

A Change Analysis Process to Characterize Software Maintenance Projects

Lionel C. Briand, Victor R. Basili, Yong-Mi Kim
Computer Science Department and Institute for Advanced Computer Studies
University of Maryland, College Park, MD, 20742

Donald R. Squier
Computer Sciences Corporation
System Sciences Division
Lanham-Seabrook, MD, 20706

Abstract

In order to improve software maintenance processes, we need to be able to first characterize and assess them. This task needs to be performed in depth and with objectivity since the problems are complex. One approach is to set up a measurement program specifically aimed at maintenance. However, establishing a measurement program requires that one understands the issues and is able to characterize the maintenance environment and processes in order to collect suitable and cost-effective data. Also, enacting such a program and getting usable data sets takes time. A short term substitute is needed.

We propose in this paper a characterization process aimed specifically at maintenance and based on a general qualitative analysis methodology. This process is rigorously defined in order to be repeatable and usable by people who are not acquainted with such analysis procedures. A basic feature of our approach is that maintenance changes are analyzed in order to understand the flaws in the change process. Guidelines are provided and a case study is shown that demonstrates the usefulness of the approach.

1 Introduction

As described in [HV92], numerous factors can affect software maintenance quality and productivity, e.g., the maintenance personnel experience profile and training, the way knowledge about the maintained systems is managed and conveyed to the maintainers and users, the maintenance organization, processes and standards in use, the initial quality of the software source code and its documentation. This last factor involves concepts such as self-descriptiveness, modularity, simplicity, consistency, expandability, and testability.

Because of the complexity of the phenomena studied, it is difficult for maintenance organizations to identify and assess the issues they have to address in order to improve the quality and productivity of their maintenance projects. Each project may encounter specific difficulties and situations that are not necessarily alike across all the organization's

maintenance projects. This may be due in part to variations in application domain, size, change frequency, and/or schedule/budget constraints. As a consequence, each project has first to be analyzed as a separate entity even if, later on, commonalities across projects may require similar solutions for improvement. Informally interviewing the people involved in the maintenance process would be unlikely to help determine accurately the real issues. Maintainers, users and owners would likely each give very different, and often contradictory, insights on the issues due to a somewhat incomplete and biased perspective.

Establishing a measurement program integrated into the maintenance process is likely to help any organization achieve an in-depth understanding of its specific maintenance issues and thereby lay a solid foundation for maintenance process improvement [RUV92]. However, defining and enacting a measurement program may take time and a short term, quickly operational substitute is needed in order to obtain a first quick insight, at low cost, into the issues to be addressed. Furthermore, defining efficient and useful measurement procedures first requires a characterization of the maintenance environment in which measurement takes place, i.e., organization structures, processes, issues, risks, etc. [BR88].

This paper presents a qualitative and inductive analysis methodology for performing objective project characterizations and thereby identifying their specific problems and needs. It is an implementation of the general qualitative analysis methodology defined in [SS92]. It encompasses a set of procedures which aids the determination of causal links between maintenance problems and flaws of the maintenance organization and process. Thus, a set of concrete steps for maintenance quality and productivity improvement can be taken based on a tangible understanding of the relevant maintenance issues. Moreover, this understanding provides a solid basis on which to define relevant software maintenance models and metrics.

Section 2 describes the phases, techniques and guidelines composing the methodology. Section 3 presents a case study of an orbit determination system maintained by the Flight Dynamics Division (FDD) of the NASA Goddard Space Flight Center for the last 26 years and still used daily for most operating satellites

This work was supported in part by NASA grant NSG-5123

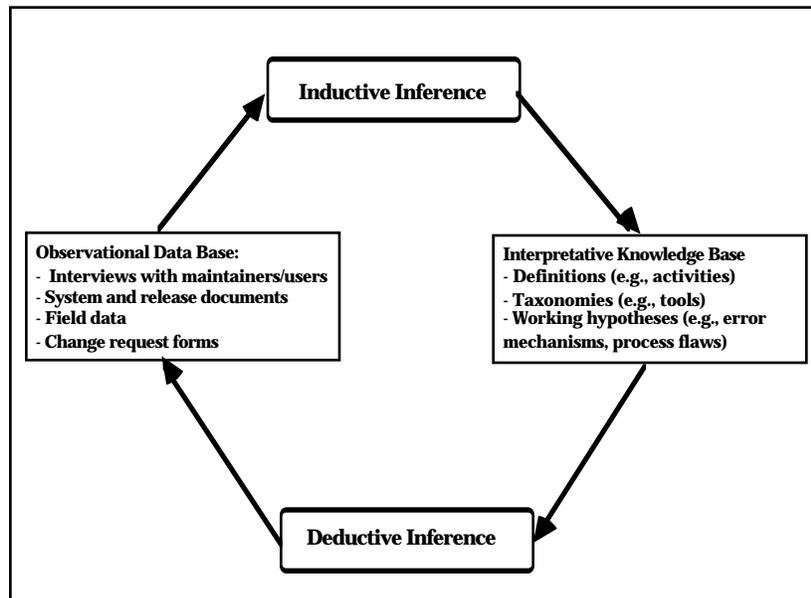


Figure 1: Qualitative Analysis Process for Software Maintenance

(GTDS: Goddard Trajectory Determination System). This study takes place in the framework of the NASA Software Engineering Laboratory (NASA-SEL), an organization aimed at improving FDD software development processes based on measurement and empirical analysis. Recently, responding to the growing cost of software maintenance, the SEL has initiated a program aimed at characterizing, evaluating and improving its maintenance processes. This paper is a first step in this direction. Section 4 outlines the main conclusions of the case study and the future research directions.

