A System Model for Managing Requirement Traceability Matrices via Statistical Artifact Change Analysis

Benjamin J. Deaver and LiGuo Huang, Southern Methodist University, Dallas

Introduction and Motivation

The system Requirement Traceability Matrix (RTM) is primarily used for ensuring that all requirements are fulfilled by the system artifact deliverables and the management of change to deliverables with respect to impact on other systems. In the systems engineering and system of systems (SoS) engineering landscapes, the RTM is a tool that is useful at time of creation, but requires constant maintenance in a frequently changing landscape to maintain the original level of validity. The dynamic nature of systems and SoS engineering landscapes requires that a RTM be constantly regenerated in order to maintain this original level of accuracy. However, in highly complex and rapidly changing systems, this becomes an unrealistic proposition. The motivation of our research is the development of a system model which will statistically determine the dependability of the RTM based on the types of changes that have taken place since the most recent generation of the RTM, thereby providing a level of dependability to change management groups working with systems and SoS developers.

As systems grow in size and complexity, the underlying burden of maintaining the RTM grows as well. Over time, the regeneration of the RTM for a dynamic system may become too cumbersome a task to perform regularly. It is the goal of our research to provide a model which will allow for us to better understand the impact of a type of change to and existing RTM in a system being actively modified.

Summary

We create a taxonomy of change based on the observed change patterns presented in three separate Open Source Software projects currently in development. The three projects we are currently working with are:

1) Gantt
2) jHotDraw
3) ReactOS

For each of the changes observed, several pieces of meta data are collected for examination. The requirement(s) affected, the order of change in relation to other changes, as well as the underlying classification of the change according to the previously identified taxonomy of change will be utilized in the generation of our statistical model of change impact to the RTM.

Once all changes between specific versions of code for specific requirements have been identified as they relate to a set of requirements which are being traced, we will be able to generate the RTM for all identified static points of code. For our testing purposes with the Gantt project, this will result in three RTMs being utilized at versions 2.0.8, 2.0.9, and 2.0.10. The changes between each version will allow us to examine the individual traces between artifacts for changes and begin the task of building a statistical model which will allow the prediction of the impact of different changes over time to the RTM.

Requirement Traceability Matrix – Gantt Open Source Software Project

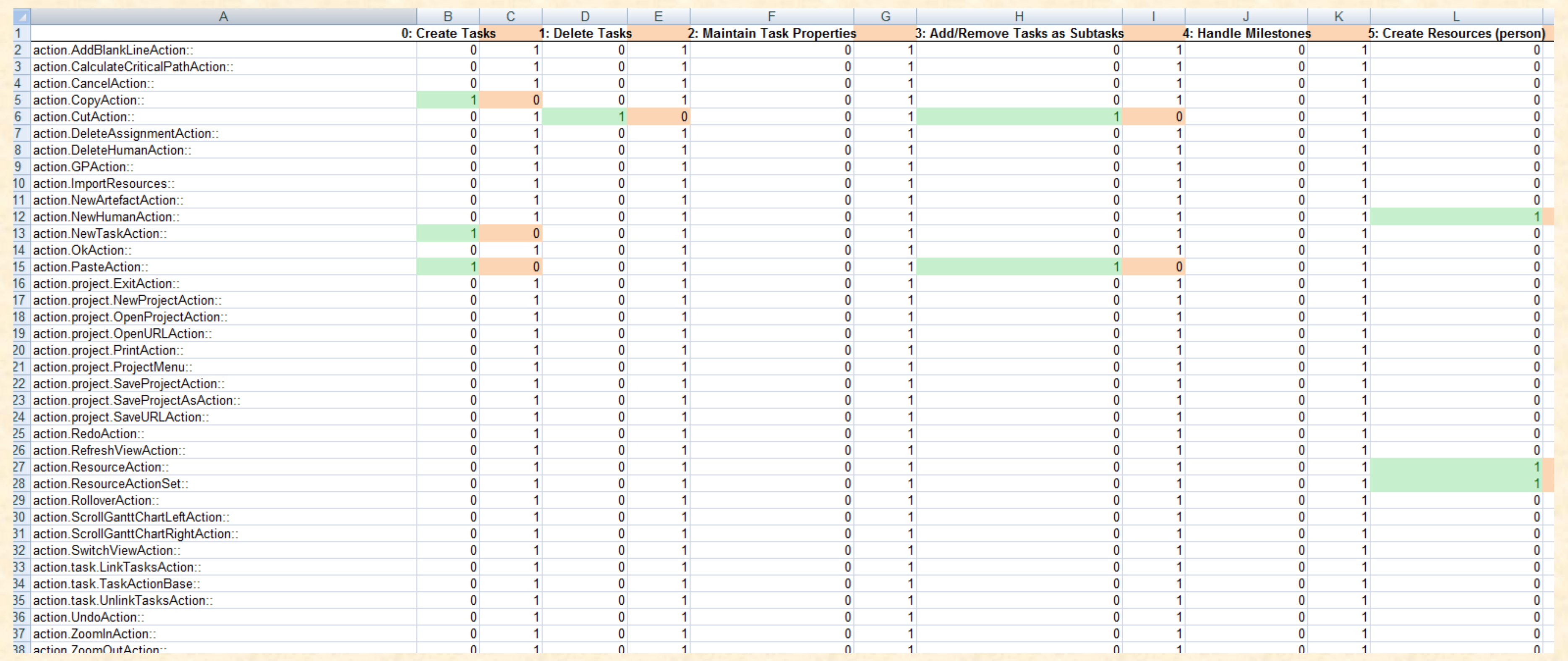
Our initial dataset for evaluation is taken from the Gantt Open Source Software Project (http://www.ganttproject.biz). The initial trace data has been provided to us by Dr. Alexander Egyed at the Institute for Systems Engineering and Automation at Johannes Kepler University. Additional traces of requirements to code for subsequent Gantt versions are being created using similar methods to the original collections performed by Dr. Egyed and other researches initially involved in the tracing of the Gantt project.

