UNIVERSITI PUTRA MALAYSIA

DEVELOPMENT AND EVALUATION OF A SOFTWARE METRICS MARKUP LANGUAGE

NG KENG YAP.

FSKTM 2006 12

DEVELOPMENT AND EVALUATION OF A SOFTWARE METRICS MARKUP LANGUAGE

By

NG KENG YAP

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirement for the Degree of Master of Science

March 2006

DEDICATION

To my parents, dearest wife and my lovely children for the endless support, encouragement and patience, and also those who made this study possible.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Master of Science

DEVELOPMENT AND EVALUATION OF A SOFTWARE METRICS MARKUP LANGUAGE

By

NG KENG YAP

March 2006

Chairman: Associate Professor Abdul Azim Abdul Ghani, PhD

Faculty: Computer Science and Information Technology

Software measurement is the dimension and/or decision criteria as to what a piece of software can provide. The output of software measurement is in the form of metric data. Metric data are important because it can be used as the input for software analysis in the software engineering field. Software engineering relies on these data to investigate many factors in software development such as cost, scheduling, affordability, quality, etc., in order to gain better control of the engineering processes. These days, people store data in different data formats, media and database technologies. These heterogeneous formats have posed many problems in data analysis, especially in terms of the integration and reusability of historical data. These problems have prompted efforts to find a data format that is compliant with the concepts of software measurement and which is applicable to any metric data.

eXtensible Markup Language (XML) is the latest platform independent and self-explanatory data model that is widely used in the world, especially significant in the heterogeneous computing environment, such as World Wide Web. Hence, XML has been chosen to be the markup language for software metrics data in order to produce Software Metrics Markup Language (SMML), which is the major output of this research. There are shortcomings of the existing data models and XML can be used to overcome these shortcomings, and further enhances the portability, extensibility and appendability of software metrics data.

The build and evaluate framework is used to ascertain that the design goals of the SMML have been archived accordingly. The SMML Toolkit and the SMML API have been built as the instruments to evaluate the viability of SMML.

The SMML vocabulary and grammar, which is synonymous to the XML elements and the elementary structure of SMML respectively are defined and implemented physically in XML schema for SMML. It determines and controls how SMML should be constructed to hold informative software metrics data.

The experimental evaluation shows that SMML is viable to be the data model for software metrics. Data can be easily stored and manipulated, either in the existing SMML model or transformed into the structured relational databases, provided that the SMML API is used.

Future research can be extended to enhancing the structure and enriching the vocabulary of SMML, and introduce ontology studies on this model, besides conducting performance tuning on the SMML API.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains

PEMBANGUNAN DAN PENILAIAN BAHASA PENANDAAN METRIK PERISIAN

Oleh

NG KENG YAP

Mac 2006

Pengerusi: **Profesor Madya Abdul Azim Abdul Ghani, PhD**

Fakulti: **Sains Komputer dan Teknologi Maklumat**

Pengukuran perisian merupakan dimensi dan/atau kriteria keputusan yang dapat dibekalkan oleh sesuatu perisian. Output pengukuran perisian pula adalah dalam bentuk data metrik. Data metrik adalah penting kerana ianya dijadikan input untuk analisis perisian dalam bidang kejuruteraan perisian. Kejuruteraan perisian pula bergantung kepada data ini untuk menyelidik faktor-faktor dalam pembangunan perisian seperti kos, penskedulan, kemampuan, kualiti dan sebagainya supaya dapat mengawal proses kejuruteraannya dengan lebih baik. Kebelakangan ini, kebanyakan orang menyimpan data dalam pelbagai format, media dan teknologi pangkalan data. Format yang heterogen ini telah menimbulkan banyak masalah dalam menganalisis data, terutamanya dari segi pengintegrasian dan penggunaan-semula data lama. Masalah ini telah merangsang usaha untuk mencari format data yang patuh dengan konsep pengukuran perisian dan boleh digunakan kepada sebarang data metrik.

eXtensible Markup Language (XML) adalah model data yang boleh merentasi pelantaran dan jelas dengan sendirinya yang digunakan secara meluas dalam dunia hari ini, terutamanya penting dalam persekitaran pengkomputeran heterogen seperti *World Wide Web*. Maka, XML telah dipilih sebagai bahasa penandaan bagi data metrik perisian untuk menghasilkan Bahasa Penandaan Metrik Perisian (SMML) iaitu output utama penyelidikan ini. Ianya tidak dapat dinafikan bahawa masih lagi terdapat kelemahan dalam model data yang sedia ada dan XML boleh digunakan untuk mengatasi kelemahan tersebut, dan lantasan itu, kemudahalihan, kebolehsambungan dan kebolehtambahan data metrik turut akan diperkukuhkan.

