Combining a CMMI SCAMPI Appraisal & Project Retrospective

2008 NDIA CMMI Technology Conference

Roger Campbell

cognence inc
Improving Software Economics

Possible Dimensions to Appraise in an Organization

- Process
- Financial
- People
- Projects
- Tools/automation

"What are the brutal facts? We've got to get a grip on the facts, what are the trends, what are the trendlines, how bad is it? Get a grip on the facts." – Jim Collins, Author of *Good to Great*

Inherent Limitations of CMMI Appraisals

- Focused on process
  - Could be a strength!
- Some high maturity organizations fall short on project performance
- Does not review/analyze organizational or project performance against previously established objectives/baselines
- Non-attribution leads to broader organizational findings
- Difficult to identify “What is going wrong in my organization, and how do I fix it?” without performance data

Project Retrospective Opportunity

- In the course of performing a SCAMPI, leverage the resources in place and the intelligence already gained to explore one or more projects and their performance in-depth
- Dimensions reviewed will be actual project performance compared to expectations/promises set at the onset – if available!
- Identify actual performance, whether it fell short, met, or exceeded expectations
- Then, map process gaps (identified through the SCAMPI) to performance gaps to quantitatively determine which processes/practices to improve first!
  - These help solve business goals based on performance data

Project Retrospective Approach

- Identify objectives set at the start of a project:
  - Scope – number of requirements to be delivered
  - Schedule – when the requirements were to be delivered
  - Cost – The expected cost of delivering the requirement
  - Quality – The expected quality of the delivered requirements
- Review actual project results at completion and identify whether they fell short, were met, or were exceeded
  - Requires significant data archeology in some cases!
  - A large amount of analysis occurs as well

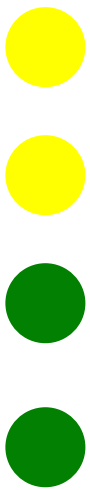
Example Project Retrospective Findings

"What you can measure you can target. And what you can target you can accomplish." – Jim Collins

cognence inc
Improving Software Economics

Retrospective Summary

ABC 1.0 experienced significant churn early on. After the development phase-gate, some schedule problems remained, although a quality software product with a majority of the planned features was delivered to the market on time and within budget.