In the Requirements Traceability Matrix for Gantt, the requirement being traced is identified as a column header. The individual classes containing executable code are the rows. For each requirement, the called classes are identified by a 1 (this class was called) or a 0 (this class was not called). This is the format of the stored Requirements Trace Matrices for the Gantt Project.

PROCEDURE

1. Identify the taxonomy of change for a given domain (Systems Engineering, SoS Engineering, Software Engineering).
2. Identify and classify changes between static versions of the product.
3. Generate Requirements Trace Matrixes for each static version of the product being analyzed.
4. Evaluate the impact of changes to the product against each of the different Requirements Trace Matrixes.
5. Utilize the collected data to develop the statistical model for RTM confidence based on applied changes to the RTM.

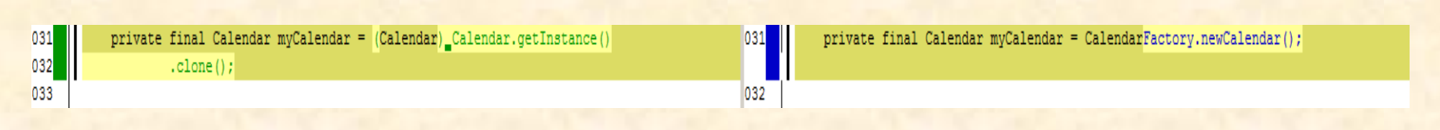
The model is a base effort that is intended to be customized for domains of engineering. Should the need for a model in the SoS space exist, it is expected that the taxonomy of change and underlying model values for Systems or Software Engineering will not fill this role effectively. The process is expected to be cross-domain.



