Experiences of Requirements Engineering for Two Consecutive Versions of a Product at VLSC

Joel So and Daniel M. Berry
University of Waterloo, Ontario, Canada

Abstract

This talk describes the experiences of the first author in leadership requirements engineering (RE) roles in the developments of two consecutive versions of one SW product in one company. The two developments differed in the amounts and the quality of their upfront RE and in their final results. The talk notes a correlation between the quantity and quality of RE and the quality of the final results in a development. The talk concludes with some lessons learned from the experiences and from the differences.

Outline

Introduction

Project Organization and Lifecycle

Delusions of Upfront RE for First Version

Decisions for the Future

Upfront RE Done Right for Second Version

Comparison of the Two Versions

Lessons Learned

Introduction

This talk is about experiences of the first author in leadership requirements engineering (RE) roles …

in two consecutive software (SW) development projects, …

for two consecutive versions of one SW product, …

carried out in one SW development company.

First Development

In the first development,

- **the RE process was poor,**
- **the shipment was late, and**
- **the product's quality was nothing to write home about!**

Between the Two Developments

Between the two developments, the management of the first development *and the executive level management of VLSC*

- **reviewed the first development,**
- **realized its RE's shortcomings,**
- **realized importance of full upfront RE, and**
- **issued a mandate to do better RE in second development.**

Second Development

In the second development,

- **the RE process was much improved,**
- **the Beta 1 shipment was on time, and**
- **the product seems to be significantly better.**

Anonymity

To keep the company name and product name anonymous,

- **the company is called ‘‘VLSC’’, for Very Large Software Company, and**
- **the product is called ‘‘PROD’’.**

If you are even minimally aware of the SW business, you have probably heard of both the company and the product. ☺

Historical Context

During Fall of 2001, first author was an intern program manager (PM) at VLSC.

A PM is part requirements engineer, part project manager, part cat herder, etc.

As PM, he witnessed release of Version 2 of PROD and kick off of development of Version 3 of PROD.

Historical Context, Cont'd

From July 2002 through August 2005, he was a full-time PM.

He contributed to rest of development of Version 3 and to first part of development of Version 4, up to Beta 1 release.

PROD Background

PROD is a popular enterprise server application in its 3rd release.

Version 3 is a complete architectural overhaul from Version 2.

As of writing of the paper, Version 4 is in the works.

