RAPID AND FLEXIBLE PRODUCT DEVELOPMENT: AN ANALYSIS OF SOFTWARE PROJECTS AT HEWLETT PACKARD AND AGILENT

by

Sharma Upadhyayula

M.S., Computer Engineering
University of South Carolina, 1991

Submitted to the System Design and Management Program in Partial Fulfillment
of the Requirements for the Degree of

Master of Science in Engineering and Management

at the
Massachusetts Institute of Technology

January 2001

© Sharma Upadhyayula. All rights reserved.
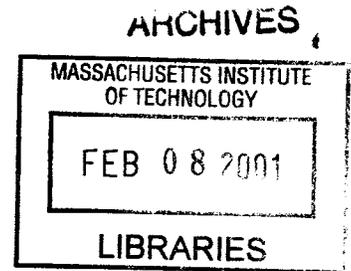
The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis
document in whole or in part.

Signature of Author....

System Design and Management Program
December 13, 2000

Certified by ....

Michael A. Cusumano

Sloan Management Review Distinguished Professor of Management

Thesis Supervisor

Accepted by...

Stephen C. Graves
Abraham Siegel Professor of Management
LFM/SDM Co-Director

Accepted by...

Paul A. Lagace
Professor of Aeronautics & Astronautics and Engineering Systems
LFM/SDM Co-Director

RAPID AND FLEXIBLE PRODUCT DEVELOPMENT: AN ANALYSIS OF SOFTWARE PROJECTS AT HEWLETT PACKARD AND AGILENT

by Sharma Upadhyayula

Abstract

Before companies started competing on Internet time, most companies involved in software product development carried out the different phases of the product development sequentially. If, during the later stages of product development (ex: coding), the company came across new information or the user needs changed then these changes would be incorporated into the next version of the product otherwise risk shipping the product late. Rapid innovation in the technological areas and the Internet has created very dynamic environment in all walks of life. In this environment, the user needs are changing very rapidly resulting in new challenges for the companies and its product development managers. They have to respond to the changing needs of the users very quickly either with rapid product releases and/or incorporating the changes into product under development. To achieve this, companies need a product development strategy that allows them to incorporate changes at any stage in the product development without affecting their time-to-market.

This thesis focuses on strategies for rapid and flexible software product development. This research will study systematically the range of approaches that producers of software and hardware use for product development.

Acknowledgements

Many people have contributed to this research project and the creation of this thesis. I am very grateful to all the people who made it possible.

Table of Contents

List of Tables

Table of Figures

Chapter 1: Introduction

1.1 Motivation:

Technologies, competitor moves, and user needs change so quickly that companies can no longer plan specifically what they will do and then proceed through long sequential product-development cycles. One approach is an iterative process that combines preliminary design goals and some design details with continual feedback from users as well as outside partners during development. Simultaneously, designers attempt to build and integrate components and prototypes as the project moves forward. Companies can also try to influence the direction toward which their products evolve by controlling architectures of their product platforms and by working with producers of complementary products. Many firms, however, have been slow to adopt the more iterative processes to product development. One reason may be that it is difficult to control such a process and know when to stop iterating. As a result, the outcomes and dates are less predictable than a sequential process, and there is likely to be less waste and rework in a sequential process. There are also few detailed case studies or statistically documented studies on how to manage an iterative development process effectively.

This research will study systematically the range of approaches that producers of software and hardware for personal computers and, especially, Internet applications use for strategic planning and product development.

Benefits:

New and deeper understanding of how firms can structure and manage iterative and cooperative approaches to product development in rapidly changing markets

Define when an incremental approach to product development, as opposed to a more sequential approach, is useful as well as difficult to introduce.

- Description of current s/w development processes

- Description of evolutionary development process

- Strategies

1.2 Existing methodologies and techniques common to software product development

1.2.1 Sequential (Waterfall) Methodology:

One of the software product development methodologies that was popular in the 70s and 80s is the sequential (waterfall) methodology. The typical sequential (waterfall) product development process consists of requirements phase, detailed design phase, module coding and testing phase,

```
┌─────────────┐
│ Requirements│
│    Phase    │
└─────────────┘
        │
        ▼
   ┌──────────────┐
   │Detailed Design│
   │    Phase      │
   └──────────────┘
           │
           ▼
      ┌──────────────┐
      │ Module Coding │
      │and Testing Phase│
      └──────────────┘
              │
              ▼
         ┌──────────────┐
         │  Integration  │
         │ Testing Phase │
         └──────────────┘
                 │
                 ▼
            ┌──────────────┐        ┌──────────┐
            │    System     │ ─────▶ │ Product  │
            │ Testing Phase │        │ Release  │
            └──────────────┘        └──────────┘
```
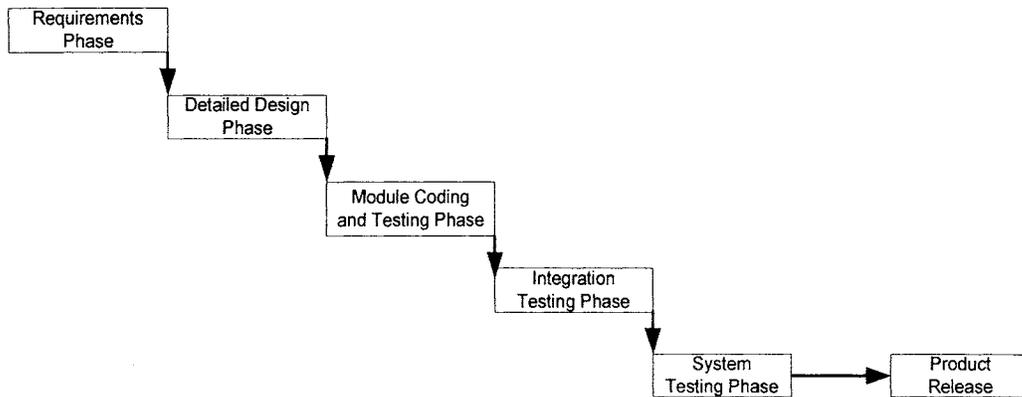
Figure 1-1 - Sequential (Waterfall) Methodology

integration testing phase and system testing phase[1] as shown in figure 1-1.

"Sequential approach to software development may require very long periods of time because they schedule work in sequence, not in parallel. Managers may also find it difficult to assess progress accurately because they tend to schedule major testing very late-often too late in the development cycle"[2]. Sequential approach has been shown to be extremely effective in stable environments but its effectiveness has been questioned in uncertain and dynamic environments[3].

---

[1] Michael A. Cusumano and Richard W. Selby, Microsoft Secrets, Free Press 1995, p 192
[2] Michael A. Cusumano and Richard W. Selby, Microsoft Secrets, Free Press 1995, p 262
[3] Alan MacCormack, Roberto Verganti, and Marco Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process", *Harvard Business School Working paper 99-118*, 1999, p 4

1.2.2 Iterative (Evolution) Methodology:

The challenge for product teams in uncertain and dynamic environments is that user needs for many types of software are so difficult to understand that it is nearly impossible or unwise to try to design the system completely in advance, especially as hardware improvements and customer desires are constantly and quickly evolving[4]. In the iterative approach, the product development cycle is divided into sub-cycles with each sub-cycle consisting of design, develop, build, test and release activities.

This methodology emphasizes the ability to respond to new information from market (customer) and technical (engineering) feedback for as long as possible during a development cycle[5]. The iterative (evolutionary) approach to product development is favored because companies usually build better products if they have the flexibility to change specifications and designs, get and incorporate market (customer) and technical (engineering) feedback, and continually test

Market (Customer) and Technical (Engineering) feedback

Specification

Design

Develop

Integration (Build/Test)

Release

Final Product Stabilization Phase
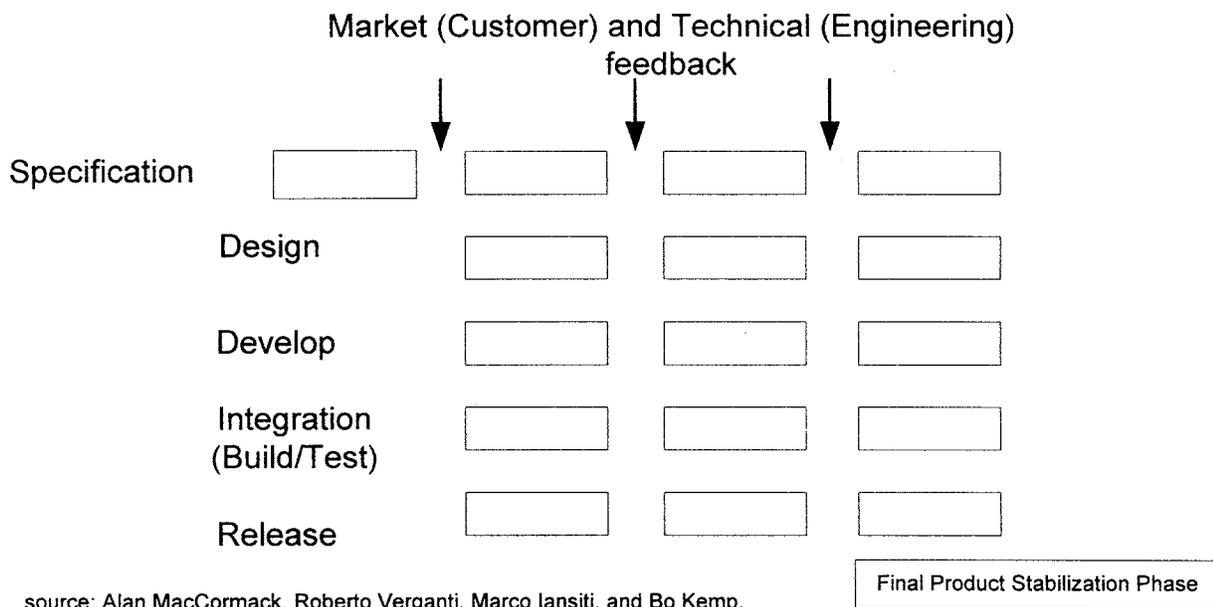
source: Alan MacCormack, Roberto Verganti, Marco Iansiti, and Bo Kemp, "Product Development Performance In Internet Software", Harvard Business School, September 1997, p 6

Figure 1-2 - Iterative (Evolutionary) Methodology

---

[4] Michael A. Cusumano and Richard W. Selby, Microsoft Secrets, Free Press 1995, p 14
[5] Alan MacCormack, Roberto Verganti, and Marco Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process", *Harvard Business School Working paper 99-118*, 1999, p 6

components as the products are evolving. The product teams also ship preliminary versions of their products, incrementally adding features or functionality over time in different product releases[6].

1.2.3 Synch and Stabilize technique:

Many product teams, in addition to the above-mentioned iterative (evolutionary) approach, also put pieces of their products together frequently. This is useful to determine what works and what does not, without waiting until the end of the project[7]. Figure 1-3 provides an overview of synch-and-stabilize development approach.

---

[6] Michael A. Cusumano and Richard W. Selby, Microsoft Secrets, Free Press 1995, p 14-15
[7] Michael A. Cusumano and Richard W. Selby, Microsoft Secrets, Free Press 1995, p 15

Planning Phase: Define product vision, specification and schedule.

---

* Vision Statement: Product and program management use extensive customer input to identify and prioritize product features.
* Specification Document: Based on vision statement, program management and development group define feature functionality, architectural issues, and component interdependencies.
* Schedule and Feature Team Formation: Based on specification document, program management coordinates and arranges feature teams that each contain approximately 1 program manager, 3-8 developers, and 3-8 testers (who work in parallel 1:1 with developers)

Development Phase: Feature development in 3 or 4 sequential subprojects that each results in a milestone release.

Program managers coordinate evolution of specification. Developers design, code and debug. Testers pair up with developers for continuous testing.
* Subproject I: Firs 1/3 of features: Most critical features and shared components..
* Subproject II: Second 1/3 of features.
* Subproject III: Final 1/3 of features: Least critical features

---

Stabilization Phase: Comprehensive internal and external testing, final product stabilization, and ship..

Program managers coordinate OEMs and ISVs and monitor customer feedback. Developers perform final debugging and code stabilization. Testers recreate and isolate errors.
* Internal Testing: Thorough testing of complete product within the company.
* External TestingI: Thorough testing of complete product outside the company by "beta" site s such as OEMs, ISVs, and end-users.
* Release PreparationI: Prepare final release of "golden master" diskettes and documentation for manufacturing.

Figure 1-3 - Overview of Synch-and-Stabilize Development Approach

Chapter 2: Research Methodology

2.1 Questionnaire Development:

A questionnaire was developed and used to allow systematic collection of data. The team involved in developing the questionnaire consisted of members from the academic community and industry members (from Hewlett Packard), along with the author. The academic community members are: Prof. Michael A. Cusumano (MIT Sloan School of Management) who is the thesis advisor to the author, Prof. Chris F. Kemerer (Katz Graduate School of Business, University of Pittsburgh), Prof. Alan MacCormack (Harvard Business School). The industry members from Hewlett Packard are Bill Crandall and Guy Cox. Both Bill Crandall and Guy Cox represented Process Consulting Group (PCG) within Hewlett Packard.

The objective of the questionnaire was to capture all pertaining information about a software development project that would provide us with:

- Will provide the ability to benchmark development practices, at Hewlett Packard and Agilent initially and subsequently (in future work) at a larger cross-section of companies globally.

- Helps in identifying variables, which contribute most to performance providing insights into approaches for rapid and flexible software product development.

Iterative (evolutionary) process was used to design and develop the questionnaire. The questionnaire consists of two parts. Part 1 of the questionnaire was focused on project description and environment, size of the project (with respect to development budget, development effort, project schedule and lines of code), origins of the software code (code from previous version, other products, off-the-shelf code, new code developed by the team), project team roles composition, design and development process, testing and debugging process, relative emphasis on different types of testing during the project, interaction with customers (a customer can be internal or external).

Part 2 of the questionnaire was focused on various project activities (requirements phase, architectural and functional design phase, detailed design and development phase, and integration and system testing phase), product development methodologies (sequential (waterfall), iterative (evolutionary) and synch-and-stabilize approach), project performance metrics (financial

performance, market performance, schedule performance, budget performance and software quality).

2.2 Data collection:

In order to facilitate an efficient data collection process, the author created a website which would allow the project team representatives to view the questionnaire and answer the questions. M.I.T through its CWIS group provides its students the ability to create forms based questionnaire and host it on web.mit.edu. After the project team representatives submit responses to the questions, the information is received as email. M.I.T's CWIS group also provides perl script, which prepares the information received as email into tab-delimited records for inclusion into a database or spreadsheet.

A more efficient process could be implemented by storing the responses directly in a database after the project team representatives submit responses to the questions. The infrastructure provided by M.I.T's Web Communications Services (WCS) group did not allow database support for the forms at the time the research was carried out. In future research work where there is a potential for large number of responses, database support for the forms (questionnaire) should be considered, if the infrastructure allowed.

Initial approach of the author was to import the data received into Microsoft Access® database. The reason for this approach was to provide, the academic and industry members of the research team, reports on the cases that were received. Due to the large number of variables being used to collect the data, the author quickly ran into issues while designing report(s) to display data for each case in its entirety. The author, realizing that using Access database may not provide various statistical analysis methods that could be used to analyze the data collected, imported all the data into SPSS® 10.0 application package.

The author used the SPSS® application to run all the statistical analysis except for a brief period of time where another statistical analysis package, Data Desk® 6.0 was used. The author started using Data Desk® because of certain usability features but realized that certain statistical information (like significance level) for some analysis was not being provided. This drawback led the author back to using the SPSS® package.

The industry members of the research team were instrumental in contacting various project team representatives to participate in the research. The goal of the research team, at the beginning of

the research project, was to collect data from 40 projects. The initial expectation of the research team was that the data collection would not be a problem since both industry members, Bill Crandall and Guy Cox, were part of Hewlett Packard and knew and/or had contacts for large number of projects. As the data collection process went live, the research team realized that the data collection process was not going as expected. To spread the word and motivate potential respondents (project team representatives) of the questionnaire, Prof. Michael A. Cusumano and Prof. Alan MacCormack spoke at Hewlett Packard and Agilent internal seminars on software product development. In addition to the above actions, by the research team, Bill Crandall also provided $50 gift certificates (towards purchases at Amazon.com) to project team representatives who participated in the research.

At the end of the data collection process, the research team received surveys for 27 projects. For the data analysis, the author used 26 project surveys for the sample. The one project that was not included in the data analysis is a very small project. The duration of this project was 1-week. The project duration was too small to study various project activities. The smallest project that was considered for data analysis in the sample was 4 months long.

After the data was collected and initial analysis was performed, the research team realized that some additional information was required in the areas % of original features implemented in the final product, bugs reported by the customer in the first 12 months after the product launch and project performance ratings (customer satisfaction rating, schedule and budget performance rating) as perceived by the project team members. Of the 26 projects in the sample, we received responses from 22 project teams for the additional questionnaire.

2.3 Variables (Context, Process and Outcome):
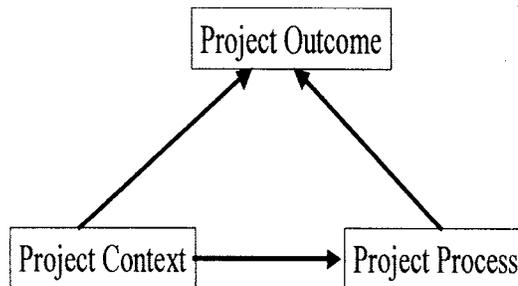
For analysis, we studied the relation between several process variables and the outcome variables.

2.3.1 Some of the contexts variables available from the research data:

- Type of Software developed:

  - Systems software

  - Application software

  - Embedded software

- Software developed for:

  - External customers

  - Internal customers

- New product development or extension of current product functionality

  - < 50% of code reuse from a previous project is assumed to be new product

- Project size:

  - Lines of code (LOC)

  - Project duration

  - Project budget

  - Effort in person years

- Team Composition:

  - Development resources

  - Testing resources

2.3.2 Some of the process variables available from the research data:

- User Orientation

- How early were the prototypes (with respect to functionality)

- How early were the system integrations (with respect to functionality)

- How early were the betas (with respect to functionality)

- Number of Sub-cycles

- Frequency of synchronization (Build frequency)

- Reviews:

    - Design reviews

    - Code reviews

- Build validation

    - Simple compile and link tests Vs. Regression tests

- Testing and debugging

    - Time spent by developers in testing Vs. Time spent by QA or testing staff

- Relative Emphasis of testing in a project

    - % Of total testing time spent in Component testing, % of total testing time spent in Integration testing, % of total testing time spent in System testing

- Testing Effort

    $AverageTestingresources / (averageDevelopment + avgTestingresources)$

- Flexibility of the process/project:

    - How late into the project schedule were the requirements changing?

    - How late into the project schedule was the team changing the design?

- How late into the project was the team able to add new code?

- Length of the first sub-cycle (which is elapsed time from project start to first system integration) – indication of time taken to implement the core/important functionality (similar to what Tom Gilb has in the evolutionary approach – Juicy bits first principle)

- Architecture Effort:

  $Architecturalresources / (development + testingresources)$

- Amount of Code Reuse

2.3.3 Some of the outcome variables available from the research data:

- Productivity (LOC per person day): productivity is defined as new lines of code developed per person day. To calculate this, total person years was used which includes project managers, arch, developers, testers etc.

  $NewLinesOfCode / (TotalPersonYears * 250)$

- % Schedule estimation error: is defined as

  $(actualprojectdurtion - estimatedprojectduration) * 100 / actualprojectduration$

- Bugginess (Average number of bugs per million Lines of code reported per month during the first 12 months after the system launch)

  $(NumberOfBugsreportedByCustomer * 1000000) / (NumberOfMonths * NewLinesOfCode)$

- Customer satisfaction Perception Rating: This variable is customer satisfaction rating as perceived by the project team.

- Schedule and Budget performance Perception Rating: This variable is schedule and budget performance rating as perceived by the project team.

- Financial return Perception Rating: This variable is a measure of financial return from the project as perceived by the project team.

A 5-point scale was used to measure customer satisfaction perception rating, schedule and budget performance perception rating and financial return perception rating, where 1= significantly below expectations, 2=below, 3=met expectations, 4=above, 5=significantly above expectations.

2.4 Generic project description (size, complexity etc):

This section summarizes the data of projects used in the sample. Table 2-1 summarizes the raw data for some of the context variables. The table shows the size of the projects in terms of actual lines of code, new code developed for the project, project duration and project development and testing resources.

Variable	Count	Mean	Median	StdDev	Min	Max
Actual LOC	26	671306	160000	1.66E+06	1320	8.50E+06
Log(Actual LOC)	26	5.1819	5.20327	0.868566	3.12057	6.92942
New Code	26	368342	57369	1.32E+06	255	6.80E+06
Log(New Code)	26	4.69759	4.75859	0.93063	2.40654	6.83251
Total Development + Testing Resources	25	11.612	6	14.2048	2	55
Total resources (in person Years)	26	27.4192	9.5	39.5967	0.2	160
Project Duration	26	18.7692	14.5	11.0247	4	45

Table 2-1 - Descriptive Statistics for Context Variables

Table 2-2 summarizes the various types of software in the sample. The different types of software in the sample are application software, system software, embedded software and other (projects with a combination of application, system and/or embedded software).

21

Software Type	Count
Application	8
System	6
Embedded	5
Other	7

Table 2-2 - Breakdown of Sample by Software Type

Table 2-3 shows the sample breakdown or grouping by customer i.e., internal customer (use) and external customer (use).

Group	Count
External Use	18
Internal Use	8

Table 2-3 - Projects Grouped by Usage

Another variable used for grouping the projects is based on whether it is a new product or a product extension. A product extension is defined, as a project with percentage of code reuse from a previous project is greater than 50%. The grouping is showed in table 2-4.

Group	Count
New Product	18
Product Extensions	8

Table 2-4 - Projects Grouped by Project Type

Tables 2-5, 2-6, 2-7 provide descriptives for various process variables. The data for some of the process variables are derived (or calculated) from the raw data provided by the project teams.

Variable	Count	Mean	Median	StdDev	Min	Max
Requirements Phase (months)	26	7.69231	5.5	7.13733	0	24
Design Phase (months)	26	10.9615	7.5	8.90609	0	33
Development Phase (months)	26	11.1538	8	8.66576	2	30
Integration Phase (months)	26	6.88462	5.5	7.33936	1	37
Stabilization Phase (months)	26	2.40385	2	2.8285	0	13
Number of Betas	26	2.61538	2	2.46701	0	10
Architectural Effort	25	0.295969	0.2	0.276657	0.02	1
% code reuse	26	0.603077	0.625	0.246021	0	0.9

Table 2-5 - Descriptive Statistics for Process Variables

Variable	Count	Mean	Median	StdDev	Min	Max
Developers testing their code (as % of total testing time)	26	0.529615	0.5	0.302793	0.07	1
QA staff testing code (as % of total testing time)	24	0.490833	0.5	0.278223	0	0.93
Component Testing (% of total testing time)	26	0.313462	0.25	0.232189	0	0.85
Integration Testing (% of total testing time)	25	0.266	0.2	0.153921	0	0.6
System Testing (% of total testing time)	26	0.426923	0.4	0.239262	0.1	1
Testing Effort	25	0.252893	0.225	0.145212	0	0.5

Table 2-6 - Descriptive Statistics for Process Variables

Variable	Count	Mean	Median	StdDev	Min	Max
% of Elapsed time at first prototype	24	33.8989	25.8333	22.9054	4.54545	83.3333
% of Elapsed time at first system integration	26	58.5214	59.7222	17.1573	25	93.3333
% of Elapsed time at first beta	19	77.8108	81.8182	17.2383	30.4348	102.778
% of functionality in first prototype	24	37.4167	36.5	25.2499	0	90
% of functionality at first system integration	24	63.0417	63.5	20.6976	15	100
% of functionality in first beta	25	91.8	95	7.04746	80	100

Table 2-7 – Descriptive Statistics for Process Variables

Table 2-8 provides a grouping of projects based on their build frequency. The other category includes projects with weekly, bi-weekly and monthly build frequency. Table 2-9 summarizes projects that have performed regression tests after developers checked changed or new code into the project build. Table 2-10 provides a breakdown of projects, which performed design reviews, and the projects that did not perform design review. Table 2-11 provides a breakdown of projects, which performed code reviews, and the projects that did not perform code review.

Group	Count
Daily Build	11
Other	15

Table 2-8 – Summary of Build Frequency

Group	Count
Regression Tests Performed	17
Regression Tests Not Performed	9

Table 2-9 – Projects grouped by whether Regression Tests were performed or not

Group	Count
Design Review Done	22
Design Review Not Done	4

Table 2-10 - Projects grouped by whether Design Review was done or not

Group	Count
Code Review Done	14
Code Review Not Done	12

Table 2-11 - Projects grouped by whether Code Review was done or not

Table 2-12 provides descriptives for some of the outcome variables. Bugginess, productivity and % schedule estimation error are derived variables.

Variable	Count	Mean	Median	StdDev	Min	Max
% of original features implemented in the final product	22	82.0455	90	19.2984	40	100
Schedule and Budget Performance Perception Rating	22	2.5	2	0.859125	1	4
Customer Satisfaction Perception Rating	22	3.5	3.5	0.672593	2	5
Financial Return Perception Rating	20	3.55	3	0.998683	1	5
Bugginess	21	464.755	12.5	2032.2	0	9333.33
Productivity	26	548.512	99.0933	2204.64	0.96	11333.3
% Schedule Estimation Error	26	73.932	40.6593	86.2291	0	340

Table 2-12 - Descriptive Statistics for Outcome Variables

Since the sample set contained several types of projects, to evaluate the significance of the mean with respect to the mean of the various groupings of the project, an analysis of variance (ANOVA) was performed. Analysis of variance was performed for the following process variables:

- % Functionality in first prototype

- % Functionality in first system integration

- % Functionality in first beta

25

- % Elapsed time till last major requirements change

- % Elapsed time till last major functional specification change

- % Elapsed time till last major code addition

- Architectural effort

- % Code reuse

- % Total testing time developers spent testing their own code

- % Total testing time QA staff spent testing code

- % Total testing time spent in component testing

- % Total testing time spent in integration testing

- % Total testing time spent in system testing

The ANOVA was performed for the above process variables under three separate groupings and they are:

- Software Use (Internal Use Vs. External Use)

- Software Type (Application S/W, System S/W, Embedded S/W and Others – combination of application, system and embedded software)

- New Products Vs. Product Extensions

Based on the ANOVA reports, it appears that in all cases (except % Code Reuse) the process variables are independent of how the projects are grouped. Appendix-A contains the ANOVA reports for the above-mentioned variables under the project groupings mentioned earlier.

