

MEASURES FOR EXCELLENCE

PURCHASING SOFTWARE INTENSIVE SYSTEMS USING QUALITY TARGETS

Copyright J.W.E Greene

QUANTITATIVE SOFTWARE MANAGEMENT LTD

7 Rue Fenoux

75015 Paris

Tel: 33-140-431210

Fax : 33-148-286249

Internet: qsm.europe@pobox.com

CompuServe: 100113,3364

www.qsm.com

93 Blythe Road

London W14 0HP

Tel: 44-207-603-9009

Fax : 44-207-602-6008

PURCHASING SOFTWARE INTENSIVE SYSTEMS USING QUALITY TARGETS

INTRODUCTION

Software projects are notorious for being late, over budget and delivered with poor reliability. This motivates the US Department of Defense (DoD) to qualify suppliers based on their CMM maturity level. DoD Airlie guidelines on purchasing software intensive systems lists "16 Critical Software Practices for Performance Based Management". (Ref. 1)

One Airlie best practice is to track defects against quality targets. This practice is becoming fundamental in both Defense and commercial purchasing organisations, especially related to complex systems where a software release is made up from separate sub-system projects. Development organisations are equally motivated to ensure software is delivered with high reliability, high availability and reduced maintenance costs.

Complex software intensive developments are now common in defence, telecommunications, air traffic control, and space. They are recognisable because of their sheer scale. Often development takes between 2– 4 years and costs in excess of 200-400 person years of development effort. A commercial example is found in Mobile Telecommunication product releases. Here a "typical" release consists of at least three sub-systems, all software intensive:

- 1 Transceiver
- 2 Transceiver Control
- 3 Control Centre

Features define the content of a release. The features are realised by mapping the requirements across the sub-systems. Normally this involves building on a large existing software base made up from the existing sub-systems that are modified and extended to engineer the new release.

Each sub-system is developed separately. Then the sub-systems are integrated and validated to provide the final product release. The challenge is to estimate, plan and manage the development of these distributed sub-systems and the total release. An overriding concern is to demonstrate the final release is highly reliable. Customers for the release are motivated to set quality targets and monitor the software reliability as development progresses to confirm the targets are met before accepting delivery.

Software defects arise in the final release validation, not only from the new software but also due to latent defects in the existing code. Substantial regression tests are run to checkout the existing software base that may amount to millions of statements. Requirement changes are common in these developments because of the long lead times involved. This adds to the complexity both at the individual sub-system level and the release.

In summary these complex developments represent major management challenges both to developers and to purchasers. Frequently the purchaser's competitive position depends on delivery (time to market) of all the new

PURCHASING SOFTWARE INTENSIVE SYSTEMS USING QUALITY TARGETS

release features by the scheduled date, within budget and with high reliability. (Ref.2)

Figure 1 illustrates the specification and development of a release where major sub-systems are developed separately and are then integrated and validated.