2 Causal Analysis of Maintenance Problems

In this section, we present a (mainly) qualitative methodology that allows for an in-depth characterization of maintenance projects at a relatively low cost. However, this approach could be easily augmented to integrate data collection and analysis and could thus provide more quantitative information (but at a higher cost).

2.1 A Qualitative Analysis Process

This characterization process is essentially an instantiation of the generic qualitative analysis process defined in [SS92]. Figure 1 illustrates at a high level our maintenance specific analysis process. It is a combination of both inductive and deductive inferences. Inductive inferences are made from the collected information. Deductive inferences occur when experimentally validating and refining our taxonomies, process models, organizational models and working

hypotheses. These deductive inferences then serve to refine the data collection process, which leads to refined and revised inductive inferences. The process continues in an iterative fashion.

We present below a general description of the process involved in preparing and performing characterizations of maintenance projects. Maintenance is defined here as any kind of enhancement, adaptation or correction performed on the software system, once in operation. At the highest level of abstraction, parts of this process are not specific to maintenance and could be used for development. However, the taxonomies and guidelines developed to support this process and presented in Section 2.2 are specifically aimed at maintenance.

Step 1 focuses on defining the organizational structures, i.e., organization entities, their communication channels and information flows. The process of producing a new release is then described and modeled in Step 2. It is important to note that we do not address here the issues related to emergency bug fixing procedures but only those relevant to regular product releases that go into configuration management. Step 3 maps generic activities into the release process in order to specify the type of work performed at each stage of the process. Then, a release (or several) has to be selected in order to define the set of changes on which the analysis will be performed (Step 4). In Step 5, relying on the work performed in Steps 1-3, information about the changes is collected and analyzed. Step 6 summarizes and abstracts from the results obtained in Step 5.

Although the steps are defined sequentially, they are really iterated within and across steps. As we learn more about the organization, we continue to refine the characterization models. The organizational and process models produced should include enough detail to allow Step 5 to be performed, but should not be so detailed as to obscure the maintenance process itself. We now define the steps in more detail:

Step 1 Identify the organizational entities with which the maintenance team interacts and the organizational structure in which maintainers operate.

1.1 Identify distinct organizational entities, i.e., what are the distinct teams involved in the maintenance project? Usually, besides the maintainers themselves, the following entities are encountered: users, owners, QA team, configuration control team, change control board. However, their roles and prerogatives can differ significantly.

1.2 Characterize the working environment of each entity, i.e., support tools (see tool taxonomy in Section 2.2), internal organizational structure.

1.3 Characterize information flows between entities, i.e., what is the type (and amount when data is available) of information, documentation, source code and other software artifacts flowing between organizational entities?

Step 2 Identify the phases involved in the creation of a new system release.

2.1 Identify the phases as defined in the environment studied. At this stage, it is important not to map an *a priori* external/generic maintenance process model and vocabulary.

2.2 Each artifact (e.g., document, source code) which is input or output of each phase has to be determined and its content carefully described (see document taxonomy in Section 2.2).

2.3 The personnel in charge of producing and validating the output artifacts of each phase have to be identified and located in the organizational structure defined in Step 1.

Step 3 Identify the generic activities involved in each phase.

3.1 Select (from the literature [C88, BC91]) or define a taxonomy of generic activities based on widely accepted definitions and used in the maintenance process. As a guideline, such a taxonomy is proposed in the next section.

3.2 Map these activities into each phase by reading the technical documents produced and interviewing the technical project leaders and maintainers about their real work habits. If possible, collect effort data for each

activity so that the importance of each activity in each phase can be assessed somewhat quantitatively.

Step 4 Select one or several past releases for analysis.

We need to select releases on which we can analyze problems as they are occurring and thereby better understand process and organization flaws. However, because of time constraints, it is sometimes more practical to work on past releases. We present below a set of guidelines for selecting them:

- Recent releases are preferable since maintenance processes and organizational structure might have changed and this would make analyses based on old releases somewhat irrelevant.

- Some releases may contain more complete documentation than others. Documentation has a very important role in detecting problems and cross-checking the information provided by the maintainers.

- The technical leader(s) of a release may have left the company whereas another release's technical leader may still be contacted. This is a crucial element since, as we will see, the causal analysis process will involve project technical leader(s) and, depending on his/her/their level of control and knowledge, possibly the maintainers themselves.

Step 5 Analysis of the problems that occurred while performing the changes of the selected releases.

For each change (i.e., error correction, enhancement, adaptation) in the selected release(s), the following information should be acquired by interviewing the maintainers and/or technical leaders and by reading the related documentation (e.g., release documents):

I1. Determine the difficulty or error-proneness of the change.

I2. Determine whether the change difficulty could have been alleviated or the error(s) resulting from the change avoided and how?

I3. Evaluate the size of the change (e.g., # components, LOCs changed, added, removed).