Chapter 3: Data Analysis

3.1 Hypothesis and data analysis:

Since the process variables were independent of project groupings, in the data analysis to evaluate various hypotheses, the data for the entire sample set was used as one group. Most of the hypothesis constructs relationship between process variables and outcome variables. The one exception is the 'percentage code reuse' variable. As part of the data analysis, Spearman Rank Correlation analysis was performed to evaluate the hypotheses. The hypotheses and analysis is focused on incremental (evolutionary) feature development, frequent synchronization and testing. To address the above-mentioned topics, detailed analysis was performed in the following areas:

Incremental (evolutionary) feature development:

- Market (customer/user) feedback. The feedback is based on the final product functionality available in the product. This is evaluated at two key milestones, which are the first prototype and first beta[8].

- Technical feedback. This is the feedback provided by the engineers (development and build). The feedback is based on the final product functionality available in the product. This is evaluated at first system integration milestone[9].

- Impact of separate development sub-cycles.

- Flexibility in project activities.

- Code reuse.

Frequent synchronization:

- Frequent synchronization.

---

[8] Alan MacCormack, Roberto Verganti, and Marco Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process", *Harvard Business School Working paper 99-118*, 1999, pp 14-15

[9] Alan MacCormack, Roberto Verganti, and Marco Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process", *Harvard Business School Working paper 99-118*, 1999, pp 14-15

Testing:

- Design and Code reviews.

- Testing (simple compile and link Vs. regression tests).

- Impact of developers and QA staff testing code.

- Relative emphasis of testing (component testing, integration testing, system testing).

- Impact of Final stabilization phase.

3.2 Impact Of Market and Technical Feedback

3.2.1 Hypothesis 1:

Obtaining market (first prototype and first beta) and technical (first system integration) feedback early in the project, with respect to functionality, allows the team to incorporate more feature changes based on the market and technical feedback. Thus the project is more flexible. This results in:

- Increased feature evolution

- Increased customer satisfaction

3.2.2 Hypothesis 2:

Incorporating more market and technical feedback, increases the schedule estimation error *(the obvious tradeoff is that as less feedback is incorporated, the schedule estimation error decreases)*.

3.2.3 Hypothesis 3:

As projects obtain technical feedback early in the project, the bugginess of the product will decrease.

3.2.4 Hypothesis 4:

As projects obtain early market feedback, the bugginess could increase as the team makes changes to incorporate the market feedback.

3.2.5 Hypothesis 5:

As projects obtain feedback early in the project, the productivity improves since it reduces potential rework (because the amount of functionality implemented is less).

The process variables used to evaluate market and technical feedbacks are:

- % Functionality implemented in first prototype

- % Functionality implemented at first system integration

- % Functionality implemented in first beta

The outcome variables used to evaluate the impact of market and technical feedbacks are:

- % Original features implemented in the final product

- Productivity

- % Schedule estimation error

- Customer satisfaction perception rating

3.2.6 Data Analysis to evaluate impact of market and technical feedback

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% final product functionality in first prototype	% final product functionality in first system integration	% final product functionality in first beta
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	.223	.348	.674**
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.345	.133	.001
	N	21	20	21	21	21	21	20	20	20
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	.672**	.134	.363
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.002	.583	.127
	N	20	20	20	20	20	20	19	19	19
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.025	.039	-.471*
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.908	.859	.020
	N	21	20	25	25	21	21	23	23	24
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.266	-.080	-.260
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.221	.716	.220
	N	21	20	25	25	21	21	23	23	24
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	.191	.128	.429
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.420	.590	.059
	N	21	20	21	21	21	21	20	20	20
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.288	-.537*	-.194
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.218	.015	.414
	N	21	20	21	21	21	21	20	20	20
% final product functionality in first prototype	Correlation Coefficient	.223	.672**	-.025	-.266	.191	.288	1.000	.476*	.482*
	Sig. (2-tailed)	.345	.002	.908	.221	.420	.218	.	.022	.020
	N	20	19	23	23	20	20	23	23	23
% final product functionality in first system integration	Correlation Coefficient	.348	.134	.039	-.080	.128	-.537*	.476*	1.000	.499*
	Sig. (2-tailed)	.133	.583	.859	.716	.590	.015	.022	.	.015
	N	20	19	23	23	20	20	23	23	23
% final product functionality in first beta	Correlation Coefficient	.674**	.363	-.471*	-.260	.429	-.194	.482*	.499*	1.000
	Sig. (2-tailed)	.001	.127	.020	.220	.059	.414	.020	.015	.
	N	20	19	24	24	20	20	23	23	24

**. Correlation is significant at the .01 level (2-tailed).

*. Correlation is significant at the .05 level (2-tailed).

Table 3-1 - Market and Technical Feedback Correlation Table
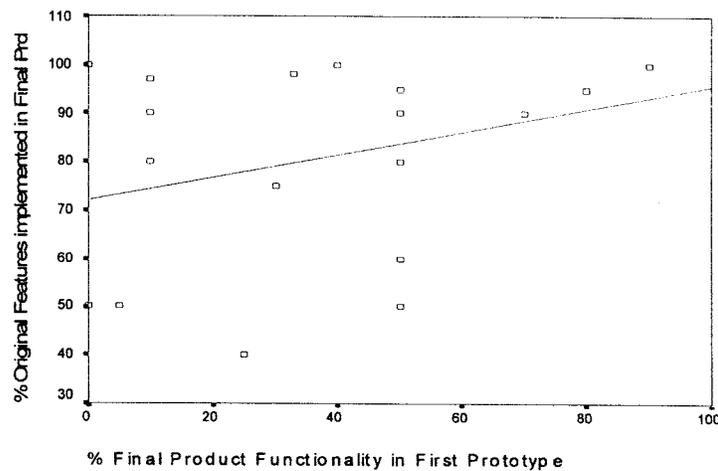
Figure 3-1- scatter plot of % final product functionality implemented in first prototype vs. % original features implemented in the final product (all projects)

Correlation between % functionality implemented in first prototype and % original features implemented in the final product: **0.223.** The correlation between these two variables is not statistically significant. The idea of having an early prototype, with respect to functionality, is to obtain and be able to incorporate customer/user feedback into the product. The lack of statistically significant correlation could be because the project teams may have implemented the market feedback in the project (with respect to functionality) and therefore final product functionality might be different than the original features that the product team started with. Potentially, the correlation may be significant if the projects completed the prototype later in the project (with respect to functionality) i.e. potentially implementing more original functionality before releasing the prototype.

% Original Features implemented in Final Prd

110 100 90 80 70 60 50 40 30

0   20   40   60   80   100   120

% Final Product Functionality in First System Integration
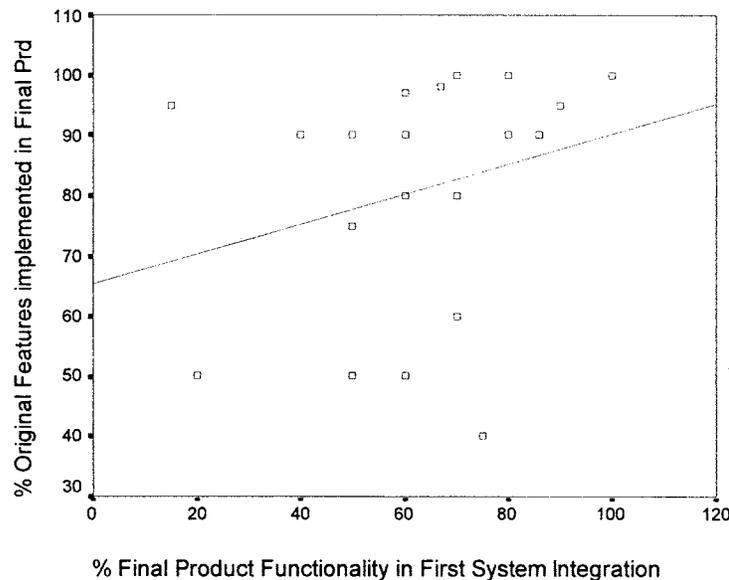
Figure 3-2 - scatter plot of % final product functionality implemented in first system integration vs. % original features implemented in the final product (all projects)

Correlation between % functionality implemented in first system integration and % original features implemented in the final product: **0.348.** The correlation between these two variables is statistically not significant. . The idea of having early system integration, with respect to functionality, is to obtain and be able to incorporate technical (engineering) feedback into the product. The lack of statistically significant correlation could be because the project teams may have incorporated the technical feedback and therefore final product functionality might be

31

different than the original features that the product team started with. Potentially, the correlation may be significant if the projects integrated the system later in the project (with respect to functionality) i.e. potentially implementing more original functionality before integrating the system.
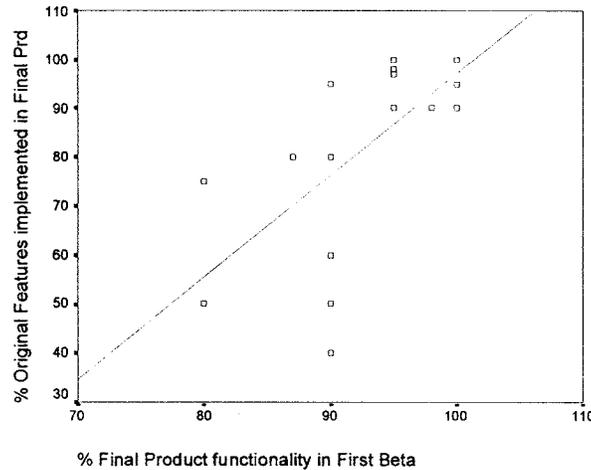
Figure 3-3 – scatter plot of % final product functionality implemented in first beta vs. % original features implemented in the final product (all projects)

Correlation between % functionality implemented in first beta and % original features implemented in the final product: **0.674.** The Correlation between the two variables is statistically significant at the 0.01 level (two-tailed). The correlation is significant because the first beta is released late in the project with respect to functionality.
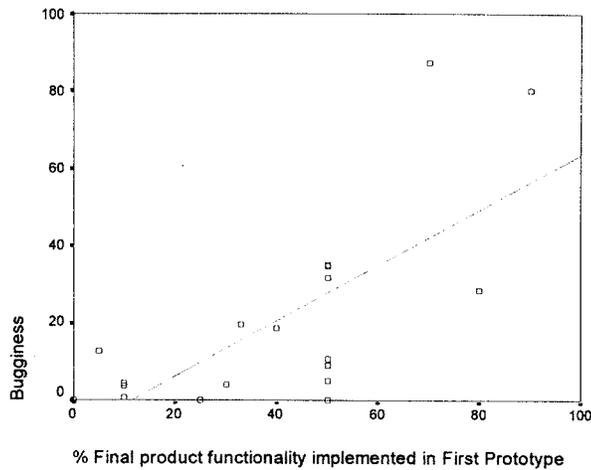
Figure 3-4 - % final product functionality implemented in first prototype Vs. Bugginess (all projects)

Correlation between % final product functionality implemented in first prototype and bugginess: **0.672.** The Correlation between the two variables is statistically significant at the 0.01 level (two-tailed). As more functionality is implemented in the first prototype it becomes difficult for the project team to incorporate market feedback and if the project team does incorporate the market feedback, then the team potential has created an environment to introduce more bugs due to the rework.

100

80

60

40

20

0

Bugginess

0   20   40   60   80   100   120

% Final Product Functionality in First System Int.
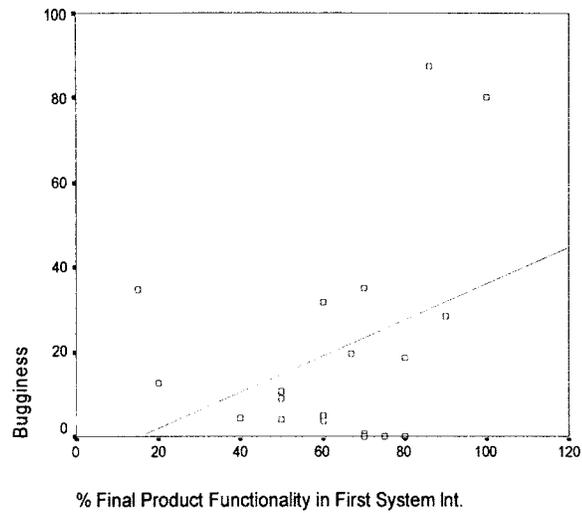
Figure 3-5 - % final product functionality implemented in first system integration Vs. Bugginess (all projects)

Correlation between % final product functionality implemented in first system integration and bugginess: **0.134.** The correlation between these two variables is statistically not significant. The possible reason for lack of statistically significant correlation between these two variables could be because bugginess is a measure of bugs reported by the end user and since the end user sees a system which has been integrated. The correlation potentially could be significant if based on the issues faced by the project team to integrate the system and the team ends up changing the functionality already implemented to resolve the integration issues. This situation could potentially lead to more bugs in the product due to the rework.

100

80

60

40

Bugginess 20

0

70   80   90   100   110

% Final Product Functionality in First Beta
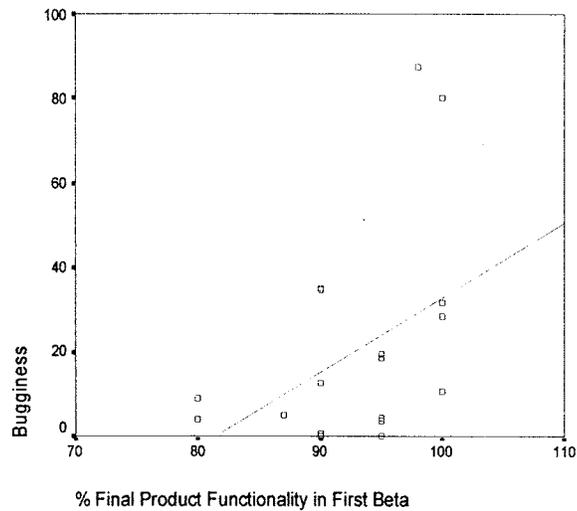
Figure 3-6 - % final product functionality implemented in first beta vs. Bugginess (all projects)

Correlation between % final product functionality implemented in first beta and bugginess: **0.363.**
The correlation between these two variables is statistically not significant. The correlation is
potentially statistically not significant because the first beta, for the projects, is released late in the
project with respect to functionality. This may result in the project team not implementing the
market feedback and if there is no rework then the team has possibly avoided opportunities to
introduce bugs due to rework.
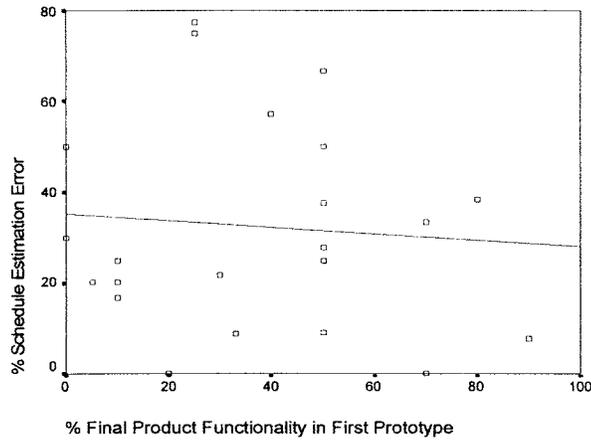
Figure 3-7 - % final product functionality implemented in first prototype vs. % schedule estimation error (all projects)

Correlation between % final product functionality implemented in first prototype and % schedule estimation error: **-0.025.** The correlation between these two variables is statistically not significant. To better understand why this correlation is statistically not significant, it would be helpful to understand the feature changes, due to customer feedback that were implemented compared to the original product functionality that the team started with.
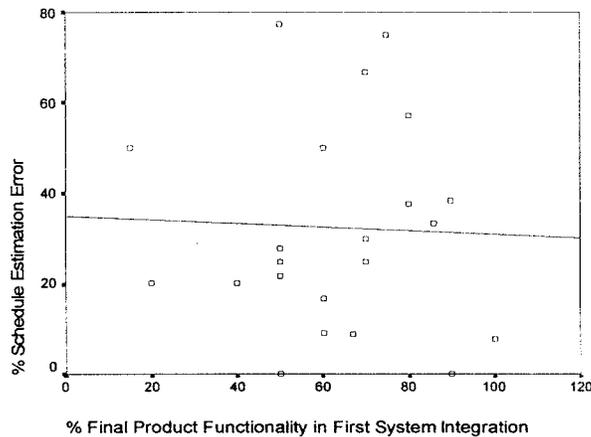
Figure 3-8 - % final product functionality implemented in first system integration vs. % schedule estimation error (all projects)

Correlation between % final product functionality implemented in first system integration and % schedule estimation error: **0.039.** The correlation between these two variables is statistically not

significant. One possible reason that this correlation is not statistically significant could be that the product did not require any rework, irrespective of the functionality implemented at the time of first system integration.
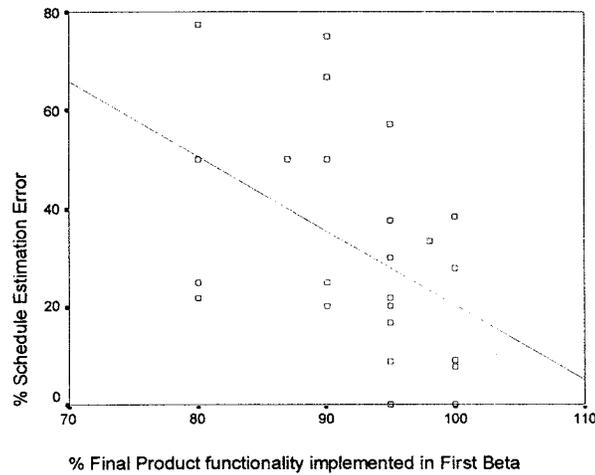
Figure 3-9 - % final product functionality implemented in first beta vs. % schedule estimation error (all projects)

Correlation between % final product functionality implemented in first beta and % schedule estimation error: **-0.471**. The Correlation between the two variables is statistically significant at the 0.05 level (two-tailed). The possible reason for the correlation to be significant could be that as the team implements more functionality in first beta, the team probably will be less inclined to incorporate customer feedback thereby reducing schedule estimation error.

37
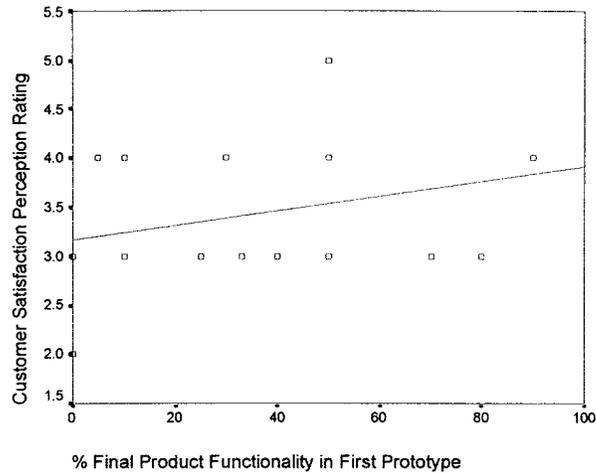
Figure 3-10 - % final product functionality implemented in first prototype vs. customer satisfaction perception rating (all projects)

Correlation between % final product functionality implemented in first prototype and customer satisfaction perception rating: **0.288.** The correlation between these two variables is statistically not significant. Even though this correlation is statistically not significant, it is a very interesting correlation i.e. as the % final product functionality implemented in first prototype is increasing so does the customer satisfaction perception rating. The basic idea of iterative (evolutionary) approach is the ability to obtain and incorporate customer feedback as the customer needs evolve. Since majority of the projects that are part of the sample are hardware dependent, it might be necessary to implement more functionality in first prototype to demonstrate the concepts to the customers. This could potentially lead to higher customer satisfaction perception rating.
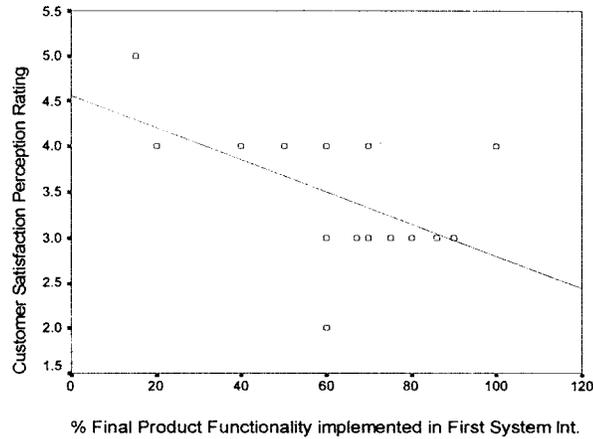
Figure 3-11 - % final product functionality implemented at first system Integration Vs. customer satisfaction perception rating (all projects)

Correlation between % final product functionality implemented at first system integration and customer satisfaction perception rating: **-0.537.** The Correlation between the two variables is statistically significant at the 0.05 level (two-tailed).
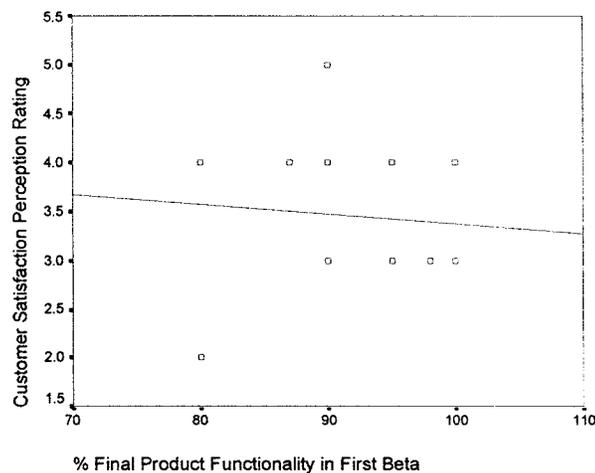
Figure 3-12 - % final product functionality implemented at first beta vs. customer satisfaction perception rating (all projects)

Correlation between % final product functionality implemented at first beta and customer satisfaction perception rating: **-0.194.** The correlation between these two variables is statistically

not significant. The correlation between these two variables is negative and as more functionality is implemented in the first beta it becomes less likely that the project team would incorporate the customer feedback from the beta resulting in a product with unmet customer needs.
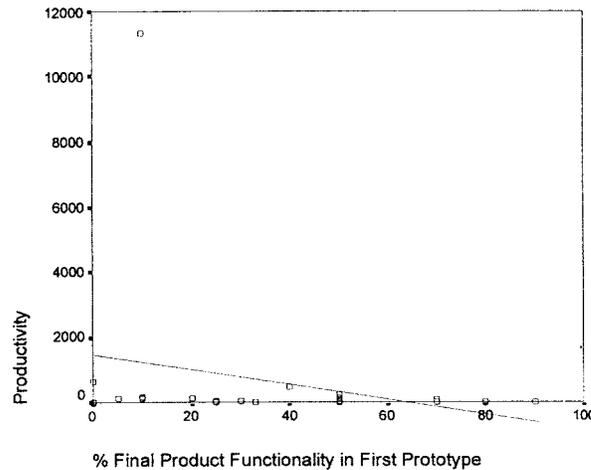
Figure 3-13 - % final product functionality implemented at first prototype vs. productivity (all projects)

Correlation between % final product functionality implemented at first prototype and productivity: **-0.266.** The correlation between these two variables is statistically not significant. The correlation suggests that as the % final product functionality implemented at first prototype is increasing the productivity of the project team is decreasing. This could be due to more rework as a result of customer feedback on the functionality that was implemented.
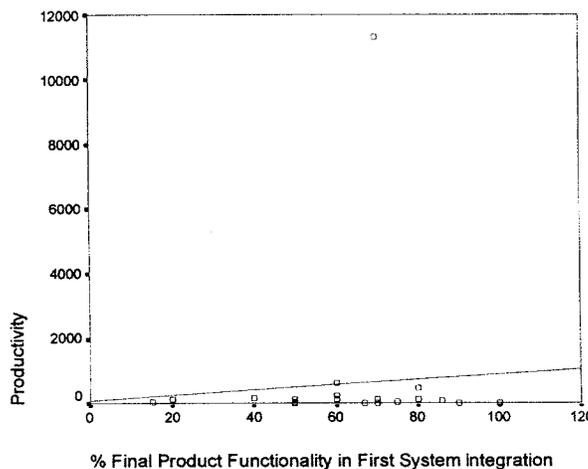
Figure 3-14 - % final product functionality implemented at first system integration vs. productivity (all projects)

Correlation between % final product functionality implemented at first system integration and productivity: **-0.080.** The correlation between these two variables is statistically not significant. The two variables have very little correlation and this may be due to smooth system integration or no major system integration issues that require rework.
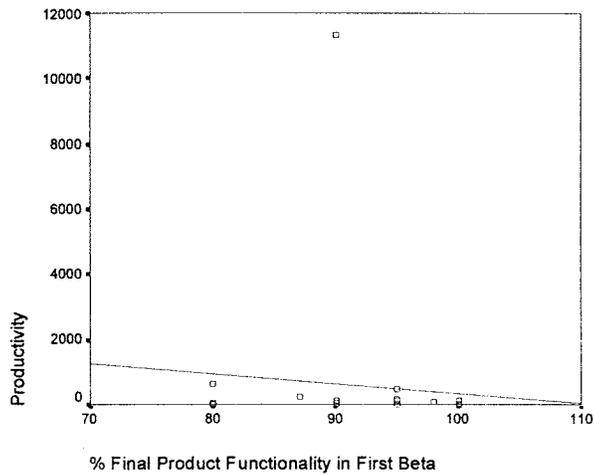
Figure 3-15 - % final product functionality implemented at first beta vs. productivity (all projects)

Correlation between % final product functionality implemented at first beta and productivity: - **0.260.** The correlation between these two variables is statistically not significant. As can be seen from the sample data the first beta was released late in the project, with respect to functionality and the project teams may not be in a position to incorporate any significant customer feedback resulting in no or very little rework. As the team does not spend any time doing rework, the team can incorporate remaining functionality resulting in new code. Since productivity is being measured as a function of total lines of code, the above mentioned scenario might result in higher productivity.

3.2.7 Sensitivity Analysis:

In our data analysis to evaluate hypothesis 1 through hypothesis 5, there are some instances where some outlier cases were observed. In order to study the effect of these outlier cases on the analysis, sensitivity analysis was performed. The correlation analysis was performed again with the data after filtering out the outlier case(s). The following correlation table (Table 3-2) contains the analysis without the outlier case(s). The scatter graph (Figure 3-16) following the table is provided for the variables with statistically significant correlation.

