Relational Metrics Model for Software Configuration Management

Charles Donald Carson, Jr.
Hassan Pournaghshband
Department of Computer Science and Software Engineering
Southern Polytechnic State University

Abstract - Changes during a software products life cycle are inevitable. These changes can correct or enhance software functionality and/or reduce costs associated with the software product. However, changes can also introduce added risks and unknowns during a software products life cycle and can result in unpredictable software behavior. Software configuration management (SCM) is a part of configuration management (CM) which supports a complex framework for monitoring, managing, and controlling changes to a software products configuration during its life cycle. In this paper, we introduce the development of a relational metrics model as a possible means for better managing and controlling software configuration change during a software products life cycle from the SCM activities, software engineering measures, and any available CM metrics. This relational approach to managing changes within a software product life cycle will result in a more effective means of validating change configurations by envisioning change from a unified quantifiable view instead of by individual change artifacts and components within the SCM framework.

1. Introduction

Software Configuration Management (SCM) is a software change management framework for managing any configuration changes during the development of a software product. A software configuration can include existing, functional and physical attributes of a software system as well as combinations of software systems [1.] "Software Configuration Management is an umbrella activity that is applied throughout the software process. Because software configuration change can occur at any time, SCM activities are developed to (1) identify change, (2) control change, (3) ensures that change is being properly implemented, and (4) report changes to others who may have an interest" [2.] Changes to a software product's configuration can typically result in revised software characteristics, and by derivation, affect the metrics that report on the software product itself. Though changes to a software product are intended to enhance its performance, and/or reduce costs, they can also introduce added risks. Software systems can comprise of multitudes of individual change artifacts and components. Furthermore, each of these change artifacts and components can involve individual dependencies and constraints. Documentation of changes in these systems can be inconsistent, incomplete, and may or may not adequately cover the individual changes in each of the change artifacts and components as a whole. It is difficult to control the software configuration change process solely by visualizing individual change artifacts and components with the expectation of accurately and consistently controlling any resulting new software configuration behavior. In this paper, we propose a relational approach to help in managing software configuration changes throughout the life cycle of a software product.

Also we will investigate the development of relations between entities and attributes derived from the various activities within the SCM framework along with the available software engineering measures and associated CM metrics.

2. Our Approach

In developing our relational metrics model, the primary functional elements of SCM, that is, configuration identification, configuration change control, configuration status accounting, and configuration audit were analyzed. Possible entities and attributes from activities within these functional elements of SCM are then derived, as well as resulting software engineering product, process, and project measures.

Primary elements of SCM activities and any resulting software engineering measures were used in establishing the data requirements of the relational metrics model as listed below in Table 1;

Table 1

SCM Activity/Software Engineering Measures	Description
Configuration identification	Identification of software configuration items.
Configuration control activity	Change process, control, and release process.
Configuration status accounting	Reports based on any configuration management data.
Configuration accounting audit	Recording and reporting the status of change requests/components in the software product.
Product measure metrics	Indirect and or direct measurements of software related activities.
Process measures metrics	Deliverables involving artifacts and or documents resulting from process activities.
Project measures metrics	Production of items used by processes in order to produce their outputs.

Through the use of an Entity Relation Diagram (ERD), we represent our established data requirements as relationships between the individual configuration item entities undergoing the actual configuration change, as well as the SCM process activities and any resulting software metrics that are expressed as attributes of these [3.] Subsequently, the ERD will then be converted into the relational metrics model itself.

In Figure 1, we have illustrated placing configuration items (CI's) that are typical in the development of a software product, such as software component, related requirements, and associated test cases under SCM control. During particular points of the CI's life cycle, the CI's undergoing any configuration change is represented as versions as reflected through the various entities illustrated in the diagram. The attributes of these entities are the actual representations of the activities and resulting software engineering metrics within the functional elements of the SCM process. In the example entity, Configuration Item Version, changes to a CI are represented as change attributes; the individual change activities within the functional elements of SCM and any artifacts from these activities. The resulting metrics are represented as attributes of that entity. An actual abstraction of the SCM process; its activities and metrics produced within each of the functional elements is essentially represented throughout the ER diagram.