I4. Assess discrepancies between initial & intermediate planning and actual effort / time.

I5. Determine the human flaw(s) (if any) that originated the error(s) or increased the difficulty related to the change. A taxonomy of human errors is proposed in Section 2.2.

I6. Determine the maintenance process flaws that led to the identified human errors (if any). A taxonomy of maintenance process flaws is proposed in Section 2.2.

I7. Try to quantify the wasted effort and/or delay generated by the maintenance process flaws (if any).

The knowledge and understanding acquired through steps 1-3 is necessary in order to understand, interpret and formalize the information of type I2, I5 or I6. As a guide for conducting interviews, templates of questions will be provided in Section 2.2.

Step 6 Establish the frequency and consequences of problems due to flaws in the organizational structure and the maintenance process by analyzing the information gathered in Step 5.

Based on these results, further complementary investigations (e.g., measurement-based), related to specific issues that have not been fully resolved by the qualitative analysis process, should be identified. Moreover, a first set of suggestions for maintenance process improvement should be devised.

For those steps which are iterative, we map the appropriate step back into the qualitative analysis process (Figure 1) showing how our characterization process fits into the general methodology. In this context, a step usually corresponds to a set of iterations of the qualitative analysis process. In each step, the *input* defines the *Observational Database* (ODB), the *output* contains the resulting characterization models that go into the *Interpretative Knowledge Base* (IKB), and the *validation* procedure helps verify the correctness of the characterization models. The pieces of information which compose the *ODB* are given in decreasing order of importance at each step. The order and content of the *ODB* varies at each step since the analysis focus shifts progressively [SS92].

Step 1: Model organizational structures

Input: maintenance standards definition document, interviews, sample of release documents, organization chart

Output: organization model (roles, agents, teams, information flow, etc.)

Validation:

- . Are all the standard documents and artifacts included in the modeled information flow?
- . Do we know who produces, validates, and certifies the standard documents and artifacts?
- . Are all the people referenced in the release documents a part of the organization scheme?

Steps 2, 3: Model process and map activities into process phases

Input: maintenance standards definition document, interviews, release documents, organization model

Output: process model

Validation:

- . Are all the people in the process model a part of the organization scheme?

. Do the documents and artifacts included in the process model match those of the information flow of the organization model?

. Is the mapping between activities and phases complete, i.e., exhaustive set of activities, complete mapping?

. Are the taxonomies of maintenance tools, methods, and activities adequate, i.e., unambiguous, disjoint and exhaustive classes?

Step 5: Perform causal analysis

Input: interviews, change request forms, release documents, organization model, process model, maintenance standards definition document

Output: causal analysis

Validation:

. Are the taxonomies of errors and maintenance process flaws adequate, i.e., unambiguous, disjoint and exhaustive classes? This is checked against actual change data and validated during interviews with maintainers.

2.2 Guidelines and Taxonomies

This section presents a set of guidelines aimed at facilitating the characterization process described in the previous section. These guidelines are mainly composed of taxonomies distinguishing maintenance activities, errors and maintenance process flaws. In addition, a set of questions which can be used during maintainers' interviews and for each change is provided.

Step 1: Identify organizational entities

Taxonomies of Maintenance Tools and Methods (Step 1.2)

The maintenance tools and methods available to maintainers can be used to understand the maintenance process, and identify potential sources of problems. The following paragraphs represent the first level of abstraction of taxonomies of environment characteristics that should be used to characterize the change framework:

- . Maintenance tools: impact analysis and planning tool; tools for automated extraction and representation of control and data flows; debugger; cross-referencer; regression testing environment (data generation, execution, and analysis of results); information system linking documentation and code.
- . Maintenance methods are characterized by the following taxonomy: rigorous impact analysis, planning, and scheduling procedures; systematic and disciplined update procedures of the user and system documentation; communication channels and procedures with the users.

A Taxonomy of Maintenance Documentation (Step 1.3)

The type of documentation related to a software system, which may be available to maintainers, can be defined by a generic taxonomy as shown below. Documentation has been described as one of the most important factors affecting the maintainability of a software system [HA93, P94]. This is why it is important to define precisely what should be contained in a complete set of documentation (either on-line or off-line) for maintenance. Also, when some of these documents appear to be missing, potential sources of maintenance problems may be identified. Based on the literature [BC91] on the subject and our own experience, we propose the following taxonomy:

- . Product-related: software requirements specifications; software design specifications; software product specifications
- . Process-related: test plans; configuration management plan; quality assurance plan; software development plan
- . Support-related: software user's manual; computer systems operator's manual; software maintenance manual; firmware support manual

Step 3: Identify the generic activities involved in each phase.

Generic Description of Maintenance Activities (Step 3.1)

<i>Acronym</i>	<i>Activity</i>
DET	Determination of the need for a change
SUB	Submission of change request
UND	Understanding requirements of changes: localization, change design prototype
IA	Impact analysis
CBA	Cost/benefit analysis
AR	Approval/rejection/priority, assignment of change request
SC	Scheduling/planning of task
CD	Change design
CC	Code changes
UT	Unit testing of modified parts i.e., has the change been implemented?
IC	Unit Inspection, Certification i.e., has the change been implemented properly and according to standards?
IT	Integration testing, i.e., does the changed part interface correctly with the reconfigured system?
RT	Regression testing, i.e., does the change have any unwanted side effects?
AT	Acceptance testing

i.e., does the new release fulfill the system requirements?