PROD Background, Cont'd

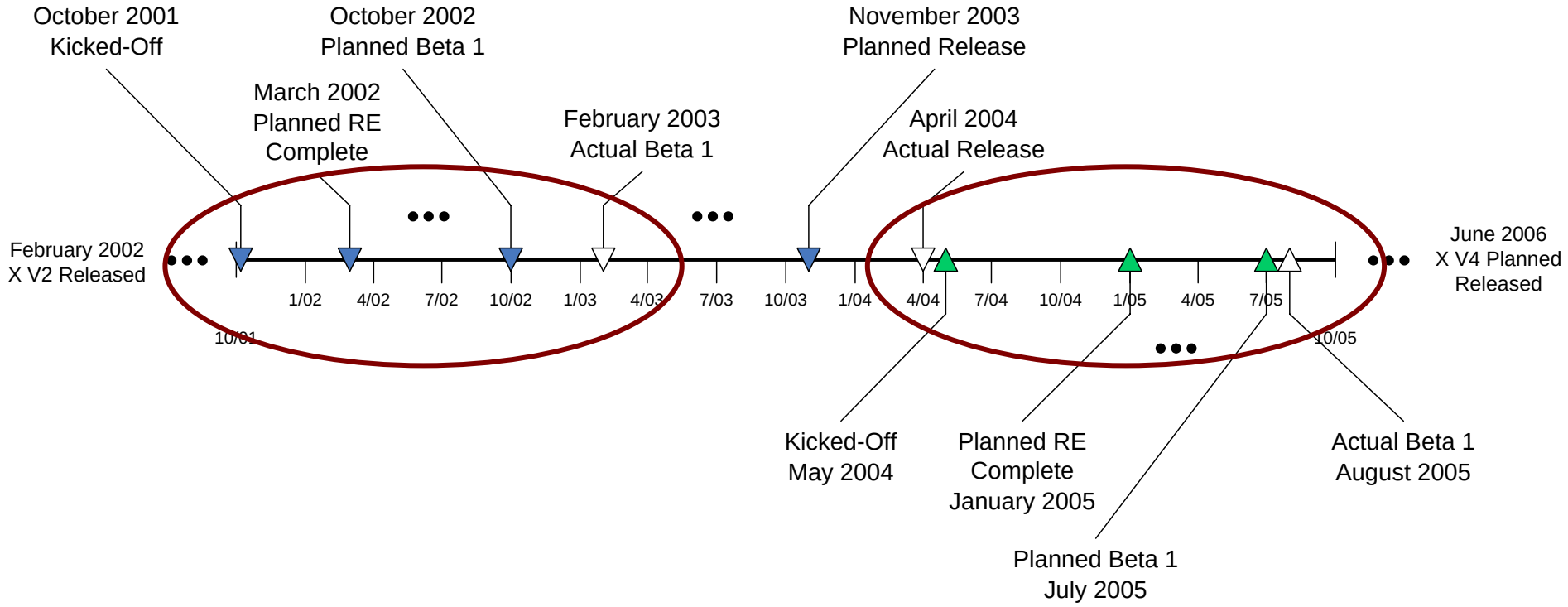
Each of Versions 3 and 4

- **has 3 major architectural components offering about 15 distinct features and**
- **required or requires about 43 person years of development.**

This talk focuses on the RE for Version 3 and Version 4.