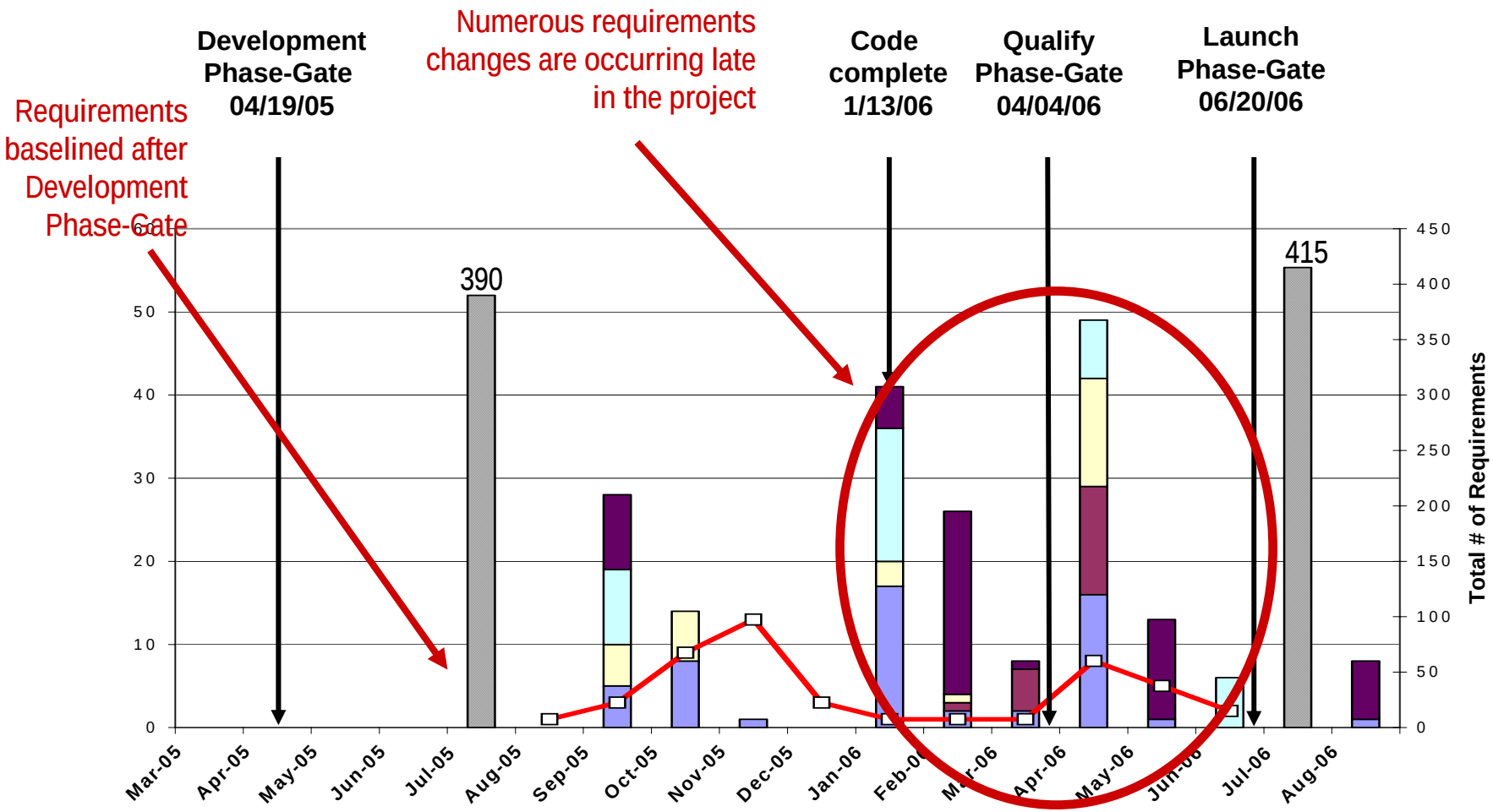


Scope	As defined by the requirements at the Development phase-gate w/changes versus actual delivered
Schedule	As determined by Qualify, Launch, and GA actuals versus targets at the Development phase-gate
Cost	As determined by planned cost targets at the Development phase-gate versus actuals
Quality	As determined by quality targets for GA at the Development phase-gate versus actuals

ABC 1.0 Scope Performance Findings

- Not all planned scope defined at the Development phase-gate delivered
  - Major scope changes were not visible in CMB change requests or in the product contract change history – these involved deferment of several features to a future release
  - No schedule/effort impacts due to changes were identified even though it was clear scope changed
  - A significant number of User Interface changes occurred during Beta testing – these were each small in impact, but together "they added up to real work"
- Detailed Requirements were not approved until July 2005, almost 3 months after the Development phase-gate approval
- Many changes occurred to requirements at the all levels
  - Customer requirements changed from 13 to 35 requirements - there is no clear CMB entry criteria or traceability why and how this occurred other than 2 CMB CRs that don't effectively provide explanations
  - Top 5 SW-related requirements docs changed from 390 to 415 requirements via 194 CRs – these changes generally well managed
  - 47 total ClearQuest software enhancement CRs – no clear traceability to Customer CMB CRs or Detailed Requirements CRs

