Multiple Dimensions of Concern in Software Testing

Stanley M. Sutton, Jr.
EC Cubed, Inc.
15 River Road, Suite 310
Wilton, Connecticut 06897
ssutton@eccubed.com

1. INTRODUCTION

Software testing is an area of software development in which multiple dimensions of concern are evident. They are reflected in the diversity of issues that are addressed in software testing, encompassing aspects of the products under test, test artifacts, and test processes. Software testing brings these concerns together in a variety of combinations and forms that apply both within testing and across the software life cycle. For this reason, an analysis of software testing from the perspective of multiple dimensions of concerns is especially illuminating. Conversely, the examination of software testing illustrates many aspects of the concern-oriented view of software in general.

The view of software testing in terms of multiple dimensions of concern is developed as follows. Section 2 presents three motivating scenarios based on testing experiences in the context of commercial software development. Section 3 describes the dimensions of concern that are present in these scenarios. Sections 4 to 6 address evidence for cross-cutting concerns, simultaneous overlapping concerns, and multiple levels and scopes of concerns. Finally, Section 7 presents a discussion of decomposition and composition based on concerns, including topics for research.

2. MOTIVATING SCENARIOS

This section presents three scenarios to motivate the analysis of software testing in terms of multiple dimensions of concern.

Scenario 1: Develop and apply general standards for graphical user interfaces (GUIs)

Prior to the development of a new family of products, general standards for graphical user interfaces were defined. However, after the product family has been expanded and has progressed through several versions, the original GUI standards are found to be outdated. Thus it is necessary to formulate new GUI standards. A multiplicity of concerns are evident in the following areas:

- The revised GUI standards apply to multiple specific products, possibly to different versions in different stages of development
- The revised GUI standards may draw on several sources: the original GUI standards, other (e.g., customer) standards, product requirements specifications and designs, the actual GUIs as they have been implemented
- The revised GUI standards address many concerns in the appearance and function of the GUIs: overall look-and-feel, common functions, the organization of menus, toolbars, and panels of various types, the degree of user control over the organization and behavior of the interface, presentation and handling of exceptions and alerts, typographical and linguistic conventions for labels and messages, and so on
- The revised GUI standards relate to and must be reconciled with multiple kinds of artifacts from throughout the software life cycle: requirements specifications, designs, code, test plans, test cases,
- The introduction of the revised GUI standards must be accommodated to ongoing software development activities, possibly at various life cycle stages, and affecting current and proposed project plans and schedules

Scenario 2: Pull together test cases and acceptance criteria from various sources into a comprehensive, product-specific test plan

With the growth of the company and expansion of the product line, it has been decided that the testing work for some products should be "outsourced." In consequence, it has been decided to create product-specific test plans for each of those products. This reflects multiple concerns in the following ways:

- Comprehensive test plans are developed for a number of specific products
- Each test plan addresses a number of areas of concern: application program interfaces (APIs), GUIs, single-user and multiple-user operation, use cases, performance, scalability, environment certification, and so on
- Various artifacts from various stages of the life cycle are drawn upon, including product specific requirements, general standards, product designs, existing test documents
- Multiple kinds of test-related documents are drawn upon and produced: generic and specific test plans, test procedures, test cases, and test reports (for example, IEEE testing documentation standards).

Scenario 3: Y2K readiness certification

In response to a request from a client, third-party certification is sought for purposes demonstrating Y2K readiness in a family of products. Third-party certification is based not on a direct examination of the products but on an audit of product development, testing, analysis, and maintenance processes. The preparation for Y2K certification reflects multiple concerns in the following ways:

- General-purpose development guidelines and practices must be abstracted and documented with respect to their relevance to Y2K readiness
- Existing test plans, procedures cases, criteria, and results must be reviewed and assessed for their relevance to Y2K readiness
- Various aspects of the products, including APIs and GUIs, algorithms, database schemas, and environmental dependencies must be examined regarding Y2K issues
- New test plans, procedures, cases, criteria, and results developed for Y2K readiness evaluation must be integrated into the overall testing process and documentation framework
- Ongoing development activities must be adapted to accommodate Y2K readiness preparation and assessment

3. MULTIPLE DIMENSIONS OF CONCERN

The scenarios described above involve a number of different dimensions of concern. Some examples are:

Products: Each specific product constitutes a concern in this dimension. For a company that produces a family of products (such as a suite of e-commerce application components) each member of the family is a concern. For a company that produces multiple families of products (e.g., IBM, Microsoft) there may also exist a dimension of concerns related to product families.

Product component kinds: Within a given product, there may be multiple component elements, each of which represents a separate concern. These may be represented by interfaces: Product-specific API, product-specific GUI, common APIs (shared across products), database APIs, CGIs, etc.

