should experience less slippage in project phase completion dates.*

Anecdotal note: The data provided in the previous tables suggest that with and increase in project size there is a commensurate increase in the benefit of using the RGM. We also expected to find that the RGM would result in an increased communication overhead during the definition phase. Based on our observations and discussions, however, no such increase was apparent.

5.2. Conjecture 2

> *Projects using the RGM should exhibit less adverse impact stemming from late discovery of requirements.*

We contend that given similar projects development efforts, if one project uses the RGM during the requirements definition phase, it will have a better success at identifying and recording customer requirements earlier than a project not employing the RGM. Consequently, the project employing the RGM will experience less adverse impact stemming from the *lat*e discovery of needed requirements. Our experience has shown that scope and requirements changes are to be expected for many projects – not that this is desirable, but simply a fact of our development process. When identified earlier in the software development process, however, these necessary changes can be corrected for a fraction of the time and cost it would take to fix them during the later phases of the development cycle [1]. By emphasizing a more complete and accurate identification of customer requirements during the requirements phase, the RGM helps minimize the adverse impact stemming from the late discovery of requirements.

During the study the project status database was also used to record changes stemming from late requirements discovery and to note their potential impact (if any) on the development effort. The project database supports the entry of three project status indicators throughout a development effort. These status indicators are associated with the project's schedule, budget and overall project "health". Project status changes are indicated by shifts in status colors. For example, the schedule status can shift from "green" to "red", indicating substantial problems ahead. There are three types of "bad" shifts: (G)reen to (Y)ellow, Green to (R)ed, and Yellow to Red. To help reason about Conjecture 2, we count the number of "bad" status changes *stemming from the late discovery of requirements*. Tables 7 and 8 capture those counts for the RGM and Non-RGM development efforts, respectively.

Table 7: RGM Group Project Status Change

Project 1	**Schedule**	**Budget**	**Overall**
G->Y	0	0	0
G->R	0	0	0
Y->R	0	0	0
Project 2	**Schedule**	**Budget**	**Overall**
G->Y	0	0	0
G->R	0	0	0

Y->R	0	0	0
Project 4	**Schedule**	**Budget**	**Overall**
G->Y	0	0	0
G->R	0	0	0
Y->R	0	0	0
Project 5	**Schedule**	**Budget**	**Overall**
G->Y	0	0	0
G->R	0	0	0
Y->R	0	0	0

Table 8: Non-RGM Group Project Status Change

Project 3	**Schedule**	**Budget**	**Overall**
G->Y	2	2	2
G->R	2	0	0
Y->R	1	0	0
Project 6	**Schedule**	**Budget**	**Overall**
G->Y	1	0	0
G->R	0	0	0
Y->R	0	0	0
Project 7	**Schedule**	**Budget**	**Overall**
G->Y	3	1	2
G->R	0	0	0
Y->R	1	0	1

5.2.1. Interpreting Status Changes

The benefits of incorporating the RGM in a "traditional" development process become apparent when one examines the data presented in Table 7 and Table 8. For all three status categories (schedule, budget and overall), the two large projects and the two small projects of the RGM group experienced *no status changes due to late discovery of requirements throughout the development effort*. On the other hand, the single large project and the two small projects for the Non-RGM group all experienced less than desirable project status changes. Project 6, the best of the three for the Non-RGM group had only one status change stemming from the discovery of late requirements, and that was from green to yellow. (Again, consistent with observations supporting Conjecture 1, this project performed better than the other Non-RGM projects.) The other two Non-RGM projects shifted to yellow several times, and each also had shifts to red.

As stated earlier, the RGM is designed to promote the identification of a more complete and accurate set of requirements early in the software development lifecycle. Consequently, we should expect (and conjecture) that

> *projects using the RGM should exhibit less adverse impact stemming from late discovery of requirements.*

The difference between the number of status changes for the RGM group projects as compared to those for the Non-RGM group projects, i.e. 0 – 18, confirms our second conjecture.

5.3. Conjecture 3

> *Projects using the RGM result in a higher satisfaction level for the project manager and business customer a greater percentage of time throughout the project to closure.*

Unlike the previous two conjectures, which are substantiated through the examination of objective data, the confirming data for Conjecture 3 are more subjective in nature. In particular, this third set of data tracks the level of satisfaction (or frustration) as directly expressed by the project manager and customer. The project manager and customer independently record their current satisfaction (or frustration) in the project status database whenever the project status changes or milestones are reached. We compare the satisfaction indicators, "yes" or "no", to determine the percentage of satisfaction versus frustration over the project development time for the Project Manager (PM) and the Business Customer (BC). Table 9 and Table 10 show the satisfaction data for the RGM group and the Non-RGM group, respectively. That is, for each project the tables indicate the total number of times the PM and BC indicate satisfaction (yes) or frustration (no). The percentage of "yes" and "no" satisfaction responses relative to the total number is also given for the PM and BC (PM % and BC %).