Requirements Changes Per Month



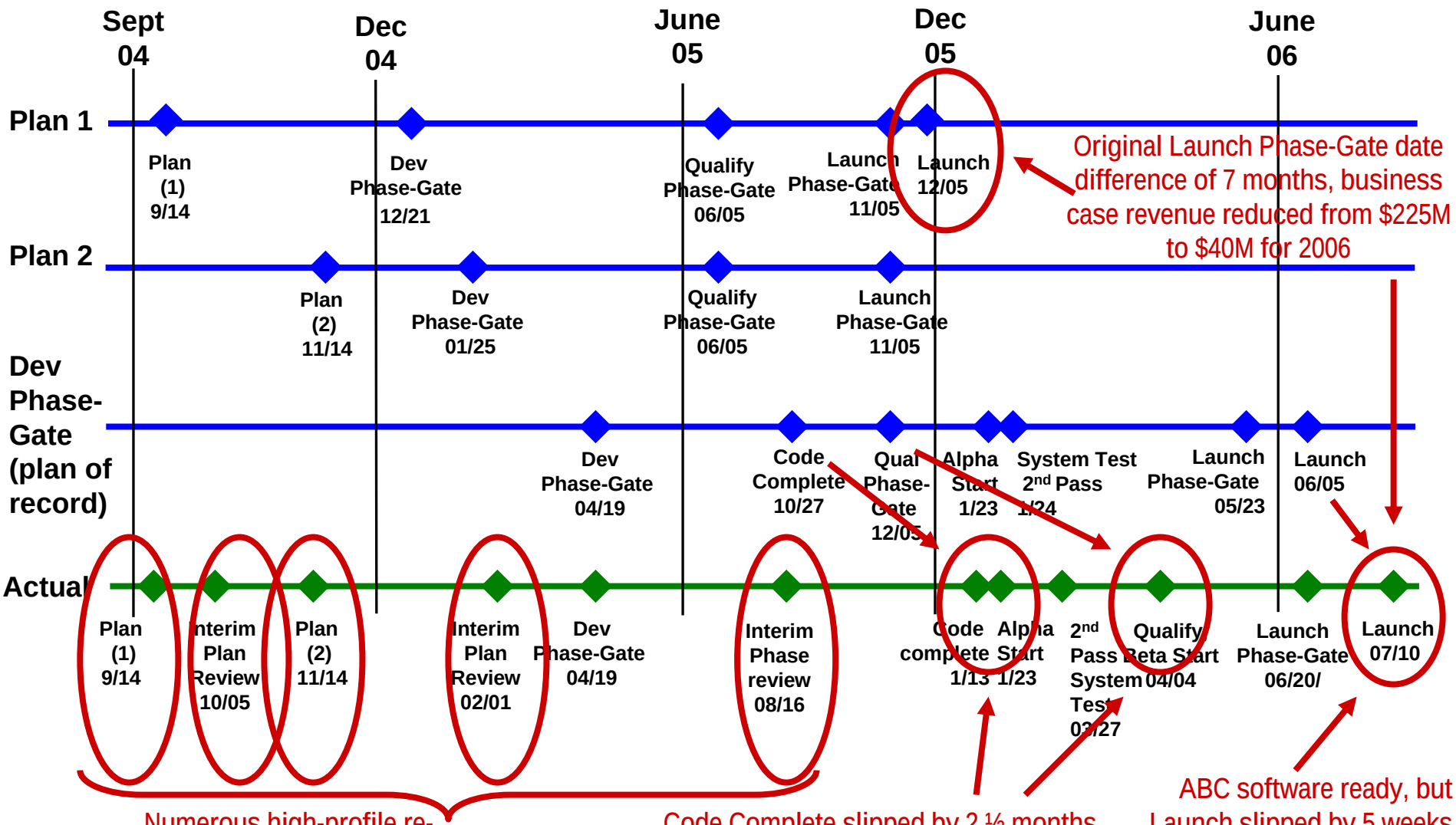
ABC 1.0 Schedule Performance Findings

- **Significant churn occurred in the ABC 1.0 planning phase**, resulting in
  - Numerous Planning reviews
  - A delay in Development Phase-Gate of 4 months from 12/21/04 to 4/19/05
  - A delay in Launch Phase-Gate of 7 months from 11/05 to 6/20/06
- **After Development Phase-Gate, interim schedule performance still suffered**
  - Code complete delay of 3 months from 10/27/05 to 1/13/06
  - System Test 2nd pass start delay of 2 months from 1/24/2006 to 3/27/2006
  - Qualify Phase-Gate delay of 4 months from 12/05 to 4/4/2006
- **Launch Phase-Gate and GA slipped by 5 weeks**, due to hardware manufacturing, not software
  - Schedule was crashed by conducting System Test 1st pass and Alpha simultaneously (baseline Development schedule planned for conducting System Test 2nd pass and Alpha simultaneously)

The 7-month Launch difference from the planned estimate resulted in the ABC 1.0 2006 business case revenue projections being reduced from $255M to $40M – a reduction of (up to) $215M

(Based on business case expected gross revenue, which considered ABC 1.0 in isolation. Actual, incremental product portfolio net revenue impact to the company is significantly less.)