	A	B	C	D	E	F	G	H	I	J	K	L
1		0: Create Tasks		1: Delete Tasks		2: Maintain Task Properties		3: Add/Remove Tasks as Subtasks		4: Handle Milestones		5: Create Resources (person)
2	action.AddBlankLineAction::	0	1	0	1	0	1	0	1	0	1	0
3	action.CalculateCriticalPathAction::	0	1	0	1	0	1	0	1	0	1	0
4	action.CancelAction::	0	1	0	1	0	1	0	1	0	1	0
5	action.CopyAction::	1	0	0	1	0	1	0	1	0	1	0
6	action.CutAction::	0	1	1	0	0	1	1	0	0	1	0
7	action.DeleteAssignmentAction::	0	1	0	1	0	1	0	1	0	1	0
8	action.DeleteHumanAction::	0	1	0	1	0	1	0	1	0	1	0
9	action.GPAction::	0	1	0	1	0	1	0	1	0	1	0
10	action.ImportResources::	0	1	0	1	0	1	0	1	0	1	0
11	action.NewArtefactAction::	0	1	0	1	0	1	0	1	0	1	0
12	action.NewHumanAction::	0	1	0	1	0	1	0	1	0	1	1
13	action.NewTaskAction::	1	0	0	1	0	1	0	1	0	1	0
14	action.OkAction::	0	1	0	1	0	1	0	1	0	1	0
15	action.PasteAction::	1	0	0	1	0	1	1	0	0	1	0
16	action.project.ExitAction::	0	1	0	1	0	1	0	1	0	1	0
17	action.project.NewProjectAction::	0	1	0	1	0	1	0	1	0	1	0
18	action.project.OpenProjectAction::	0	1	0	1	0	1	0	1	0	1	0
19	action.project.OpenURLAction::	0	1	0	1	0	1	0	1	0	1	0
20	action.project.PrintAction::	0	1	0	1	0	1	0	1	0	1	0
21	action.project.ProjectMenu::	0	1	0	1	0	1	0	1	0	1	0
22	action.project.SaveProjectAction::	0	1	0	1	0	1	0	1	0	1	0
23	action.project.SaveProjectAsAction::	0	1	0	1	0	1	0	1	0	1	0
24	action.project.SaveURLAction::	0	1	0	1	0	1	0	1	0	1	0
25	action.RedoAction::	0	1	0	1	0	1	0	1	0	1	0
26	action.RefreshViewAction::	0	1	0	1	0	1	0	1	0	1	0
27	action.ResourceAction::	0	1	0	1	0	1	0	1	0	1	1
28	action.ResourceActionSet::	0	1	0	1	0	1	0	1	0	1	1
29	action.RolloverAction::	0	1	0	1	0	1	0	1	0	1	0
30	action.ScrollGanttChartLeftAction::	0	1	0	1	0	1	0	1	0	1	0
31	action.ScrollGanttChartRightAction::	0	1	0	1	0	1	0	1	0	1	0
32	action.SwitchViewAction::	0	1	0	1	0	1	0	1	0	1	0
33	action.task.LinkTasksAction::	0	1	0	1	0	1	0	1	0	1	0
34	action.task.TaskActionBase::	0	1	0	1	0	1	0	1	0	1	0
35	action.task.UnlinkTasksAction::	0	1	0	1	0	1	0	1	0	1	0
36	action.UndoAction::	0	1	0	1	0	1	0	1	0	1	0
37	action.ZoomInAction::	0	1	0	1	0	1	0	1	0	1	0
38	action.ZoomOutAction::	0	1	0	1	0	1	0	1	0	1	0

Sample Change Patterns Identified between Code Versions

Class	From Version	To Version	Change Pattern	Description of Change
CalendarFactory.java	2.0.8	2.0.9	New Method	2.0.9 Contains new method newCalendar(). Does not exist in previous versions.
WeekendCalendarImpl.java	2.0.8	2.0.9	Calling Dependency Change	2.0.9 Code calls new CalendarFactory.Java class for calendar creation. Calendar object being created remains the same, only the method of creating changes.



```
031    private final Calendar myCalendar = (Calendar) Calendar.getInstance()
032            .clone();
033
```

```
031    private final Calendar myCalendar = CalendarFactory.newCalendar();
032
```

Change Pattern	Calling Dependency Change
Class	WeekendCalendarImpl.java
From Version	2.0.8
To Version	2.0.9
Description of Change	2.0.9 Contains new method newCalendar(). Does not exist in previous versions.
Impact	Black box impact - none White box impact - New method of creating a calendar object utilizing factory class
Reqs. Impacted	2: Maintain Task Properties (high likelihood of impact) 10: Change Task Begin/End Times manually with user changes (high likelihood of impact) 11: Change Task Begin/End Times automatically with dependency changes (high likelihood of impact)

The above data identifies a set of two sample changes as seen between versions 2.0.8 and 2.0.9. While a change itself is unique and seen exactly one time, the type of change being seen is not unique. Those types of change, "New Method" and a subsequent "Calling Dependency Change" are identified and associated to the meta data of the identified change for further evaluation.

The identification of the specific taxonomy of change has been performed previously to this experiment with the assistance of a significant number of graduate students reading through code changes. The suggested taxonomy of change for the Software Engineering Domain of RTM impact analysis has been limited to:

1) Method name change
2) Method parameter change
3) If Statement Changes
4) Variable Changes
5) Constraint Changes (public, private, protected)
6) Calling Dependency Changes
7) New / Deleted Method

As changes are identified between versions of the Gantt system, individual changes must be examined in greater detail in order to properly collect the meta-data needed for further analysis.