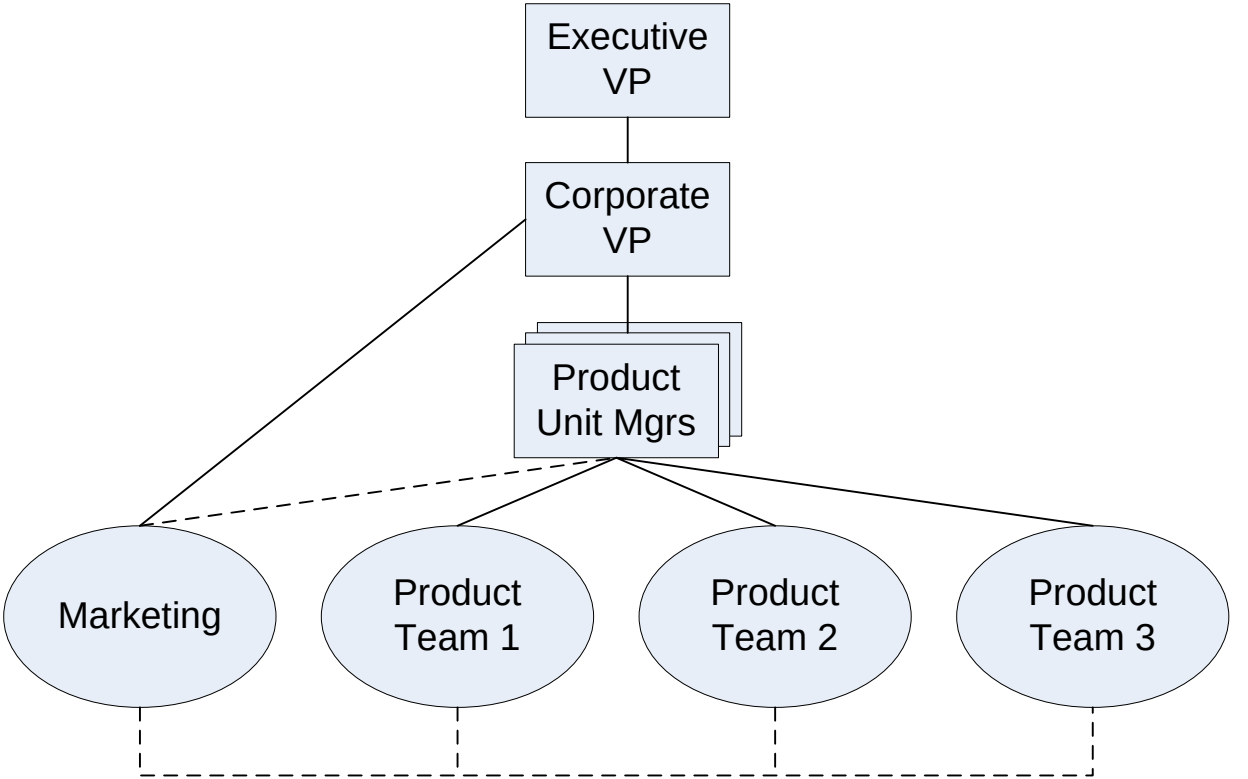
Timelines

Next slide shows a single timeline for the two developments up to each's Beta 1 release.


October 2001
Kicked-Off
October 2002
Planned Beta 1
November 2003
Planned Release
March 2002
Planned RE
Complete
February 2003
Actual Beta 1
April 2004
Actual Release
February 2002
X V2 Released
June 2006
X V4 Planned
Released
1/02
4/02
7/02
10/02
1/03
4/03
7/03
10/03
1/04
4/04
7/04
10/04
1/05
4/05
7/05
10/01
10/05
Kicked-Off
May 2004
Planned RE
Complete
January 2005
Actual Beta 1
August 2005
Planned Beta 1
July 2005

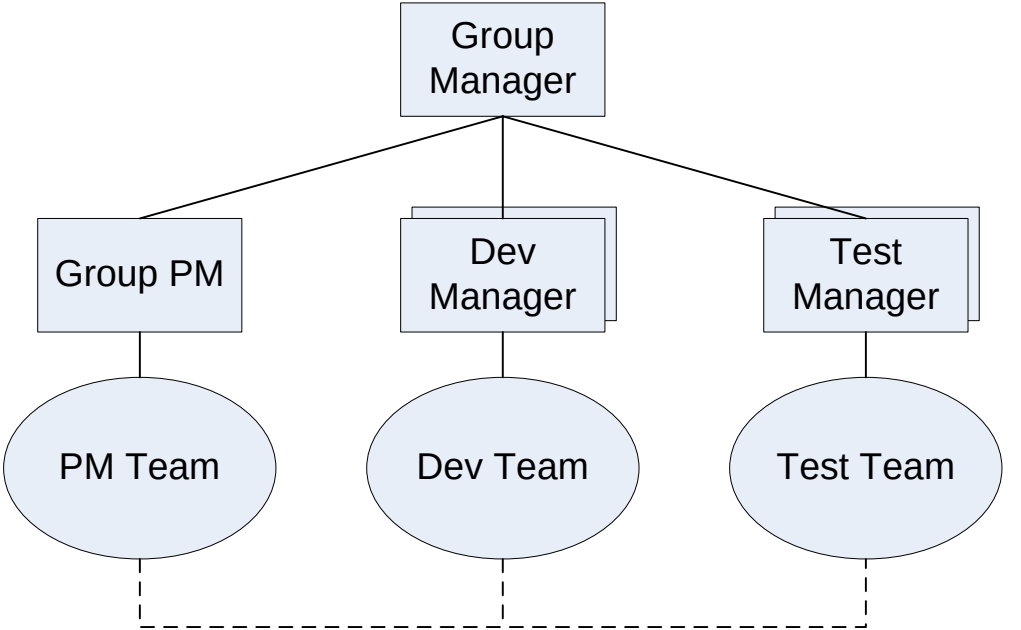
Project Organization

Next slide shows an organizational chart for the enterprise server division in VLSC:


Executive VP
Corporate VP
Product Unit Mgrs
Marketing
Product Team 1
Product Team 2
Product Team 3

Project Organization, Cont'd

Next slide shows an organizational chart for the typical product team.


Group Manager
Group PM
Dev Manager
Test Manager
PM Team
Dev Team
Test Team

Project Organization, Cont'd

The typical product team for the development of any version of any large-scale SW product at VLSC has,

- **excluding its managers, about**
- **20 PMs,**
- **15 developers, and**
- **about 50 testers,**

totalling about 85 people.