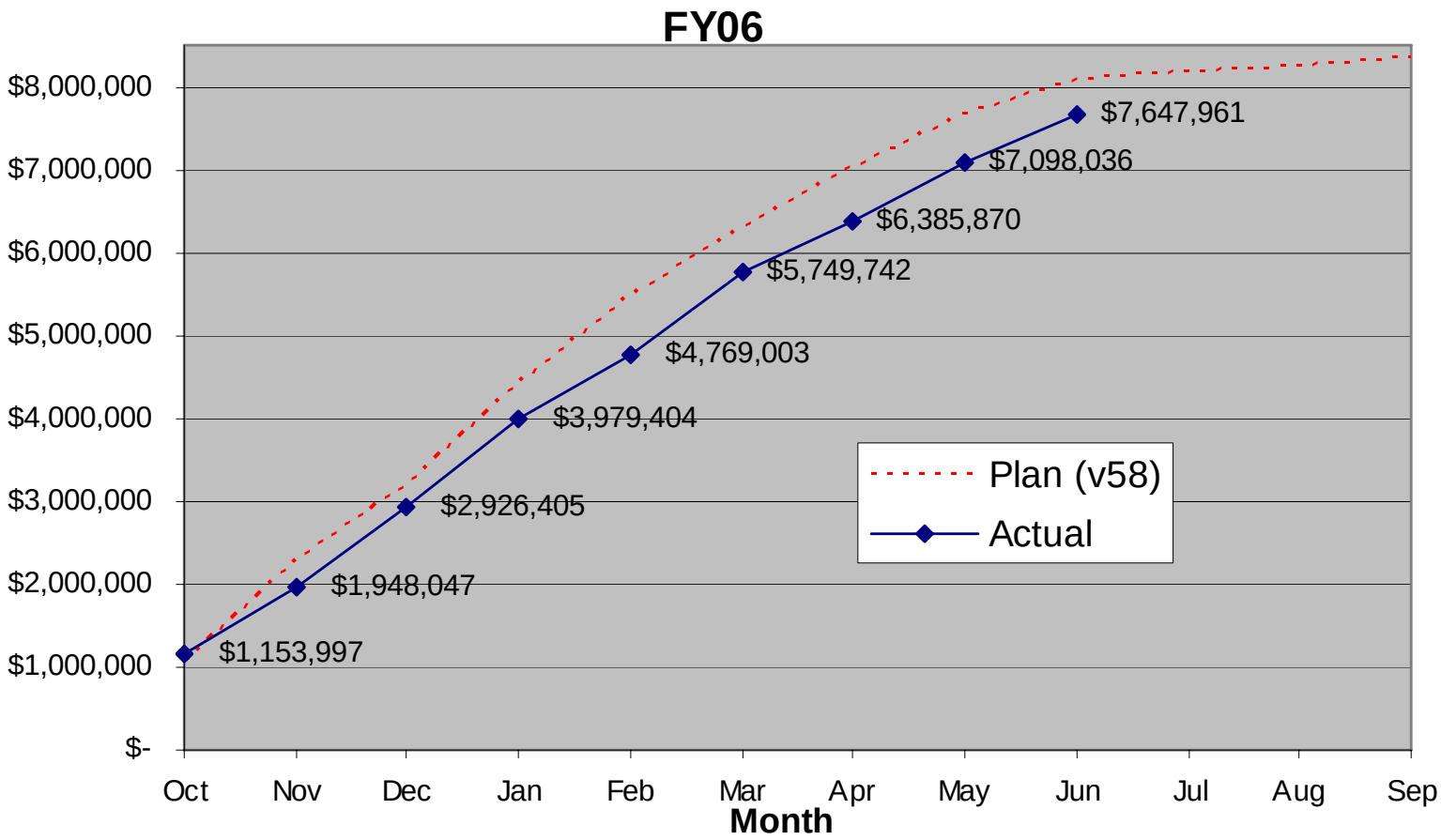
ABC 1.0 Chronology



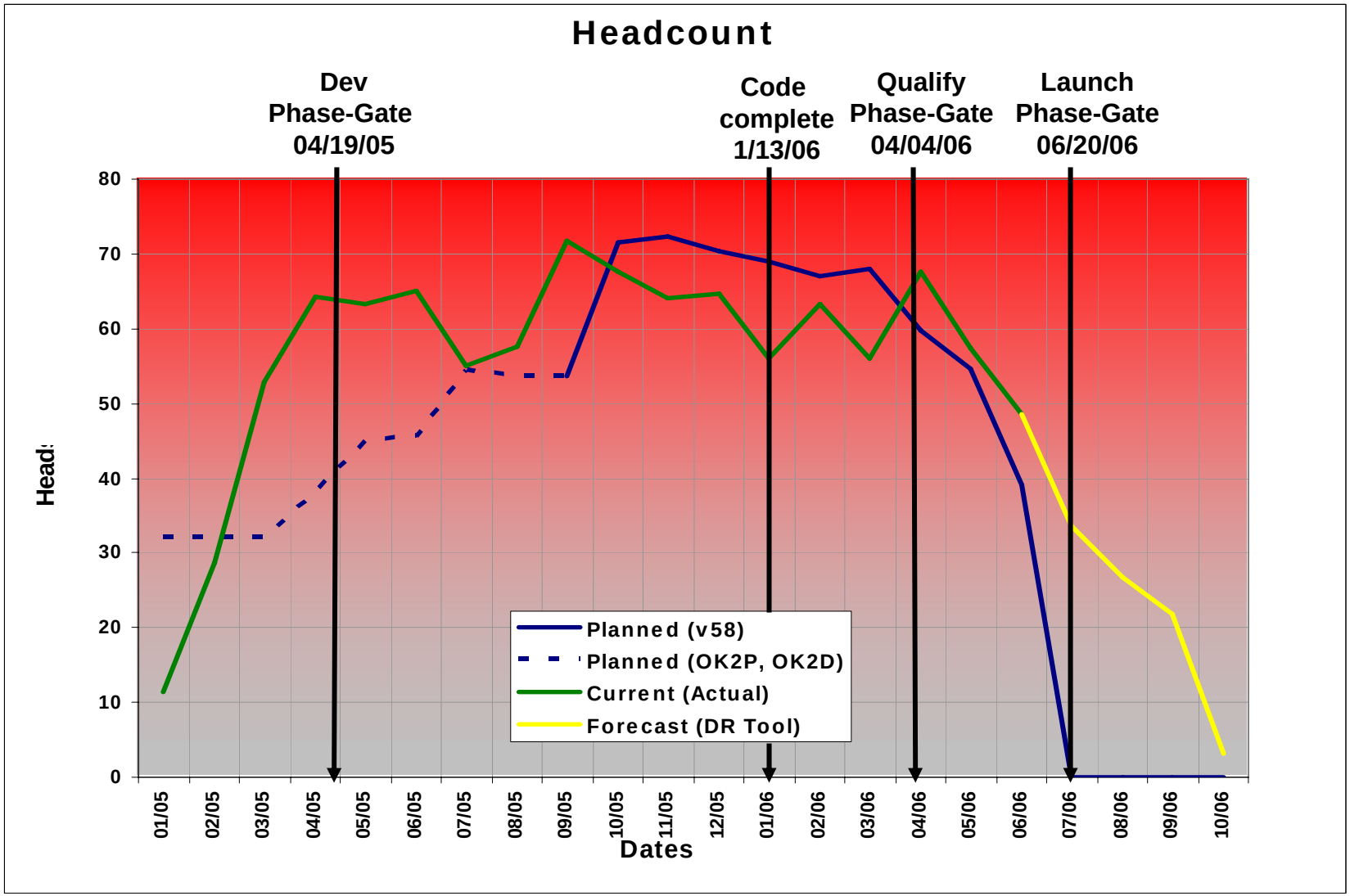
ABC 1.0 Cost Performance Findings

- Unable to account for resources/costs specific to ABC 1.0
  - Work on multiple Phase 1 Releases are not differentiated (i.e., 1.0 vs. 1.1).
  - Unable to account for actual resources during Plan Phase – No actual data prior to February 2005
- 2006 costs are on target; project costs prior to v58 plan are difficult to determine actuals versus estimates
- Effort actuals overran plan during the Plan and Qualify Phases