USD	Update system and user documentation
SA	Checking conformance to standards; quality assurance procedures
IS	Installation
PIR	Post-installation review of changes
EDU	Education/training regarding the application domain/system

All these activities usually contain an overhead of communication (meeting and release document writing) with owners, users, management hierarchy and other maintainers, which should be estimated. This is possible through data collection or by interviewing maintainers (e.g., Delphi method).

Step 5: Perform causal analysis

Questions asked for each change in selected release(s) (Items 11-14)

The following list describes a set of questions for which answers can be provided by maintainers and/or release standard documents. These questions attempt to capture the information necessary for the identification of maintenance process flaws.

1 - Description of the change

1.1 Localization

- . subsystem(s) affected
- . module(s) affected
- . inputs/outputs affected

1.2 Size

- . LOCs deleted, changed, added
- . Modules examined, deleted, changed, added

1.3 Type of change

- . Preventive changes: improvement of clarity, maintainability or documentation.
- . Enhancement changes: add new functions, optimization of space/time/accuracy
- . Adaptive changes: adapt system to change of hardware and/or platform
- . Corrective changes: corrections of development errors.

2 - Description of the change process

2.1 effort, elapsed time

2.2 maintainer's expertise and experience

- . How long has the person been working on the system?
- . How long has the person been working in this application domain?

2.3 Did the change generate a change in any document?
Which document(s)?

3 - Description of the problem

3.1 Were some errors committed?

- . Description of the errors (see taxonomies below)
- . Perceived cause of the errors:
maintenance process flaw(s)

3.2 Difficulty

- . What made the change difficult?
- . What was the most difficult activity associated with the change?

3.3 How much effort was wasted (if any) as a result of maintenance process flaws?

3.4 What could have been done to avoid some of the difficulty or errors (if any)?

Taxonomies of human errors (Item I5)

Note that we are exclusively referring to errors occurring during the maintenance process, not errors resulting from the development process.

. Error origin: when did the misunderstanding occur?

- . Change requirements analysis
- . Change localization analysis
- . Change design analysis
- . Coding

. Error domain: what caused it?

. Lack of application domain knowledge:
operational constraints (user interface, performance), mathematical model

. Lack of system design or implementation knowledge:
data structure or process dependencies, performance or memory constraints, module interface inconsistency

- . Ambiguous or incomplete requirements
- . Language misunderstanding <semantic, syntax>
- . Schedule pressure
- . Existing uncovered fault
- . Oversight.

Determining the origin and cause of the errors will help determine their possible causal relationships to maintenance process flaws in the taxonomy presented below.

Taxonomy of Maintenance Process flaws (Item I6)

. Organizational flaws:

- . communication: interface problems, information flow "bottlenecks" in the communication between the maintainers and the

- . users
- . management hierarchy
- . quality assurance (QA) team
- . configuration management team

. roles:

- . prerogatives and responsibilities are not fully defined or explicit
- . incompatible responsibilities, e.g., development and QA

. process conformance: no effective structure for enforcing standards and processes

. Maintenance methodological flaws

- . Inadequate change selection and priority assignment process
- . Inaccurate methodology for planning of effort, schedule, personnel
- . Inaccurate methodology for impact analysis
- . Incomplete, ambiguous protocols for transfer, preservation and maintenance of system knowledge
- . Incomplete, ambiguous definitions of change requirements
- . Lack of rigor in configuration (versions, variations) management and control
- . Undefined / unclear regression testing success criteria.

. Resource shortages

- . Lack of financial resources allocated, e.g., necessary for preventive maintenance, unexpected problems unforeseen during impact analysis.
- . Lack of tools providing technical support (see previous tool taxonomy)
- . Lack of tools providing management support (i.e., impact analysis, planning)

. Low quality product(s)

- . Loosely defined system requirements
- . Poor quality design, code of maintained system
- . Poor quality system documentation
- . Poor quality user documentation

. Personnel-related issues

- . Lack of experience and/or training with respect to the application domain
- . Lack of experience and/or training with respect to the system requirements (hardware, performance) and design
- . Lack of experience and/or training with respect to the users' operational needs and constraints

In order to demonstrate the feasibility and usefulness of the above approach, we present the following case study.

3 A Case Study

This case study is intended to provide actual examples and results of the change causal analysis process described in previous sections. We first present the maintained system used as a case study. Then, the specific maintenance organization and process are described in detail according to the template provided in Section 2.1. Examples of change causal analyses are shown and the lessons learned resulting from this analysis process are presented.

3.1 System History and Description

GTDS is a 26 year old, 250 KLOC, FORTRAN orbit determination system. It is public domain software and, as a consequence, has a very large group of users all over the world. Usually, 1 or 2 releases are produced every year in addition to mission specific versions that do not go into configuration management right away (but are integrated later on to a new version by going through the standard release process). Like most maintained software systems, very few of the original developers are still present in the organization, but the turnover of the maintenance team is low compared to other maintenance organizations. However, turnover still remains a crucial issue in this environment.

3.2 Modeling of the Maintenance Organization and Processes

During the process of building a new release of GTDS, different organizational entities interact in different ways. By performing Step 1 of the characterization process described in Section 2.1, two types of entities and five types of interactions (i.e., differentiated according to the purpose of the information flow) were identified.