Rangka kerja 'membina dan menilai' telah digunakan untuk memastikan matlamat reka bentuk SMML tercapai. SMML Toolkit dan SMML API turut dibangunkan sebagai peralatan untuk menguji kebolehlaksanaan SMML.

Suku kata dan tatabahasa SMML, atau masing-masing lebih dikenali sebagai unsur XML dan struktur unsuran SMML telah didefinisikan dan dilaksanakan secara fizikal dalam bentuk skema XML untuk SMML. Ia menentukan dan mengawal bagaimana SMML perlu dibentuk untuk memegang data metrik perisian yang informatif.

Penilaian secara eksperimen menunjukkan bahawa SMML boleh dijadikan sebagai model data metrik perisian. Data juga boleh disimpan dan dimanipulasikan dengan mudah, sama ada dalam model SMML asal ataupun selepas ianya ditransformasikan ke pangkalan data hubungan yang berstruktur, sekiranya SMML API digunakan.

Penyelidikan seterusnya bolehlah diperluaskan untuk memperkukuhkan lagi struktur dan memperkayakan suku kata SMML, serta memperkenalkan pengajian ontologi ke atas model ini selain daripada melakukan penalaan prestasi ke atas SMML API.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the kindness of several individuals and parties. I would like to express my gratitude and deepest appreciation to my supervisor, Assoc. Prof. Dr. Abdul Azim Abdul Ghani, not only for his kindness and considerable mentoring all the time, but also for sponsoring my Master of Science study through the Graduate Research Assistant (GRA) scheme for his research. Assoc. Prof. Dr. Hj. Ali Mamat and Mrs. Hazura Zulzalil also deserve a great deal of thanks for offering their guidance and insightful comments in the aspects of XML and software quality assurance, respectively.

I pay tribute to the Web-based Software Certification Model research group members who gave me the greatest pleasure of working closely together with them, notably Mrs. Hazura Zulzalil, Mrs. Arina Kamaruddin, Miss Kamsiah Mohamed and Mrs. Zaida Zaini.

I am also deeply indebted to Prof. Barbara Kitchenham, Luis Olsina and Alan Brain for providing me with their research data, references and literature materials.
My special thanks go to my dearest wife, Siew-Luan, my children Haw-Wen and Yie-Syuen for their understanding and patience throughout the course of my research and the preparation of this dissertation. I would also like to express my uttermost appreciation to my parents and family members for their kindness and support.

To all others who have helped me in one way or another, may God reward you for your good deeds.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANSI	American National Standardisation Institute
API	Application Programming Interface
attrib.	attributes
CMF	Content Management Framework
CMM	Capability Maturity Model
DA Pam	Department of Army Pamphlet
DB	Database
DDL	Data Definition Language
DOM	Document Object Model
DSS	Decision Support System
EIS	Executive Information System
E-R	Entity-Relationship
ES	Expert System
ETL	Extraction, Transformation and Loading
GQM	Goal/Question/Metric
GUI	Graphical User Interface
HFS	Hierarchical File System
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
INRIA	*Institut National pour la Recherche en Informatique et Automatique* (National Institute for Research in Computer Science and Control, France)

IRPA	Intensification of Research in Priority Areas
ISO	International Organisation for Standardisation
JAXP	Java API for XML Processing
JDBC	Java Database Connectivity
JDOM	Java Document Object Model
JVM	Java Virtual Machine
KLOC	Kilo Lines of Codes
MIT	Massachusetts Institute of Technology (USA)
MOSTI	Ministry of Science, Technology and Innovation (Malaysia)
MPMS	Metrics-based Project Management System
OASIS	Organisation for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity
OLAP	On-Line Analytical Processing
O-O	Object-Oriented
O-R	Object-Relational
OS	Operating System
POJO	Plain Old Java Object
POM	Project Object Model
PSM	Practical Software Measurement
RDBMS	Relational Database Management System
SAX	Simple API for XML
SDLC	Software Development Life Cycle
SEE	Software Engineering Environment
SEI	Software Engineering Institute (Carnegie Mellon University, USA)

SMML	Software Metrics Markup Language
SQL	Structured Query Language
UML	Unified Modelling Language
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	eXtensible Stylesheet Language Transformation