The team may use also outsourced resources.

VLSC's SW Development Lifecycle

Officially, each SW development at VLSC follows a waterfall model whose milestones are:

M_0: finishing functional specification,

M_1: finishing design specification,

M_2: finishing code development,

M_3: finishing testing of the entire system,

Lifecycle, Cont'd

M_i: Beta test *i* release to early adopters who have agreed to evaluate the release,

FR: final release of retail version to market, out of product team's hands

Each release starts with steps leading to a new M_0, and eventually to FR.

Beta Test Release

The steps and prerequisites for preparing and doing a Beta test release to early adopters are:

1. **all reported bugs in M_i build have been resolved,**
2. **M_i build passes all test plan tests,**
3. **M_i build is released via the Web or other mechanism, and**
4. **early adopters provide feedback.**

Delusions of Upfront RE

From kick off until well into design of Version 3, …

the PMs were wearing SW architect's hats when they should have been wearing requirements engineer's hats.

Architecture Thinking

Recall: Version 3 was to be a major architectural restructuring of Version 2.

Therefore, several designs and prototypes were floating around from even before kick off.

The PMs were so focused on architecture and code, that they were barely thinking at requirements level.

Causes of Architecture Thinking

The focus on architecture came from a lack of:

1. **direction from management to focus on requirements**

2. **a clear product vision, which would specify goals, the why that motivates the what.**

Immersion in Development

The PMs were immersed in development, putting code before requirements, because of perceptions that

- **working out requirements before coding**
  - **wasted time and**
  - **delayed starting real work, and**
- **beginning coding sooner shortend the lifecycle.**

Upfront RE Was Mandated

Project schedule mandated upfront RE,

But, mostly only lip service was paid to this RE.

What Was Really Happening

Requirements gathering and specification were lumped into one step.

Thus, first ideas were taken as final requirements without carefully considering whether the ideas made a consistent whole.

Barely any real attempt to gather requirements from real customers and users.

Really Happenings, Cont'd

Instead, the PMs put on customer's hats and wrote down what they *thought* a real customer would want

The PMs did not check if any real customer really wanted what the PMs wrote down.

Really Happenings, Cont'd

At scheduled time of the M_0 milestone, the PMs took what they had and signed off on it.

After all, working code existed!

No one really cared about the requirements specification (RS).

A Telling Statistic

A telling statistic: the process leading to M_0 had very few bugs reported.

A low bug count in the normally bug-laden step of distilling disparate people's fuzzy ideas into a consistent, complete, and concrete whole is a *bad* sign.

True Bug Status

Undoubtedly, the RS was loaded with bugs.

Since they were not found during RE, they were lurking to be committed into buggy designs and code.

These bugs would be found only

- **during testing or**
- **after shipping, by customers, the best bug finders in the universe.**

Cart Before the Horse

In fact, steps leading to M_1 and M_2 were well underway before M_0.

Much of RE that did happen in project happened in short bursts of RE during the steps leading to M_1 or M_2, …

whenever designing or coding could not proceed without resolving a requirements issue.

Another Sign of Trouble

In VLSC, a change in a product's internal code name indicates a massive realignment of the product's scope and requirements.

PROD's internal code name changed 3 times late during Version 3's development, …

and the 3rd was even after Beta 1 release.

Slippage

Beta 1 was released 4 months late.

As at most SW companies, at VLSC, everything ships eventually, and the slips just accumulate.

Final release of Version 3 slipped an additional 2 months.

Thus, the total delay was 6 months in a 25 month project, about 25%.

Questionable Quality

Version 3's quality was questionable.

Several quick fixes were issued in first month of general availability.

Planning for a patch package began immediately.

Postmortem

Product team asked itself, ‘‘What went wrong?’’ not to repeat bad history.

The PMs realized that they had not done enough upfront RE,

- **even though RE and its milestone, M_0, were scheduled, and**
- **the PMs had signed M_0 as completed.**

Instead

The PMs had a salute-the-flag attitude about the RE that they should have been doing:

- **PROD's vision was not defined up front;**
- **targeted scenarios were not identified and finalized up front.**

Informality, Imagination, and TBD

The PMs' requirements gathering was very informal.