The entities, teams and groups, are represented in Figure 2 by boxes and ellipses, respectively. Teams are persistent organizational structures; groups are composed of members of several different teams, and are dynamic entities in the sense that they only exist when group members meet. These groups have been designed to facilitate communication between teams and decision making.

In the five interaction types identified, information was used for the following purposes:

- . decision: decision based on information provided
- . review: review of documents
- . approval: approval of documents or plans
- . transformation: supplied information product is transformed into another information product
- . information: dissemination of information

Teams:

- . *Testers:* they present acceptance test plans, perform acceptance test and provide change requests to the maintainers when necessary.
- . *Owners/Users:* they suggest, control and approve performed changes.
- . *Product Assurance Organization (PAO):* They control maintainers' work, e.g., conformance to standards, attend release meetings, audit delivery packages. They have a different management from the maintenance team.
- . *Configuration Management (FDCM):* They integrate updates into the system. Coordinate the production and release of versions of the system. Provide tracking of change requests.
- . *Maintenance management:* They grant preliminary approvals of maintenance change requests and release definitions.
- . *Maintainers:* They analyze changes, make recommendations, perform changes, perform unit and change validation testing after linking the modified units to the existing system, perform validation and regression testing after they get back the recompiled system from the *FDCM* team.
- . *Configuration Control Board and Configuration Management Office (CCB/CMO):* They are officially responsible for all changes to configured software and the allocated budget. Their goal is to ensure that the production of new releases is consistent with the long-term goals of the organization. It is composed of high-level managers.

Groups:

- . *Software Management Planning Board (SMPB):* Their main goal is to address management issues that run across maintenance projects. For example, they help resolve conflicts between owners and maintainers and review release planning documents. Also, they allow task leaders and higher level managers to exchange relevant information about the evolution of their respective systems. However, the *SMPB* has no official function. The board is composed of the task leader, section manager, department manager, and operation manager.
- . *GTDS user's group:* It is a forum for discussion of technical issues but has no official function. It is composed of users, maintainers, and testers.

The process described below represents our understanding of the working process for a release of GTDS and the mapping into standard generic activities. This combines the information gained from Steps 2 and 3 of the characterization process. Phases, their associated inputs/outputs and activities are presented below. Activity acronyms are used as defined in Section 2.2. In this case, each phase milestone in a release is represented by the discussion, approval and distribution of a specific release document.

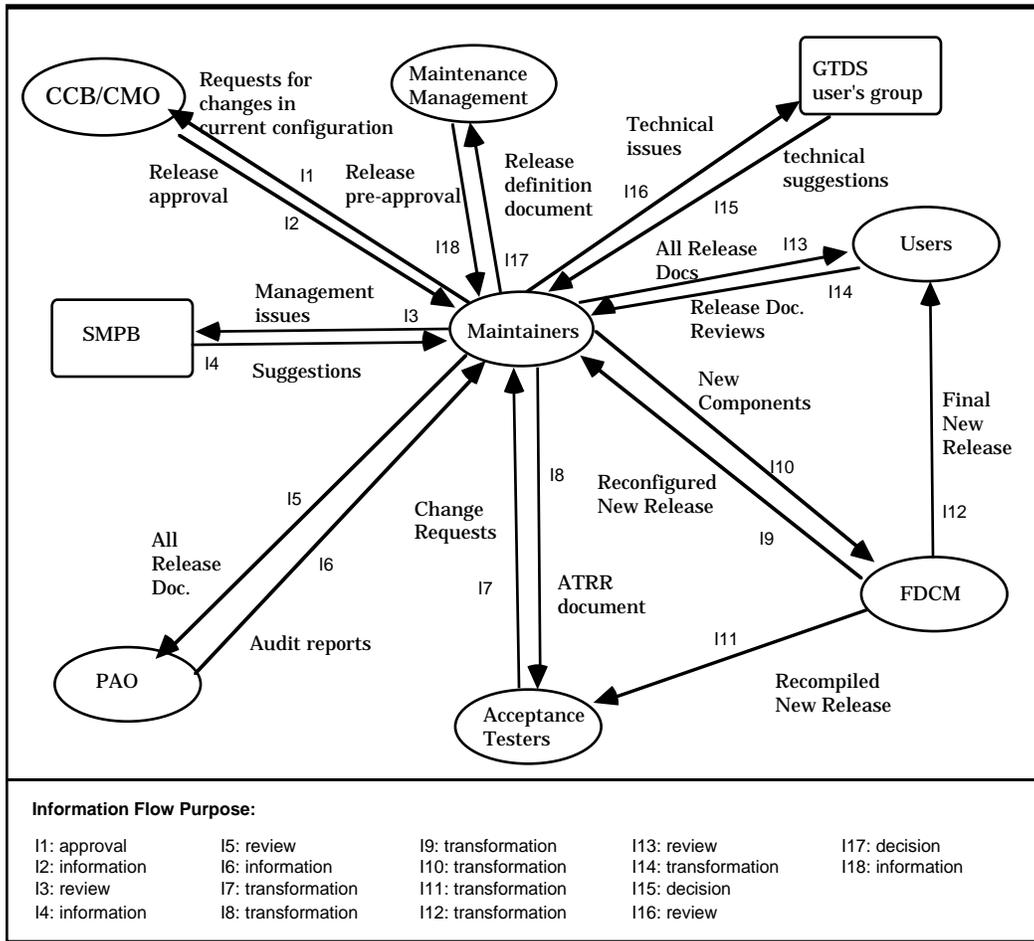


Figure 2: Information Flow within the Maintenance Organization

Phase 1. Change analysis

Input: change requests from software owner and priority list

Output: Release Content Review (RCR) document which contains change design analysis, prototyping, and cost/benefit analysis that may result in changes in the priority list that will be discussed with the software owner/user.