41

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% final product functionality in first prototype	% final product functionality in first system integration	% final product functionality in first beta
% Original Features implemented	Correlation Coefficient	1.000	.275	-.250	-.255	.301	-.071	.194	.373	.639**
	Sig. (2-tailed)	.	.255	.288	.277	.197	.767	.425	.116	.003
	N	20	19	20	20	20	20	19	19	19
Bugginess (per mil LOC)	Correlation Coefficient	.275	1.000	-.032	.039	.278	.245	.664**	.198	.357
	Sig. (2-tailed)	.255	.	.898	.875	.249	.311	.003	.432	.146
	N	19	19	19	19	19	19	18	18	18
% Schedule Estimation Error	Correlation Coefficient	-.250	-.032	1.000	.226	-.190	-.060	-.055	.022	-.493*
	Sig. (2-tailed)	.288	.898	.	.287	.423	.802	.809	.923	.017
	N	20	19	24	24	20	20	22	22	23
Productivity	Correlation Coefficient	-.255	.039	.226	1.000	-.496*	-.071	-.202	-.124	-.234
	Sig. (2-tailed)	.277	.875	.287	.	.026	.765	.367	.583	.283
	N	20	19	24	24	20	20	22	22	23
Schedule and Budget Perf. perception rating	Correlation Coefficient	.301	.278	-.190	-.496*	1.000	.072	.247	.112	.464*
	Sig. (2-tailed)	.197	.249	.423	.026	.	.762	.308	.647	.046
	N	20	19	20	20	20	20	19	19	19
Customer satisfaction perception rating	Correlation Coefficient	-.071	.245	-.060	-.071	.072	1.000	.255	-.545*	-.222
	Sig. (2-tailed)	.767	.311	.802	.765	.762	.	.292	.016	.361
	N	20	19	20	20	20	20	19	19	19
% final product functionality in first prototype	Correlation Coefficient	.194	.664**	-.055	-.202	.247	.255	1.000	.518*	.491*
	Sig. (2-tailed)	.425	.003	.809	.367	.308	.292	.	.013	.020
	N	19	18	22	22	19	19	22	22	22
% final product functionality in first system integration	Correlation Coefficient	.373	.198	.022	-.124	.112	-.545*	.518*	1.000	.521*
	Sig. (2-tailed)	.116	.432	.923	.583	.647	.016	.013	.	.013
	N	19	18	22	22	19	19	22	22	22
% final product functionality in first beta	Correlation Coefficient	.639**	.357	-.493*	-.234	.464*	-.222	.491*	.521*	1.000
	Sig. (2-tailed)	.003	.146	.017	.283	.046	.361	.020	.013	.
	N	19	18	23	23	19	19	22	22	23

**. Correlation is significant at the .01 level (2-tailed).

*. Correlation is significant at the .05 level (2-tailed).

**Table 3-2 - Market and Technical Feedback Correlation Table – without the outlier in productivity**
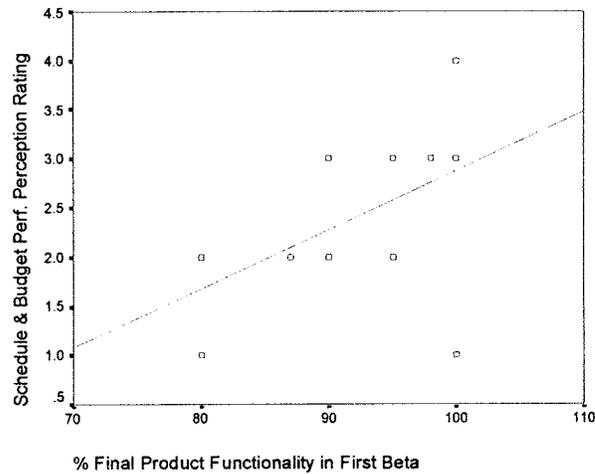
Figure 3-16 - % final product functionality implemented at first beta vs. schedule & budget performance perception rating (without outlier in productivity)

Correlation between % final product functionality implemented at first beta and schedule & budget performance perception rating: **0.464.** The Correlation between the two variables is statistically significant at the 0.05 level (two-tailed). As has been mentioned earlier, even without the outlier if the % final product functionality implemented at first beta is high then that results in smaller schedule estimation error or better schedule performance perception rating.

3.2.8 Observations based on the data analysis for market and technical feedback:

Hypothesis Number	Summary of hypothesis	Observations
1	Obtaining early market and technical feedback results in increased feature evolution and customer satisfaction.	Increased feature evolution is statistically significant with functionality in first beta. Early technical feedback and customer satisfaction perception rating are significantly correlated.
2	Incorporating more market and technical feedback increases schedule estimation error.	Schedule estimation error is statistically significant with % of functionality in final product at first beta.
3	Obtaining early technical feedback reduces bugginess	The relation between these two variables is statistically not significant.
4	Bugginess increases as project teams implement early market feedback	Bugginess is statistically significant with functionality in first prototype.
5	Productivity increases due to reduced rework as project teams obtain early feedback	The relation between these two variables is statistically not significant.

Table 3-3 - Summary of hypotheses on impact of market and technical feedback

- The analysis validates our hypothesis that as the projects obtain feedback early in the project (with respect to functionality), there is more feature evolution. As can be seen from the analysis, as the % final product functionality at key milestones (first prototype,

system integration and first beta) increases, so does the % of original features implemented in the final product, making it more inflexible and reducing the ability to incorporate market and technical feedback

- For customer satisfaction perception rating, it can be seen that our hypothesis is validated. As can be seen from the analysis, as the % final product functionality in first system integration increases, the customer satisfaction perception rating decreases. This could be because the customer has less influence in the features that will be available in the final product.

- From the analysis we observe that there is very little correlation of % functionality at first prototype and system integration with schedule estimation error. Our hypothesis holds true for % functionality in first beta, as the functionality is increasing, schedule estimation error is decreasing.

3.3 Impact of Separate Development Sub-Cycles

3.3.1 Hypothesis 6:

Dividing the development phase of the project into separate development sub-cycles that built and tested a subset of the final product functionality, allows the team to be:

- More flexible (increased feature evolution)

- Deliver a high quality product

- Improves the productivity of the team (*section 14.12 The productivity of evolutionary delivery*, Principles Of Software Engineering Management, Tom Gilb, 1988).

3.3.2 Hypothesis 7:

A high level architectural specification (without implementation details) provides for more flexible product development while detailed architectural specification (lot of rules) tend to create a rigid environment (*chapter 4, page 244* - Microsoft Secrets, Michael Cusumano and Richard Selby, 1998).

The process variables used to evaluate the impact of separate development sub-cycles are:

- Number of sub-cycles

- Architecture effort

The outcome variables used to evaluate the impact of separate development sub-cycles are:

- % Original features implemented in the final product

- Bugginess

- Productivity

3.3.3 Data Analysis to evaluate the impact of separate development sub-cycles

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Number Of Subcycles	Architectural Effort
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.165	.475*
	Sig. (2-tailed)		.204	.327	.212	.233	.834	.476	.029
	N	21	20	21	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.119	.068
	Sig. (2-tailed)	.204		.927	.816	.348	.191	.618	.775
	N	20	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.028	.233
	Sig. (2-tailed)	.327	.927		.361	.390	.822	.895	.263
	N	21	20	25	25	21	21	25	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	.334	-.227
	Sig. (2-tailed)	.212	.816	.361		.085	.561	.103	.276
	N	21	20	25	25	21	21	25	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.094	.199
	Sig. (2-tailed)	.233	.348	.390	.085		.891	.685	.388
	N	21	20	21	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.047	.033
	Sig. (2-tailed)	.834	.191	.822	.561	.891		.839	.889
	N	21	20	21	21	21	21	21	21
Number Of Subcycles	Correlation Coefficient	-.165	-.119	-.028	.334	-.094	.047	1.000	-.408*
	Sig. (2-tailed)	.476	.618	.895	.103	.685	.839		.043
	N	21	20	25	25	21	21	25	25
Architectural Effort	Correlation Coefficient	.475*	.068	.233	-.227	.199	.033	-.408*	1.000
	Sig. (2-tailed)	.029	.775	.263	.276	.388	.889	.043	
	N	21	20	25	25	21	21	25	25

*. Correlation is significant at the .05 level (2-tailed).

Table 3-4 - Correlation Table For Separate Development Sub-Cycles
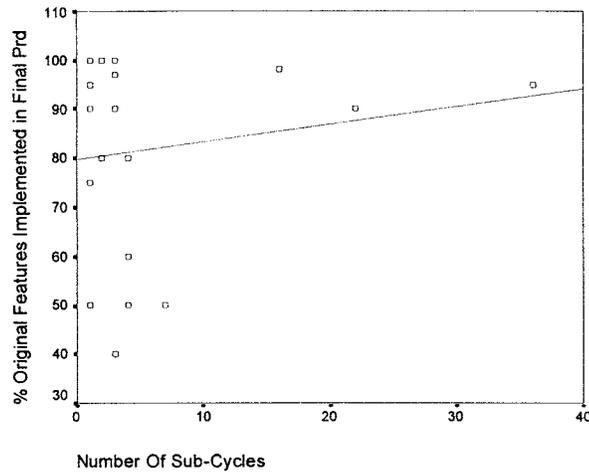
Figure 3-17 - Number of Sub-cycles Vs. % original features implemented in the final product (all projects)

Correlation between number of sub-cycles and % original features implemented in the final product: **-0.165.** The correlation between these two variables is statistically not significant. % Original features implemented in the final product could mean that either the project team has not implemented all the functionality due to schedule constraints or the project team may have incorporated feature changes as a result of customer and technical feedback. Further information about the type and/or amount of changes incorporated in the product would be helpful in understanding this relation better.
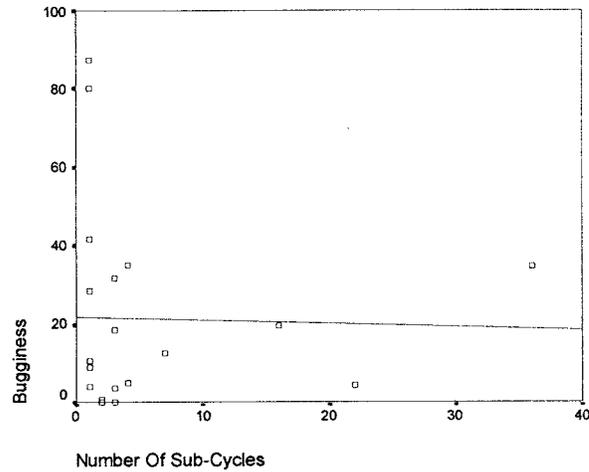
Figure 3-18 - Number of sub-cycles vs. bugginess (all projects)

Correlation between number of sub-cycles and bugginess: **-0.119.** The correlation between these two variables is statistically not significant. One of the possible reasons that we see a negative correlation between these two variables could be that as the project teams divide their project development cycle into more sub-cycles, it provides them with the opportunity to discover bugs and correct them as they test the new code at the end of each sub-cycle. This could be a reason why the end user is encountering low number of bugs after the final product release.
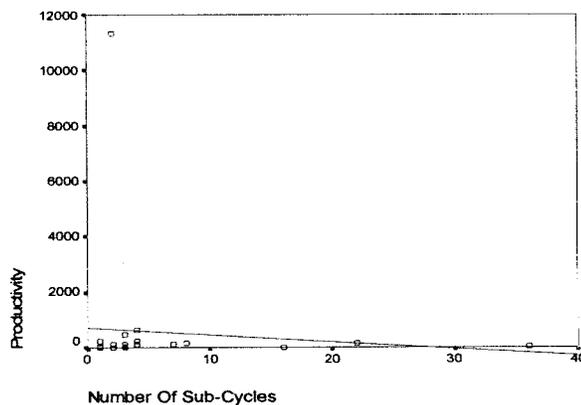
Figure 3-19 - Number of sub-cycles vs. productivity (all projects)

Correlation between number of sub-cycles and productivity: **0.334.** The correlation between these two variables is statistically not significant. As can be seen from the scatter plot in figure 3-19, there is an outlier case. Sensitivity analysis was performed by filtering out the outlier case

48

and it is seen that the correlation becomes statistically significant, as shown later in this section (figure 3-23).
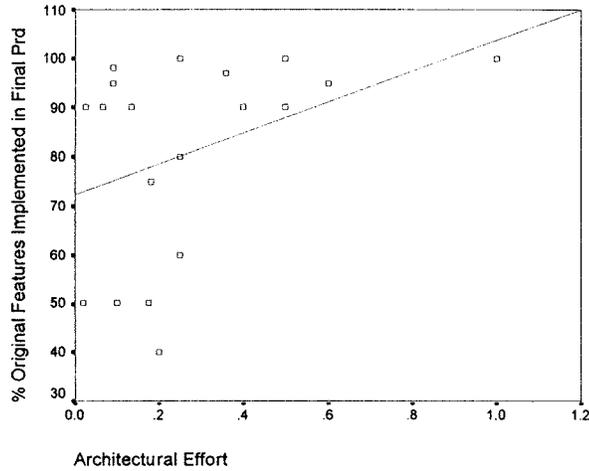
Figure 3-20 – Architecture Effort vs. % original features implemented in the final product (all projects)

Correlation between architecture effort and % original features implemented in the final product: **0.475.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). One of the reasons for this correlation could be that as the project team spends more effort in creating detailed architecture based on the original specifications, thus creating more rules on how the product will be implemented. This potentially leaves little room for the project team to incorporate customer feedback; therefore the team would end up implementing more of the original features.
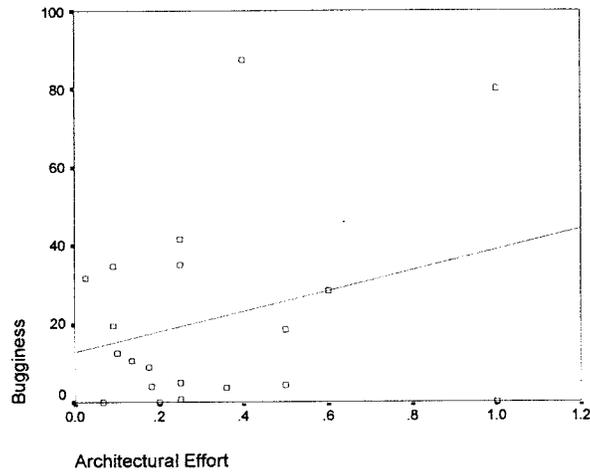
Figure 3-21 - Architecture Effort Vs. bugginess (all projects)

Correlation between architecture effort and bugginess: **0.068.** The correlation between these two variables is statistically not significant. The positive correlation could be because of rework by the project team after spending a lot off effort on architecture, which may create an environment of inflexibility as far as incorporating market feedback into the product. By trying to rework the implementation based on market feedback, with lot off implementation rules (due to detailed architecture) could result in increased bugs.
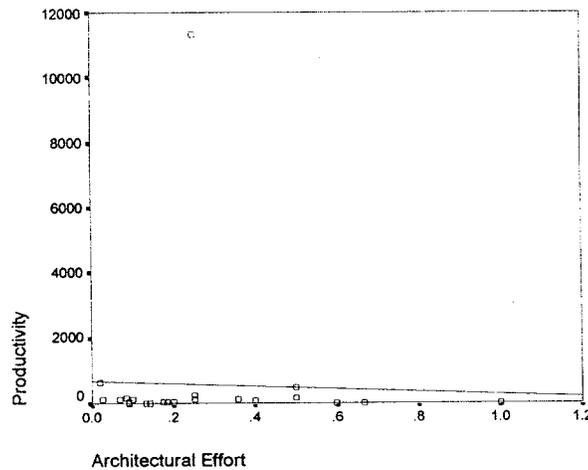
Figure 3-22 - Architecture Effort Vs. productivity (all projects)

Correlation between architecture effort and productivity: **-0.227.** The correlation between these two variables is statistically not significant. One of the reasons that could be influencing this relation is the fact that if the project team spends lot off effort on architecture effort then they could have designed clean interfaces between modules reducing the need for lot off new code required for the modules to interact with each other. This has a direct bearing on the productivity since it is measured in terms of lines of code.

3.3.4 Sensitivity Analysis:

In our data analysis to evaluate hypothesis 6 and hypothesis 7, there are some instances where some outlier cases were observed for number of sub-cycles, productivity and architectural effort variables. In order to study the effect of these outlier cases on the analysis, sensitivity analysis was performed. The correlation analysis was performed again with the data after filtering out the outlier case(s). The following correlation table (Table 3-4) contains the analysis without the outlier case(s). The scatter graphs (Figure 3-23 and Figure 3-24) following the table are provided for the variables with statistically significant correlation.

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Number Of Subcycles	Architectural Effort
% Original Features implemented	Correlation Coefficient	1.000	.340	-.234	-.002	.310	-.182	-.152	.390
	Sig. (2-tailed)	.	.198	.365	.992	.226	.486	.561	.122
	N	17	16	17	17	17	17	17	17
Bugginess (per mil LOC)	Correlation Coefficient	.340	1.000	.038	.057	.133	.000	-.175	.161
	Sig. (2-tailed)	.198	.	.888	.833	.622	1.000	.516	.551
	N	16	16	16	16	16	16	16	16
% Schedule Estimation Error	Correlation Coefficient	-.234	.038	1.000	.250	-.164	-.067	-.200	.444*
	Sig. (2-tailed)	.365	.888	.	.275	.529	.799	.385	.044
	N	17	16	21	21	17	17	21	21
Productivity	Correlation Coefficient	-.002	.057	.250	1.000	-.478	-.033	.448*	-.141
	Sig. (2-tailed)	.992	.833	.275	.	.052	.900	.041	.543
	N	17	16	21	21	17	17	21	21
Schedule and Budget Perf. perception rating	Correlation Coefficient	.310	.133	-.164	-.478	1.000	-.105	-.071	.260
	Sig. (2-tailed)	.226	.622	.529	.052	.	.688	.787	.313
	N	17	16	17	17	17	17	17	17
Customer satisfaction perception rating	Correlation Coefficient	-.182	.000	-.067	-.033	-.105	1.000	-.051	.154
	Sig. (2-tailed)	.486	1.000	.799	.900	.688	.	.845	.554
	N	17	16	17	17	17	17	17	17
Number Of Subcycles	Correlation Coefficient	-.152	-.175	-.200	.448*	-.071	-.051	1.000	-.287
	Sig. (2-tailed)	.561	.516	.385	.041	.787	.845	.	.206
	N	17	16	21	21	17	17	21	21
Architectural Effort	Correlation Coefficient	.390	.161	.444*	-.141	.260	.154	-.287	1.000
	Sig. (2-tailed)	.122	.551	.044	.543	.313	.554	.206	.
	N	17	16	21	21	17	17	21	21

*. Correlation is significant at the .05 level (2-tailed).

Table 3-5- Correlation Table For Separate Development Sub-Cycles – without the outliers for number of sub-cycles, productivity and architectural effort
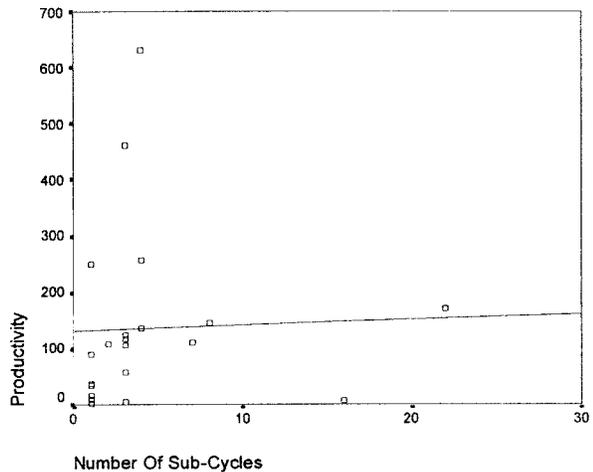
Figure 3-23 - Number of sub-cycles vs. productivity - without the outliers for number of sub-cycles, productivity and architectural effort

Correlation between number of sub-cycles and productivity: **0.448.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). This essentially validates our hypothesis.
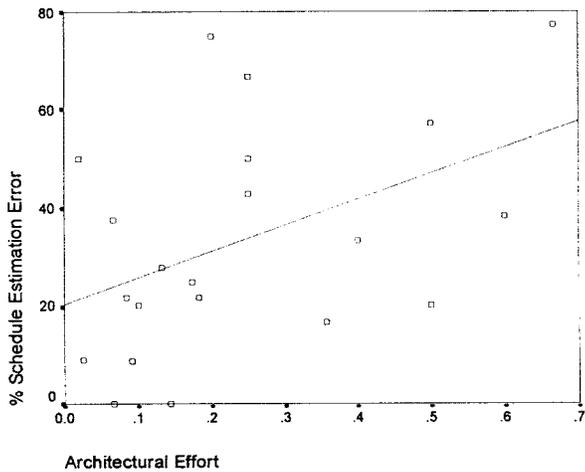
Figure 3-24 - Architecture Effort Vs. % schedule estimation error - without the outliers for number of sub-cycles, productivity and architectural effort

Correlation between architecture effort and % schedule estimation error: **0.444.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). This seems to be an interesting correlation and one of the possible reasons is that if the project team spends too much effort in creating a detailed architecture thus creating lot off rules on how to implement the product. By creating a detailed architecture instead of high-level architecture the product team may have to rework the detailed architecture and the product implementation as they obtain customer feedback. This in turn could further delay the product launch thus increasing the schedule estimation error.

3.3.5 Observations based on the data analysis for separate development sub-cycles:

Hypothesis Number	Summary of hypothesis	Observations
6	Dividing the development phase into sub-cycles allows the team to be more flexible, deliver high quality product and improve the productivity.	The correlation between sub-cycles and various outcome variables is statistically not significant.
7	High-level architecture specification provides for more flexible product development measured in terms of feature evolution.	The relation between architectural effort and % of features implemented in final product is statistically significant.

Table 3-6 Summary of hypotheses on the impact of separate development sub-cycles

- From the analysis we observe that as the architecture effort is increasing so is the % of original features implemented in the final product. This validates our hypothesis.

3.4 Flexibility in Project Activities

3.4.1 Hypothesis 8:

Evolutionary development also allows great flexibility in project activities. This allows the project team to work in uncertain environment with requirements changes, design changes and consequently implement new features (add new code) very late in the product development cycle. As we mentioned earlier, flexible projects have high feature evolution.

The process variables that are used to measure the flexibility in project activities are:

- % Elapsed project duration till the last major requirement changes

- % Elapsed project duration till the last major functional design changes

- % Elapsed project duration till the last major code addition for new features (excluding any bug fixes)

Some of the outcome variables that we will study in relation to the process variables are:

- % Original features implemented in the final product

- Bugginess

- Productivity

3.4.2 Data analysis to evaluate flexibility in project activities

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% Elapsed prj duration till last major req change	% Elapsed prj duration till last major func spec change	% Elapsed prj duration till last major code addtn.
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.547*	-.439	-.254
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.012	.060	.266
	N	21	20	21	21	21	21	20	19	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.068	.175	.157
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.783	.487	.510
	N	20	20	20	20	20	20	19	18	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	.127	-.038	.322
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.553	.864	.117
	N	21	20	25	25	21	21	24	23	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.086	.017	.062
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.690	.937	.768
	N	21	20	25	25	21	21	24	23	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.162	-.040	.099
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.495	.872	.668
	N	21	20	21	21	21	21	20	19	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.373	.113	.259
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.105	.646	.258
	N	21	20	21	21	21	21	20	19	21
% Elapsed prj duration till last major req change	Correlation Coefficient	-.547*	-.068	.127	-.086	-.162	.373	1.000	.596**	.135
	Sig. (2-tailed)	.012	.783	.553	.690	.495	.105	.	.003	.530
	N	20	19	24	24	20	20	24	23	24
% Elapsed prj duration till last major func spec change	Correlation Coefficient	-.439	.175	-.038	.017	-.040	.113	.596**	1.000	.317
	Sig. (2-tailed)	.060	.487	.864	.937	.872	.646	.003	.	.141
	N	19	18	23	23	19	19	23	23	23
% Elapsed prj duration till last major code addtn.	Correlation Coefficient	-.254	.157	.322	.062	.099	.259	.135	.317	1.000
	Sig. (2-tailed)	.266	.510	.117	.768	.668	.258	.530	.141	.
	N	21	20	25	25	21	21	24	23	25

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-7 – Correlation Table For Variables to Evaluate Flexibility in Project Activities
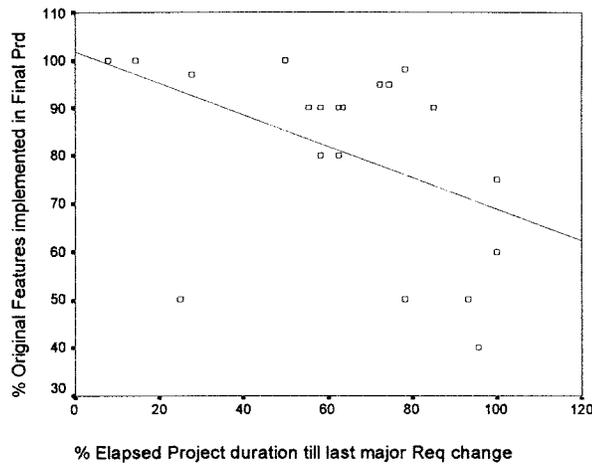
Figure 3-25 - % elapsed time from project start till last major requirements change vs. % original features implemented in the final product (all projects)

Correlation between % elapsed time from project start till last major req. change and % original features implemented in the final product: **-0.547.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). One of the obvious reasons for this is that as the product requirements change late into the project, either due to technological reasons or customer feedback, there will be significant feature evolution.
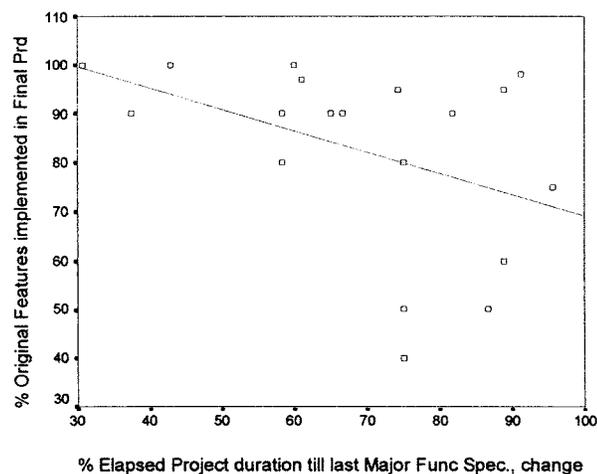
Figure 3-26 - % elapsed time from project start till last major functional spec change vs. % original features implemented in the final product (all projects)

Correlation between % elapsed time from project start till last major functional specification change and % original features implemented in the final product: **-0.439.** The correlation

56

between these two variables is statistically not significant. Just as in the requirements case, if there are design changes late into the project that could result in changes to the original list of features that the team started the project with. Alternately, the design could change without significantly impacting the original feature list and this may explain why the correlation is statistically not significant yet the negative correlation tells that there will feature evolution as the team changes the product design late into the project.
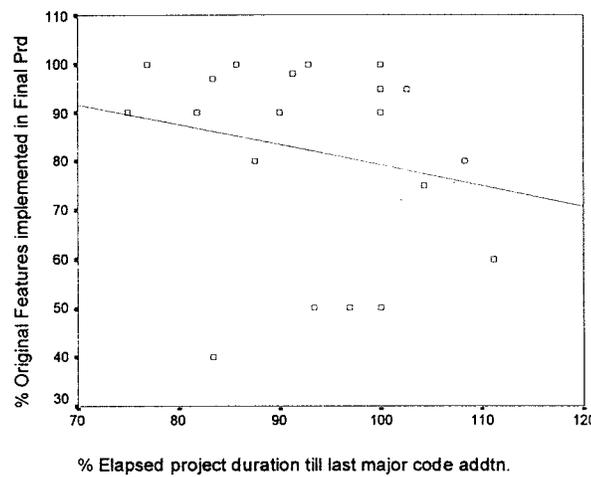
**Figure 3-27 - % elapsed time from project start till last major code addition
vs. % original features implemented in the final product (all projects)**

Correlations between % elapsed time from project start till last major code addition and % original features implemented in the final product: **-0.254.** The correlation between these two variables is statistically not significant. Just as in the requirements and functional specification cases, if there are major code additions late into the project that could result in changes to the original list of features that the team started the project with. Alternately, major code could be added without significantly impacting the original feature list i.e., the features may not be changing but the team may be delayed and just adding the code for the features on the original list and this may explain why the correlation is statistically not significant yet the negative correlation tells that there will be either feature evolution and/or reduction in the features of the product due to schedule delays as the team adds new code late into the project.
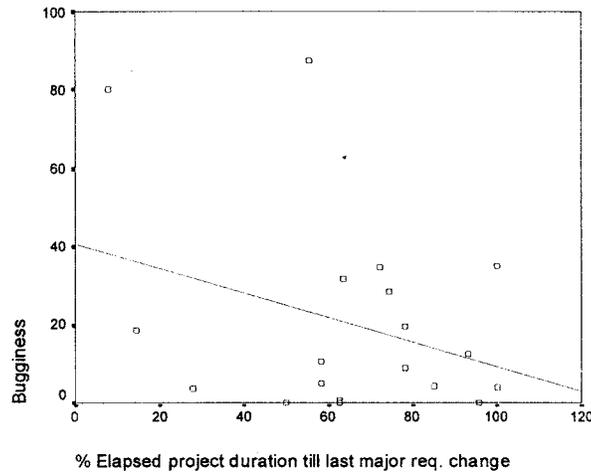
Figure 3-28 - % elapsed time from project start till last major requirements change vs. bugginess (all projects)

Correlation between % elapsed time from project start till last major req. change and bugginess: -**0.068.** The correlation between these two variables is statistically not significant. The ability of the product team to handle changes late into the project depends on how the team has architected the various modules within the product. If it is a clean architecture where the changes are localized to a single module then the product team may achieve good product quality (low number of bugs) even with all the late changes in requirements on the other hand if the architecture is not clean then all the late changes in requirements may decrease the product quality (high number of bugs).