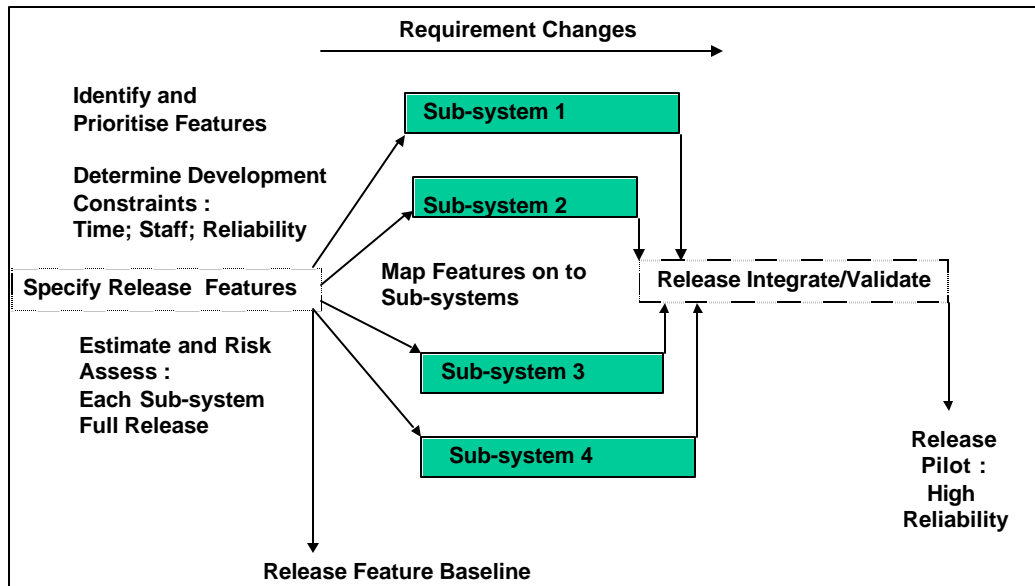


Figure 1: Release- Sub-systems

BACKGROUND TO THE CASE STUDY

The case study is from the point of view of the purchaser who requires software planning and progress data as specified under a formal commercial contract with the supplier. This contract requires the supplier to provide high-level development estimates and plans, on-going progress data and sets delivery acceptance criteria based on quantified quality targets.

The estimate and plan data is first used to check the proposed schedules for each sub-system and the release are realistic and that each development represents value for money. Once accepted the plan data provides the baseline to monitor progress.

The contract requires the supplier to provide progress data every two weeks. The defect information consists of software defect numbers and classes. This defect data is supplied once individual modules within each sub-system are formally handed over by programmers for integration. Additional progress data relates to staff numbers, module status and code production, milestones, and integration and validation test cases.

The supplier of these advanced telecommunications products develops sub-systems at different locations. Each location provides the progress data up to

PURCHASING SOFTWARE INTENSIVE SYSTEMS USING QUALITY TARGETS

the point when the individual sub-system is passed over for the final product release integration and validation.

The case study shows how the defect data at the sub-system and release level is used to provide evidence of the reliability throughout development and to forecast the remaining defects.

RELIABILITY MODEL and THE BASIC DATA

The defect monitoring uses the Rayleigh function to forecast the discovery rate of defects as a function of time throughout the software development process. Empirical evidence shows the Rayleigh model closely approximates the actual profile of defect data collected from software development efforts. Our research at QSM shows there is a solid theoretical basis for its use in software reliability modelling. (Ref. 3) Independent analysis by IBM confirms the Rayleigh function as a sound basis for defect modelling. (Ref. 6)

The generic form of the Rayleigh model is “tuned” using the actual defect data reported. The model then forecasts the defects that remain and the key dates when specific levels of reliability will be achieved. Where very high reliability is required this is the point in time when 99.9 % of the theoretical software defects have been discovered.

Simple extensions of the model provide other useful information. For example, defect priority classes are specified as percentages of the total. This allows the model to predict defects by severity classes over time. The tuning for the defect classes is again made using the actual reported defects and adjusted as development progresses. In the case study the reported software defect classes are:

- 1 Critical
2. Major
3. Minor

INDIVIDUAL SUB-SYSTEM PROGRESS DATA: SOFTWARE DEFECTS

A monthly summary of the progress data from the three sub-systems is shown below in Figure 2. Note that sub-system 1 development begins about 1 year ahead of the other two sub-systems. This is often the case due to a particular sub-system having more features and being more large and complex than other sub-systems.

The defects are reported once the integration tests for each sub-system begin. Each month the columns show the total defects and the breakout in to the three software defect classes.

RELEASE CONSOLIDATED DEFECTS

The data in Figure 2 is consolidated for all sub-systems and is used to track the overall defects for the product release. This continues for each sub-system until it completes and is delivered for final release integration, load tests and validation.

**PURCHASING SOFTWARE INTENSIVE SYSTEMS
USING QUALITY TARGETS**

Defects found during the final release integration are identified by sub-system and communicated back to the developers for fixing. Each defect continues to be classified according to severity.