ABC Phase 1 Cost Performance



ABC Phase 1 Headcount Performance



Note: Plan v58 = FY06

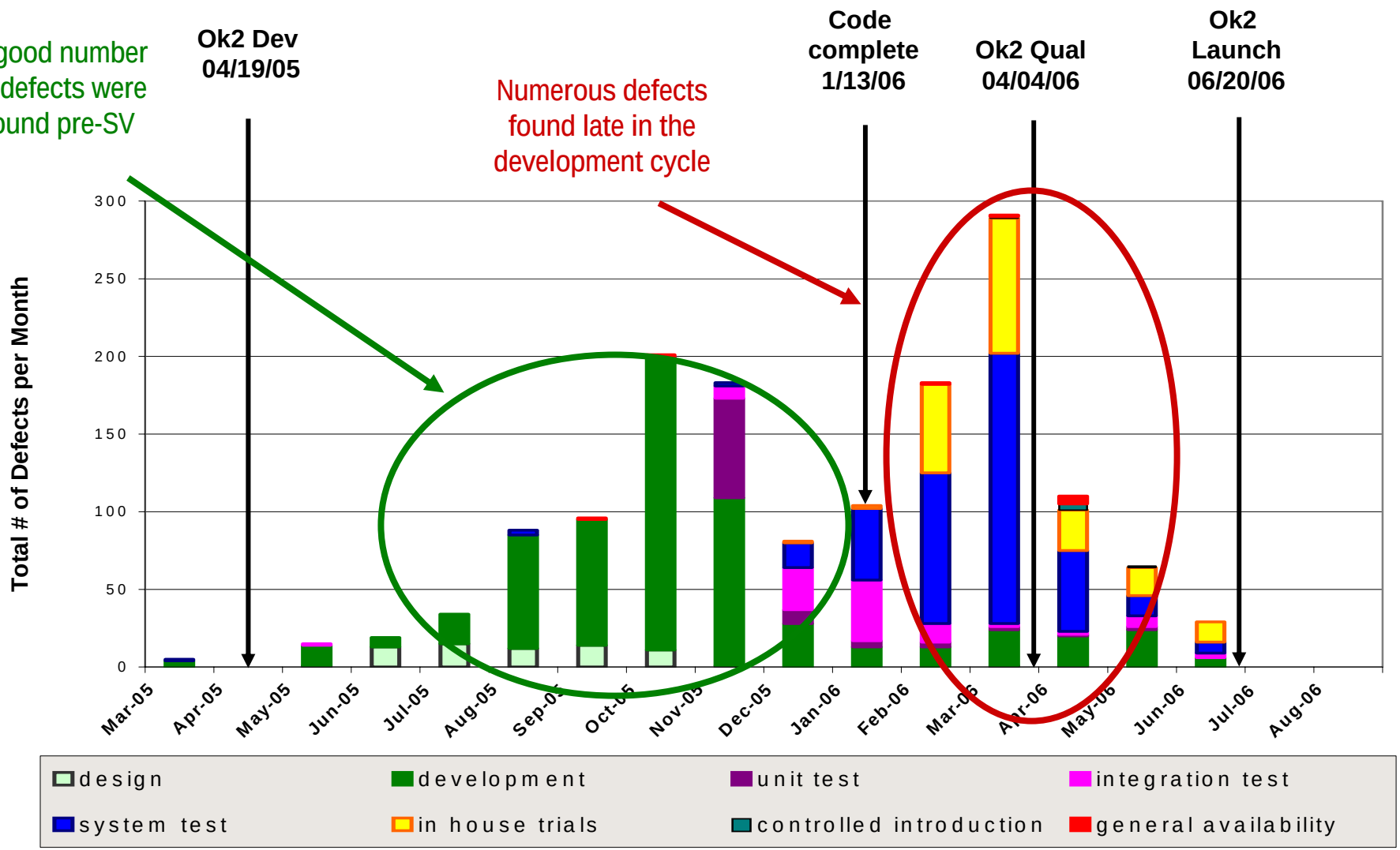
ABC 1.0 Quality Performance Findings

- Most quality targets established at the Development phase-gate were met at Launch
  - 0 open Sev 1 & Sev 2 CRs
  - 73 open Sev 3/4 CRs versus <100 target
  - 99.15% system tests executed at Launch phase-gate versus 100% target
  - 95.6% system tests ran and passed at Launch phase-gate versus 100% target
- Critical/high severity defect detection occurring late in the product development lifecycle
  - System test & Alpha performed concurrently
  - Approximately 2/3rds of all critical/high severity defects detected in system test or later
  - Almost as many critical/high severity defects reported in (Alpha) phase as were in system test
- 73 open CRs were deferred to later releases (1.1, 1.2) as they were not thought to be critical to the 1.0 release
- SQM recently implemented to track overall project quality progress weekly toward Launch. Overall SQM target met at Launch.