We have organized any metrics produced from the various activities within the SCM process into the respective areas of the software engineering measurements. From this type of organization, a better understanding can be realized regarding the measurements produced from the various activities within the functional activities of the SCM process and how these impact the software product and the SCM process itself.

Typical metrics within configuration management can be correlated with each of the software engineering measures based on the product, process, and project measurement types and what these might represent. Correlation of CM metrics and these software engineering measures are discussed in [1.] From the analysis of the individual activities derived from functional elements of SCM process and the correlation of the CM metrics with its respective software engineering measures, we have established an understanding of the SCM process as a whole.

Through correlating the entities and their respective attributes we have indicated that the conceptual model provides a sufficient representation of the overall activities within the functional elements of the SCM process.

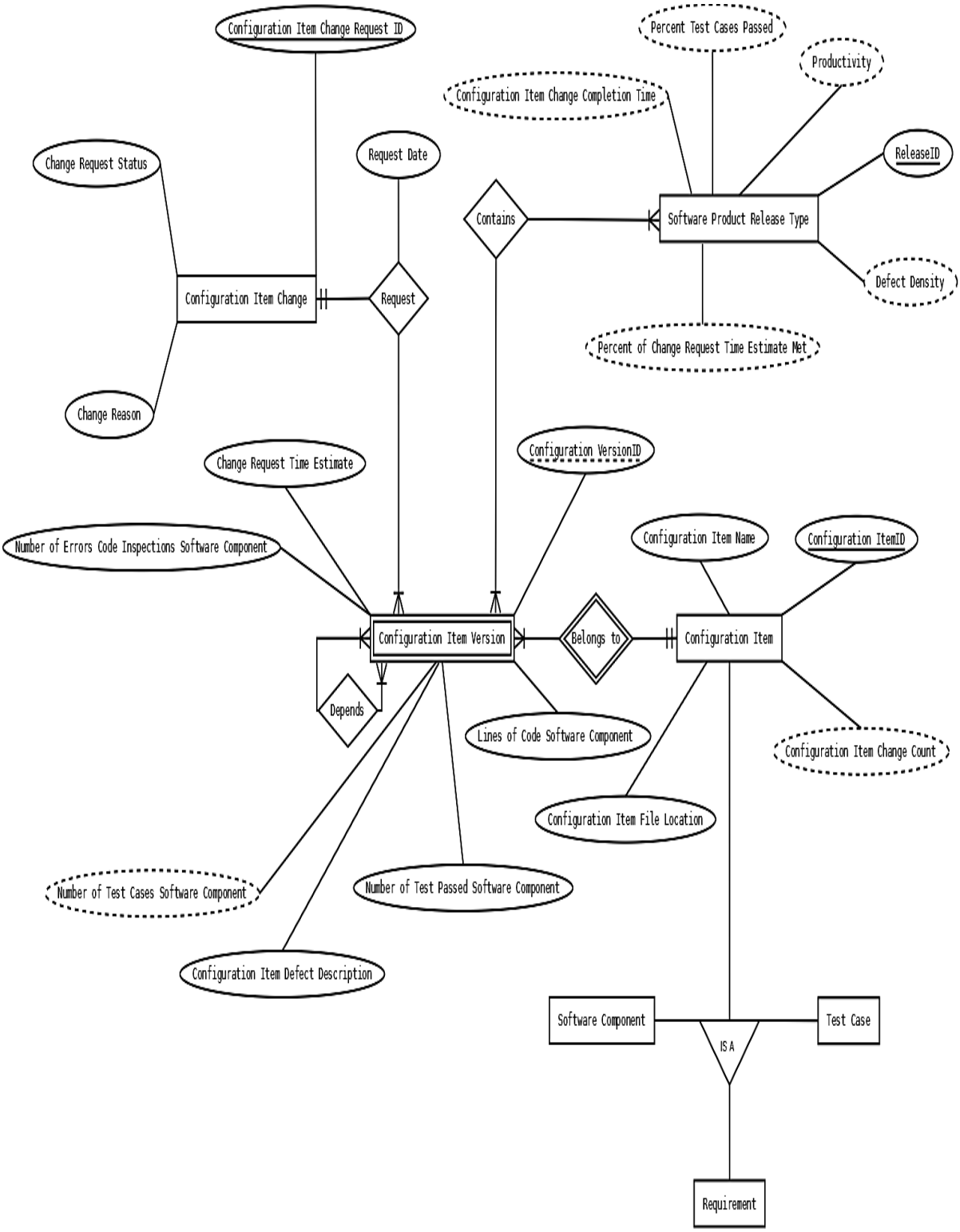

Figure 1 - ER Diagram of Relational Metrics Model

Furthermore, the activities within the functional elements of the SCM process provides a means of identifying what CI's in the various software configuration baselines are undergoing changes at various points in time. The conceptual relation metrics model through the use of relations provides an overall view of the SCM process by envisioning change from a unified quantifiable view instead of by individual change artifacts and components within the SCM framework [1.] Figure 2 shows the relation metrics data model resulting from the conversion of each entity and its respective attributes represented in the ER diagram into individual relations.

3. Traditional SCM Approach and the Relational Metrics Model

We have found that the traditional approach to SCM consists mainly of individually managing the individual change artifacts and components within a software products lifecycle. However, this presents a level of difficulty, possible inconstancies, and is susceptible to being error prone. Furthermore, it is difficult to control the software configuration change process solely by visualizing individual change artifacts and components with the expectation of accurately and consistently controlling any resulting new software configuration behavior. Though various SCM tools exist, these tools can tend to focus only on the SCM aspect and does not fully integrate any software engineering measures which can be crucial in providing any important SCM activity metrics.