100

80

60

40

Bugginess 20

0

30   40   50   60   70   80   90   100

% Elapsed project duration till last major func spec change
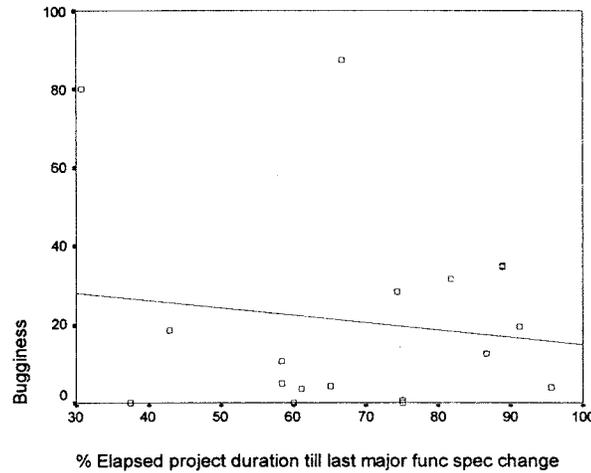
Figure 3-29 - % elapsed time from project start till last major functional spec change vs. bugginess (all projects)

Correlation between % elapsed time from project start till last major functional specification change and bugginess: **0.175.** The correlation between these two variables is statistically not significant. The ability of the product team to handle changes late into the project depends on how the team has architected the various modules within the product. If it is a clean architecture where the functional specification changes are localized to a few modules then the product team may achieve good product quality (low number of bugs) even with all the late changes in the design on the other hand if the architecture is not clean then all the late changes in design will decrease the product quality (high number of bugs).
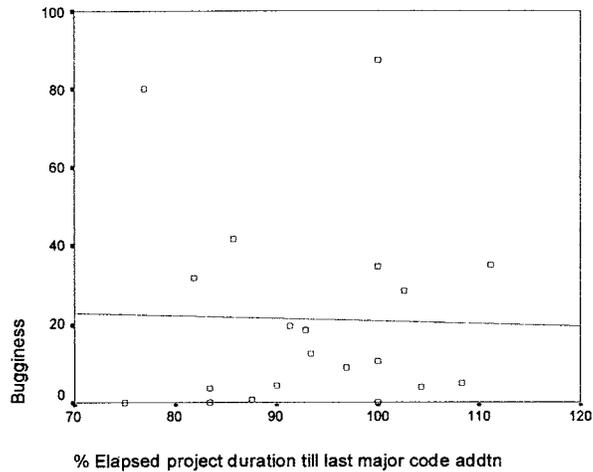
Figure 3-30 - % elapsed time from project start till last major code addition vs. bugginess (all projects)

Correlations between % elapsed time from project start till last major code addition and bugginess: **0.157.** The correlation between these two variables is statistically not significant. As mentioned earlier the ability to handle changes depends on the product architecture. Even with clean architecture any time the team adds or modifies code there is always an opportunity to introduce bugs. This is what the positive correlation between these two variables, is telling us.
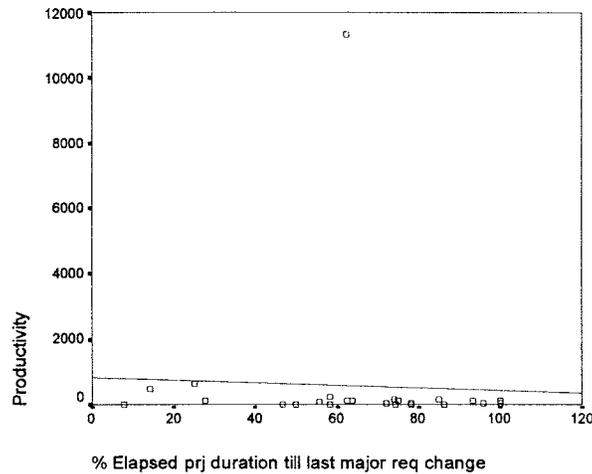
Figure 3-31 - % elapsed time from project start till last major requirements change vs. productivity (all projects)

Correlation between % elapsed time from project start till last major req. change and productivity: **-0.086.** The correlation between these two variables is statistically not significant. The two variables have very little correlation between them. Productivity is measured as a function of uncommented lines of code. The total lines of code developed can be impacted by variety of factors when the product requirements are changing late into the project. Some of the factors could be that the team may find an efficient algorithm to implement functionality based on the new information, which may reduce the lines of code, required, less feature implementation due to schedule constraints etc.

61
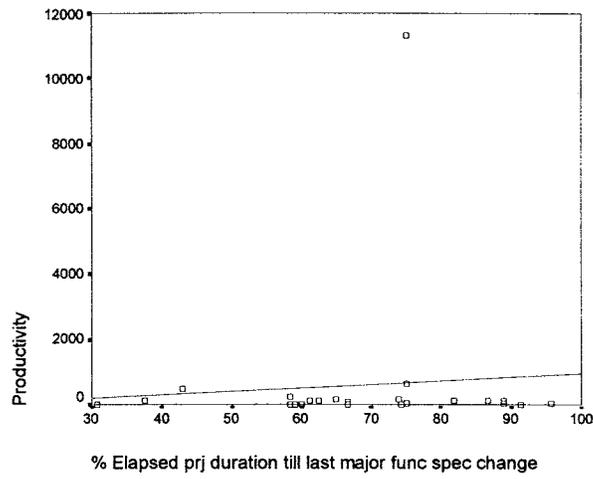
Figure 3-32 - % elapsed time from project start till last major functional spec change vs. productivity (all projects)

Correlation between % elapsed time from project start till last major functional specification change and productivity: **0.017.** The correlation between these two variables is statistically not significant. The two variables are barely correlated and the reasons for lack of correlation are similar to the reasons for lack of correlation between % elapsed time for project start till last major requirements change and productivity.
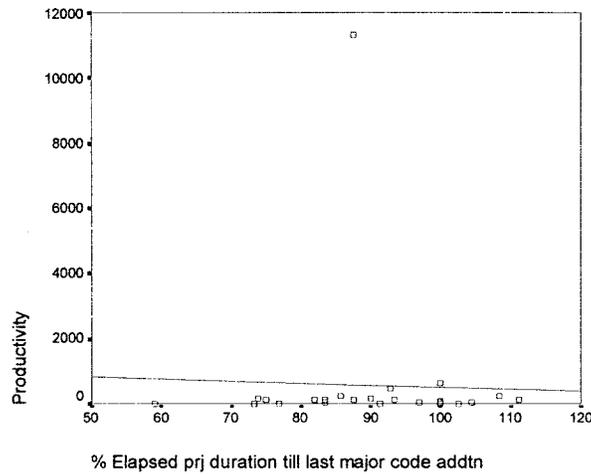
Figure 3-33 - % elapsed time from project start till last major code addition vs. productivity (all projects)

Correlations between % elapsed time from project start till last major code addition and productivity: **0.062.** Correlation between these two variables is statistically not significant. The reasons for lack of correlation are the same as in the previous two cases.

3.4.3 Sensitivity Analysis:

In our data analysis to evaluate hypothesis 8, there are some instances where some outlier cases were observed for productivity variables. In order to study the effect of these outlier cases on the analysis, sensitivity analysis was performed. The correlation analysis was performed again with the data after filtering out the outlier case(s). The following correlation table (Table 3-6) contains the analysis without the outlier case(s). There are no scatter graphs following the table because there were no correlations between process and outcome variables, which were statistically significant.

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% Elapsed prj duration till last major req change	% Elapsed prj duration till last major func spec change	% Elapsed prj duration till last major code addtn.
% Original Features implemented	Correlation Coefficient	1.000	.275	-.250	-.255	.301	-.071	-.565*	-.419	-.272
	Sig. (2-tailed)	.	.255	.288	.277	.197	.767	.012	.083	.246
	N	20	19	20	20	20	20	19	18	20
Bugginess (per mil LOC)	Correlation Coefficient	.275	1.000	-.032	.039	.278	.245	-.106	.192	.114
	Sig. (2-tailed)	.255	.	.898	.875	.249	.311	.677	.461	.642
	N	19	19	19	19	19	19	18	17	19
% Schedule Estimation Error	Correlation Coefficient	-.250	-.032	1.000	.226	-.190	-.060	.151	-.022	.320
	Sig. (2-tailed)	.288	.898	.	.287	.423	.802	.491	.923	.128
	N	20	19	24	24	20	20	23	22	24
Productivity	Correlation Coefficient	-.255	.039	.226	1.000	-.496*	-.071	-.071	-.035	.095
	Sig. (2-tailed)	.277	.875	.287	.	.026	.765	.747	.877	.658
	N	20	19	24	24	20	20	23	22	24
Schedule and Budget Perf. perception rating	Correlation Coefficient	.301	.278	-.190	-.496*	1.000	.072	-.156	-.046	.128
	Sig. (2-tailed)	.197	.249	.423	.026	.	.762	.525	.855	.590
	N	20	19	20	20	20	20	19	18	20
Customer satisfaction perception rating	Correlation Coefficient	-.071	.245	-.060	-.071	.072	1.000	.373	.123	.241
	Sig. (2-tailed)	.767	.311	.802	.765	.762	.	.116	.626	.306
	N	20	19	20	20	20	20	19	18	20
% Elapsed prj duration till last major req change	Correlation Coefficient	-.565*	-.106	.151	-.071	-.156	.373	1.000	.602**	.141
	Sig. (2-tailed)	.012	.677	.491	.747	.525	.116	.	.003	.521
	N	19	18	23	23	19	19	23	22	23
% Elapsed prj duration till last major func spec change	Correlation Coefficient	-.419	.192	-.022	-.035	-.046	.123	.602**	1.000	.329
	Sig. (2-tailed)	.083	.461	.923	.877	.855	.626	.003	.	.135
	N	18	17	22	22	18	18	22	22	22
% Elapsed prj duration till last major code addtn.	Correlation Coefficient	-.272	.114	.320	.095	.128	.241	.141	.329	1.000
	Sig. (2-tailed)	.246	.642	.128	.658	.590	.306	.521	.135	.
	N	20	19	24	24	20	20	23	22	24

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-8 - Correlation Table For Variables to Evaluate Flexibility in Project Activities – without the outlier for Productivity

3.4.4 Observations based on the data analysis for variables to evaluate flexibility in project activities:

Hypothesis Number	Summary of hypothesis	Observations
8	Evolutionary development allows flexibility in product development allowing the project team to make requirements, functional changes and add code for new features late into the project.	The relation between flexibility in requirements change and % of original features in final product is statistically significant.

Table 3-9 Summary of hypothesis on flexibility in project activities

- The first hypothesis in this area is validated by the analysis. From the analysis we observe that as the % of elapsed time (for requirements change, functional design change, and code addition) increases, the % of original features implemented in the final product is decreasing significantly. This essentially is because more feedback is being incorporated.

3.5 Impact of Code Reuse

3.5.1 Hypothesis 9:

More code reuse results in:

- Reduced product bugginess

- Reduced feature evolution due to potential inflexibility introduced due to existing code (I.e. code being reused)

- Reduces schedule estimation error.

The process variable that is used to track the code reuse is:

- % New code developed by the team

Some of the outcome variables that we will study to understand the impact of code reuse are:

- Bugginess

- % Of original features implemented in the final product

- % Schedule estimation error

3.5.2 Data analysis to evaluate impact of code reuse

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% Code Reuse
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	.245
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.285
	N	21	20	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	.168
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.480
	N	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	.034
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.870
	N	21	20	25	25	21	21	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.069
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.743
	N	21	20	25	25	21	21	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.251
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.272
	N	21	20	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.096
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.678
	N	21	20	21	21	21	21	21
% Code Reuse	Correlation Coefficient	.245	.168	.034	-.069	-.251	.096	1.000
	Sig. (2-tailed)	.285	.480	.870	.743	.272	.678	.
	N	21	20	25	25	21	21	25

Table 3-10 – Correlation Table For Code Reuse Measures
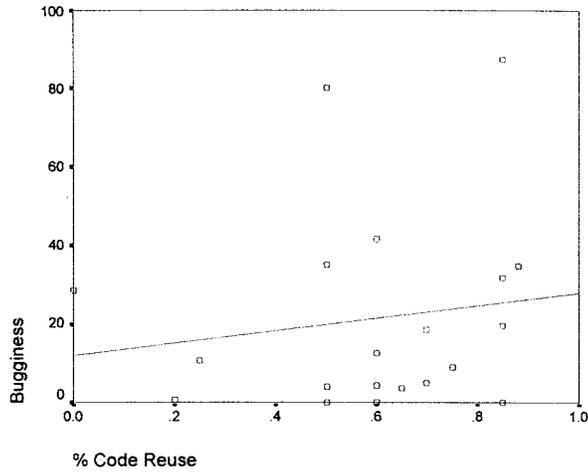
Figure 3-34 - % code reuse vs. Bugginess (all projects)

Correlation between % code reuse and Bugginess: **0.168.** Correlation between these two variables is statistically not significant. One of the reasons for the positive correlation could be because the team members may not be completely familiar with code being reused or the assumptions made while developing the previous code resulting in increased bugs reported by the customer.
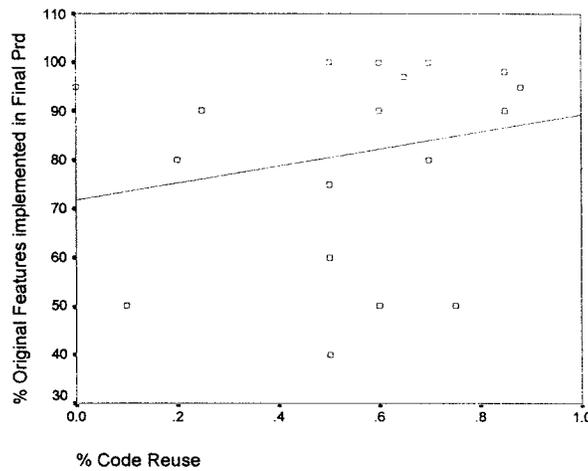
Figure 3-35 - % code reuse vs. % original features implemented in the final product (all projects)

Correlation between % code reuse and % original features implemented in the final product: **0.245.** The correlation between these two variables is statistically not significant. There is a positive correlation between these two variables indicating that more % of original features would be implemented as the % code reuse increases. When a project team reuses code, they essentially are reusing functionality that has already been implemented and this leaves little flexibility for the team to incorporate customer feedback without changing the code being reused.
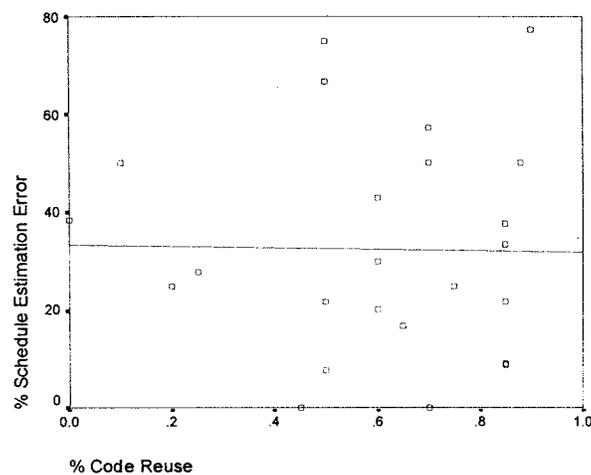
Figure 3-36 - % code reuse vs. % schedule estimation error (all projects)

Correlation between % code reuse and % schedule estimation error: **0.034.** The correlation between these two variables is statistically not significant. Typically one would expect with more % code reuse, the schedule estimation error would be less since part of the code is already implemented. On the flip side, as mentioned earlier, if the project team members are not familiar with the code being reused or the assumptions made while developing the previous code, then the project team may face quality issues which might impact the schedules and increase the %el schedule estimation error.

Observations based on the data analysis for impact of code reuse:

Hypothesis Number	Summary of hypothesis	Observations
9	More code reuse results in reduced bugginess, reduced feature evolution and reduced schedule estimation error.	The correlation, between the process and outcome variables being used to measure the impact of code reuse, is statistically not significant.

Table 3-11 Summary of hypothesis on impact of code reuse

There are no significant observations made since the data analysis did not yield any statistically significant correlation between process and outcome variables.

3.6 Impact of Frequent Synchronization

3.6.1 Hypothesis 10:

"Doing daily (frequent) builds gives rapid feedback to the project team about how the product is progressing. This makes sure that the basic functions of the evolving product are working correctly most of the time" (*chapter 5, pages 268-269* - Microsoft Secrets, Michael Cusumano and Richard Selby, 1998). This results in:

- Higher productivity

- Higher customer satisfaction

- Bugginess could be lower. An alternate hypothesis is that frequent synchronization will eliminate any system integration surprises but does not necessarily mean that it reduces bugginess in individual components.

3.6.2 Hypothesis 11:

"Knowing where the team is, with respect to the project, makes the overall product development process more visible and predictable" (*chapter 5, page 276* – Microsoft Secrets, Michael Cusumano and Richard Selby, 1998). This results in:

- Reduced schedule estimation error

The process variable that was used to measure the frequency of synchronization is:

- Build frequency (daily or other – weekly, biweekly, monthly etc.)

- •Some of the outcome variables that we will study to understand the impact of frequent synchronization approach:

- Productivity

- Bugginess

- % Original features implemented

- % Schedule estimation error

3.6.3 Data analysis to evaluate impact of frequent synchronization

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Build Frequency (Daily - 1; Other - 0)
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.127
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.584
	N	21	20	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	.000
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	1.000
	N	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.142
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.499
	N	21	20	25	25	21	21	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	.102
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.628
	N	21	20	25	25	21	21	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.135
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.559
	N	21	20	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.194
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.399
	N	21	20	21	21	21	21	21
Build Frequency (Daily - 1; Other - 0)	Correlation Coefficient	-.127	.000	-.142	.102	-.135	.194	1.000
	Sig. (2-tailed)	.584	1.000	.499	.628	.559	.399	.
	N	21	20	25	25	21	21	25

Table 3-12 – Correlation Table For Frequent Synchronization Measure
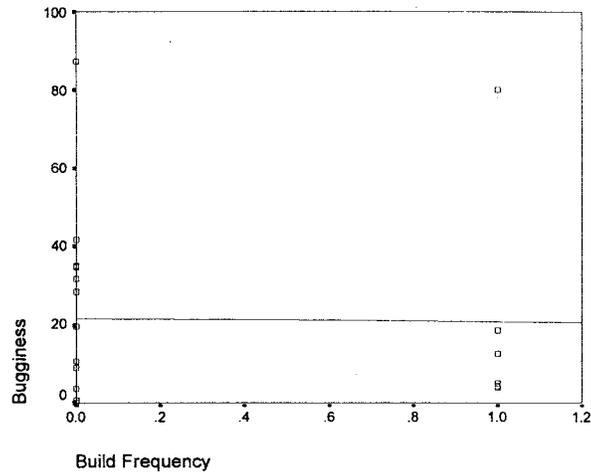
Figure 3-37 - Build Frequency Vs. bugginess (all projects)

Correlations between build frequency and bugginess: **0.** The correlation between these two variables is statistically not significant. The reason that these two variables are not correlated is probably because bugs as reported by the customer may not be the right parameter to evaluate. The parameter that should be used is the number of bugs found during the sub-cycles. This piece of data has not been captured in the questionnaire and should be obtained in future research.
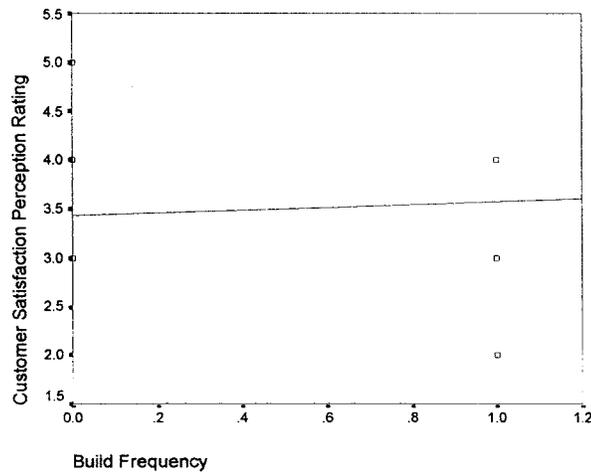
Figure 3-38 - Build Frequency Vs. customer satisfaction perception rating (all projects)

Correlations between build frequency and customer satisfaction perception rating: **0.194.** The correlation between these two variables is statistically not significant. Frequent synchronization might have provided the project team with opportunities to resolve any issues found during the product development and synchronization process resulting in a higher customer satisfaction perception rating.
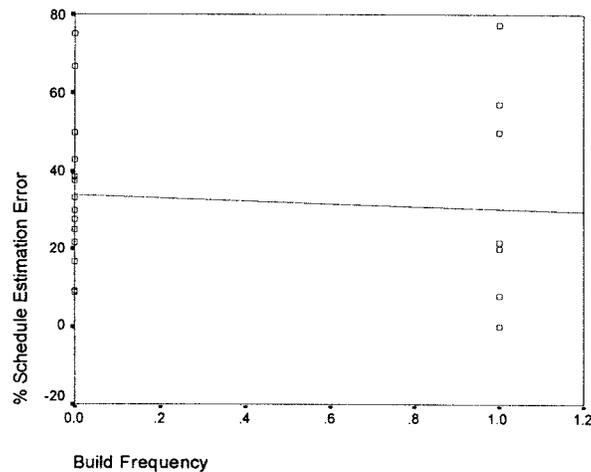
Figure 3-39 - Build Frequency vs. % schedule estimation error (all projects)

Correlations between build frequency and % schedule estimation error: **-0.142.** The correlation between these two variables is statistically not significant. The reason for negative correlation between these two variables could be because frequent synchronization might have provided the project team with opportunities to resolve any issues found during the product development and synchronization process early enough in the project thus reducing schedule estimation error.
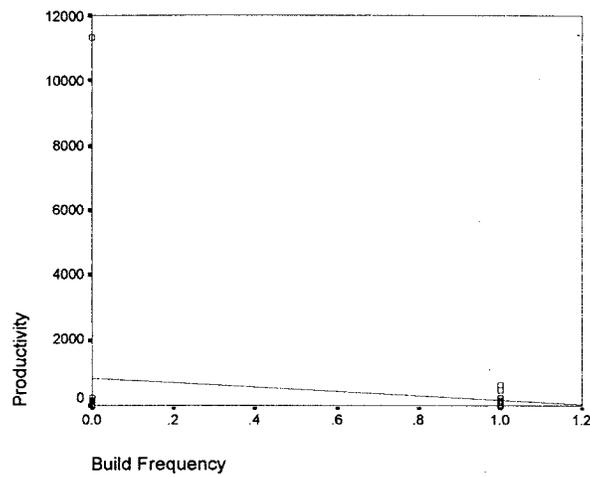
Figure 3-40 - Build Frequency vs. productivity (all projects)

Correlations between build frequency and productivity: **0.102.** The correlation between these two variables is statistically not significant. The correlation between these two variables is positive indicating that the frequent synchronization improves productivity. The potential reason could be that frequent synchronization would flush out any implementation issues or bugs immediately when new code is integrated into the product. This reduces the possibility of a team member continuing to extend the problematic area with new code, which at the end of the project might require more time to fix reducing the time available for the team to implement new features.

3.6.4 Sensitivity Analysis:

In our data analysis to evaluate hypothesis 10 and hypothesis 11, there are some instances where some outlier cases were observed for productivity variables. In order to study the effect of these outlier cases on the analysis, sensitivity analysis was performed. The correlation analysis was performed again with the data after filtering out the outlier case(s). The following correlation table (Table 3-9) contains the analysis without the outlier case(s). There are no scatter graphs following the table because there were no correlations between process and outcome variables, which were statistically significant.