- No critical/high severity customer-found defects reported to-date

Quality Targets & Measures

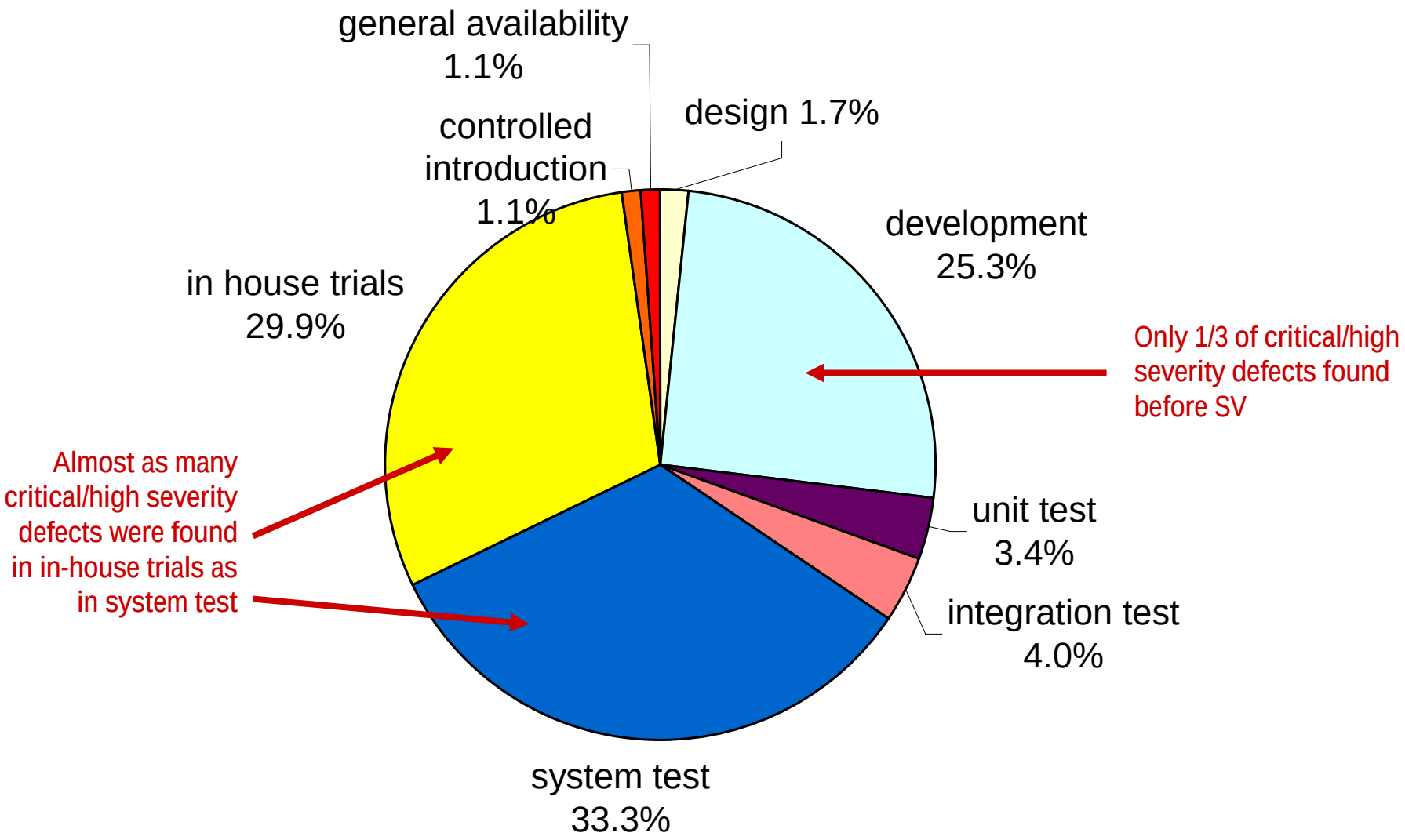
Criteria	Qualify Phase-Gate Goal	Launch Phase-Gate Goal	Current	Steady State Goal
Open critical Severity 1 CRs	0 or downward trend	zero	Zero	zero
Open critical Severity 2 CRs	0 or downward trend	zero	Zero	zero
Total open CR's	<200 severity 3-4 and downward trend	<100 severity 3-4 and downward trend	73 as of 19 June. Defer into 1.1/1.2	<50 severity 3-4
% System tests executed	100% 1st pass	100% 2nd Pass	99.15% Soak up/downgrade and non-critical audio remain	100%
% System tests passed of executed	80% in 1st pass	100% 2nd Pass	95.6% Issues sev 3, 4 and deferred into 1.1/1.2	100%
In-circuit and functional test in use at contract manufacturer	No	Yes	Yes	Yes
HW Group Test in use at contract manufacturer	Yes	Yes	Yes	Yes
DOA claims	n/a	n/a	n/a	< 0.5%
Product return rate	n/a	n/a	n/a	< 0.5%
Trouble call rate	n/a	n/a	n/a	< 0.1%