Activities: UND, IA, CBA, CD, some CC, UT and IT for prototyping

Phase 2. RCR meeting

Input: draft of Release Content Review document proposed by maintainers is discussed, i.e., change priority, content of release.

Output: Updated Release Content Review document

Activities: AR, SA (QA engineers are reviewing the release documents and attending the meeting)

Phase 3. Solution analysis

Input: updated Release Content Review document

Output: devise technical solutions based on prototyping analysis they performed in Step 1, Release Design Review (RDR) document.

Activities: SC, CD, CC, UT (prototyping), (preparation of test strategy for) IT (based mainly on functional testing: equivalence partitioning)

Phase 4. RDR meeting

Input: RDR documentation

Output: approved (and possibly modified) RDR documentation

Activities: review and discuss CC, UT, (plan for) IT, SA

Phase 5. Change implementation and test

Input: RDR and prototype solutions (phases 1, 3)

Output: changes are completed and unit tested; change validation test is performed with new reconfigured system (integration test); formal inspections are performed (when quality of code and design allows it) on new or extensively modified components; some (usually superficial) regression testing is performed on

the new system to insure minimal quality before AT; a report with the purpose of demonstrating that the system is ready for AT is produced: Acceptance Test Readiness Review document (ATRR)

Activities: CC, UT, IC, IT, RT, USD, SA

Phase 6. ATRR meeting

Input: Acceptance Test Readiness Review document

Output: The changes are discussed and validated and the testing strategy used is discussed. The acceptance test team presents its acceptance testing plan.

Activities: review the current output of IT, SA

Phase 7. Acceptance test

Input: the new GTDS release and all release documentation

Outputs: A list of Software Change Requests (SCRs) is provided to the maintainers. These changes correspond to inconsistencies between the new reconfigured release and the general system requirements.

Activities: UND, RT, AT

Steps 1, 2, and 3 of our characterization process required several iterations before there was sufficient validation of the resulting characterization of the organization, phases and documents. As part of Step 2, for each of the standard documents generated for the studied GTDS releases, we determine who produces it, who approves it, and what additional relevant information and data they contain. When doing so, we have to look for possible inconsistencies between the organization model (Step 1) and the identified producers/approvers of the documents.

. Document 1: Release Content Review (RCR):

Producer: maintenance team

Approvers: users, maintenance management, CCB

Content:

- . change requirement description
- . description of defect (if any) that originated the change
- . design of a prototype solution
- . schedule, effort plans
- . impact analysis assessment

. Document 2: Release Design Review (RDR):

Producer: maintenance team

Approvers: users, CCB

Content:

- . identification of modified units
- . proposed definitive solution
- . rough cost/schedule estimates

- . testing guidelines: mainly equivalence partitioning classes
- . definition of test success criteria

. Document 3: Acceptance Test Readiness Review (ATRR):

Producers: maintenance team, acceptance test team (test plan)

Approvers: CCB, testers

Content:

- . results of test cases and benchmarks (regression testing)
- . screen printouts, short reports
- . acceptance test plan

. Document 4: Delivery package:

Producer: maintenance team

Approvers: CCB

Content:

- . cause of error (if any)
- . effort breakdown: analysis, design, code, test
- . number of components examined, modified, added, deleted.
- . number of LOCs modified, added, deleted

As specified in Step 4 of our process, we selected a release for analysis. This release was quite recent, most of the documentation identified in Step 2 was available, and most importantly, the technical leader of the release was available for additional insights and information.

Step 5 involved a causal analysis of the problems observed during maintenance and acceptance test of the releases studied. These problems were linked back to a precise set of issues belonging to taxonomies presented in Section 2.2. Figure 3 summarizes Step 5 as instantiated for this case study.

In order to illustrate Step 5, we provide below an example of causal analysis for *one* of the changes in the selected release. Implementation of this change resulted in 11 errors that were found by the acceptance test team, 8 of which had to be corrected before final delivery could be made. In addition, a substantial amount of rework was necessary. Typically, changes do not generate so many subsequent errors, but the flaws that were present in this change are representative of maintenance problems in GTDS. In the following paragraphs, we discuss only *two* of the errors generated by the change studied.