Testing document kinds: Within the realm of test documentation, different kinds of documents represent different concerns (and the separation of these concerns is the motivation for the different kinds of documents). Kinds of document (and concern) represented include test plans, test procedures, test cases, and test reports. Some of these can be made generic or specific according to whether they are developed for product families or specific products.

Testing concerns: Various testing documents are themselves highly structured in to standard sections. Each of these sections reflects a more specific concern with respect to the more general concern of the overall document. Thus a standard test plan (representing a concern with respect to the dimension of testing documentation) is itself divided into sections (items to be tested, environment, resources, etc.) that reflect specific concerns regarding test planning. Similarly, test procedures are divided into sections that reflect specific test procedure concerns.

Product issues: Each product addresses a number of different issues that may be categorized with respect to concerns in a number of different dimensions. Software testing in effect attempts to verify that a product satisfactorily addresses the issues with which it is concerned. In this dimension, the concerns of testing parallel those of the product. Some general categories of concern in this dimension are functionality, architecture, appearance, performance, scalability, robustness, environment, Y2K compliance, and so on. Note that these issues tend to span the software life cycle and to be reflected in artifacts produced from multiple phases of development (requirements, design, code, analysis, testing, documentation, etc.).

Processes: Software testing is really a collection of processes and activities, each of which can be seen to address a different set of concerns involving different elements from other dimensions. Testing in the traditional sense may include planning, procurement, development, setup, execution, monitoring, evaluation, and reporting (among others). Each of these is concerned with a different phase of the testing life cycle. Testing construed more broadly may include processes for dynamic testing (testing in the narrow sense), static analysis, simulation, inspection, reviews, and so on, each of which may address different aspects of a product and entail different "testing" life cycles.

Other dimensions can probably be elaborated based on the described scenarios and other possible scenarios.

4. CROSS-CUTTING CONCERNS

The dimensions of concerns described above include many concerns that are cross-cutting with respect to other dimensions of concern. In other words, concerns in one dimension may be correlated with concerns in another dimension, although this correlation may vary from great to slight. Correspondingly, an artifact or activity that manifests one concern may more or less strongly manifest other, cross-cutting concerns.

Many product issues arise in testing (e.g., function, look and feel, performance, Y2K readiness) that cut across life-cycle phases and are reflected in both the activities and artifacts associated with each phase. Issues that are verified in testing are specified in requirements, addressed in design, realized in code, and so on. The role and representation of specific issues typically varies between phases.

Product issues also cut across products and product groupings (e.g., families, specific products, and components). Some concerns (e.g., functionality) may vary from product to product, while others (e.g., Y2K readiness, look and feel) may be constant across products.

Test cases (that reflect particular product issues from requirements) may be referenced in multiple test plans, exercised in multiple test executions, and applied to multiple products.

Testing processes must be defined with respect to a number of dimensions, for example, to evaluate specific products, satisfy given testing-process requirements, assess various product issues, and use and produce various testing documents. In contrast, concerns related to the environment in which testing occurs may be largely independent of concerns related to the environment in which design or coding occurs.

5. SIMULTANEOUS OVERLAPPING CONCERNS

Testing combines elements from throughout the software life cycle and so combines their concerns. For example, consider a specific test case. It reflects product issues that cut across the life cycle, beginning with a requirement specification. It is represented as a document that may be incorporated as a section in one or more test plans. These test plans may apply to one or more products and they may be exercised one or more times. The test case thus holds a position with respect to the product-issue dimension, the test-plan (and more general test-document) dimension, the tested-product dimension, and the testing process (i.e., test execution) dimension. A change to the test case has implications in each of these dimensions. Conversely, a change in any one of these dimensions may affect the test case or its relevance in the dimension.

For a second example, consider the execution of a testing process (i.e., the actual testing of a software product). This process can be ``localized" in the multi-dimensional space of concerns with respect to the product dimension (the product under test), the execution-environment dimension, the product-issues dimension, the project-management dimension, the test plan, procedure, and case dimensions, and so on. A change in any one of these dimensions may have a significant impact on the validity of the tests.

6. MULTIPLE LEVELS AND SCOPES OF CONCERN

Consideration of the concerns of software testing (and the more general software life cycle into which testing fits) reveals that concerns are organized into dimensions that have a variety of levels and scopes. The occurrence of multiple levels of concern should not be surprising since many elements or aspects of software and software development are organized hierarchically. For example:

- Testing documents are composed of sections and subsections and may incorporate by reference other (also hierarchically structured) documents. Hierarchically organized documents are found across the life cycle.

- Testing processes may be aggregated into testing programs and are composed of subprocesses that are composed of steps that are composed of substeps, and so on. Such a hierarchical organization typically applies to processes both as they are defined and as they are executed. This organization is also typical of processes throughout the life cycle. Indeed, testing itself is one component within a higher-level life cycle.