Software engineering metrics which consists of product, process, and project measures can be derived from any of the various SCM activities. These measures can help in quantifying change within the SCM activity framework and help drive any necessary improvements in the SCM process as well of improvements to individual artifacts of the software project and the software project itself.

Lastly, in comparing the proposed Relational Metrics Model to traditional SCM approaches, we believe there exits an advantage to managing change configuration from a unified view perspective through interrelations of the SCM activities and the software engineering metrics rather than by individual change artifacts and components as presented by traditional SCM approaches and tools. Although change activities within traditional SCM approaches track changes and any artifacts affected, however questions arise in quantifying the various areas of change in order to identify, correlate, and prioritize changes. Through an abstracted view of software products change activities and related metrics, software project managers for example, can anticipate and project issues and establish a "preemptive" approach rather than a "reactive" approach which is a prevailing factor in many software projects presently.

4. Conclusions

In this study, we proposed the development of a relational model as a possible means for managing and controlling software configuration change during a software products life-cycle from the SCM activities and the available software engineering metrics. Current approaches to SCM focus primarily on individually managing the individual change artifacts and components within a software products lifecycle which is susceptible to errors as well as not providing an abstracted view of a software products change activities. We demonstrated that our relational approach to managing changes within a software product lifecycle will result in a more effective means of validating change configurations by visualizing change from a unified quantifiable view rather than by individual change artifacts and components within the SCM framework.