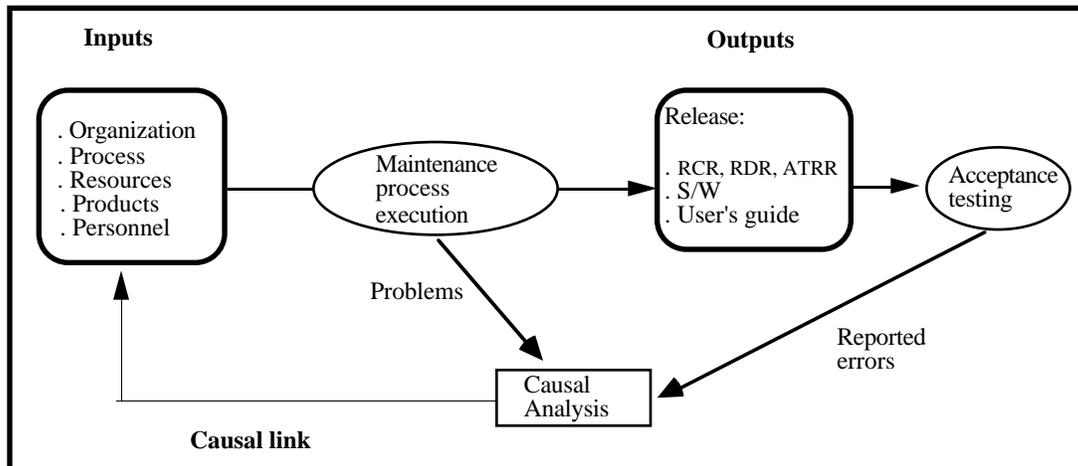


Figure 3: Causal Analysis in GTDS

- . Increased difficulty related to change (rework)

. *Description:* Initially, users requested an enhancement to existing GTDS capabilities (change 642). The enhancement involved vector computations performed over a given time span. This enhancement was considered quite significant by the maintainers, but users failed to supply adequate requirements and did not attend the RCR meeting. Users did not report their dissatisfaction with the design until ATRR meeting time, at which time requirements were rewritten and maintainers had to perform rework on their implementation. This change took a total of 3 months to implement, of which at least 1 month was attributed to several flaws in the process.

. *Maintenance process flaw(s):* organizational: a lack of clear definitions of the prerogatives/duties of users with respect to release document reviews and meetings (roles), and a lack of enforcement of the release procedure (process conformance); maintenance methodological flaw: incomplete, ambiguous definitions of change requirements.

- . Errors caused by change 642

The implementation of the change itself resulted in an error (A1044) found at the acceptance test phase. When the correction to A1044 was tested, an error (A1062) was found that could be traced back to both 642 and A1044.

A1044

. *Description:* Vector computations at the endpoints of the time span were not handled correctly. But in the requirements it was not clear whether the endpoints should be considered when implementing the solution.

- . *Error origin:* change requirement analysis

- . *Error domain:* ambiguous and incomplete requirements

. *Maintenance process flaw(s):* organizational: communication between users and maintainers, due in part to a lack of defined standards for writing change requirements; maintenance methodological flaw: incomplete, ambiguous definitions of change requirements.

A1062

. *Description:* One of the system modules in which the enhancement change was implemented has two processing modes for data. These two modes are listed in the user manual. When run in one of the two possible processing modes, the enhancement generated a set of errors, which were put under the heading A1062. At the phase these errors were found, the enhancement had already successfully passed the tests for the other processing mode. The maintainer should have designed a solution to handle both modes correctly.

- . *Error origin:* change design analysis.

. *Error domain:* lack of application domain knowledge.

. *Maintenance process flaw(s):* personnel-related: lack of experience and/or training with respect to the application domain.

The next section presents in detail the results of performing Step 6.

3.3 Lessons Learned about the Studied Maintenance Project

The lessons learned are classified according to the taxonomy of maintenance flaws defined in Section 2.2. By performing an overall analysis of the change causal

analysis results (Step 6), we abstracted a set of issues classified as follows:

Organization

- . There is a large communication cost overhead between maintainers and users, e.g., release standard documentation, meetings, management forms. In an effort to improve the communication between all the participants of the maintenance process, non-technical, communication-oriented activities have been emphasized. At first glance, this seems to represent about 40% (rough approximation) of the maintenance effort. This figure seems excessive, especially when considering the apparent communication problems (next paragraph).
- . Despite the number of release meetings and documents, disagreements and misunderstandings seem to disturb the maintenance process until late in the release cycle. For example, design issues that should be settled at the end of the RDR meeting keep emerging until acceptance testing is completed.

As a result, it seems that the administrative process and organization scheme should be investigated in order to optimize communication and sign-off procedures, especially between users and maintainers.

Process

- . The tools and methodologies used have been developed by maintainers themselves and do not belong to a standard package provided by the organization. Some ad hoc technology transfer seems to take place in order to compensate for the lack of a global, commonly agreed upon strategy.
- . The task leader has been involved in the maintenance of GTDS for a number of years. His expertise seems to compensate for the lack of system documentation. He is also in charge of the training of new personnel (some of the easy changes are used as an opportunity for training). Thus, the process relies heavily on the expertise of one or two persons.
- . The fact that no historical database of changes exists makes some changes very difficult. Maintainers very often do not understand the semantics of a piece of code added in a previous correction. This seems to be partly due to emergency patching for a mission which was not controlled and cleaned up afterwards (this has recently been addressed), a high turnover of personnel, and a lack of written requirements with respect to performance, precision and platform configuration constraints.
- . For many of the complex changes, requirements are often ambiguous and incomplete, from a maintainer's perspective. As a consequence, requirements are often unstable until very late in the release process. While prototyping might be necessary for some of them, it is not recognized as such by the users and maintainers.