Often, they only imagined what a real user *would* want.

Thus, the signed-off RS was neither concise nor complete, leaving much room for error and misinterpretation.

PMs left much to be fleshed out by implementers, with a high risk of their not implementing the PMs' intents.

Delusions

Much of so-called RE time was spent doing things other than RE, …

things that PMs perceived as more important than RE.

The PMs' signature on the M_0 milestone reflected self delusion, not fact.

Compounded Problems

PMs did costing and scheduling based on an incomplete and incorrect RS.

When true requirements began to emerge during coding,

- **the old costs and schedule were invalidated**
- **the new requirements cost considerably more to implement than if they had been found during RE.**

Compounded Compounding

New requirements forced major, costly restructuring, …

throwing off costing and scheduling even more!

Version 3 Development Sum Up

Thus, letting the implementation cart get before the RE horse led to

- **incorrect costing and scheduling,**
- **substantial re-engineering, and**
- **finally, a 25% schedule slip.**

Executional Catastrophe

Version 3's development and release were viewed internally as an executional catastrophe despite its positive growth in sales revenue.

Heads rolled:

- **both division VPs,**
- **1 PROD product unit manager,**
- **2 general managers, and**
- **others**

Cause of Catastrophe

VLSC's *executive level* management recognized that the failure was caused primarily by insufficient and incorrect upfront RE.

How often do you see high-level executives in a big corporation thinking about good RE?

Nu!?!?

Decisions for the Future

VLSC's executive level management decided to address the problems at the executive level first and …

then to proceed from top to bottom.

Management's Commitments

Management committed to, and made itself accountable for,

- **defining,**
- **communicating, and**
- **upholding**

the product vision

- **up front and**
- **throughout**

the project.

Management's Provisions

Management provided resources to make proper upfront RE possible, including

- **a longer M_0 cycle,**
- **more PMs, and**
- **more funding to the whole project.**

Management's Decrees

Management

- **issued new RE mandates,**
- **described new RE processes, and**
- **offered new incentives to do fuller RE.**

New Elicitation Stage & Milestone

Management introduced a new M_{-1} milestone for focused requirements elicitation in advance of writing an RS for M_0.

Focus for New Milestone

The focus for M_{-1} is on customer-centric requirements:

- **convening customer and technical advisory boards,**
- **surveying customers, and**
- **having early-adopting external customers sign off on the RS.**

The M_{-1} stage is customer- and market-driven, not engineering-driven.

More Formality

Management sought to replace the too informal RS process of the past.

Management mandated a whole bunch of new procedures.

New Mandates

Management

- **had VLSC's RS templates updated.**
- **demanded formal**
  - **use-case modeling,**
  - **security modeling, and**
  - **pseudo-UML functional modeling,**

  focusing on scenarios, both
  - **goal based and**
  - **user based.**

Mandates, Cont'd

- **instituted and enforced a rigid RS review and sign-off process, including the early-adopter RS feedback and sign off.**
- **demanded accountability for sign off of M_0 exit criteria.**
- **decreed that**
  - **M_0 sign off is *before* beginning steps leading to M_1, and**
  - **M_1 sign off is *before* beginning steps leading to M_2.**

Idle Developers & Testers?

Developers and testers were put to work reviewing during RE, to avoid idling them.

Developers and testers

1. **could evaluate technical feasibility of proposed requirements,**
2. **could improve their understanding of the PROD requirements, and**
3. **were thus prevented from premature coding and test case design.**

New Special Team

Management even introduced a special team to

- **examine all scenarios,**
- **ensure overall cohesiveness of RE artifacts,**
- **do RE for any orphan issue that did not have natural home, and**
- **build and conduct tests of PROD with real-world data and loads.**

Cultural Changes

VLSC's management realized that some cultural changes were essential.