Month	SUB-SYSTEM 1						SUB-SYSTEM 2						SUB-SYSTEM 3					
	Staff	Code	Defects				Staff	Code	Defects				Staff	Code	Defects			
			Total	Crit	Maj	Min			Total	Crit	Maj	Min			Total	Crit	Maj	Min
1	2																	
2	1	4500																
3	2	4641																
4	2	5522																
5	3	9802																
6	3	16760																
7	4	19833																
8	6	22682																
9	7	30831																
10	9	31642																
11	10	42242																
12	5.5	44402					1	1500										
13	8.5	50611	15		7	8	5	3300					4.5	721				
14	12	53589	38	2	13	23	8	7769					6.5	1537				
15	11	56211	55		26	29	15.5	16126					13	2592				
16	10.5	58193	78		33	45	20	19716	10	2	5	3	14.5	3879	23	1	15	7
17	10	60368	13		7	6	11	19716	35	10	18	7	18	5871	53	6	29	18
18	9.5	62520	18		3	15	12	19716	24	1	15	8	19.5	5871	36	4	24	8
19	8.5	62984	24		6	18	12.5	22334	54	3	21	30	18	6046	37	3	24	10
20	7.5	63322	6		2	4	10	23850	38	6	11	21	15.5	6492	64	6	36	22
21	6	63789	2			2	6.5	25468	63	1	12	40	11	6575	60	1	25	34

Figure 2: Sub-system Progress Data

FORECASTING THE OUTSTANDING DEFECTS

Figure 3 shows the current cumulative defects found by month 21. A total of 750 actual defects are reported. This data is used to tune the defect model and forecast the number of defects that remain to be found and fixed. The results show that a further 600 defects of all classes remain to be found. High reliability for the full release is achieved around month 33.

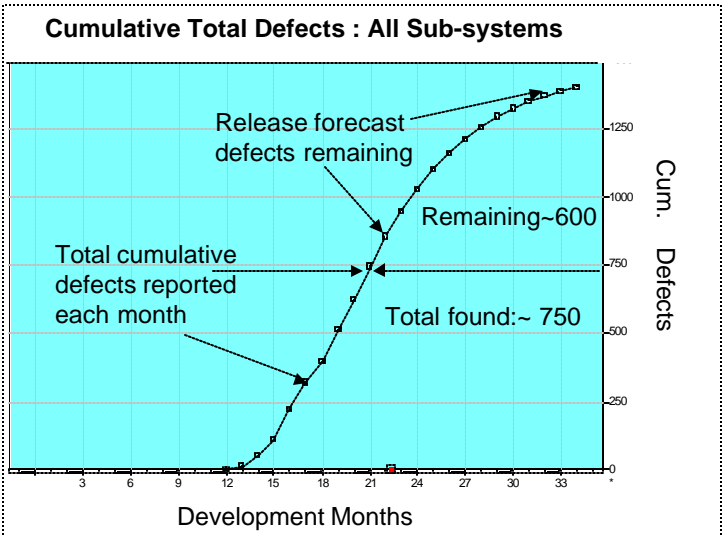


Figure 3: Cumulative Actual Defects and Forecast Remaining Defects

**PURCHASING SOFTWARE INTENSIVE SYSTEMS
USING QUALITY TARGETS**

QUALITY TARGETS: MEAN TIME TO DEFECT

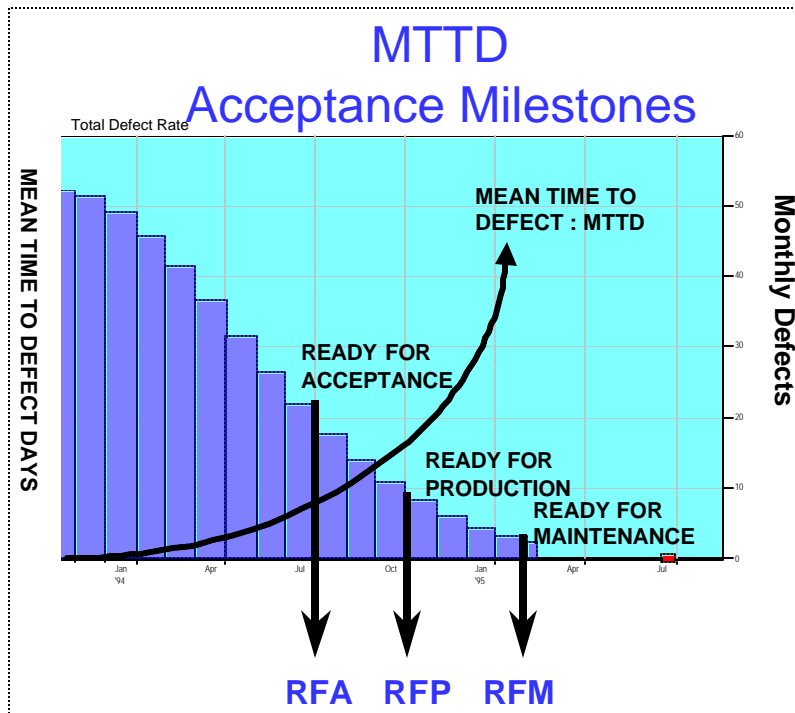
Purchasing sets the contractual quality targets in terms of the software mean time to defect (MTTD). The MTTD is simply the average time between software failures. This is calculated as the reciprocal of the average number of defects within a given period. For instance if 20 defects are found each working month of 20 days then on average the MTTD is one day between each defect.