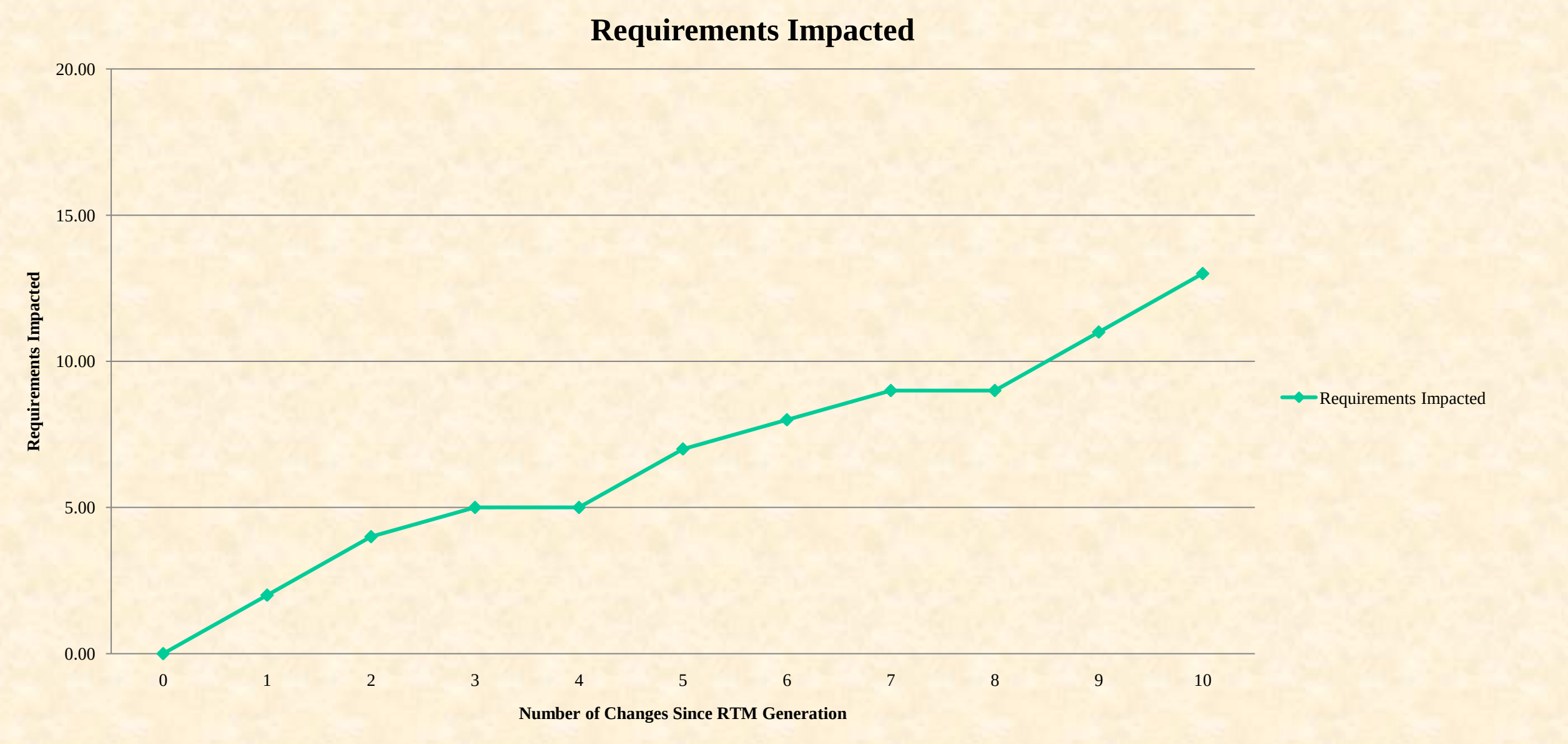
Examinations of individual changes will need to be done in greater detail in order to truly understand the impact of change to the requirement(s) in questions.

As the requirement(s) being affected are identified, we can begin to understand the effect of overlap and any other possibly affected requirements in the product being evaluated. The likelihood of impact is a purely subjective analysis being performed by individuals who are experienced in creating requirements traces. The impacted requirement(s) can be derived from the trace data captured at each of the stable versions.

The Value of the Requirements Traceability Matrix

Kannenberg et al identify the underlying necessity of the Requirements Traceability Matrix and the underlying effect on project management, process visibility, verification and validation, as well as project maintainability. Over time, the Requirements Traceability Matrix provides significant insight in to the internal workings of the relationships between requirements and deliverable artifacts.