- **No more guessing or asking others what the requirements are:**
  - **Look it up first in the RS.**
  - **If RS does not answer, ask the PMs and FIX the RS.**
  - **If RS is not clear, clarify with PMs and FIX the RS.**

Cultural Changes, Cont'd

- **Organizational shift**

  from being purely development driven

  to being balanced between being
  - **requirement driven,**
  - **development driven, and**
  - **testing driven.**

- **Monetary incentives for more collaboration among feature-centered teams.**

Acceptance of Cultural Changes

No one denied that there had been a catastrophe

No one contested the assessment of catastrophe's cause.

Thus, contrary to normal situation, the cultural changes were fairly easy to impose.

Additional Confirmation

An organizational health survey for 2003 and 2004 showed that lack of upfront RE was a major cause of low morale among Version 3's developers and testers.

Constant changes and reengineering due to continual requirements changes demoralizes developers and testers.

VLSC's management thus felt confident of the correctness of their new mandates.

Upfront RE Done Right

The new processes identified in the postmortem were instituted and enforced by management.

Consulted Real Customers

This time, requirements came from real customers that wanted the next version of PROD to meet their needs.

A new user-experience team got the job of

- **identifying,**
- **naming, and**
- **modeling**

each kind of PROD user as a personna.

Complete & Unambiguous RS

This time,

- **the PMs really tried to write a complete and unambiguous RS.**

- **the RS**
  - **was scenarios centric and**
  - **followed a standard template.**

Special Team

The special team proved effective in

- **encouraging collaboration among feature-centered teams and**
- **integrating islands of functionality into a coherent system.**

Real Accountability

This time, no one signed off on a document unless he or she was willing to be held accountable for the document.

This time, the PMs did not sign off on M_0 until all PMs felt that

- **the customers had been wrung dry of requirements and**
- **the requirements that were scoped into Version 4 made a consistent whole.**

Early Bug Finding & Fixing

This time, the team and the early adopters found and fixed many bugs in the RS while the RS was been written.

This time, no one signed off on M_0 until all these bugs were fixed.

Change Management with Teeth

This time, the PMs differentiated between

- **changes stemming from incomplete or incorrect requirements and**
- **changes stemming from scope creep.**

A new requirement was included only if it lay within the scope of Version 4's vision.

Changes After Sign Off

After M_0 sign off, any proposed requirements change was treated as a formal change request with its usual heavy bureaucracy.

The Change Bureaucracy

For any formal change request,

- **The PMs had to make a case for the request to the PROD group manager, who got to make the call.**
- **Considerable justification,**
  - **a big increase in income or**
  - **a big savings in expenses or time,**

  was needed get the change into Version 4.

PMs New Understanding

The PMs now understood the product vision and …

knew when to punt a request to a later version.

Results

As of the writing of the paper, the project for Version 4 was proceeding well.

- **The team exited M_0 less than one month late in January 2005.**
- **Each of M_1, M_2, and M_3 was essentially on time, leaving the total slip at one month.**
- **Beta 1 release was in August 2005, only 1 month late.**

Results, Cont'd

- **Beta 2 release was actually expected to realign with the original schedule due to the high-quality code of the Beta 1 release.**
- **The original final shipping date was still on target.**

Other Good Signs

The organizational health survey for 2005 showed marked improvements in team morale and work–life balance.

Customer feedback for Beta 1 release was very positive with very few reported bugs.