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Build Frequency (Daily - 1; Other - 0)
% Original Features implemented	Correlation Coefficient	1.000	.275	-.250	-.255	.301	-.071	-.138
	Sig. (2-tailed)	.	.255	.288	.277	.197	.767	.561
	N	20	19	20	20	20	20	20
Bugginess (per mil LOC)	Correlation Coefficient	.275	1.000	-.032	.039	.278	.245	-.062
	Sig. (2-tailed)	.255	.	.898	.875	.249	.311	.801
	N	19	19	19	19	19	19	19
% Schedule Estimation Error	Correlation Coefficient	-.250	-.032	1.000	.226	-.190	-.060	-.141
	Sig. (2-tailed)	.288	.898	.	.287	.423	.802	.512
	N	20	19	24	24	20	20	24
Productivity	Correlation Coefficient	-.255	.039	.226	1.000	-.496*	-.071	.171
	Sig. (2-tailed)	.277	.875	.287	.	.026	.765	.425
	N	20	19	24	24	20	20	24
Schedule and Budget Perf. perception rating	Correlation Coefficient	.301	.278	-.190	-.496*	1.000	.072	-.099
	Sig. (2-tailed)	.197	.249	.423	.026	.	.762	.679
	N	20	19	20	20	20	20	20
Customer satisfaction perception rating	Correlation Coefficient	-.071	.245	-.060	-.071	.072	1.000	.171
	Sig. (2-tailed)	.767	.311	.802	.765	.762	.	.472
	N	20	19	20	20	20	20	20
Build Frequency (Daily - 1; Other - 0)	Correlation Coefficient	-.138	-.062	-.141	.171	-.099	.171	1.000
	Sig. (2-tailed)	.561	.801	.512	.425	.679	.472	.
	N	20	19	24	24	20	20	24

*. Correlation is significant at the .05 level (2-tailed).

Table 3-13 - Correlation Table For Frequent Synchronization Measure – without the outlier for productivity

75

3.6.5 Observations based on the data analysis for impact of frequent synchronization:

Hypothesis Number	Summary of hypothesis	Observations
10	Doing daily (frequent) builds gives rapid feedback to the project team about how the product is progressing	The correlation, between the process and outcome variables being used to measure the impact of frequent builds, is statistically not significant
11	Knowing where the team is, with respect to the project, makes the overall product development process more visible and predictable.	The correlation, between build frequency and schedule estimation error, is statistically not significant.

Table 3-14 Summary of hypotheses on impact of frequent synchronization

There are no significant observations made about the hypotheses since the data analysis did not yield any statistically significant correlation between process and outcome variables. A possible reason why there are no significant observations regarding frequency of synchronizations is that there is insufficient variance between building daily and building weekly or monthly, as opposed to a traditional waterfall method where projects build the whole system only in the final integration phase.

3.7 Impact of Design and Code Reviews

3.7.1 Hypothesis 12:

"Design reviews identify any consistency problems earlier than the later testing activities that require a running product" (*chapter 5, page 303* – Microsoft Secrets, Michael Cusumano and Richard Selby, 1998). With the pressure of short development cycles and uncertain environments, it is not clear if it is better to spend more time up front doing more reviews and design work or to devise better ways of checking for the problems later.

- This could result in more feature evolution due to design changes for incorporating changing requirements and/or market and technical feedback.

- As design reviews are done to reduce consistency problems, this potentially has neutral or positive impact on bugginess (I.e. reduce bugginess).

- Design reviews potentially could create an opportunity to introduce more delay.

3.7.2 Hypothesis 13:

Code review helps in early detection of bugs

- Reducing the bugginess of the product

- Reducing schedule estimation error since it takes lot more time to track and fix bugs at a late stage in the product development cycle. Alternately, reviews introduce opportunity for more delay.

The process variables that are used to track design/code review are:

- Design review done or not

- Number of design reviews

- Code review done or not

- Number of people reviewing the code

Some of the outcome variables that were study to understand the impact of design and code reviews are:

- % Of original features implemented in the final product

- Bugginess

- Schedule estimation error

3.7.3 Data analysis to evaluate the impact of design and code reviews

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Design Review (Yes-1; No - 0)	Number of Design Reviews	Code Review (Yes- 1; No - 0)	Number of People reviewing code
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.182	-.101	-.303	-.243
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.429	.662	.182	.288
	N	21	20	21	21	21	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.499*	-.358	-.410	-.338
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.025	.121	.073	.144
	N	20	20	20	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	.091	.082	.123	.168
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.666	.696	.558	.421
	N	21	20	25	25	21	21	25	25	25	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.242	-.144	-.123	-.167
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.244	.494	.558	.426
	N	21	20	25	25	21	21	25	25	25	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	.249	.362	-.184	-.118
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.276	.107	.425	.610
	N	21	20	21	21	21	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.100	-.023	-.027	.005
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.667	.920	.908	.983
	N	21	20	21	21	21	21	21	21	21	21
Design Review (Yes-1; No - 0)	Correlation Coefficient	-.182	-.499*	.091	-.242	.249	.100	1.000	.666**	.053	.130
	Sig. (2-tailed)	.429	.025	.666	.244	.276	.667	.	.000	.802	.535
	N	21	20	25	25	21	21	25	25	25	25
Number of Design Reviews	Correlation Coefficient	-.101	-.358	.082	-.144	.362	-.023	.666**	1.000	-.070	-.064
	Sig. (2-tailed)	.662	.121	.696	.494	.107	.920	.000	.	.738	.761
	N	21	20	25	25	21	21	25	25	25	25
Code Review (Yes- 1; No - 0)	Correlation Coefficient	-.303	-.410	.123	-.123	-.184	-.027	.053	-.070	1.000	.925**
	Sig. (2-tailed)	.182	.073	.558	.558	.425	.908	.802	.738	.	.000
	N	21	20	25	25	21	21	25	25	25	25
Number of People reviewing code	Correlation Coefficient	-.243	-.338	.168	-.167	-.118	.005	.130	-.064	.925**	1.000
	Sig. (2-tailed)	.288	.144	.421	.426	.610	.983	.535	.761	.000	.
	N	21	20	25	25	21	21	25	25	25	25

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-15 - Correlation Table For Design and Code Review Measure
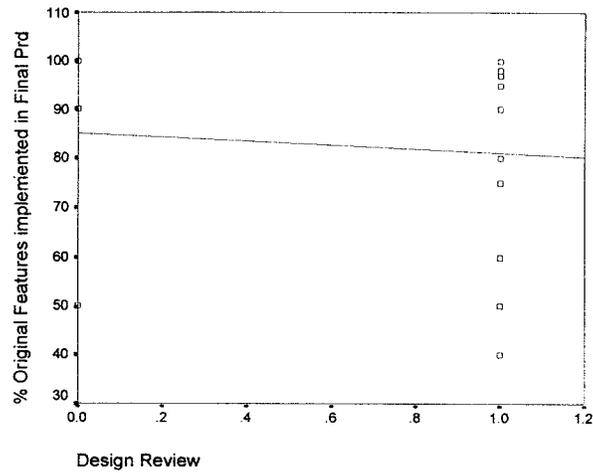
Figure 3-41- Design Review done or not vs. % Original Features implemented in final product (all projects)

Correlation between Design review done(1) or not (0) and % original features implemented in final product: **-0.182.** The correlation between these two variables is statistically not significant. The negative correlation could be because design review might provide technical feedback to the team, which may result in changing some of the features from the original list.
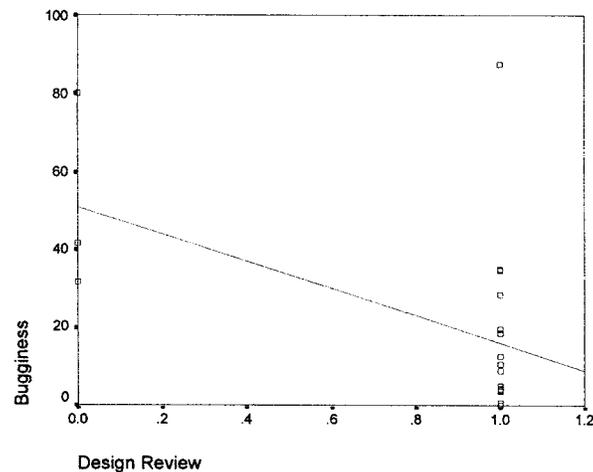
Figure 3-42 - Design Review done or not Vs. Bugginess (all projects)

Correlation between Design review and Bugginess: **-0.499.** The Correlation between these two variables is statistically significant at the 0.01 level (two-tailed). This validates our hypothesis

and the reason for this negative correlation is because design reviews will provide technical (engineering) feedback to the team and prevent potential situations, well into the project, where the project team discovers the engineering issues and has to change their implementation possibly leading to more bugs.
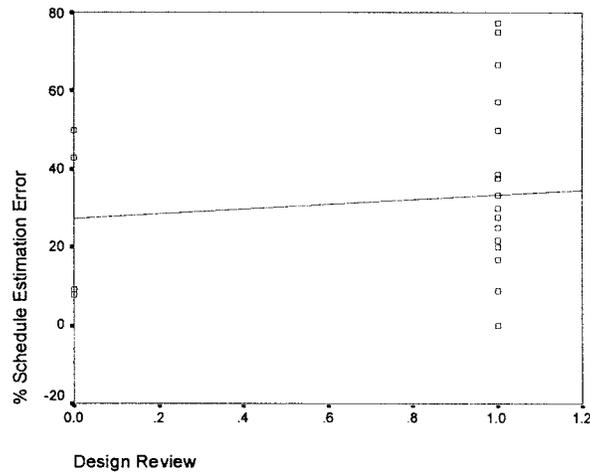
Figure 3-43 - Design review vs. % schedule estimation error (all projects)

Correlation between Design review and % schedule estimation error: **0.091.** The correlation between these two variables is statistically not significant. The correlation is practically non-existent between these two variables. There could be other factors affecting the schedule, for example several of our cases have hardware dependencies.
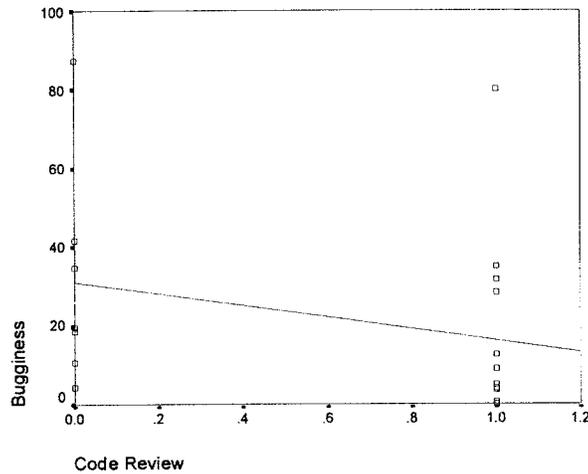
Figure 3-44 - Code Review done or not vs. Bugginess (all projects)

Correlation between Code review and Bugginess: **-0.410.** The correlation between these two variables is statistically not significant. The reason for this negative correlation is because code reviews will provide technical (engineering) feedback to the team (ex: potential logical errors in an algorithm or possible constraints with a particular implementation and prevent potential situations, well into the project, where the project team discovers the engineering issues and has to change their implementation possibly leading to more bugs.
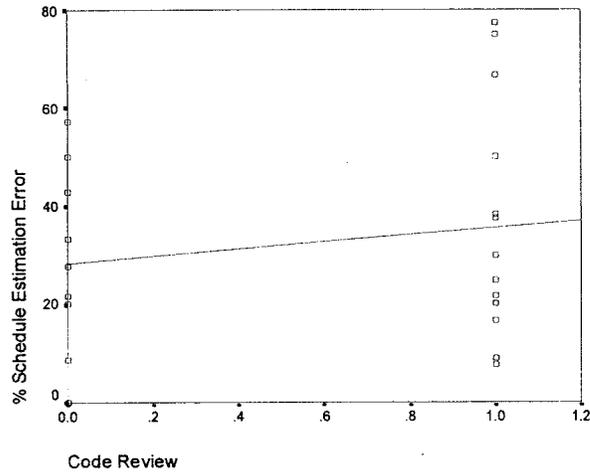
Figure 3-45 - Code Review done or not vs. % schedule estimation error (all projects)

Correlation between Code review and % schedule estimation error: **0.123.** The correlation between these two variables is statistically not significant. Depending on when the code reviews are done and what types of issues are found i.e., if the code reviews find incorrect implementations that require rework then it impacts the project schedule, increasing schedule estimation error.

3.7.4 Observations based on the data analysis for impact of Design and Code review:

Hypothesis Number	Summary of hypothesis	Observations
12	Design reviews identify any consistency problems earlier than the later testing activities that require a running product	The correlation between design review and bugginess is statistically significant.
13	Code review helps in early detection of bugs.	The correlation between code review and bugginess is statistically not significant.

Table 3-16 Summary of hypotheses on impact of design and code review

- Analysis shows that having design reviews reduces the bugginess of the product validating our hypothesis.

3.8 Impact of simple compile and link test vs. regression testing

3.8.1 Hypothesis 14:

"PRINCIPLE: Continuously test the product as you build it". Too many software producers emphasize product testing primarily at the end of the development cycle, when fixing bugs can be extraordinary difficult and time-consuming (*chapter 5, pages 294-295* – Microsoft Secrets, Michael Cusumano and Richard Selby, 1998). Running regression tests, each time developers check changed or new code into the project build, improves product quality.

The process variable that was used to track the regression testing is:

- Regression test done or simple compile and link test done.

The outcome variable that we will study to understand the impact of regression testing is:

- Bugginess

3.8.2 Data analyses to evaluate Impact of simple compile and link test vs. regression testing

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	Regression Test (Yes - 1; No - 0)
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.152
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.511
	N	21	20	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.531*
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.016
	N	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.066
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.756
	N	21	20	25	25	21	21	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.107
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.611
	N	21	20	25	25	21	21	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.253
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.269
	N	21	20	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	.157
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.497
	N	21	20	21	21	21	21	21
Regression Test (Yes - 1; No - 0)	Correlation Coefficient	-.152	-.531*	-.066	-.107	-.253	.157	1.000
	Sig. (2-tailed)	.511	.016	.756	.611	.269	.497	.
	N	21	20	25	25	21	21	25

*. Correlation is significant at the .05 level (2-tailed).

Table 3-17- Correlation Table For Regression Test Measure
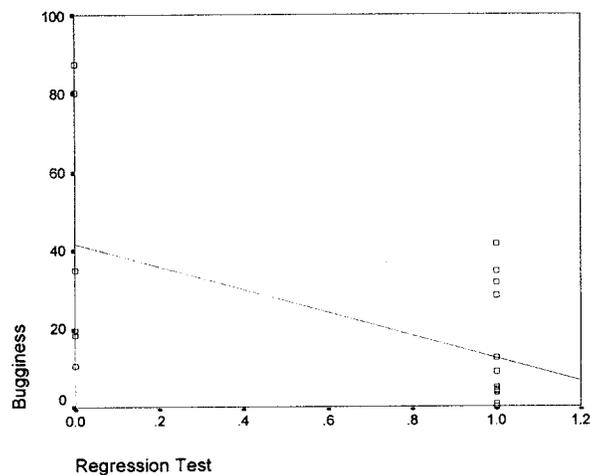
Figure 3-46 - Running Regression Test or not Vs. Bugginess (all projects)

Correlation between Running Regression Test (1) or not (0) and Bugginess: **-0.531.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed)

3.8.3 Observations based on the data analysis for impact of simple compile and link test vs. regression testing:

Hypothesis Number	Summary of hypothesis	Observations
14	Running regression tests, each time developers check changed or new code into the project build, improves product quality.	The correlation between regression tests and bugginess is statistically significant.

Table 3-18 Summary of hypothesis on impact of simple compile and link test vs. regression testing

- Analysis validates our hypothesis that projects running regression test reduces the bugginess of the product. As we see from the analysis, for projects running regression test the number of bugs reported by customers is dropping.

85

3.9 Relative emphasis of developers testing vs. QA staff testing code

3.9.1 Hypothesis 15:

Knowledge of the code and product features help in testing the product. When developers, with intimate knowledge of the code and features, spend more time testing their code, the product bugginess decreases but the productivity also decreases.

3.9.2 Hypothesis 16:

As testing effort increases, the bugginess of the product decreases.

The process variables that are used to track the time spent by developers and QA staff testing the code are:

- % Of total testing time developers tested their own code

- % Of total testing time separate QA staff tested code.

- Testing effort

Some of the outcome variables that were used to understand the impact of time spent by developers and QA testing code are:

- Bugginess

- Productivity

3.9.3 Data analysis for Relative emphasis of developers testing vs. QA staff testing code

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% of total testing time dev. tested their own code	% of total testing time QA tested code	Testing Effort
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	-.044	.107	.291
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.850	.645	.200
	N	21	20	21	21	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.462*	.375	-.029
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.040	.103	.904
	N	20	20	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.012	-.053	.072
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.955	.801	.733
	N	21	20	25	25	21	21	25	25	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.134	.105	.114
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.525	.616	.587
	N	21	20	25	25	21	21	25	25	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.088	.141	-.049
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.704	.543	.834
	N	21	20	21	21	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	-.116	.040	-.433*
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.615	.864	.050
	N	21	20	21	21	21	21	21	21	21
% of total testing time dev. tested their own code	Correlation Coefficient	-.044	-.462*	-.012	-.134	-.088	-.116	1.000	-.961**	-.328
	Sig. (2-tailed)	.850	.040	.955	.525	.704	.615	.	.000	.109
	N	21	20	25	25	21	21	25	25	25
% of total testing time QA tested code	Correlation Coefficient	.107	.375	-.053	.105	.141	.040	-.961**	1.000	.299
	Sig. (2-tailed)	.645	.103	.801	.616	.543	.864	.000	.	.147
	N	21	20	25	25	21	21	25	25	25
Testing Effort	Correlation Coefficient	.291	-.029	.072	.114	-.049	-.433*	-.328	.299	1.000
	Sig. (2-tailed)	.200	.904	.733	.587	.834	.050	.109	.147	.
	N	21	20	25	25	21	21	25	25	25

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-19- Correlation Table For Developers and QA testing Code
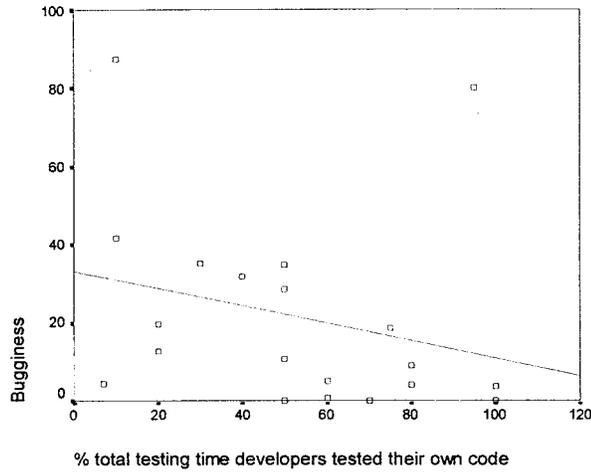
Figure 3-47 - % of total testing time developers tested their own code vs. bugginess (all projects)

Correlation between % of total testing time developers tested their own code and bugginess: -0.462. The Correlation between the two variables is statistically significant at the 0.05 level (two-tailed). The reasoning for this is explained in hypothesis 15.
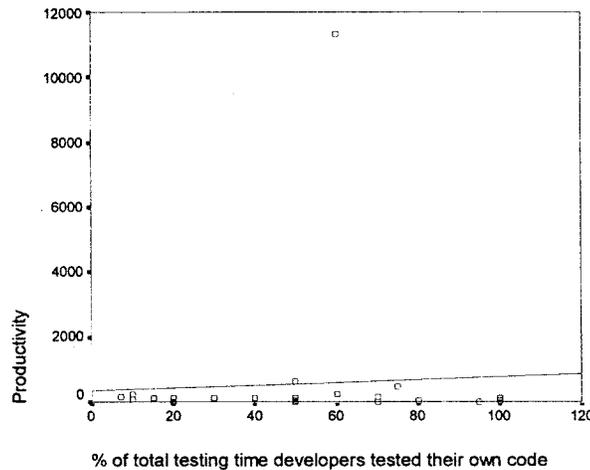
Figure 3-48 - % of total testing time developers tested their own code vs. productivity (all projects)

Correlation between % of total testing time developers tested their own code and productivity: -0.134. The correlation between these two variables is statistically not significant. The correlation between these two variables is negative implying that as the % of total testing time developers tested their own code increased, the productivity decreased. The correlation between

these two variables is not significant even after the outlier productivity case was filtered out. One reason could be because the productivity is a function of lines of code and there are many other factors that might impact the lines of code written for a product.
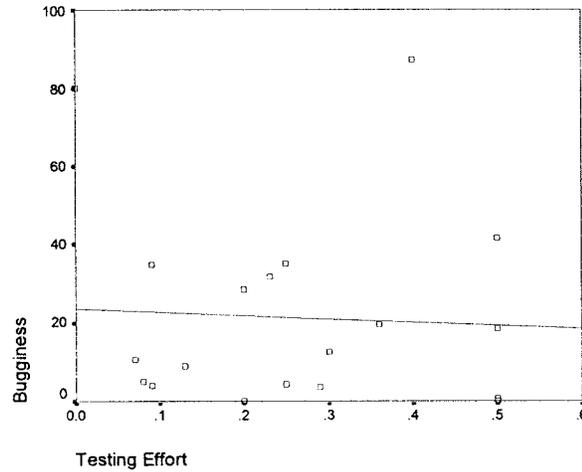
Figure 3-49 - Testing effort vs. bugginess (all projects)

Correlation between testing effort and bugginess: **-0.029.** The correlation between these two variables is statistically not significant. Even though the correlation is negative, it is very small.
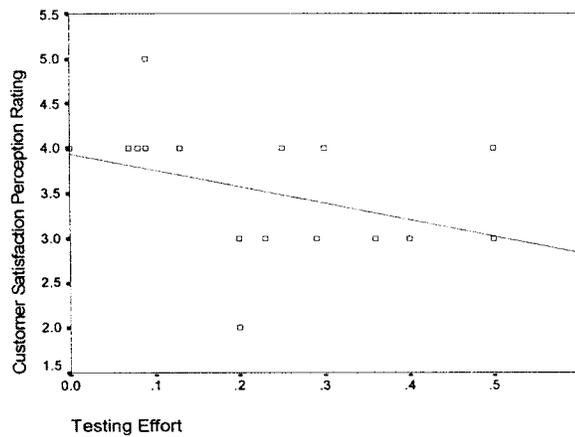
Figure 3-50 - Testing effort vs. customer satisfaction perception rating (all projects)

Correlation between testing effort and customer satisfaction perception rating: **-0.433.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed).

This correlation is not significant when the outlier for productivity is removed as seen from table 3-13.

3.9.4 Sensitivity Analysis:

In our data analysis to evaluate hypothesis 15 and hypothesis 16, there are some instances where some outlier cases were observed for productivity variables. In order to study the effect of these outlier cases on the analysis, sensitivity analysis was performed. The correlation analysis was performed again with the data after filtering out the outlier case(s). The following correlation table (Table 3-13) contains the analysis without the outlier case(s). There are no scatter graphs following the table because there are no correlations between process and outcome variables, which were statistically significant.

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% of total testing time dev. tested their own code	% of total testing time QA tested code	Testing Effort
% Original Features implemented	Correlation Coefficient	1.000	.275	-.250	-.255	.301	-.071	-.027	.086	.348
	Sig. (2-tailed)		.255	.288	.277	.197	.767	.912	.718	.133
	N	20	19	20	20	20	20	20	20	20
Bugginess (per mil LOC)	Correlation Coefficient	.275	1.000	-.032	.039	.278	.245	-.456*	.380	.071
	Sig. (2-tailed)	.255		.898	.875	.249	.311	.050	.108	.774
	N	19	19	19	19	19	19	19	19	19
% Schedule Estimation Error	Correlation Coefficient	-.250	-.032	1.000	.226	-.190	-.060	.006	-.067	.083
	Sig. (2-tailed)	.288	.898		.287	.423	.802	.978	.754	.701
	N	20	19	24	24	20	20	24	24	24
Productivity	Correlation Coefficient	-.255	.039	.226	1.000	-.496*	-.071	-.168	.130	.011
	Sig. (2-tailed)	.277	.875	.287		.026	.765	.432	.544	.960
	N	20	19	24	24	20	20	24	24	24
Schedule and Budget Perf. perception rating	Correlation Coefficient	.301	.278	-.190	-.496*	1.000	.072	-.097	.140	-.108
	Sig. (2-tailed)	.197	.249	.423	.026		.762	.683	.556	.649
	N	20	19	20	20	20	20	20	20	20
Customer satisfaction perception rating	Correlation Coefficient	-.071	.245	-.060	-.071	.072	1.000	-.106	.038	-.409
	Sig. (2-tailed)	.767	.311	.802	.765	.762		.658	.875	.073
	N	20	19	20	20	20	20	20	20	20
% of total testing time dev. tested their own code	Correlation Coefficient	-.027	-.456*	.006	-.168	-.097	-.106	1.000	-.965**	-.387
	Sig. (2-tailed)	.912	.050	.978	.432	.683	.658		.000	.061
	N	20	19	24	24	20	20	24	24	24
% of total testing time QA tested code	Correlation Coefficient	.086	.380	-.067	.130	.140	.038	-.965**	1.000	.346
	Sig. (2-tailed)	.718	.108	.754	.544	.556	.875	.000		.098
	N	20	19	24	24	20	20	24	24	24
Testing Effort	Correlation Coefficient	.348	.071	.083	.011	-.108	-.409	-.387	.346	1.000
	Sig. (2-tailed)	.133	.774	.701	.960	.649	.073	.061	.098	
	N	20	19	24	24	20	20	24	24	24

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-20 - Correlation Table For Developers and QA testing Code Measure – without the outlier for productivity

3.9.5 Observations based on analysis of developers and QA testing code:

Hypothesis Number	Summary of hypothesis	Observations
15	When developers, with intimate knowledge of the code and features, spend more time testing their code, the product bugginess decreases but the productivity also decreases.	The correlation between % of total testing time developers tested their own code and bugginess is statistically significant. The correlation between % of total testing time developers tested their own code and productivity is statistically not significant.
16	As testing effort increases, the bugginess of the product decreases.	The correlation between testing effort and bugginess is statistically not significant.

Table 3-21 Summary of hypotheses on impact of developers and QA staff testing code

- Analysis validates our hypothesis that when developers spend more time testing their own code then the product bugginess is decreased but affects the productivity.