CHAPTER 1

INTRODUCTION

1.1 Background

All engineering disciplines employ methods that are based on models and theories. Software engineering is no exception. Software engineering describes the collection of techniques that apply an engineering approach to the construction and support of software products. It encompasses a range of activities, including managing, costing, planning, modelling, analysing, specifying, designing, implementing, testing and maintaining the software products. It is a relatively new discipline that still requires further research and a great deal of empirical improvements to strengthen and make it more rigorous for the industry. Software is an abstract object. Hence, the techniques in software engineering are somewhat different from those in other engineering disciplines. Most of the measurement techniques and processes in software engineering are still undergoing extensive research in order to make software more measurable, transparent and easier to control throughout the software development life cycle (SDLC). You cannot control what you cannot measure (DeMarco, 1982), as software engineering is all about justifying things that we can measure in order to seize control of the process and product quality. Therefore, measurement plays a vital role in successful software engineering.

Software measurement is only as good as the data collected and analysed. In other words, we cannot make good assessments of a software product with bad data.

> “Data should be collected with a clear purpose in mind. Not only a clear purpose but also a clear idea as to the precise way in which they will be analysed so as to yield the desired information. It is astonishing that men, who in other respects are clear-sighted, will collect absolute hotchpotches of data in the blithe and uncritical belief that analysis can get something out of it.” (Moroney, 1950).”

Good data come from good measurements. A good measurement must be carefully defined, clear-cut and specific to the purpose. Besides, the presentation of data on paper or other human-readable formats must be readily understandable. This is especially important when data collection, analysis and transfer to the database are carried out by different groups of people. Storing metrics data in a database without a clear definition of the database fields will pose problems on occasions when many users inject and extract data into and from the database. Any data in a relational database are meaningless until all fields have been properly defined and documented to include the measurement protocol, units used, collector’s information, attributes, etc. These are known as software metrics metadata. Kitchenham and Hughes (2001) have mentioned the importance of software measurement metadata as a means of ensuring trustworthy data for further analysis to provide the rational grounds for decision-making in software engineering. Software metadata make the purpose of the data collection clear, keep the data collection in line with its purpose and provide useful information to the users during data analysis.

Software Metrics Markup Language (SMML) has been designed to serve precisely this purpose by creating a model to hold software metrics data in a more descriptive way. It is pioneered as part of my supervisor's research project. The project is fully funded by the Ministry of Science, Technology and Innovation (MOSTI) of Malaysia, under its Intensification of Research in Priority Areas (IRPA) programme. The research project, entitled 'Web-based Software Certification Model', was an attempt to create a quality certification model for web-based software. Software certification is a process involving software measurement and classification by quality against certain predefined and 'recognised' rules and scales. SMML acts as a data model for handling data chunks during the software certification process.

This said, it is worth pointing out that software certification is not the only purpose SMML serves. Rather, SMML has been designed to support data handling in all aspects of software engineering. As we know, software engineering has been undertaken across diverse computing environments. A piece of software may vary from others in terms of the programming language used, the underlying platform, the quality attributes between them, etc. To collect metrics data from these diverse software environments and store them in a single metrics repository is a daunting task. Some researchers have proposed a general layer between the various software engineering environments and the relational database systems. With respect to this matter, we will also expose the weaknesses of applying such a general layer later in this dissertation.

There is no standard data format for storing metrics data to date as we can see that most of the software engineering tools currently use different data models to hold their data.

Of the software measurement tools developed so far, many implement their own proprietary formats, while others are confusing to use due to insufficient metadata. Storing software metrics data in proprietary formats or different relational data structures inevitably complicates data integration and integral analysis of the metrics data. This dissertation will focus on how eXtensible Markup Language (XML), which is a platform-independent data format, can play a pivotal role in transforming metrics data in a single repository and, ultimately, how it can be used to invent a platform-independent Software Metrics Markup Language (SMML) which will most likely be the metrics data format of the future.

1.2 Problem Statement

Software measurement should be an integral part of software engineering. The effectiveness and efficiency of a software engineering exercise is very much dependent upon the way we measure, manage and manipulate the metrics data, as well as how we use the metrics data in decision-making. Data are collected in organisations that exercise software engineering practices in the process of software development. They generally store data in a repository or in whatever format convenient to them. Several studies have shown the necessity to store metrics data in a database system, preferably a relational database management system (RDBMS) (Fenton and Pfleeger, 1998). We have found that several issues can arise when the aforementioned methods are used to manage data, as outlined below:

- Issues with portability,