Comparison of the Two Projects

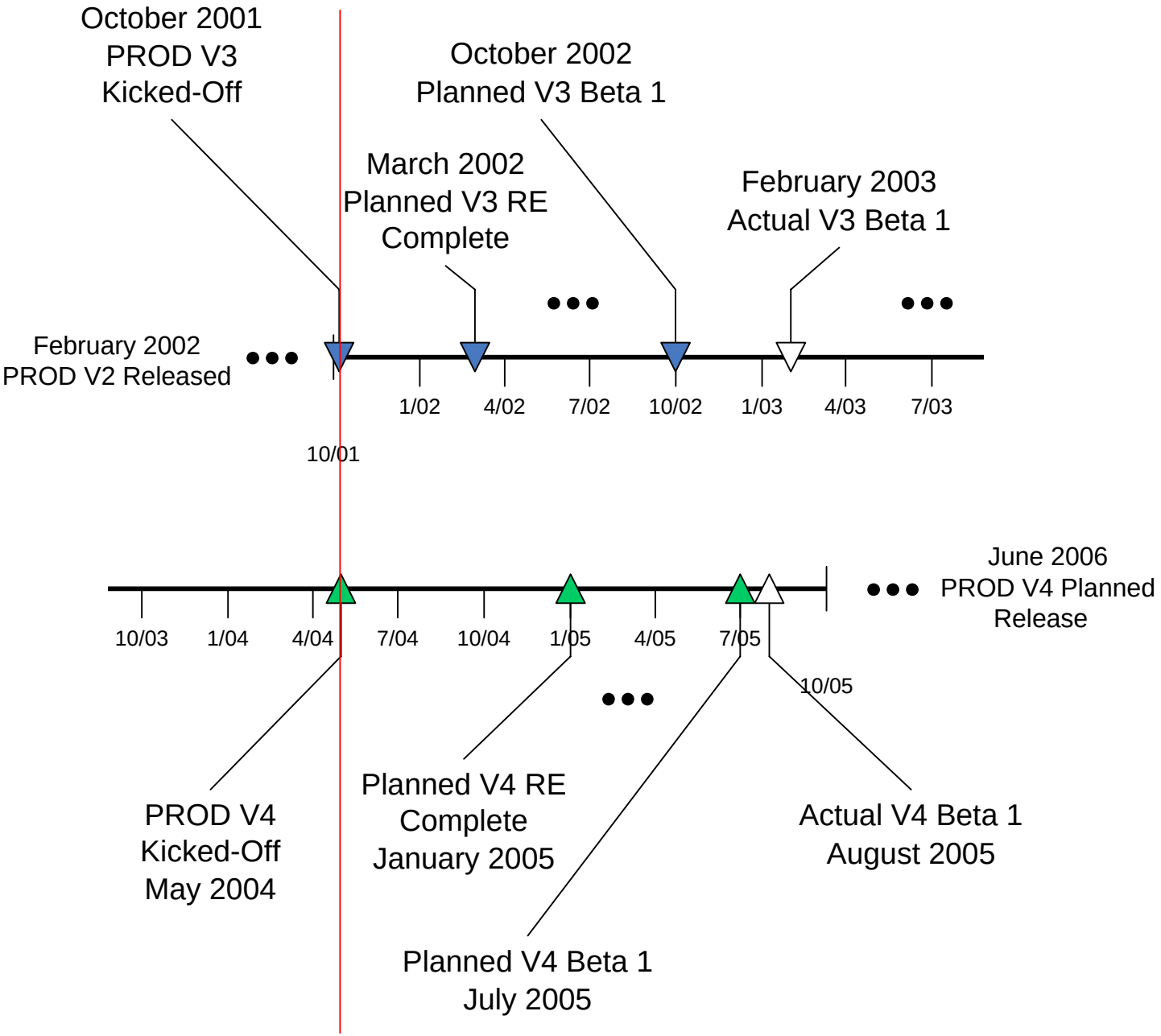
A disclaimer is necessary about the results.

- **The first author no longer has access to the internal bug database.**
- **The results shown are based on only**
  - **archived status e-mail messages and**
  - **informal consultation with old colleagues.**
- **Therefore, there may be inaccuracy.**

Timelines

The next slide shows the timelines of the two developments up to their Beta 1 releases, side by side to allow direct comparison of their histories.

Note that the final date, 8.5/03, of the top timeline is the beginning date of the bottom timeline.


October 2001
PROD V3
Kicked-Off
October 2002
Planned V3 Beta 1
March 2002
Planned V3 RE
Complete
February 2003
Actual V3 Beta 1
February 2002
PROD V2 Released
1/02
4/02
7/02
10/02
1/03
4/03
7/03
10/01
June 2006
PROD V4 Planned
Release
10/03
1/04
4/04
7/04
10/04
1/05
4/05
7/05
10/05
PROD V4
Kicked-Off
May 2004
Planned V4 RE
Complete
January 2005
Actual V4 Beta 1
August 2005
Planned V4 Beta 1
July 2005

The table on the next slide compares the two project by several project variables. In the last column, ‘‘+’’ means ‘‘increase’’, and ‘‘-’’ means ‘‘decrease’’.