Moreover, there is no well defined standard for expressing change requirements in a way suitable for both maintainers and users.

Products

- . System documentation other than the user's guide is not fully maintained and not trusted by maintainers. Source code is currently the only reliable source of information used by maintainers.
- . GTDS has a large number of users. As a consequence, the requirements of this system are varied with respect to the hardware configurations on which the system must be able to run, the performance and precision needs, etc. However, no requirement analysis document is available and maintained in order to help the maintainers devise optimal change solutions.
- . Because of budget constraints, there is no document reliably defining the hardware and precision requirements of the system. Considering the large number of users and platforms on which the system runs, and the rapid evolution of users' needs, this would appear necessary in order to avoid confusion while implementing changes.

People

- . There is a lack of understanding of operational needs and constraints by maintainers. Release meetings were supposed to address such issues but they seem to be inadequate in their current form.
- . Users are mainly driven by short term objectives which are aimed at satisfying particular mission requirements. As a consequence, there is a very limited long term strategy and budget for preventive maintenance. Moreover, the long term evolution of the system is not driven by a well defined strategy and maintenance priorities are not clearly identified.

As a general set of recommendations and based on the analysis presented in this paper, we suggest the following set of actions:

- . A standard (that may simply contain guidelines and checklists) should be set up for change requirements. Both users and maintainers should give their input with respect to the content of this standard.
- . The conformance to the defined release process should be improved, e.g., through team building, training. In other words, the release documents and meetings should more effectively play their specified role in the process, e.g., the RDR meeting should settle all design disagreements and inconsistencies.
- . Those parts of the system that are highly convoluted as a result of numerous modifications should be redesigned and documented for more productive and reliable maintenance. Technical task leaders should be able to point out the sensitive system units.

4 Conclusion

Characterizing and understanding software maintenance processes and organizations are necessary, if effective management decisions are to be made and if adequate resource allocation is to be provided. Also, in order to plan and efficiently organize a measurement program—a necessary step towards process improvement [BR88]—, we need to better characterize the maintenance environment and its related issues. The difficulty of performing such a characterization stems from the fact that the people involved in the maintenance process, who have the necessary information and knowledge, cannot perform it because of their inherently partial perspective on the issues and the tight time constraints of their projects. Therefore, a well defined characterization process, which is cost-effective, objective, and applicable by outsiders, needs to be devised.

In this paper, we have presented such an empirically refined characterization process which has allowed us to gain an in-depth understanding of the maintenance issues involved in a particular project, the GTDS project. We have been able to gather objective information on which we can base management and technical decisions about the maintenance process and organization. Moreover, this process is general enough to be followed in most maintenance organizations.

However, such a qualitative analysis is a priori limited since it does not allow us to quantify precisely the impact of various organizational, technical, and process related factors on maintenance cost and quality. Thus, the planning of the release is sometimes arbitrary, monitoring its progress is extremely difficult, and its evaluation remains subjective.

Hence, there is a need for a data collection program for GTDS and across all the maintenance projects of our organization. In order to reach such an objective, we will base the design of such a measurement program on the results provided by this study. In addition, we need to model more rigorously the maintenance organization and processes so that precise evaluation criteria can be defined [SB94].

This approach will be used to analyze several other maintenance projects in the NASA-SEL in order to better understand project similarities and differences in this environment. Thus, we will be able to build better models of the various classes of maintenance projects.

Acknowledgments

We are grateful to Steve Condon, Walcelio Melo, Carolyn Seaman, Barbara Swain and Jon Valett for reviewing early drafts of this paper. We also would like to thank Amy Bleich for helping us to analyze the release documents.

References

- [BC91] K. Bennett, B. Cornelius, M. Munro, D. Robson, "Software Maintenance", *Software Engineering Reference Book*, Chapter 20, Butterworth-Heinemann Ltd, 1991
- [BR88] V. Basili and H. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments", *IEEE Trans. Software Eng.*, 14 (6), June, 1988.
- [C88] N. Chapin, "The Software Maintenance Life-Cycle", CSM'88, Phoenix, Arizona, 1988.
- [HA93] C. Hartzman, C. Austin, "Maintenance Productivity: Observations Based on an Experience in a Large System Environment", CASCON'93, Toronto, Canada, 1993
- [HV92] M. Hariza, J.F. Voidrot, E. Minor, L. Pofelski, and S. Blazy, "Software Maintenance: An analysis of Industrial Needs and Constraints", CSM'92, Orlando, Florida.
- [P94] D. Parnas, "Software Aging", ICSE 16th, May 1994, Sorrento, Italy.
- [RUV92] D. Rombach, B. Ulery and J. Valett, "Toward Full Cycle Control: Adding Maintenance Measurement to the SEL", *Journal of systems and software*, May 1992.
- [SB94] C. Seaman, V. Basili, "OPT: An Approach to Organizational and Process Improvement", AAAI 1994 Spring Symposium Series, Stanford University, March 1994.
- [SS92] A. Shelly, E. Sibert, "Qualitative Analysis: A Cyclical Process Assisted by Computer", *Qualitative Analysis*, pp 71-114, Oldenbourg Verlag, Munich, Vienna, 1992