No milestone dates are set for delivery; acceptance (and payment) is linked to MTTD.

Defects are tracked (and forecast) throughout the development of each sub-system as well as the entire release. Release acceptance begins when concrete evidence exists that the full release meets the MTTD acceptance criteria. Quality continues to be improved by the supplier after the first delivery.

In practice three MTTD quality targets are set (Figure 4 illustrates):

1. RFA: Ready for Acceptance, marks the first delivery to the purchaser to start a factory acceptance test (typically this is a MTTD of 5 days)
2. RFP: Ready for Production, is the point at which the release is introduced at a single location as a pilot (MTTD 15 days)
3. RFM: Ready for Maintenance, during the pilot the final defects are found and fixed. This marks the start of rolling out the new highly reliable release. (MTTD 30 days)



**Figure 4:
MTTD
Quality
Targets**

OBSERVATIONS AND CONCLUSIONS

The results shown here for a large-scale software intensive development demonstrate that it is practical to treat the individual sub-systems as software developments in their own right. The sub-systems are consolidated and their final integration and validation are included as part of the full release development. The full release behaves as a single large development.

Reliability depends on finding and fixing the software defects originating from the sub-systems. Tracking the sub-system defect behaviour reveals the contribution that factors such as time pressure and software size make to the defects found in each sub-system and later during the release integration and validation tests. These insights offer ways to improve reliability and avoid the premature delivery of the sub-systems and the release (Ref. 5). Software reliability quality targets, as measured by the MTTD, determine when release delivery takes place.

In practice the management techniques outlined here form part of a software control office. (Ref. 4) This office continuously evaluates and reports on all current developments. Management ensures that completed releases are only accepted and payments made when specific quality targets are achieved.

Future release plans are evaluated to determine if these are realistic (Ref.2). Disasters are avoided by sizing the features to determine what can be developed within specific constraints and the corresponding level of risk assessed. In this way a realistic baseline-planning estimate is produced. The contractual progress data is used to detect variance against the quantified baseline plan and initiates immediate corrective action.

These two aspects, a realistic estimate and continuous visibility during development, are fundamental to the successful management of software projects by development and purchasing organisations. While true for every software development this becomes even more important when complex software intensive systems are involved.

Jim Greene is Managing Director of Quantitative Software Management Europe in Paris, France: telephone 33-140431210; fax 33-148286249. He has over 35 years experience in software engineering, with a particular interest in management methods used by development and purchasing organisations based on the quantification of software development.

Ref.1. DoD Airlie guidelines "16 Critical Software Practices for Performance Based Management" see www.spmn.com

Ref. 2. Geerhard W. Kempff "Managing Software Acquisition" *Managing System Development* July 1998 Applied Computer Research Inc. P.O. Box 82266, Phoenix, AZ, USA.

Ref. 3. : L.H. Putnam "Measures For Excellence: Reliable Software, On Time, Within Budget:" Prentice Hall New York 1992

Ref.4 J.W.E. Greene, The Software Control Office EC2 Software Engineering Conference Toulouse 1991

Ref. 5: J.W.E. Greene "Avoiding the Premature Delivery of Software" QSM paper see www.qsm.com 1996

Ref. 6 : S.H. Kan Modeling and Software Development Quality IBM Systems Journal Vol 30 No.3 1991