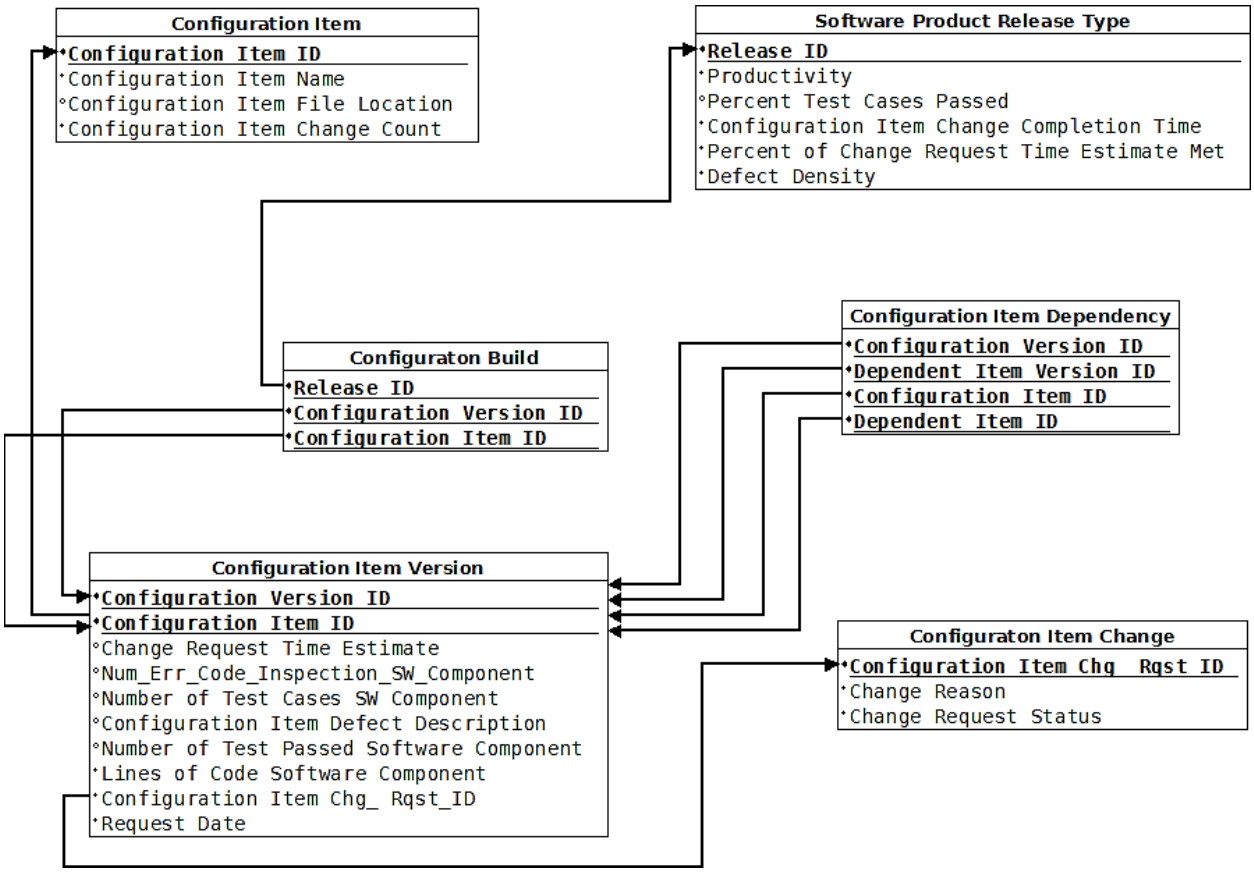

Figure 2- Relation Metrics Data Model

References

[1].J.Keyes, “Software Configuration Management,” Auerbach Publications, 2004.

[2]. R.Pressman, “Software Engineering-A Practitioner's Approach,” Seventh Edition, McGraw-Hill, 2010.

[3]. Gornik, D. (2003). Enity Relationship Modeling With UML. Retrieved from www.ibm.com/developerworks/rational/library/content/03July/2500/2715/2785_uml.pdf

[4].Software Metrics (http://www.cs.ucl.ac.uk/staff/A.Finkelstein/advmsc/11.pdf)

[5]. Harrington, J. (2002). Relational Database Design, San Diego, California: Morgan Kaufmann Publishers

[6]. IEEE Standards: "IEEE 828-2012: STANDARD FOR CONFIGURATION MANAGEMENT IN SYSTEMS ANS SOFTWARE ENGINEERING."