3.10 Relative emphasis of component testing vs. integration testing vs. system testing

3.10.1 Hypothesis 17:

More emphasis on component testing (% of total testing time spent in component testing) reduces product bugginess (but does not necessarily mean there would be no issues with system integration).

3.10.2 Hypothesis 18:

More emphasis on integration testing (% of total testing time spent in integrating testing) reduces product bugginess and reduces schedule estimation error due to less integration issues at the end of the product development cycle. Alternately, if the team is spending increased amount of time in integrating testing, it could be because the team may be facing integration issues thus affecting the project schedule and increasing the schedule estimation error.

3.10.3 Hypothesis 19:

More emphasis on system testing may find and help resolve bugs that would not be apparent in component testing leading to improved customer satisfaction.

The process variables that are used to track the emphasis of testing are:

- % Of total testing time spent in component testing

- % Of total testing time spent in integration testing

- % Of total testing time spent testing the complete system.

Some of the outcome variables that we will study to understand the impact of different emphasis of testing are:

- Bugginess

- Schedule estimation error

- Customer satisfaction perception rating

3.10.4 Data analysis for relative emphasis of component testing vs. integration testing vs. system testing

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% of total testing time spent testing components	% of total testing time spent on integration testing	% of total testing time spent on system testing
% Original Features implemented	Correlation Coefficient	1.000	.297	-.225	-.284	.272	-.049	.011	-.211	.220
	Sig. (2-tailed)	.	.204	.327	.212	.233	.834	.964	.358	.337
	N	21	20	21	21	21	21	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.297	1.000	-.022	-.056	.222	.305	-.381	-.230	.428
	Sig. (2-tailed)	.204	.	.927	.816	.348	.191	.098	.329	.060
	N	20	20	20	20	20	20	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.225	-.022	1.000	.191	-.198	-.052	-.082	.414*	-.200
	Sig. (2-tailed)	.327	.927	.	.361	.390	.822	.696	.040	.338
	N	21	20	25	25	21	21	25	25	25
Productivity	Correlation Coefficient	-.284	-.056	.191	1.000	-.385	-.135	-.155	.179	.003
	Sig. (2-tailed)	.212	.816	.361	.	.085	.561	.459	.392	.987
	N	21	20	25	25	21	21	25	25	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.272	.222	-.198	-.385	1.000	.032	-.286	.016	.338
	Sig. (2-tailed)	.233	.348	.390	.085	.	.891	.209	.945	.134
	N	21	20	21	21	21	21	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.049	.305	-.052	-.135	.032	1.000	-.131	-.253	.222
	Sig. (2-tailed)	.834	.191	.822	.561	.891	.	.572	.268	.333
	N	21	20	21	21	21	21	21	21	21
% of total testing time spent testing components	Correlation Coefficient	.011	-.381	-.082	-.155	-.286	-.131	1.000	-.206	-.810**
	Sig. (2-tailed)	.964	.098	.696	.459	.209	.572	.	.322	.000
	N	21	20	25	25	21	21	25	25	25
% of total testing time spent on integration testing	Correlation Coefficient	-.211	-.230	.414*	.179	.016	-.253	-.206	1.000	-.279
	Sig. (2-tailed)	.358	.329	.040	.392	.945	.268	.322	.	.177
	N	21	20	25	25	21	21	25	25	25
% of total testing time spent on system testing	Correlation Coefficient	.220	.428	-.200	.003	.338	.222	-.810**	-.279	1.000
	Sig. (2-tailed)	.337	.060	.338	.987	.134	.333	.000	.177	.
	N	21	20	25	25	21	21	25	25	25

*. Correlation is significant at the .05 level (2-tailed).

**. Correlation is significant at the .01 level (2-tailed).

Table 3-22 – Correlation Table For Emphasis of Testing
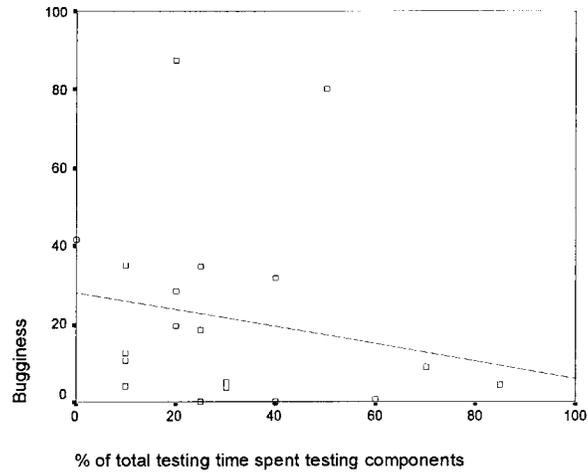
Figure 3-51 - % of total testing time spent in component testing vs. Bugginess (all projects)

Correlation between % of total testing time spent in component testing and Bugginess: **-0.381.**
The correlation between these two variables is statistically not significant. The negative
correlation implies that if the team spends more % of total testing time in component testing, the
bugginess of the product decreases which seems to be logical since the team would be spending
considerable time testing each component.
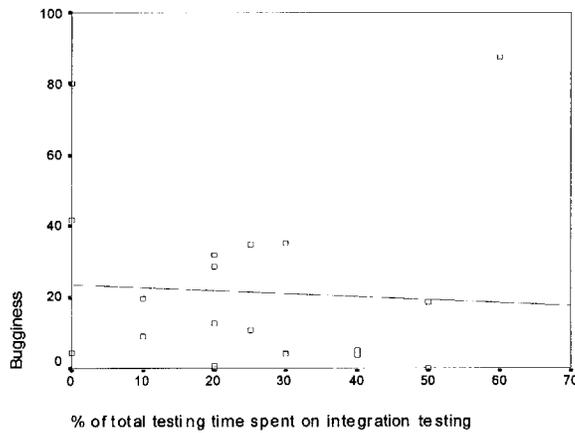
Figure 3-52 - % of total testing time spent in integration testing vs. bugginess (all projects)

Correlation between % of total testing time spent in integration testing and bugginess: **-0.230.**
The correlation between these two variables is statistically not significant.
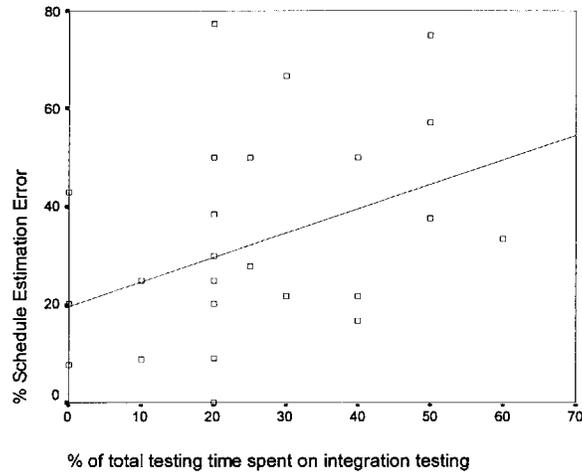
Figure 3-53 - % of total testing time spent in integration testing Vs. Schedule Estimation Error (all projects)

Correlation between % of total testing time spent in integration testing and Schedule Estimation Error: **0.414.** The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). The reason for the positive correlation between these two variables could be because the team if the team spends more time in integrating testing could be due to integrating issues being encountered by the team. The above scenario would impact the project schedule, increasing schedule estimation error.
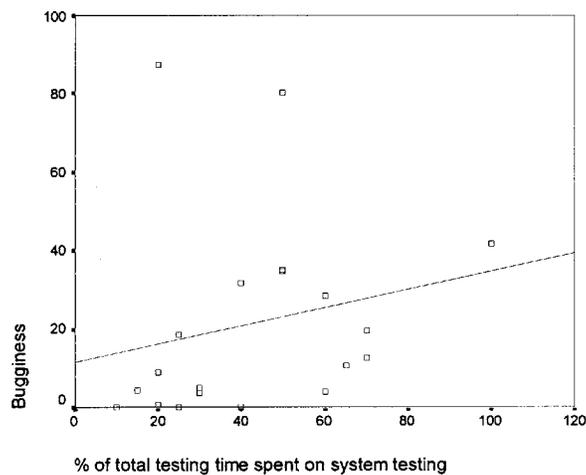
Figure 3-54 - % of total testing time spent in system testing Vs. Bugginess (all projects)

Correlation between % of total testing time spent in system testing and Bugginess: **0.428.** The
correlation between these two variables is statistically not significant. The correlation is a very
interesting one because as the % of total testing spends in system testing increases the bugginess
is also increasing. One would think that the bugginess would go down. The question to consider,
to better understand this relation, is whether the team is spending less time in other testing areas
like component testing and integration testing when they spend more time, as % of total testing
time, system testing. If that is the case then one possible explanation for this positive correlation
is that the bugs in components may not have been completely identified and resolved.
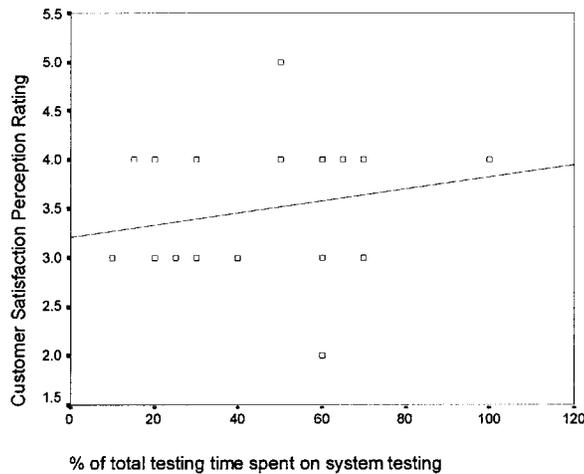
**Figure 3-55 - % of total testing time spent in system testing Vs. Customer satisfaction perception
rating (all projects)**

Correlation between % of total testing time spent in system testing and customer satisfaction
perception rating: **0.222.** The correlation between these two variables is statistically not
significant. The interesting part about this correlation is that when viewed with the previous case
(figure 3-54), it is puzzling in the sense that a system with increased bugginess seems to have
high customer satisfaction perception rating. The only explanation that seems to be reasonable is
that the features implemented meet the customer needs and the positive user experience because
of this may be shadowing the inconvenience caused by the bugs.

3.10.5 Observations based on analysis of Relative Emphasis of Testing:

Hypothesis Number	Summary of hypothesis	Observations
17	More emphasis on component testing (% of total testing time spent in component testing) reduces product bugginess.	The correlation between % of total testing time spent in component testing and bugginess is statistically not significant.
18	More emphasis on integration testing (% of total testing time spent in integrating testing) reduces product bugginess and reduces schedule estimation error due to less integration issues at the end of the product development cycle	The correlation between % of total testing time spent in integrating testing and bugginess is statistically not significant. The correlation between % of total testing time spent in integrating testing and schedule estimation error is statistically significant.
19	More emphasis on system testing may find and help resolve bugs that would not be apparent in component testing leading to improved customer satisfaction.	The correlation between % of total testing time spent in system testing and bugginess is statistically not significant.

Table 3-23 Summary of hypotheses on impact of relative emphasis of testing

- For integration testing, analysis shows that the % schedule estimation error is increasing with increase in relative emphasis on integration testing. This validates our alternate hypothesis.

3.11 Impact of Final Stabilization Phase

3.11.1 Hypothesis 20:

If the project team has enough time for final product stabilization phase then they have completed the project on time. This results in:

- Lower schedule estimation error

3.11.2 Hypothesis 21:

The project team may decide to spend time on final product stabilization versus making late design changes that incorporate market and technical feedback. This may result in increased % of original features implemented in the final product. Alternately, if the project team is incorporating market and technical feedback then the project team will have less time for final product stabilization phase

3.11.3 Data analysis for Impact of Final Stabilization Phase

		% Original Features implemented	Bugginess (per mil LOC)	% Schedule Estimation Error	Productivity	Schedule and Budget Perf. perception rating	Customer satisfaction perception rating	% Prj Duration spent in stabilization phase	% final product functionality in first prototype	% final product functionality in first system integration	% final product functionality in first beta
% Original Features implemented	Correlation Coefficient	1.000	.236	-.247	-.253	.227	-.071	.360	.244	.330	.680**
	Sig. (2-tailed)	.	.303	.268	.255	.311	.753	.100	.286	.144	.001
	N	22	21	22	22	22	22	22	21	21	21
Bugginess (per mil LOC)	Correlation Coefficient	.236	1.000	.064	-.160	.323	.348	-.285	.575**	.151	.183
	Sig. (2-tailed)	.303	.	.781	.489	.154	.122	.210	.008	.525	.441
	N	21	21	21	21	21	21	21	20	20	20
% Schedule Estimation Error	Correlation Coefficient	-.247	.064	1.000	.114	-.111	-.006	-.437*	-.037	.063	-.498*
	Sig. (2-tailed)	.268	.781	.	.578	.623	.978	.026	.864	.769	.011
	N	22	21	26	26	22	22	26	24	24	25
Productivity	Correlation Coefficient	-.253	-.160	.114	1.000	-.450*	-.178	-.042	-.212	-.098	-.138
	Sig. (2-tailed)	.255	.489	.578	.	.035	.429	.839	.321	.648	.510
	N	22	21	26	26	22	22	26	24	24	25
Schedule and Budget Perf. perception rating	Correlation Coefficient	.227	.323	-.111	-.450*	1.000	.096	.051	.142	.154	.276
	Sig. (2-tailed)	.311	.154	.623	.035	.	.670	.822	.540	.506	.226
	N	22	21	22	22	22	22	22	21	21	21
Customer satisfaction perception rating	Correlation Coefficient	-.071	.348	-.006	-.178	.096	1.000	-.458*	.260	-.500*	-.247
	Sig. (2-tailed)	.753	.122	.978	.429	.670	.	.032	.256	.021	.280
	N	22	21	22	22	22	22	22	21	21	21
% Prj Duration spent in stabilization phase	Correlation Coefficient	.360	-.285	-.437*	-.042	.051	-.458*	1.000	.054	.383	.540**
	Sig. (2-tailed)	.100	.210	.026	.839	.822	.032	.	.802	.064	.005
	N	22	21	26	26	22	22	26	24	24	25
% final product functionality in first prototype	Correlation Coefficient	.244	.575**	-.037	-.212	.142	.260	.054	1.000	.466*	.472*
	Sig. (2-tailed)	.286	.008	.864	.321	.540	.256	.802	.	.022	.020
	N	21	20	24	24	21	21	24	24	24	24
% final product functionality in first system integration	Correlation Coefficient	.330	.151	.063	-.098	.154	-.500*	.383	.466*	1.000	.449*
	Sig. (2-tailed)	.144	.525	.769	.648	.506	.021	.064	.022	.	.028
	N	21	20	24	24	21	21	24	24	24	24
% final product functionality in first beta	Correlation Coefficient	.680**	.183	-.498*	-.138	.276	-.247	.540**	.472*	.449*	1.000
	Sig. (2-tailed)	.001	.441	.011	.510	.226	.280	.005	.020	.028	.
	N	21	20	25	25	21	21	25	24	24	25

**. Correlation is significant at the .01 level (2-tailed).

*. Correlation is significant at the .05 level (2-tailed).

Table 3-24- Correlation Table For Final Product Stabilization Phase
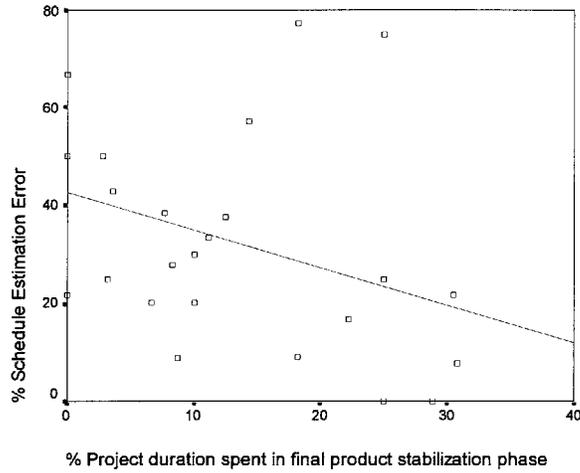
Figure 3-56- % project duration spent in stabilization phase vs. % schedule estimation error (all projects)

Correlation between % project duration spent in stabilization phase and % schedule estimation error: **-0.437**. The Correlation between these two variables is statistically significant at the 0.05 level (two-tailed). This validates our hypothesis. The reason for the project team to have an increased % project duration spent in final product stabilization phase could be because the team would have completed implementation of all the features. This would lead to on time delivery of the product thus reducing schedule estimation error.
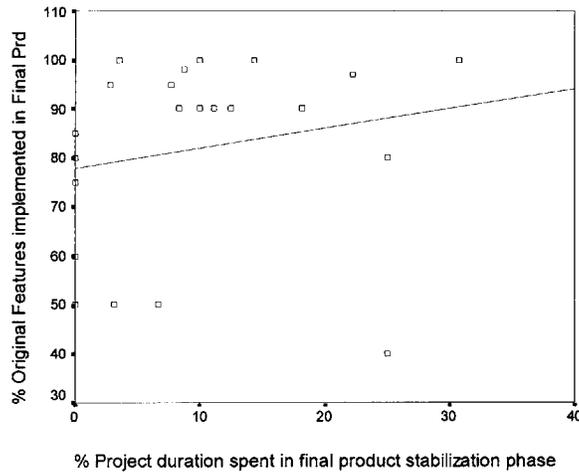
Figure 3-57 - % project duration spent in stabilization phase vs. % Original features implemented in final product (all projects)

100

Correlation between % project duration spent in stabilization phase and % Original features implemented in final product: **0.360.** The correlation between these two variables is statistically not significant. The reason for the correlation is the same as in the previous case.
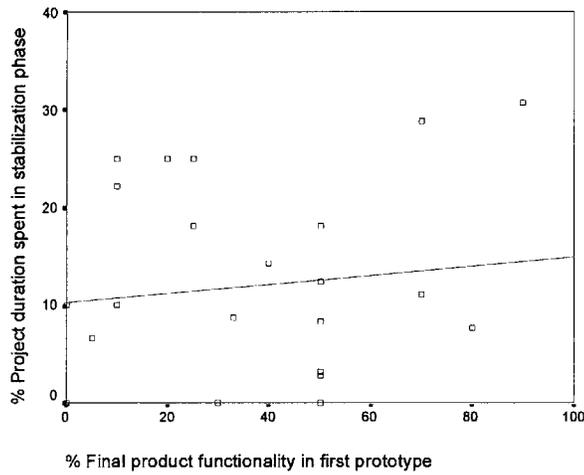
Figure 3-58 - % project duration spent in stabilization phase vs. % final product functionality in first prototype

Correlation between % project duration spent in stabilization phase and % Final product functionality in first prototype: **0.054.** The correlation between these two variables is statistically not significant and there seems to be very little correlation.
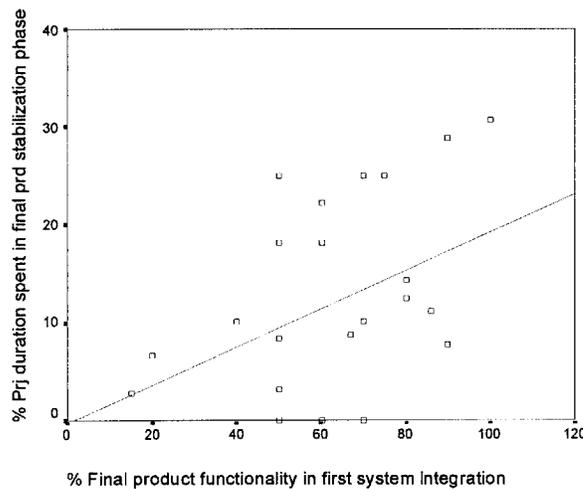
Figure 3-59 - % project duration spent in stabilization phase vs. % final product functionality in first system integration

101

Correlation between % project duration spent in stabilization phase and % final product functionality in first system integration: **0.383**. Even though the correlation is statistically not significant but the correlation is stronger than the correlation at the first prototype.
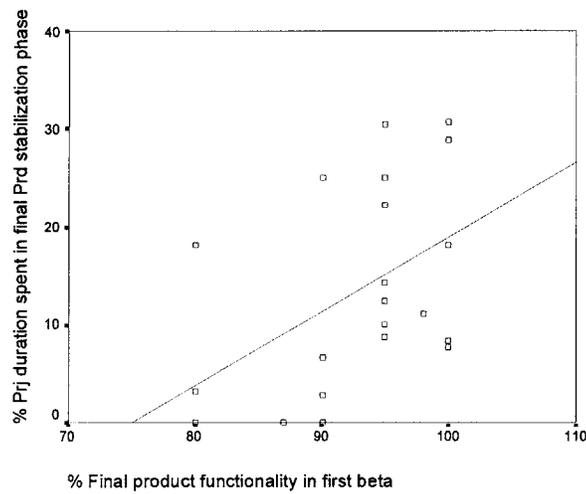
Figure 3-60 - % project duration spent in stabilization phase vs. % final product functionality in first beta (all projects)

Correlation between % project duration spent in stabilization phase and % final product functionality in first beta: **0.540**. The Correlation between these two variables is statistically significant at the 0.01 level (two-tailed). Compared to the previous two cases, the correlation is stronger and also significant. Based on this observation it appears that the project team should implement the functionality of the product at a steady rate between the three major milestones.

3.11.4 Observations based on analysis of Final product stabilization phase variables:

Hypothesis Number	Summary of hypothesis	Observations
20	If the project team has enough time for final product stabilization phase then they have completed the project on time. This results in lower schedule estimation error.	The correlation between % project duration spent in stabilization phase and % schedule estimation error is statistically significant.
21	The project team may decide to spend time on final product stabilization versus making late design changes that incorporate market and technical feedback. This may result in increased % of original features implemented in the final product.	The correlation between % project duration spent in stabilization phase and % final product functionality in first beta is statistically significant.

Table 3-25 Summary of hypotheses on impact of final product stabilization phase

- Analysis shows that an increase in the duration for final product stabilization phase is decreasing the % schedule estimation error. This validates our hypothesis.

- Our second hypothesis is also validated which states that as the % of original features implemented in the final product increases, so is the duration for final product stabilization phase.

Chapter 4: Conclusions

4.1 Current state of project practices:

This research has analyzed software projects at Hewlett Packard and Agilent. The current software product development practices at these firms are very diverse. Some of the projects use sequential (waterfall) approach while some others are leaning towards iterative (evolutionary) approach. Based on the data analysis here is a summary of current state of project practices:

- One of the observations based on the data analysis was that about 50% of the projects did not have project requirements available at the design start time.

- On average the projects were reusing about 60% of code from various sources including previous versions of the products. This is a significant amount of code reuse.

- On average the proportion of resources (development + testing) allocated to full time QA was 25%. This does not include the testing undertaken by the developers.

- About 92% of the projects had some sort of prototype. On average the first prototypes had about 37% of final product functionality implemented. The range of final product functionality implemented at the first prototype is 0 to 90%. These prototypes were completed 33% of the way through the product development cycle and this ranged from 4 to 83%.

- On average at the time of first system integration of the system the team had implemented about 63% of final product functionality. The range of the final product functionality implemented at the first system integration is 15 to 100%. The first system integration, on average, occurred 58% of the way through the product development cycle and this ranged from 25 to 93%.

- About 73% of the projects had a beta release. On average the first beta had about 92% of final product functionality implemented. The range of final product functionality implemented at the first beta is 80 to 100%. The first beta was released about 78% of the way through the product development cycle and this ranged from 30 to 100%.

- In the area of daily builds, 11 projects built daily while the other 15 ranged from weekly to monthly.

4.2 Practices for flexible product development:

There are some commonly used product development approaches in practice, such as sequential (waterfall) approach, iterative (evolutionary) approach, iterative approach combined with synch-and-stabilize approach. Based on the data analysis, some of the important factors that influence a flexible product development strategy are:

- With increased competition and constantly changing technological landscape, there is increased burden on the project teams to deliver a product that meets the customer needs in the first try itself. To achieve this the project teams should obtain customer feedback (both feedback on the prototype and feedback on the beta release of the product) early in the project, with respect to functionality. This provides the project team the opportunity to incorporate the customer feedback without extensive rework.

- The project teams should be aware of the fact that obtaining and incorporating customer feedback results in feature evolution, which may be significantly different than their original feature list. This will impact the project schedules. Allowing time, for obtaining and incorporating customer feedback, in the project schedule is critical so that the project team does not find itself in a situation where they have to cut corners in various project activities. As our data analysis shows that incorporating more % of final product functionality in the first beta reduces the schedule estimation error but the tradeoff is that the project team may not be in a situation to incorporate the customer feedback obtained during the beta phase. One approach to solve this dilemma would be to obtain customer feedback more frequently even before first beta and this could be achieved by utilizing the iterative (evolutionary) approach. Before the project team adopts the iterative (evolutionary) approach, they should put in place a process that would help them manage the feedback process. Another option is for the project teams to release earlier betas or prototypes.

- It has always been a point of discussion as to how much architectural effort should be put into the product design. From the analysis, the conclusion is that the project team should put in high-level architecture and quickly move to the implementation. This provides the

team with the flexibility to incorporate the customer feedback without being bogged down by rigid rules developed through detailed architecture and design. The team should keep in mind partition of the architecture if they are interested in incorporating customer feedback. Clean interfaces between modules could help the team in localizing the changes to the product implementation when they are incorporating the customer feedback.

- As has been shown by many experts in the software product development area, our analysis shows that design reviews help improve the product quality. In the iterative (evolutionary) approach one of the areas that the design review could help is evaluating how the architecture is partitioned and whether the interfaces between various modules clean and independent. As was mentioned earlier, clean interfaces could help the project team localize the changes to the product implementation due to customer feedback.

- The data analysis shows repeatedly that the project is in good shape, with respect to project schedule, if the product has higher % of final product functionality implemented in the product at first beta release. As was mentioned earlier, to reduce the amount of changes to the product after first beta release due to customer feedback, the project team should get continuous feedback through out the product development. The basic idea here is that if the project team involves the customer actively through out the product development cycle, the feature evolution will reduce significantly by first beta release. With this approach the project team potentially will be in a situation where they are able to implement higher % of final product functionality by first beta thus reducing the schedule estimation error. This will also allow the project team with time for final product stabilization phase where the project team will have the opportunity to improve the quality of the product.

- The proportion of testing (QA) staff and the development staff should be balanced. The testing staff should work closely with the development staff right from the beginning of the product development cycle so that the testing staff is equally knowledgeable about the product, its design and implementation to carry out an effective testing strategy. This will potentially reduce the % of total testing time spent by the developers in testing their own code but at the same time not impact the quality of the product because the testing staff is equally familiar with the product usage and its design and implementation. This

will allow the users to spend more time implementing new features for the product and in turn have higher productivity.