As systems grow in size and complexity, the ability of project managers to maintain insight into the internal workings of the project relationships is diminished. Eventually, the effort required to maintain such relationships is too great for the relationships to be effectively maintained. However, as the complexity of the effort increases, the need for the additional insight into the relationships between requirements and deliverables as well as the overlap seen between them becomes more important, as the overlap can grow in scope.



The number of requirements affected by a number of changes in a system tends to increase at a rate greater than 1-1. We will normally see in a system that there is overlap between requirements and deliverable elements. Our observations indicate that this relationship becomes more prone to overlap as systems undergo increased iterations of development. This supports our original suppositions that the underlying value of the Requirements Traceability Matrix increases as the system becomes more complex. However, the underlying effort often precludes the development of the additional versions of the RTM as the value becomes highest.

It is our hope that the statistical model of impact to the Requirements Traceability Matrix will allow for the empirical analysis of change to determine the most effective time for generation of a new Requirements Traceability Matrix, thereby offsetting some of the effort involved with the generation by capitalizing on the highest value attained.

Impact to Systems Engineering

Over time, we anticipate seeing several specific impacts to Systems Engineering and, subsequently, Systems of Systems Engineering as well as Software Engineering. After examining the types and availability of trace data, based on our proof-of-concept study we envision that this research will positively impact the domain of systems engineering at many levels. As the taxonomy of changes can be modified for the scope under investigation, it is essentially a scalable solution that will be applicable and useful for requirement traceability management from the scope of individual system to the SoS.
Specific impacts include:

1) A comprehensive understanding of the types of changes which impact a system the most. The taxonomy of change will classify the types of system implementation changes. While all changes impact a system, some will undoubtedly impact the system more than others. The knowledge of the level of impact associated with change types will be beneficial to system or SoS change management during development and evolution.

2) A systematic vision on the impact of system changes to the RTM. While an RTM may exist within the parameters of acceptable validity before a change is introduced, it will be possible to predict the impact of the next change to the RTM validity, thereby proactively driving regeneration of the RTM if required. This will reduce the effort to having a valid RTM as well as reduce the speculation of RTM validity at any given time.

3) Knowledge of RTM dependability. The RTM is assumed to be correct at a time t = 0. Based on the number and types of changes that have been applied since t = 0, what is the dependability of the RTM while being used to manage future change?

The potential for additional interesting questions exists as well. Value based effects, the ability to affect the usefulness of IR methods for automated RTM generation, and other impacts on RTM studies that can benefit from a knowledge of the impact of change will certainly yield interesting research opportunities moving forward.

REFERENCES

Antoniol, G., Canfora, G., Casazza, G., and De Lucia, A., "Information Retrieval Models for Recovering Traceability Links between Code and Documentation", in Proceedings IEEE International Conference on Software Maintenance (ICSM'00), San Jose, CA, October 11-14 2000, pp. 40-51.

Egyed, A. A Scenario-driven Approach to Traceability. In *Proc. of the 23e ICSE* p. 123-132, Toronto, Ontario, Canada, May 12-19, 2001.

Egyed, A., Biffl, S., et al. A Value-based Approach for Understanding Cost-benefit Trade-offs During Automated Software Traceability. In *3rd Int'l Workshop on Traceability in Emerging Forms of Soft Engr*. p. 2-7, ACM Press: Long Beach, CA, 2005.

Heindl, M. and Biffl, S.: "A Process for Value-based Requirements Tracing - A Case Study on the Impact on Cost and Benefit," Proceedings of the European Software Engineering Conference and Foundations of Software Engineering (ESEC/FSE), Lisboa, Portugal, September 2005.

Heindl, Matthias, and Stefan Biffl. A Case Study on Value-Based Requirements Tracing. Proc. of the 10th European Software Engineering Conference. Lisbon, Portugal, 2005: 60-69.

Kannenberg, A., Saiedian, H. Why Software Requirements Traceability Remains a Challenge. CrossTalk, Vol. 22, No. 5, 2009, p. 14-21. Palmer, J.D. "Traceability." Software Requirements Engineering. Richard H. Thayer and Merlin Dorfman, eds. New York: IEEE Computer Society Press, 1997.

R. Wieringa, C. Ebert, Guest Editors' introduction: RE'03: practical requirements engineering solutions, IEEE Software 21 (2004) 16–18.