Table 9: RGM Group Project Satisfaction

Project 1	**PM**	**PM %**	**BC**	**BC %**
Yes	5	100%	5	100%
No	0	0%	0	0%
Total	5		5	
Project 2	**PM**	**PM %**	**BC**	**BC %**
Yes	5	100%	5	100%
No	0	0%	0	0%
Total	5		5	
Project 4	**PM**	**PM %**	**BC**	**BC %**
Yes	6	100%	6	100%
No	0	0%	0	0%
Total	6		6	

Project 5	PM	PM %	BC	BC %
Yes	3	100%	3	100%
No	0	0%	0	0%
Total	3		3	

Table 10: Non-RGM Group Project Satisfaction

Project 3	PM	PM %	BC	BC %
Yes	8	57%	9	64%
No	6	43%	5	36%
Total	14		14	
Project 6	**PM**	**PM %**	**BC**	**BC %**
Yes	9	100%	9	100%
No	0	0%	0	0%
Total	9		9	
Project 7	**PM**	**PM %**	**BC**	**BC %**
Yes	4	44%	6	67%
No	5	56%	3	33%
Total	9		9	

5.3.1. Interpreting Satisfaction Levels

Table 9 shows that the PM and the BC were satisfied with the RGM projects 100% of the time. This is not surprising since the four projects (a) met or were close to their expected delivery dates (see Table 1), and (b) had no "bad" status changes throughout their development lifecycle (see Table 7). Overall, this shows that the projects using the RGM succeeded in meeting customer expectations. The satisfaction data for the Non-RGM group projects is provided in Table 10 and conveys a picture similar to the one formed when analyzing the data for Conjecture 2. That is, Project 6 faired better than the other two projects by garnering 100% satisfaction from both the PM and BC. For the other two Non-RGM projects, however, the BC was frustrated at least 1/3 of the time. An even higher level of frustration is indicated by the PMs for these two projects – they reported being frustrated with the project progress almost 50% of the time.

Hence, based on the data provided in Tables 9 and 10 we can state that (at least for this study)

The RGM promotes an increased level of satisfaction among project managers and business customers.

Anecdotal Note: During post-project reviews for the two RGM group projects (4 & 5), the business customer and other stakeholders stated that their expectations was were always met or exceeded. With respect to RGM methods, the *Silent Parking Lot* approach was judged by those involved as one of the best tools used during requirements elicitation and other meetings. The feeling of "not forgetting" necessary items, as well as "being heard without speaking" was articulated as a major benefit.

6. Conclusion

The data presented in this study contrasting RGM and Non-RGM development efforts has led to a confirmation of three conjectures. That is, the RGM

- helps in decreasing project slippage,
- reduces the detrimental impact stemming from late requirements discovery, and
- reduce the frustration levels for project managers and their business customers.

We further note that the elements of the RGM employed during this study was only a subset of those available – yet even this subset was effective in promoting better requirements definition. We expect that employing the full set of RGM activities and procedures will add additional strengthen and structure the all-too-often *ad hoc* requirements definition processes.

Structuring, monitoring, and controlling the requirements definition process are all goals of the RGM. Although still in its infancy, the RGM represents one additional step towards defining a comprehensive and integrated approach that provides effective support for the requirements generation process.

7. References

[1] J. D. Arthur, M. K. Gröner, K. J. Hayhurst, and C. M. Holloway, "Evaluating the Effectiveness of Independent Verification and Validation," *IEEE Computer*, vol. 32, pp. 79-83, 1999.

[2] B. Boehm, "A Spiral Model for Software Development and Enhancement," *Computer*, vol. 21, pp. 61-72, 1988.

[3] E. Bravo, "The hazards of leaving out the users," in *Participatory Design: Principles and Practices*, D. Schuler and A. Namioka, Eds. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., Publishers, 1993, pp. 3-11.

[4] M. K. Gröner and J. D. Arthur, "An Operational Model Supporting The Generation of Requirements That Capture Customer Intent," 17th Annual Pacific Northwest Software Quality Conference, Portland OR, Oct. 1999, pp. 286-302.

[5] P. Mambrey and B. Schmidt-Belz, "Systems Designers and Users: Fictions and Facts," in *Systems Design for, with, and by the user*, U. Briefs, C. Ciborra, and L. Schneider, Eds.: North-Holland Publishing Company, 1983, pp. 61-69.

[6] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," presented at WESCON, 1970.

[7] I. Sommerville, *Software Engineering*, 5th ed. Reading, Mass.: Addison-Wesley Publishing Co., 1996.