- Tested products may be organized into families, individual products, versions, components, and interfaces.

- Product requirements (from which test cases and acceptance criteria are derived) may be divided into API and GUI requirements. GUI requirements may be divided into performance and appearance requirements. Appearance requirements may be divided into look-and-feel, color, layout, and so on.

Product code (depending on the programming language used) may also have a hierarchical organization, for example, libraries or archives, packages, files, classes, members, and so on.

These examples focus on hierarchical structures, but it is possible that there are other organizations of concerns in parallel with the other organizations commonly found in software artifacts and activities. For example, concerns may be organized into various arrangements of versions and configurations. (As with software in general, different modes of organization may serve different purposes.)

Given that concerns exist on multiple levels, with higher levels being more inclusive or general than lower levels, it follows that the scope of concerns will vary according to the level of the dimension on which they are found. For example, some concerns, such as certain product issues, may span the life cycle. Other concerns, such as the choice of an appropriate testing approach (dynamic testing versus static analysis, etc.), are relevant only within a single life-cycle phase. Still others, such as the choice of test environments or test data sets for dynamic testing, are relevant within a restricted scope of a single approach. Note that in a space of multiple dimensions, scopes may be defined

with respect to multiple dimensions. The example above describes scopes with respect to life cycle phases, but scopes may also be defined in other dimensions such as spaces of products or issues.

In light of the above, it also seems that a concern at one level can be viewed as a dimension from a lower level. The products of a company may be organized into families, and each family may be considered a concern within the overall product dimension. A family may be organized into products, and each product may be considered a concern within the product-family dimension. A product may be organized into components, and each component may be considered a concern within the product dimension. A similar view may be taken of test plans that are hierarchically organized. Thus, dimensions and concerns can (at least sometimes) be described in terms of a recursive structure.

7. DECOMPOSITION AND COMPOSITION BASED ON CONCERNS

As illustrated in the testing scenarios presented in Section 2, testing is a domain in which information, artifacts, and activities are naturally organized according to dimensions of concern. Moreover, in the typical scenario, multiple dimensions of concern are relevant. Different dimensions of concern may play a central or primary role in different scenarios, but typical scenarios entail the combination of concerns from many dimensions.

As a practical matter, the full range of testing artifacts and activities that may be needed in a software development organization, or even within a software project, can be difficult to anticipate. (For example, the Y2K problem went unanticipated for decades, testing concerns may evolve as products and infrastructure evolve, and new customers may raise new quality assurance issues.) Even when the need for testing activities or artifacts can be anticipated, due to time and resource constraints these activities and artifacts may only be developed ``on demand", i.e., which finally needed, and then only under some pressure. Despite these significant challenges, there are also rich opportunities for effective development of activities and artifacts within the testing domain. These arise from the fact that testing serves as multi-dimensional nexus for a wide range of concerns at a variety of scopes and scales.

For example, general standards documents may be applied to derive specific test cases and acceptance criteria. Product-specific requirements also serve as a basis for specific test cases and acceptance criteria, as can information about product-specific designs and code. Test plans for one member of a product family may be adapted for another member of the family. (Alternatively, a generic test plan for a product family may be specialized for individual members of the family.) Test methods and test cases may be reused in multiple test plans and test executions. Results from different kinds of tests applied across a product family may be reorganized into individual product test reports covering multiple product issues. Results obtained in a test for one purpose (say, general functionality) may be relevant to a later test for a different purpose (say, Y2K readiness). Results gathered from a variety of tests may be combined and organized to help determine the scope of work for a maintenance or upgrade development cycle.

In these and other cases, the ability to selectively decompose and compose different sorts of elements from different sorts of artifacts or activities can contribute greatly to the efficiency and quality of software testing in particular and software development in general. Due to the semantically rich and diverse nature of software testing, a variety of kinds of support for decomposition and composition are needed. Automated techniques can be helpful, for example, in assessing test coverage of code or in performing automated analyses. However, semantic complexities imply that people must necessarily be involved to a large degree in many cases. This involvement is likely to extend into effecting decomposition and composition activities and to formulating and evaluating composition rules. Human judgment may be needed, for example, in specializing a generic test plan to a specific product, in abstracting existing test cases that are relevant to a specific issue, in making qualitative judgments about the adequacy of test coverage, in formulating test cases to address critical aspects of architecture or implementation, and so on.

To better support people in the carrying out decomposition and composition activities, research is necessary in a number of areas. These include the structuring of documents to facilitate composition and decomposition according to concerns, the representation of concerns within documents (e.g., via intrinsic organization or markup languages), tools for querying, browsing, and viewing documents by concern, tools to support the abstraction and combination elements according to concerns, and tools to support the review, evaluation, and documentation of documents according to the correctness and consistency of concerns and their representations. Of course, research is also needed into methods and processes to support concern-based testing and concern-based software development.