Having mentioned the various factors affecting a flexible product development strategy, it is important for the project managers to realize that the strategies for product development will typically differ based on the type of project being implemented. It is the project manager's responsibility to customize the software development approach to the project at hand. Some of the factors that should be considered when customizing the development strategy are:

- Is the product being developed in a segment with mature technology?

- Are there any uncertainties in the product requirements that the team has to address as the product is being developed?

- How many sub-cycles should the product development cycle be divided into?

- How early (with respect to functionality) in the project should the project team start getting customer feedback?

- How often should the various modules in the system be integrated?

- What are the various dependencies that the project has that impact the product implementation (ex: hardware availability)?

Addressing these issues will help the project team put in a process and a product development strategy that would help them develop and deliver products that benefit all stakeholders.

4.3 Limitations of the research:

There are several hypotheses that have not been validated from the data analysis. One of the reasons for this is the context or the project environment. For example, in the case of frequent synchronization we should further evaluate the project context to understand if the outcome of the data analysis is being impacted by such things as hardware availability and the dynamics associated with it. Since several projects had dependencies on hardware availability, depending on how much and when the project team adds new code the team may not be building daily. In contrast an application software project does not have such hardware dependencies and could be

synchronized frequently with daily builds. In the sample only 8 projects are application software while the remaining 18 have hardware dependencies because they are embedded software or system (device drivers) software. One way to evaluate all projects in similar context, for the projects with hardware dependencies, some additional data should be collected which would allow the research team to discount the impact of the hardware dependencies on the project schedule.

4.4 Next Steps:

The current research study was a pilot study. The academic and industry members of the research team plan to expand this study globally. The study will be performed at various organizations to further gain better insight into software product development with market and technical uncertainties and also to further validate the findings from the pilot study.

4.5 Areas for inclusion in the survey instrument (addition for future surveys):

Architecture:

- Is it modular or monolithic? This has an impact on whether the team has the flexibility to incorporate feedback easily.

Feature churn:

- What % of original features are implemented in the final product?

- What % of features in final product are new features (not in the original features list) or changes due to market and technical feedback?

Rework:

- Rework due to changes in architecture

- Rework due to technical feedback or technological changes

- Rework due to customer feedback

Group expertise:

- Expertise in the functional (domain) area

- Technical expertise

Product quality:

- Number of bugs found during various milestones or sub-cycles

- Reason for bugs

  o Because of rework due to customer feedback

  o Because of rework due to technical feedback

Customer feedback:

- How many numbers of customers were used for obtaining feedback for prototype and beta releases?

- What % of customer feedback was incorporated in the final product?

- Does the project team have a process/infrastructure to keep track of the customer feedback obtained and incorporated into the product?

Appendix-A One Way ANOVA (Analysis Of Variance) Reports

Dependent Variables: **% Functionality in First Prototype**
% Functionality in First System Integration
% Functionality in First Beta

Independent Variable (Factor): **S/W Use Type (External Use or Internal Use)**

Descriptives

		% Functionality in First Prototype			% Functionality in First System Integration			% Functionality in First Beta		
		Internal Use	External Use	Total	Internal Use	External Use	Total	Internal Use	External Use	Total
N		8	16	24	8	16	24	8	17	25
Mean		33.7500	39.2500	37.4167	70.0000	59.5625	63.0417	90.6250	92.3529	91.8000
Std. Deviation		26.6927	25.1860	25.2499	15.1186	22.6155	20.6976	7.2887	7.0882	7.0475
Std. Error		9.4373	6.2965	5.1541	5.3452	5.6539	4.2249	2.5769	1.7191	1.4095
95% Confidence Interval for Mean	Lower Bound	11.4343	25.8293	26.7546	57.3606	47.5115	54.3018	84.5315	88.7085	88.8910
	Upper Bound	56.0657	52.6707	48.0788	82.6394	71.6135	71.7815	96.7185	95.9974	94.7090
Minimum		.00	.00	.00	40.00	15.00	15.00	80.00	80.00	80.00
Maximum		80.00	90.00	90.00	90.00	100.00	100.00	100.00	100.00	100.00

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Functionality in First Prototype	.007	1	22	.933
% Functionality in First System Int.	1.200	1	22	.285
% Functionality in First Beta	.014	1	23	.908

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Functionality in First Prototype	Between Groups	(Combined)		161.333	1	161.333	.245	.626
		Linear Term	Unweighted	161.333	1	161.333	.245	.626
			Weighted	161.333	1	161.333	.245	.626
	Within Groups			14502.500	22	659.205		
	Total			14663.833	23			
% Functionality in First System Integration	Between Groups	(Combined)		581.021	1	581.021	1.379	.253
		Linear Term	Unweighted	581.021	1	581.021	1.379	.253
			Weighted	581.021	1	581.021	1.379	.253
	Within Groups			9271.937	22	421.452		
	Total			9852.958	23			
% Functionality in First Beta	Between Groups	(Combined)		16.243	1	16.243	.318	.578
		Linear Term	Unweighted	16.243	1	16.243	.318	.578
			Weighted	16.243	1	16.243	.318	.578
	Within Groups			1175.757	23	51.120		
	Total			1192.000	24			

Dependent Variables: **% Elapsed time till Last Major Requirements Change**
% Elapsed time till Last Major Functional Spec., Change
% Elapsed time till Last Major Code Addition

Independent Variable (Factor): **S/W Use Type (External Use or Internal Use)**

Descriptives

		% Elapsed time till Last Major Requirements Change			% Elapsed time till Last Major Functional Spec Change			% Elapsed time till Last Major Code addition		
		Internal Use	External Use	Total	Internal Use	External Use	Total	Internal Use	External Use	Total
N		8	17	25	8	16	24	8	18	26
Mean		63.3723	65.9439	65.1210	67.7423	69.7947	69.1105	94.8790	88.8419	90.6994
Std. Deviation		29.7896	24.2515	25.5425	18.4421	16.5601	16.8327	11.0357	12.8825	12.4510
Std. Error		10.5322	5.8819	5.1085	6.5203	4.1400	3.4360	3.9017	3.0364	2.4418
95% Confidence Interval for Mean	Lower Bound	38.4675	53.4750	54.5776	52.3243	60.9704	62.0027	85.6530	82.4355	85.6704
	Upper Bound	88.2770	78.4129	75.6644	83.1603	78.6189	76.2184	104.1051	95.2482	95.7285
Minimum		14.29	7.69	7.69	37.50	30.77	30.77	75.00	59.09	59.09
Maximum		100.00	100.00	100.00	88.89	95.65	95.65	111.11	108.33	111.11

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Elapsed time till Last Major Req. Change	.276	1	23	.605
% Elapsed time till Last Major Funcional Spec Change	.108	1	22	.746
% Elapsed time till Last Major Code Addition	.470	1	24	.500

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Elapsed time till Last Major Req. Change	Between Groups	(Combined)		35.978	1	35.978	.053	.820
		Linear Term	Unweighted	35.978	1	35.978	.053	.820
			Weighted	35.978	1	35.978	.053	.820
	Within Groups			15622.115	23	679.222		
	Total			15658.093	24			
% Elapsed time till Last Major Funcional Spec Change	Between Groups	(Combined)		22.465	1	22.465	.076	.785
		Linear Term	Unweighted	22.465	1	22.465	.076	.785
			Weighted	22.465	1	22.465	.076	.785
	Within Groups			6494.327	22	295.197		
	Total			6516.793	23			
% Elapsed time till Last Major Code Addition	Between Groups	(Combined)		201.864	1	201.864	1.319	.262
		Linear Term	Unweighted	201.864	1	201.864	1.319	.262
			Weighted	201.864	1	201.864	1.319	.262
	Within Groups			3673.808	24	153.075		
	Total			3875.672	25			

111

Dependent Variables: **Architectural Effort**
% Code Reuse

Independent Variable (Factor): **S/W Use Type (External Use or Internal Use)**

Descriptives

		Architectural Effort			% Code Reuse		
		Internal Use	External Use	Total	Internal Use	External Use	Total
N		7	18	25	8	18	26
Mean		.3124	.2896	.2960	.4625	.6656	.6031
Std. Deviation		.2261	.2998	.2767	.3215	.1817	.2460
Std. Error		8.546E-02	7.065E-02	5.533E-02	.1137	4.282E-02	4.825E-02
95% Confidence Interval for Mean	Lower Bound	.1033	.1405	.1818	.1937	.5752	.5037
	Upper Bound	.5215	.4387	.4102	.7313	.7559	.7024
Minimum		.02	.03	.02	.00	.25	.00
Maximum		.60	1.00	1.00	.85	.90	.90

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
Architectural Effort	.147	1	23	.705
% Code Reuse	6.828	1	24	.015

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
Architectural Effort	Between Groups	(Combined)		2.619E-03	1	2.619E-03	.033	.858
		Linear Term	Unweighted	2.619E-03	1	2.619E-03	.033	.858
			Weighted	2.619E-03	1	2.619E-03	.033	.858
	Within Groups			1.834	23	7.975E-02		
	Total			1.837	24			
% Code Reuse	Between Groups	(Combined)		.228	1	.228	4.266	.050
		Linear Term	Unweighted	.228	1	.228	4.266	.050
			Weighted	.228	1	.228	4.266	.050
	Within Groups			1.285	24	5.353E-02		
	Total			1.513	25			

Dependent Variables: **% Total Testing Time Developers Tested Their Code**
% Total Testing Time QA Staff Tested Code

Independent Variable (Factor): **S/W Use Type (External Use or Internal Use)**

Descriptives

		% Total Testing Time Developers tested their own Code			% Total Testing Time QA Staff tested Code		
		Internal Use	External Use	Total	Internal Use	External Use	Total
N		8	18	26	8	18	26
Mean		52.7500	53.3333	53.1538	42.250	46.667	45.308
Std. Deviation		27.8093	31.7620	30.0436	26.709	31.762	29.834
Std. Error		9.8321	7.4864	5.8920	9.443	7.486	5.851
95% Confidence Interval for Mean	Lower Bound	29.5008	37.5385	41.0190	19.921	30.872	33.258
	Upper Bound	75.9992	69.1282	65.2887	64.579	62.462	57.358
Minimum		7.0	10	7.0	.0	.0	.0
Maximum		100	100	100	93	90	93

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Developers tested their own Code	1.367	1	24	.254
% Total Testing Time QA Staff tested Code	1.667	1	24	.209

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Developers tested their own Code	Between Groups	(Combined)		1.885	1	1.885	.002	.965
		Linear Term	Unweighted	1.885	1	1.885	.002	.965
			Weighted	1.885	1	1.885	.002	.965
	Within Groups			22563.500	24	940.146		
	Total			22565.385	25			
% Total Testing Time QA Staff tested Code	Between Groups	(Combined)		108.038	1	108.038	.117	.735
		Linear Term	Unweighted	108.038	1	108.038	.117	.735
			Weighted	108.038	1	108.038	.117	.735
	Within Groups			22143.500	24	922.646		
	Total			22251.538	25			

113

Dependent Variables: **% Total Testing Time Spent in Component Testing**
% Total Testing Time Spent in Integration Testing
% Total Testing Time Spent in System Testing

Independent Variable (Factor): **S/W Use Type (External Use or Internal Use)**

Descriptives

		% Total Testing Time Spent in Component Testing			% Total Testing Time Spent in Integration Testing			% Total Testing Time Spent in System Testing		
		Internal Use	External Use	Total	Internal Use	External Use	Total	Internal Use	External Use	Total
N		8	18	26	8	18	26	8	18	26
Mean		31.250	31.389	31.346	26.875	25.000	25.577	40.6250	43.6111	42.6923
Std. Deviation		27.223	22.083	23.219	16.677	16.088	15.958	21.6197	25.4261	23.9262
Std. Error		9.625	5.205	4.554	5.896	3.792	3.130	7.6437	5.9930	4.6923
95% Confidence Interval for Mean	Lower Bound	8.491	20.407	21.968	12.933	17.000	19.131	22.5505	30.9670	33.0283
	Upper Bound	54.009	42.371	40.724	40.817	33.000	32.022	58.6995	56.2552	52.3563
Minimum		5	.0	.0	.0	.0	.0	15	10	10
Maximum		85	70	85	50	60	60	70	100	100

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Spent in Component Testing	.241	1	24	.628
% Total Testing Time Spent in Integration Testing	.001	1	24	.978
% Total Testing Time Spent in System Testing	.063	1	24	.803

ANOVA

			Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Spent in Component Testing	Between Groups	(Combined)	.107	1	.107	.000	.989
		Linear Term Unweighted	.107	1	.107	.000	.989
		Weighted	.107	1	.107	.000	.989
	Within Groups		13477.778	24	561.574		
	Total		13477.885	25			
% Total Testing Time Spent in Integration Testing	Between Groups	(Combined)	19.471	1	19.471	.074	.788
		Linear Term Unweighted	19.471	1	19.471	.074	.788
		Weighted	19.471	1	19.471	.074	.788
	Within Groups		6346.875	24	264.453		
	Total		6366.346	25			
% Total Testing Time Spent in System Testing	Between Groups	(Combined)	49.386	1	49.386	.083	.776
		Linear Term Unweighted	49.386	1	49.386	.083	.776
		Weighted	49.386	1	49.386	.083	.776
	Within Groups		14262.153	24	594.256		
	Total		14311.538	25			

Dependent Variables: **% Functionality in First Prototype**

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

PFUNCPTY

		App S/W	System S/W	Embedded S/w	Other	Total
N		8	6	4	6	24
Mean		35.6250	35.8333	25.0000	49.6667	37.4167
Std. Deviation		16.3527	35.8353	28.8675	21.5097	25.2499
Std. Error		5.7816	14.6297	14.4338	8.7813	5.1541
95% Confidence Interval for Mean	Lower Bound	21.9538	-1.7735	-20.9347	27.0936	26.7546
	Upper Bound	49.2962	73.4401	70.9347	72.2397	48.0788
Minimum		10.00	5.00	.00	25.00	.00
Maximum		50.00	90.00	50.00	80.00	90.00

Test of Homogeneity of Variances

PFUNCPTY

Levene Statistic	df1	df2	Sig.
3.089	3	20	.050

ANOVA

PFUNCPTY

			Sum of Squares	df	Mean Square	F	Sig.
Between Groups	(Combined)		1557.792	3	519.264	.792	.512
	Linear Term	Unweighted	321.918	1	321.918	.491	.491
		Weighted	374.083	1	374.083	.571	.459
		Deviation	1183.708	2	591.854	.903	.421
Within Groups			13106.042	20	655.302		
Total			14663.833	23			

Dependent Variables: **% Functionality in First System Integration**

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

PFUNCSI

		App S/W	System S/W	Embedded S/W	Others	Total
N		8	6	4	6	24
Mean		64.3750	60.0000	48.7500	73.8333	63.0417
Std. Deviation		11.1604	30.3315	23.9357	14.6754	20.6976
Std. Error		3.9458	12.3828	11.9678	5.9912	4.2249
95% Confidence Interval for Mean	Lower Bound	55.0447	28.1690	10.6630	58.4325	54.3018
	Upper Bound	73.7053	91.8310	86.8370	89.2342	71.7815
Minimum		50.00	20.00	15.00	50.00	15.00
Maximum		80.00	100.00	70.00	90.00	100.00

Test of Homogeneity of Variances

PFUNCSI

Levene Statistic	df1	df2	Sig.
2.126	3	20	.129

ANOVA

PFUNCSI

			Sum of Squares	df	Mean Square	F	Sig.
Between Groups	(Combined)		1585.500	3	528.500	1.279	.309
	Linear Term	Unweighted	96.416	1	96.416	.233	.634
		Weighted	114.083	1	114.083	.276	.605
		Deviation	1471.417	2	735.708	1.780	.194
Within Groups			8267.458	20	413.373		
Total			9852.958	23			

Dependent Variables: **% Functionality in First Beta**

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

PFUNCBTA

		App S/W	System S/W	Embedded S/W	Others	Total
N		8	6	5	6	25
Mean		89.6250	93.3333	92.0000	93.0000	91.8000
Std. Deviation		7.1502	7.5277	7.5829	7.2111	7.0475
Std. Error		2.5280	3.0732	3.3912	2.9439	1.4095
95% Confidence Interval for Mean	Lower Bound	83.6473	85.4335	82.5846	85.4324	88.8910
	Upper Bound	95.6027	101.2332	101.4154	100.5676	94.7090
Minimum		80.00	80.00	80.00	80.00	80.00
Maximum		100.00	100.00	100.00	100.00	100.00

Test of Homogeneity of Variances

PFUNCBTA

Levene Statistic	df1	df2	Sig.
.004	3	21	1.000

ANOVA

PFUNCBTA

			Sum of Squares	df	Mean Square	F	Sig.
Between Groups	(Combined)		60.792	3	20.264	.376	.771
	Linear Term	Unweighted	25.836	1	25.836	.480	.496
		Weighted	31.867	1	31.867	.592	.450
		Deviation	28.924	2	14.462	.268	.767
Within Groups			1131.208	21	53.867		
Total			1192.000	24			

Dependent Variables: **% Elapsed Time till Last Major Requirements Change**
% Elapsed Time till Last Major Functional Spec., Change
Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

		% Elapsed Time At Last Major Req Change					% Elapsed Time At Last Major Func. Spec Change				
		App S/W	System S/W	Embed ded S/W	Other	Total	App S/W	System S/W	Embed ded S/W	Other	Total
N		8	6	5	6	25	7	6	5	6	24
Mean		72.4077	60.0783	55.8937	68.1375	65.1210	67.6407	67.6443	71.2271	70.5278	69.1105
Std. Deviation		12.9774	38.1791	19.9150	30.1560	25.5425	16.2126	22.6128	12.5068	18.4111	16.8327
Std. Error		4.5882	15.5866	8.9062	12.3111	5.1085	6.1278	9.2316	5.5932	7.5163	3.4360
95% Confidence Interval for Mean	Lower Bound	61.5583	20.0118	31.1660	36.4907	54.5776	52.6465	43.9136	55.6978	51.2065	62.0027
	Upper Bound	83.2570	100.1449	80.6214	99.7842	75.6644	82.6349	91.3750	86.7564	89.8491	76.2184
Minimum		58.33	7.69	25.00	14.29	7.69	37.50	30.77	58.33	42.86	30.77
Maximum		95.83	100.00	73.91	100.00	100.00	83.33	95.65	88.89	91.30	95.65

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Elapsed Time At Last Major Req Change	3.933	3	21	.023
% Elapsed Time At Last Major Func Spec Change	.338	3	20	.798

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Elapsed Time At Last Major Req Change	Between Groups	(Combined)		1057.642	3	352.547	.507	.682
		Linear Term	Unweighted	96.548	1	96.548	.139	.713
			Weighted	137.904	1	137.904	.198	.661
			Deviation	919.738	2	459.869	.661	.527
	Within Groups			14600.451	21	695.260		
	Total			15658.093	24			
% Elapsed Time At Last Major Func Spec Change	Between Groups	(Combined)		62.472	3	20.824	.065	.978
		Linear Term	Unweighted	47.557	1	47.557	.147	.705
			Weighted	45.073	1	45.073	.140	.713
			Deviation	17.400	2	8.700	.027	.973
	Within Groups			6454.320	20	322.716		
	Total			6516.793	23			

Dependent Variables: **% Elapsed Time till Last Major Code Addition**

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

PLSTCDAD

		App S/W	System S/W	Embedded S/W	Others	Total
N		8	6	5	7	26
Mean		90.0450	86.8785	94.7826	91.8060	90.6994
Std. Deviation		10.9038	11.4170	11.6664	16.6549	12.4510
Std. Error		3.8551	4.6610	5.2174	6.2949	2.4418
95% Confidence Interval for Mean	Lower Bound	80.9292	74.8971	80.2968	76.4028	85.6704
	Upper Bound	99.1608	98.8599	109.2684	107.2092	95.7285
Minimum		75.00	73.33	73.91	59.09	59.09
Maximum		108.33	104.35	100.00	111.11	111.11

Test of Homogeneity of Variances

PLSTCDAD

Levene Statistic	df1	df2	Sig.
.179	3	22	.910

ANOVA

PLSTCDAD

			Sum of Squares	df	Mean Square	F	Sig.
Between Groups	(Combined)		182.957	3	60.986	.363	.780
	Linear Term	Unweighted	62.613	1	62.613	.373	.548
		Weighted	46.573	1	46.573	.277	.604
		Deviation	136.384	2	68.192	.406	.671
Within Groups			3692.715	22	167.851		
Total			3875.672	25			

Dependent Variables: **Architectural Effort**
                     % Code Reuse

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

	Architectural Effort					% Code Reuse				
	App S/W	System S/W	Embedded S/W	Others	Total	App S/W	System S/W	Embedded S/W	Others	Total
N	7	6	5	7	25	8	6	5	7	26
Mean	.1475	.3803	.2655	.3939	.2960	.6625	.5500	.5360	.6286	.6031
Std. Deviation	9.358E-02	.3386	.4126	.2087	.2767	.2167	7.746E-02	.3510	.3134	.2460
Std. Error	3.537E-02	.1382	.1845	7.889E-02	5.533E-02	7.662E-02	3.162E-02	.1570	.1185	4.825E-02
95% Confidence Lower Bound	6.092E-02	2.497E-02	-.2468	.2009	.1818	.4813	.4687	.1001	.3387	.5037
Interval for Mean Upper Bound	.2340	.7356	.7778	.5870	.4102	.8437	.6313	.9719	.9184	.7024
Minimum	.03	.10	.02	.09	.02	.20	.45	.10	.00	.00
Maximum	.25	1.00	1.00	.67	1.00	.85	.65	.88	.90	.90

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
Architectural Effort	2.114	3	21	.129
% Code Reuse	2.524	3	22	.084

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
Architectural Effort	Between Groups	(Combined)		.269	3	8.962E-02	1.200	.334
		Linear Term	Unweighted	.133	1	.133	1.778	.197
			Weighted	.149	1	.149	1.996	.172
			Deviation	.120	2	5.993E-02	.803	.461
	Within Groups			1.568	21	7.467E-02		
	Total			1.837	24			
% Code Reuse	Between Groups	(Combined)		7.220E-02	3	2.407E-02	.367	.777
		Linear Term	Unweighted	4.827E-03	1	4.827E-03	.074	.789
			Weighted	5.667E-03	1	5.667E-03	.087	.771
			Deviation	6.653E-02	2	3.327E-02	.508	.609
	Within Groups			1.441	22	6.550E-02		
	Total			1.513	25			

Dependent Variables: **% Total Testing Time Developers Tested Their Code**
% Total Testing Time QA Staff Tested Code

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others –**
combination of application, system and embedded software)

Descriptives

		% Total Testing Time Developers Test Their Code					% Total Testing Time QA Staff Test Code				
		App S/W	System S/W	Embed ded S/W	Others	Total	App S/W	System S/W	Embed ded S/W	Others	Total
N		8	6	5	7	26	8	6	5	7	26
Mean		63.1250	62.0000	58.0000	30.7143	53.1538	36.875	38.000	42.000	63.571	45.308
Std. Deviation		29.3911	39.2683	10.9545	23.8797	30.0436	29.391	39.268	10.954	28.094	29.834
Std. Error		10.3913	16.0312	4.8990	9.0257	5.8920	10.391	16.031	4.899	10.619	5.851
95% Confidence	Lower Bound	38.5534	20.7904	44.3983	8.6293	41.0190	12.303	-3.210	28.398	37.589	33.258
Interval for Mean	Upper Bound	87.6966	103.2096	71.6017	52.7993	65.2887	61.447	79.210	55.602	89.554	57.358
Minimum		15	7.0	50	10	7.0	.0	.0	30	25	.0
Maximum		100	100	70	75	100	85	93	50	90	93

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Developers Test Their Code	2.436	3	22	.092
% Total Testing Time QA Staff Test Code	2.678	3	22	.072

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Developers Test Their Code	Between Groups	(Combined)		4907.081	3	1635.694	2.038	.138
		Linear Term	Unweighted	3689.788	1	3689.788	4.597	.043
			Weighted	3760.299	1	3760.299	4.685	.042
			Deviation	1146.782	2	573.391	.714	.501
	Within Groups			17658.304	22	802.650		
	Total			22565.385	25			
% Total Testing Time QA Staff Test Code	Between Groups	(Combined)		3278.949	3	1092.983	1.267	.310
		Linear Term	Unweighted	2545.927	1	2545.927	2.952	.100
			Weighted	2586.601	1	2586.601	2.999	.097
			Deviation	692.348	2	346.174	.401	.674
	Within Groups			18972.589	22	862.390		
	Total			22251.538	25			

Dependent Variables: **% Total Testing Time Spent in Component Testing**
% Total Testing Time Spent in Integration Testing

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

		% Total Testing Time Spent in Component Testing					% Total Testing Time Spent in Integration Testing				
		App S/W	System S/W	Embedded S/W	Others	Total	App S/W	System S/W	Embedded S/W	Others	Total
N		8	6	5	7	26	8	6	5	7	26
Mean		35.000	42.500	23.000	23.571	31.346	29.375	18.333	26.000	27.143	25.577
Std. Deviation		22.520	31.265	10.954	22.120	23.219	15.222	16.021	8.216	21.381	15.958
Std. Error		7.962	12.764	4.899	8.360	4.554	5.382	6.540	3.674	8.081	3.130
95% Confidence Interval for Mean	Lower Bound	16.173	9.689	9.398	3.114	21.968	16.649	1.521	15.799	7.369	19.131
	Upper Bound	53.827	75.311	36.602	44.029	40.724	42.101	35.146	36.201	46.917	32.022
Minimum		5	10	10	.0	.0	10	.0	20	.0	.0
Maximum		70	85	40	70	85	50	40	40	60	60

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Spent in Component Testing	1.896	3	22	.160
% Total Testing Time Spent in Integration Testing	1.719	3	22	.192

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Spent in Component Testing	Between Groups	(Combined)		1624.670	3	541.557	1.005	.409
		Linear Term	Unweighted	1041.594	1	1041.594	1.933	.178
			Weighted	889.525	1	889.525	1.651	.212
			Deviation	735.146	2	367.573	.682	.516
	Within Groups			11853.214	22	538.782		
	Total			13477.885	25			
% Total Testing Time Spent in Integration Testing	Between Groups	(Combined)		448.281	3	149.427	.555	.650
		Linear Term	Unweighted	.339	1	.339	.001	.972
			Weighted	1.108	1	1.108	.004	.949
			Deviation	447.173	2	223.586	.831	.449
	Within Groups			5918.065	22	269.003		
	Total			6366.346	25			

Dependent Variables: % **Total Testing Time Spent in System Testing**

Independent Variable (Factor): **Project Type (Application, System, Embedded, Others – combination of application, system and embedded software)**