Variable	*Version 3 Project*	*Version 4 Project*	*Change from V3 to V4*
Planned Upfront RE Time	**6 months**	**9 months**	**+ 50%**
Planned Time to Beta 1 Release	**13 months**	**15 months**	**+ 15%**
RE as a Percentage of Time to Beta 1	**46%**	**60%**	**+ 30%**
Number of PMs in RE	**18**	**25**	**+ 39%**
Number of Marketing Product Managers in RE	**3**	**8**	**+ 166%**
M_0 Spec Bugs Opened & Resolved	**468**	**984**	**+ 110%**
M_1 Bugs Opened & Resolved			
Change Requests Against M_0	**117**	**38**	**- 67%**
Design Defects	**525**	**314**	**- 40%**

Variable	*Version 3 Project*	*Version 4 Project*	*Change from V3 to V4*
Beta 1 Bugs Opened & Resolved			
Change Requests Against M_0	**104**	**19**	**- 82%**
Change Requests Against M_1	**126**	**23**	**- 78%**
Code Defects	**409**	**376**	**- 8%**
Time to Beta 1 Release	**17 months**	**16 months**	**- 6%**
Slippage in Beta 1 Release	**4 months**	**1 month**	**- 75%**
Beta 1 Hot Fixes Issued in First Month after Release	**11**	**0**	**- 100%**

Effects of Increased Upfront RE

The increase in allocated upfront RE time and personnel led to

- **increased RE efficiency,**
- **increased early discovery and resolution of requirements-related issues,**
- **decreased downstream change requests,**
- **decreased downstream bug counts, and**
- **decreased number of quick fixes to Beta 1 release within first month after release.**

Effects, Cont'd

The number of quick fixes to the Beta 1 release in the first month after release dropped from

11 in Version 3

to

0 in Version 4.

Conclusion

We conclude that the Version 4 Beta 1 release was of significantly higher quality than the Version 3 Beta 1 release.

Future Possibility

As of the writing of the paper:

- **The increase in allocated upfront RE time and personnel may yet lead to a shorter downstream development cycle for Version 4.**
- **Version 4's Beta 1 release was completed only 6% faster than that of Version 3.**

Future Possibility, Cont'd

- **However, if the current trend continues, the final release of Version 4 should ship on schedule, …**

  for 13% faster overall completion time than for Version 3.

What Actually Happened

Information obtained in first author's recent informal lunch with former workmates:

- **Version 4 was released to manufacturing at the end of March 2006 for early May 2006 general availability *2* months ahead of the schedule known at the time of the writing of the paper.**

- **User reviews are much more positive than those of Version 3.**

Early Release?

The reason for the early release seems to be a combination of

- **the better RE and the high quality Beta 1 release**
- **business pressures to release early, perhaps at the cost of cutting out some minor features in final release, relaxing some release criterion, or both.**

Lessons Learned

In reality, Lessons 2 through 4 are sublessons of Lesson 1, but each should be stated on its own.

Lesson 1

Don’t be delusional about upfront RE.

Upfront RE must be done right, not just done.

Processes must be in place to ensure that RE time is used actually for RE, and not for designing or coding.

Lesson 2

Upfront RE is extremely process-oriented.

Upfront RE should be viewed as a process with discrete deliverables, rather than as a single task.

The process should include review, sign-off, and accountability mechanisms.

Lesson 3

Proper upfront RE yields tangible gains, among which are more stable and predictable downstream development cycles and higher quality designs and code.

It potentially results in shorter downstream development cycles, and happier engineers.

Lesson 4

You are never really done with RE, but you do have to finish at some point, in order to move on to designing and coding.

The trick is finish neither too soon nor too late.

Use the product vision to help you decide when you have done enough.

Define clear, measurable exit criteria for each milestone.