Defects Found Per Month



cognence inc
Improving Software Economics

Critical & High Severity Defects Detected by Phase



Key CMMI Concepts That Can Help

Scope

- Develop better customer-oriented requirements
- Better manage customer requirements changes via CMB and CRs
- Establish bi-directional traceability from all requirements to project work products and plan to better assess and document impact of changes to the project

Schedule

- Improve planning and tracking for Concept and Planning phases of projects
- Improve tracking (WBS, risks, resources, etc.) of actuals on projects
- Conduct Process and Product Quality Assurance (PPQA) reviews of project plans prior to phase-gate reviews

Cost

- Track and report actual effort and cost throughout the project, not just at phase gates

Quality

- Perform more rigorous design & interface reviews
- Perform more rigorous code inspections and analyze/report code review data
- Implement a Process and Product Quality Assurance (PPQA) capability
- Develop automated unit, integration, and system test harnesses (non-CMMI)

Key Findings Mapped to Product Dev Lifecycle

Program/Project Management

+ PM tracking & visibility through PM System
+ Common product dev life cycle for managing programs
+ Phase Gate Reviews at key points through the product development life
- No standardized WBSs
- Not all project activities are accounted for in project plans and schedules
- Lack of clearly defined roles, responsibilities and program infrastructure at all levels

Concept & Plan Phases

+ Product Management infrastructure in place
+ UI Prototyping, Focus Groups
+ Dedicated Systems Engineering Group
- Churn in early phases leading to early schedule slippage
- Reqs doc too technical, doesn't convey user requirements
- Reqs not complete at Dev Phase-Gate

Develop & Qualify Phases

+ System Test Organization
+ Alpha/Beta Test
+ High Quality Product Delivery to Launch
- Focus on System test to catch defects, versus earlier prevention
- Lack of development connectedness to "success" of ABC
- Changes to customer-level requirements unclear

Process Management & Support

+ Software Configuration Management Environment (ClearCase, ClearQuest)
- Organizational processes are referenced, but not followed as documented
- Project information stored in numerous repositories
- No consistent organization/program/project measurement capability

Thank You!

Roger D. Campbell

(720) 971-7652

Roger_Campbell@cognence.com

cognence inc

Improving Software Economics