Descriptives

PSYSTST

		App S/W	System S/W	Embedded S/W	Others	Total
N		8	6	5	7	26
Mean		35.6250	39.1667	51.0000	47.8571	42.6923
Std. Deviation		22.9031	24.5798	11.4018	31.8665	23.9262
Std. Error		8.0975	10.0347	5.0990	12.0444	4.6923
95% Confidence Interval for Mean	Lower Bound	16.4775	13.3717	36.8429	18.3855	33.0283
	Upper Bound	54.7725	64.9616	65.1571	77.3288	52.3563
Minimum		10	10	40	10	10
Maximum		70	70	65	100	100

Test of Homogeneity of Variances

PSYSTST

Levene Statistic	df1	df2	Sig.
1.839	3	22	.170

ANOVA

PSYSTST

			Sum of Squares	df	Mean Square	F	Sig.
Between Groups	(Combined)		1005.973	3	335.324	.554	.651
	Linear Term	Unweighted	847.971	1	847.971	1.402	.249
		Weighted	798.734	1	798.734	1.321	.263
		Deviation	207.239	2	103.619	.171	.844
Within Groups			13305.565	22	604.798		
Total			14311.538	25			

Dependent Variables: **% Functionality in First Prototype**
% Functionality in First System Integration
% Functionality in First Beta

Independent Variable (Factor): **New Project or Product Extension**

Descriptives

		% Functionality in First Prototype			% Functionality in First System Integration			% Functionality in First Beta		
		Prd Extension	New Product	Total	Prd Extension	New Product	Total	Prd Extension	New Product	Total
N		7	17	24	7	17	24	8	17	25
Mean		37.1429	37.5294	37.4167	55.8571	66.0000	63.0417	93.7500	90.8824	91.8000
Std. Deviation		23.6039	26.5991	25.2499	24.0862	19.1409	20.6976	3.4949	8.1462	7.0475
Std. Error		8.9214	6.4512	5.1541	9.1037	4.6424	4.2249	1.2356	1.9757	1.4095
95% Confidence Interval for Mean	Lower Bound	15.3129	23.8534	26.7546	33.5812	56.1586	54.3018	90.8282	86.6940	88.8910
	Upper Bound	58.9728	51.2054	48.0788	78.1331	75.8414	71.7815	96.6718	95.0707	94.7090
Minimum		10.00	.00	.00	15.00	20.00	15.00	87.00	80.00	80.00
Maximum		70.00	90.00	90.00	86.00	100.00	100.00	98.00	100.00	100.00

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Functionality in First Prototype	.014	1	22	.905
% Functionality in First System Integration	.376	1	22	.546
% Functionality in First Beta	7.274	1	23	.013

ANOVA

			Sum of Squares	df	Mean Square	F	Sig.
% Functionality in First Prototype	Between Groups	(Combined)	.741	1	.741	.001	.974
		Linear Term Unweighted	.741	1	.741	.001	.974
		Weighted	.741	1	.741	.001	.974
	Within Groups		14663.092	22	666.504		
	Total		14663.833	23			
% Functionality in First System Integration	Between Groups	(Combined)	510.101	1	510.101	1.201	.285
		Linear Term Unweighted	510.101	1	510.101	1.201	.285
		Weighted	510.101	1	510.101	1.201	.285
	Within Groups		9342.857	22	424.675		
	Total		9852.958	23			
% Functionality in First Beta	Between Groups	(Combined)	44.735	1	44.735	.897	.353
		Linear Term Unweighted	44.735	1	44.735	.897	.353
		Weighted	44.735	1	44.735	.897	.353
	Within Groups		1147.265	23	49.881		
	Total		1192.000	24			

Dependent Variables: **% Elapsed time till Last Major Requirements Change**
% Elapsed time till Last Major Functional Spec., Change
% Elapsed time till Last Major Code Addition

Independent Variable (Factor): **New Project or Product Extension**

Descriptives

		% Elapsed Time at Last Major Req Change			% Elapsed Time at Last Major Func Spec Change			% Elapsed Time at Last Major Code Addition		
		Prd Extension	New Product	Total	Prd Extension	New Product	Total	Prd Extension	New Product	Total
N		8	17	25	8	16	24	8	18	26
Mean		63.7877	65.7484	65.1210	64.2391	71.5462	69.1105	89.7600	91.1170	90.6994
Std. Deviation		17.5060	29.0387	25.5425	14.4930	17.8170	16.8327	12.3643	12.8231	12.4510
Std. Error		6.1893	7.0429	5.1085	5.1241	4.4543	3.4360	4.3714	3.0224	2.4418
95% Confidence Interval for Mean	Lower Bound	49.1523	50.8181	54.5776	52.1227	62.0522	62.0027	79.4231	84.7402	85.6704
	Upper Bound	78.4232	80.6787	75.6644	76.3556	81.0403	76.2184	100.0968	97.4938	95.7285
Minimum		27.78	7.69	7.69	37.50	30.77	30.77	73.91	59.09	59.09
Maximum		85.00	100.00	100.00	88.89	95.65	95.65	108.33	111.11	111.11

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Elapsed Time at Last Major Req Change	2.709	1	23	.113
% Elapsed Time at Last Major Func Spec Change	1.013	1	22	.325
% Elapsed Time at Last Major Code Addition	.004	1	24	.952

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
% Elapsed Time at Last Major Req Change	Between Groups	(Combined)		20.912	1	20.912	.031	.862
		Linear Term	Unweighted	20.912	1	20.912	.031	.862
			Weighted	20.912	1	20.912	.031	.862
	Within Groups			15637.181	23	679.877		
	Total			15658.093	24			
% Elapsed Time at Last Major Func Spec Change	Between Groups	(Combined)		284.767	1	284.767	1.005	.327
		Linear Term	Unweighted	284.767	1	284.767	1.005	.327
			Weighted	284.767	1	284.767	1.005	.327
	Within Groups			6232.025	22	283.274		
	Total			6516.793	23			
% Elapsed Time at Last Major Code Addition	Between Groups	(Combined)		10.199	1	10.199	.063	.803
		Linear Term	Unweighted	10.199	1	10.199	.063	.803
			Weighted	10.199	1	10.199	.063	.803
	Within Groups			3865.473	24	161.061		
	Total			3875.672	25			

Dependent Variables: **Architectural Effort**
% Code Reuse

Independent Variable (Factor): **New Project or Product Extension**

Descriptives

		Architectural Effort			% Code Reuse		
		Prd Extension	New Product	Total	Prd Extension	New Product	Total
N		8	17	25	8	18	26
Mean		.2268	.3285	.2960	.7600	.5333	.6031
Std. Deviation		.1742	.3131	.2767	.1093	.2595	.2460
Std. Error		6.160E-02	7.593E-02	5.533E-02	3.864E-02	6.117E-02	4.825E-02
95% Confidence Interval for Mean	Lower Bound	8.118E-02	.1675	.1818	.6686	.4043	.5037
	Upper Bound	.3725	.4895	.4102	.8514	.6624	.7024
Minimum		.07	.02	.02	.60	.00	.00
Maximum		.50	1.00	1.00	.88	.90	.90

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
Architectural Effort	2.319	1	23	.141
% Code Reuse	3.229	1	24	.085

ANOVA

				Sum of Squares	df	Mean Square	F	Sig.
Architectural Effort	Between Groups	(Combined)		5.622E-02	1	5.622E-02	.726	.403
		Linear Term	Unweighted	5.622E-02	1	5.622E-02	.726	.403
			Weighted	5.622E-02	1	5.622E-02	.726	.403
	Within Groups			1.781	23	7.742E-02		
	Total			1.837	24			
% Code Reuse	Between Groups	(Combined)		.285	1	.285	5.559	.027
		Linear Term	Unweighted	.285	1	.285	5.559	.027
			Weighted	.285	1	.285	5.559	.027
	Within Groups			1.229	24	5.119E-02		
	Total			1.513	25			

126

Dependent Variables: **% Total Testing Time Developers Tested Their Code**
% Total Testing Time QA Staff Tested Code

Independent Variable (Factor): **New Project or Product Extension**

Descriptives

		% Total Testing Time Developers Tested Their Code			% Total Testing Time QA Staff Tested Code		
		Prd Extension	New Product	Total	Prd Extension	New Product	Total
N		8	18	26	8	18	26
Mean		45.2500	56.6667	53.1538	54.750	41.111	45.308
Std. Deviation		32.7185	29.0537	30.0436	32.718	28.417	29.834
Std. Error		11.5677	6.8480	5.8920	11.568	6.698	5.851
95% Confidence Interval for Mean	Lower Bound	17.8967	42.2186	41.0190	27.397	26.980	33.258
	Upper Bound	72.6033	71.1147	65.2887	82.103	55.242	57.358
Minimum		7.0	10	7.0	.0	.0	.0
Maximum		100	100	100	93	90	93

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Developers Tested Their Code	.052	1	24	.822
% Total Testing Time QA Staff Tested Code	.138	1	24	.714

ANOVA

			Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Developers Tested Their Code	Between Groups	(Combined)	721.885	1	721.885	.793	.382
		Linear Term Unweighted	721.885	1	721.885	.793	.382
		Weighted	721.885	1	721.885	.793	.382
	Within Groups		21843.500	24	910.146		
	Total		22565.385	25			
% Total Testing Time QA Staff Tested Code	Between Groups	(Combined)	1030.261	1	1030.261	1.165	.291
		Linear Term Unweighted	1030.261	1	1030.261	1.165	.291
		Weighted	1030.261	1	1030.261	1.165	.291
	Within Groups		21221.278	24	884.220		
	Total		22251.538	25			

Dependent Variables: **% Total Testing Time Spent in Component Testing**
% Total Testing Time Spent in Integration Testing
% Total Testing Time Spent in System Testing

Independent Variable (Factor): **New Project or Product Extension**

Descriptives

		% Total Testing Time Spent in Component Testing			% Total Testing Time Spent in Integration Testing			% Total Testing Time Spent in System Testing		
		Prd Extension	New Product	Total	Prd Extension	New Product	Total	Prd Extension	New Product	Total
N		8	18	26	8	18	26	8	18	26
Mean		30.625	31.667	31.346	34.375	21.667	25.577	35.0000	46.1111	42.6923
Std. Deviation		22.903	24.010	23.219	18.792	13.284	15.958	17.9284	25.8705	23.9262
Std. Error		8.097	5.659	4.554	6.644	3.131	3.130	6.3387	6.0977	4.6923
95% Confidence Interval for Mean	Lower Bound	11.478	19.727	21.968	18.665	15.061	19.131	20.0115	33.2460	33.0283
	Upper Bound	49.772	43.606	40.724	50.085	28.273	32.022	49.9885	58.9762	52.3563
Minimum		10	.0	.0	.0	.0	.0	15	10	10
Maximum		85	70	85	60	50	60	70	100	100

Test of Homogeneity of Variances

	Levene Statistic	df1	df2	Sig.
% Total Testing Time Spent in Component Testing	1.597	1	24	.218
% Total Testing Time Spent in Integration Testing	1.793	1	24	.193
% Total Testing Time Spent in System Testing	2.146	1	24	.156

ANOVA

			Sum of Squares	df	Mean Square	F	Sig.
% Total Testing Time Spent in Component Testing	Between Groups	(Combined)	6.010	1	6.010	.011	.918
		Linear Term Unweighted	6.010	1	6.010	.011	.918
		Weighted	6.010	1	6.010	.011	.918
	Within Groups		13471.875	24	561.328		
	Total		13477.885	25			
% Total Testing Time Spent in Integration Testing	Between Groups	(Combined)	894.471	1	894.471	3.923	.059
		Linear Term Unweighted	894.471	1	894.471	3.923	.059
		Weighted	894.471	1	894.471	3.923	.059
	Within Groups		5471.875	24	227.995		
	Total		6366.346	25			
% Total Testing Time Spent in System Testing	Between Groups	(Combined)	683.761	1	683.761	1.204	.283
		Linear Term Unweighted	683.761	1	683.761	1.204	.283
		Weighted	683.761	1	683.761	1.204	.283
	Within Groups		13627.778	24	567.824		
	Total		14311.538	25			

SOFTWARE DEVELOPMENT PROCESS STUDY

by

MIT Sloan School of Management

Katz Graduate School of Business, University of Pittsburgh

Harvard Business School

This survey has two fundamental objectives for the current study of software development process:

- To identify and document best-known methods for increasing performance in software development, such as speed, flexibility, and quality.
- To identify and understand what types of approaches to software development work best in different types of projects.

This research, an industry-academia cooperative effort, is sponsored by HP's Product Generation Solutions team with the goal of understanding how to keep HP and Agilent's product generation processes ahead of the curve in the Internet age. Survey results will be published in academic and industry publications, including a master's thesis. HP and Agilent will get an early look at the results. You are being asked to provide information from a specific software development project.

All project-specific identifying data will be kept confidential by the researchers; only summary results and project data that cannot be matched to a specific project will be included in the publications.

Contact Information:

HP/Agilent Contacts:

- Bill Crandall, (650) 857-6543 or telnet 857-6543, bill_crandall@hp.com

- Guy Cox, (650) 857-8980 or telnet 857-8980, guy_cox@hp.com

Academic Contacts:

- Prof. Michael Cusumano (MIT Sloan School of Management), cusumano@mit.edu
- Prof. Chris F. Kemerer (Katz Graduate School of Business, University of Pittsburgh), ckemerer@katz.pitt.edu
- Prof. Alan MacCormack (Harvard Business School), amaccormack@hbs.edu

Student Contact: (responsible for maintaining the research questionnaire and data collection)

- Sharma Upadhyayula, supadhy@mit.edu

Some reference material that would be helpful in filling out the Survey:

- Project data sheets
- Project schedules
- Project resource plans

- Project results
- Project checkpoint presentations

Name of the project you are describing in this questionnaire (including version number, if any):

Today's date:

Name of the person filling out this form:

Your role on the project (e.g., project manager, lead architect, developer, etc.):

Your email address (in the event that there are questions):

Your phone number (in the event that there are questions):

If you wish to be provided with a summary of the results of this research, please indicate that here (select one): ⊙ Yes ○ No

Part 1

1.1 Project Description and Environment:

In this section you will be answering questions about the main software deliverable from the project.

- *A 'project' here is the entire effort devoted toward delivering a specific software deliverable where the activity was separately managed and tracked from other software deliverables.*
- *The software deliverable from a project might be a product or a service. In particular, it might be a new release of a previously existing piece of software.*

Throughout this survey the focus will generally be on the software <u>project</u>, but some questions will ask about the software deliverable, the <u>product or service</u>. When questions ask about the product or service, they are referring only to the version created by this project.

1.1.1 Into what category (type and primary customer) does the deliverable fall? (check one if possible. If multiple categories, please check the primary category only)

For example: HP Unix is systems software sold primarily to enterprises. Microsoft Office is applications software sold both to enterprises and individuals. Yahoo's search engine software for its web site is applications software primarily for customer service (i.e. it is not primarily sold to enterprises or individuals). Control software for HP printers is embedded software sold both to enterprises and individuals. A Cisco router software project would be embedded software sold primarily to enterprises.

	Sold Primarily	Sold Primarily	Primarily For In-House
Systems Software	☐	☐	☐
Applications Software	☐	☐	☐
Embedded Software	☐	☐	☐

1.1.2 Outline briefly the main functions of the software:

1.2 Size of the Project:

Budget and Schedule

1.2.1 What was the software development *budget* for the project in *dollars* (please give budget in $M)? ☐ $M

1.2.2 What was the software development *budget* for the project in *effort* in Person-years? ☐

1.2.3 What was the original software development *schedule* (duration in calendar months)? ☐

Software

1.2.4 What programming language (e.g. C, C++, HTML, Assembly) was the software primarily written in? ☐

1.2.5 Please estimate the size of the delivered software in source lines of code: ☐

1.2.6 Does this figure include comments? (select one) ☐ Yes ☑ No

1.2.7 If "yes", estimate percentage of comments here: ☐ %

1.2.8 What was the origin of the software code in the finished release according to the following categories?

Category	Percentage of Code
Off-the-shelf code retained from the previous version of this product	☐
Off-the-shelf code from other sources	☐
New code developed for this product in other project team(s) (e.g. core code, components)	☐
New code developed for this product in this project team	☐
TOTAL	**100%**

1.3 Project Team Roles Composition:

1.3.1 What was the structure of the software development team?

Position	Average Staff (number of people)	Peak Staff (number of people)	Total staff Resources (person-years)
Project Management (includes project managers and directors, but not team or technical leads)			
Architecture and Design			
Development/Programming			
Testing (QA/QE) & Integration			
Project Support (e.g., configuration management, documentation, etc.)			
Other:			
TOTAL			

1.4 Design and Development Process:

Specifications – Architecture, Functional, and Detailed Design

1.4.1 Did the team have an *architectural specification* (i.e., a document that provided a high level description of the subsystems and interfaces of the eventual product or service)? Select one:

     ⊙ Yes
     ○ No

    1.4.2 If "yes," what percentage of the architectural specification was completed before the team started coding?

    1.4.3 If "yes," and if the architectural specification was adopted from a previous project, what percentage of the architectural specification was modified before the team started coding?

    1.4.4 How long were the architectural specifications for this system or product in terms of pages?

1.4.5 Did the team write a *functional specification* (i.e., a document that described how features worked but not the underlying structure of the code or modules)? Select one:

     ⊙ Yes
     ○ No

1.4.6 If "yes," what percentage of the functional specification was completed before the team started coding?

1.4.7 How long were the functional specifications for this system or product in terms of pages?

1.4.8 Did the team write a _detailed design specification_ (i.e. a document that provides the structure of the modules and an outline of algorithms where needed)? Select one: ⊙ Yes ○ No

1.4.9 If "yes," what percentage of the detailed design specification was completed before the team started coding?

1.4.10 How long were the detailed design specifications for this system or product in terms of pages?

Development

1.4.11 Were there any design reviews done?   ⊙ Yes ○ No

1.4.12 If yes, please note approximate dates: (mm/yy)

Builds

1.4.13 During the development phase, how frequently was the system "built" (i.e., how often were design changes, including bug fixes, integrated into the code base and then recompiled, e.g. daily, twice per week, weekly, twice per month, once per month, once at end of development phase)?
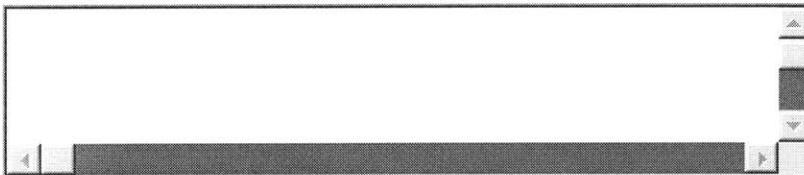
1.4.14 How many people typically review another person's code before it can be checked into the system build?   People

1.4.15 Was any type of integration or regression test (as opposed to a simple compile and link test) run each time developers checked changed or new code into the project build?   ⊙ Yes ○ No

1.4.16 If yes, how long did the integration test usually take to run?   Hours

1.4.17 When the product was "built," how long did it take (in hours) to get feedback on the performance of the system using the most comprehensive set of system tests assembled during the project (whether these were manual or   Days OR   Hours

automated)?

1.5 Testing and Debugging Process:

1.5.1

Responsibility for Testing	Percentage of Total Testing Time
Developers tested their own code	
Separate QA or testing staff tested code	
TOTAL	100%

1.5.2 What was the **relative emphasis** on different types of testing during the project?

Focus of Testing	Percentage of Total Testing Time
Component testing (testing individual features or blocks of code)	
Integration testing (testing several blocks of code integrated together)	
System testing (testing the complete product)	
TOTAL	100%

1.5.3 Approximately what percentage of the test cases run on the product or system were automated? [   ] %

1.6 Interaction with Customers (A customer can be internal or external):

1.6.1 Estimate the percentage of the final product functionality which existed in the design at the following project events (assume the functionality in the design is 0% at the start of the project

and 100% at the time the product is launched):

Project Event	Percentage of Final Product Functionality
The first prototype shown to customers (even if only a mock-up)	
The first system integration (even if modules only partially complete)	
The first beta version (the initial full version for external customer use)	

Part 2

Please consider the following definitions:

Requirements Planning: Phase that outlines the project goals

Architectural and Functional Design: Phase that outlines the high-level system design and a functional description of the product

Detailed Design and Development: Phase that covers detailed design, coding, unit-level testing, and debugging.

Integration and System Testing: Phase that integrates and stabilizes modules, and tests the performance of the whole system.

Development Sub-Cycle: A typical software development cycle consists of "Design", "Develop", "Build", "Test" and "Release" activities. Some projects might not have all the five activities for each sub-cycle.

Please consider the following general model of a software development project -- note that your organization may not track all these steps, or may use slightly different terminology.

2.1.1 Please fill in the dates for the following events on your project in the format MM/YY.

Activity Number	Activity Description	Activity Date
1	Project start date	
2	Requirements specification document first available on	
3	Last major change to requirements specification	
4	Architecture design start date	
5	Functional design start date	
6	Last major change to the functional design specification (e.g. feature complete milestone)	
7	Development start date	
8	Last addition of new code, excluding bug fixes (e.g. code complete)	
9	First system integration test date	
10	Final system test date	
11	System launch date	

2.1.2 When was the first prototype of any sort shown to customers (e.g. a mock-up of the user interface)?

2.1.3 How many beta versions, if any, did you release to customers?

2.1.4 If you released a beta version, when was the first beta version released to customers?

2.1.5 For projects that included hardware development, at which point during the project did the hardware platform for which the software was designed become available and stable?

2.1.6 Did you divide the Development phase of the project into separate development sub-cycles that built and tested a subset of the final product's functionality?   ⦿ Yes ⦾ No

2.1.7 If "yes," how many separate development sub-cycles were there on this project?

2.1.8 If "yes", after which sub-cycle was the first alpha released?

2.1.9 If "yes", after which sub-cycle was the first beta released?

2.1.10 How were the size and deliverables for each development sub-cycle determined?
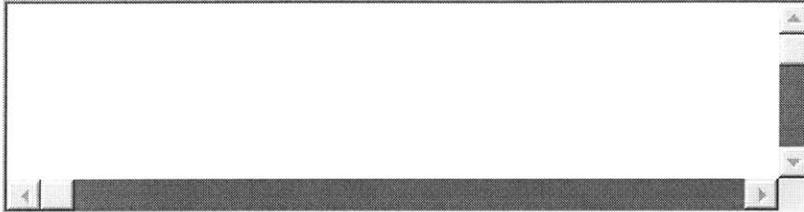
2.2 Project Performance Metrics:

Financial Performance

2.2.1 If you sold your product, please estimate the total dollar revenues that the product generated in the first 12 months after shipment of the final release, including extra charges for design changes, if applicable. If your product included charges for hardware, please estimate *revenues solely attributable to the software part* of the product (for example, as tracked by your internal accounting procedures).

Actual market revenues: **OR**

If you sold your product in-house at a special transfer price, please estimate what the revenues generated from the product would have been using market prices.

Estimated market revenues:

2.2.2 Are the product revenues from: ⊙ Hardware + Software **OR** ○ Software Only

Market Performance

2.2.3 If you sold the results of your project in the market, please estimate the increase/decrease in market or user share of your product in the first 12 months after release:

(Note: if <12 months have passed, note the number of months here: months)

Schedule Performance

2.2.4 What was the (A) actual duration of the project: months

2.2.5 What was the (B) schedule for the project estimated at the end of the requirements planning phase [____] months

Budget Performance

2.2.6 Please provide an answer to <u>either</u> 1 or 2 below.

2.2.7 What was the (A) actual outcome expenditure for the project (in $million): [____]

2.2.8 What was the (B) budget (in $million) for the project established during the up-front planning phase: [____]

Quality

Software quality is often thought of as the relative absence of defects, or 'bugs'. Most organizations have mechanisms in place for testers and customers to report bugs, (e.g. 'software problem reports'). The following questions ask about these bugs in terms of their volume and timing.

2.2.9 Estimate the approximate peak (maximum) number of 'bugs open' (i.e., bugs that were reported but not yet fixed) and the approximate average number of 'bugs open' during the following periods.

Period	Peak Bugs Open	Average Bugs Open
Between the start of coding and the first system integration	[____]	[____]
If there was a beta release, please answer 1 and 2 below:		
1 Between the first system integration and the first beta release	[____]	[____]
2 Between the first beta release and the system launch	[____]	[____]
If there was <u>no</u> beta release, then please answer 3 below:		
3 Between the first system integration and the system launch	[____]	[____]

If there was a beta release, please answer the following question:

2.2.10 Estimate the proportion of all bugs discovered **after** the first beta version that came from the following sources?

Source	Percentage of Bugs Found After First Beta
Bugs found by development engineers themselves	
Bugs found by QA and test engineers during testing activities	
Bugs found by customers using the beta release	
TOTAL	100%

Follow on questions used to gather information on % original features implemented and project performance perception ratings:

1. What percentage of the features that were implemented in the final product were contained in the original functional specification?_____

2. Please indicate the extent to which you perceive the project met expectations in terms of:

- Schedule and budget performance _____

- Customer satisfaction with the end-product _____

- Financial returns from the project as a whole _____

(Answer using a 5-point scale, where 1= Significantly below, 2=Below, 3=Met expectations, 4=Above, 5=Significantly above)

3 (a). Estimate the number of bugs reported by customers (end users) in the first 12 months after the system was launched: _____

Note: If less than 12 months have passed since the system launch, please note the number of months here: _____

References

1. Michael A. Cusumano and Richard W. Selby, <u>Microsoft Secrets</u>, Free Press 1995

2. Alan MacCormack, Roberto Verganti, and Marco Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process", *Harvard Business School Working paper 99-118*, 1999

3. Alan MacCormack, Roberto Verganti, Marco Iansiti, and Bo Kemp, "Product Development Performance In Internet Software", *Harvard Business School*, September 1997

4. Michael A. Cusumano and David B. Yoffe, <u>Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft</u>, Free Press, 1998

5. Alan MacCormack and Roberto Verganti, "Managing the Sources Of Uncertainty: Matching Process and Context in New Product Development", *EIASM Final draft*

6. Nancy Staudenmayer and Michael A. Cusumano, "Alternative Designs for Product Component Integration", *Sloan working paper #4021*, April 1998

7. Tom Gilb, <u>Principles of Software Engineering Management</u>, Addison Wesley, 1988

8. Michael A. Cusumano and Chris F. Kemerer, "A Quantitative Analysis of U.S. and Japanese Practice and Performance in Software Development", *Management Science*, November 1990, pp 1384-1406

9. Michael A. Cusumano and Richard W. Selby, "How Microsoft Builds Software", *Communications of the ACM*, June 1997, Vol.40 No.6, pp 53-61

10. Ian Sommerville, <u>Software Engineering</u>, 4th Edition, Addison Wesley, 1992