MODELLING THE QUANTIFICATION OF VERACITY REQUIREMENTS
TECHNICAL DEBT

Judith Perera, PhD Researcher, The University of Auckland
Supervisors: AP Ewan Tempero, Dr Kelly Blincoe, Dr Yu-Cheng Tu
RBOP Update --- Veracity Spearhead 2 day get together, October 2023
AGENDA

- Motivation
- Research Roadmap
- RE 23 Paper
- Discussion
MOTIVATION
Software Requirements play a crucial role in the development of a software systems.

Gathering requirements right and delivering the product that offers the best value to the end-users, is essential for the success of software systems.
Stakeholders make sub-optimal decisions concerning requirements during gathering, formalizing and implementing requirements --- e.g., missing to capture important user needs, introducing ambiguities when formalizing the requirements, insufficient implementation of requirements.

Requirements Technical Debt (RTD) captures the consequences of such sub-optimal decisions (in terms of costs).

Accumulating RTD can result in developing the wrong product for the customer or in delays in product delivery.

Therefore, RTD must be managed.

To manage RTD, it must be quantified.

But how can we quantify RTD?

We establish a theoretical foundation focusing on RTD and its quantification in the RE 23 paper.
Software Requirements concerning **Veracity** --- That is, software requirements related to **truth, trust, authenticity, provenance and integrity of data and human interactions**

Veracity is becoming a main concern in the present-day **software systems** and software requirements concerning veracity play a crucial role in the development of software systems.
RTD applies to all types of software requirements: functional and non-functional

Software requirements concerning veracity can be either functional or non-functional or both, depending on the software system

We hypothesize that the impact of RTD could be much higher in the case of software requirements concerning veracity and the quantification of RTD could support decision-making to better manage RTD

Motivation --- How is this applicable to Veracity?
RESEARCH ROADMAP
Step 1 (Completed): Establishing a theoretical foundation for Requirements Technical Debt (RTD) quantification
- Systematic Mapping Study (SMS) completed
- Conceptual model (developed)
- RE 23 paper (published)
- Invitation to extend as a Journal paper for the REJ Special Issue (paper in writing)

Step 2 (in Progress): Anonymous Online Questionnaire for Practitioners
- Study Goal: Understand if and how practitioners formally or informally quantify RTD, and how quantifying could support decision-making for better managing RTD for software requirements in general and for software requirements concerning veracity
- Target audience: software professionals from different domains
- Study Design (done)
- Ethics application (submitted)
- Currently obtaining feedback from experts on the questionnaire

Step 3: Modelling the quantification of veracity RTD
- Understand the implications for software requirements concerning veracity from questionnaire data
- Refining the conceptual model to capture specifics for veracity
RE 23 PAPER
Quantifying Requirements Technical Debt

A SYSTEMATIC MAPPING STUDY AND A CONCEPTUAL MODEL

Judith Perera, Ewan Tempero, Yu-Cheng Tu and Kelly Blincoe

The original talk was presented at the 31st IEEE International Requirements Engineering Conference in Hannover, Germany held from 4-8th Sep 23
**RQ1:** What can we learn about quantifying RTD from approaches proposed in RTD literature?

--- answered via the **SMS**

**RQ2:** How can we model the quantification of RTD?

--- answered via the **development of the conceptual model**
RQ1: SMS

- We followed recommendations given by Kitchenham et al. and Petersen et al.; Search, Article Screening, Reference snowballing, Data Extraction, Analysis and Synthesis
- 7 articles were finally included out of 87 articles obtained from the digital databases: SCOPUS, IEEE, ACM, SpringerLink, ScienceDirect
- Search Query and Incl/Excl criteria focused exclusively on RTD literature and quantification of RTD

RQ2: Development of the Conceptual Model

- In part, by examining what constitutes RTD quantification informed by its code-related counter part in our previous work
- and in part, by examining the literature captured via our SMS

Replication package for this work: https://zenodo.org/record/7754413
Previous work: https://arxiv.org/abs/2303.06535
Avgeriou et al.’s work that informed our previous work: https://drops.dagstuhl.de/opus/volltexte/2016/6693/
Results

- **RQ1: Learnings from literature**
  - Different authors discuss different definitions of RTD --- there is no commonly agreed formal definition
  - There is no commonly agreed way to quantify RTD
  - *Other findings included* --- Concepts related to RTD quantification, Metrics, Tools and Supported RTDM activities, RTDM strategies, RTD causes, indicators and consequences of RTD, challenges associated with RTD (*Details are in the paper*).

- This led us to formally define RTD and model RTD quantification via RQ2

- **RQ2: Conceptual model**
  - Enabled a deeper understanding of RTD and its quantification --- *key observations in later slides*
  - Enabled the comparison of RTD and software code-related TD quantification
  - Enables the comparison of different RTD quantification approaches and serves as a reference point to develop new quantification approaches.
Ernst et. al, presents the **earliest definition** on RTD (in 2012) — the distance between the optimal requirements specification and the actual system implementation, under domain assumptions and constraints.

Lenarduzzi and Fucci provide a **more comprehensive definition** of RTD by extending the definition of RTD by Ernst to include upstream activities involving the elicitation of requirements and their translation into specifications — three types of RTD: Type 0, 1, and 2, based on Incomplete Users’ needs, Requirement Smells, and Mismatch implementation, respectively.

Abad et. al, --- trade-offs in the System Requirements Specification (SRS) that result from intentional strategic decisions or unintentionally due to the changes in the context

Other definitions included --- insufficient, incomplete or outdated requirements (intentionally or unintentionally), failures in SRS, poor or partial implementation of requirements
“Requirements Technical Debt (RTD) captures the consequences of sub-optimal decisions made concerning requirements, either deliberately (for strategic gains) or inadvertently (due to changes in context), during the identification, formalization, and implementation of requirements.”
The Requirements Technical Debt Quantification Model (RTDQM) --- A conceptual model that captures the concepts related to RTD quantification and the relationships between them

Model concepts and relationships were informed by the SMS literature and our previous work
- We identified 57 concepts related to RTD quantification from the primary studies
- Based on our previous work, we categorized them into categories: process or time, cost, benefit, and probability which contained 32, 16, 1, 2 concepts, respectively
- An abstracted set of 14 concepts went into the conceptual model
Results (Model development): RTD Quantification Model (RTDQM)

<table>
<thead>
<tr>
<th>RTD Quantification Concept</th>
<th>Informed by Literature</th>
<th>Informed by TDQM counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>density (d)</td>
<td>groundness (g)</td>
</tr>
<tr>
<td>User Need</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Requirements Engineering Step</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Total Cost of a RE Step</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>(Formalized) Requirement</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>RTD Item</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>RTD Rectifying Step</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Cost of Rectifying (or remediating)</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>RTD Interest</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>New Code Cost associated with RTD</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Rework Cost associated with RTD</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>RE Cost associated with RTD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Benefit of Rectifying</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Benefit of taking RTD</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RTD Interest Probability</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

RTD Quantification Concepts — d - Num. of sources, g - Num. of concepts extracted from a source that relates to a RTD concept

- The Technical Debt Quantification Model (TDQM) in our previous work: https://arxiv.org/abs/2303.06535
Results (Model development): RTD Quantification Model (RTDQM)

Legend
- Process/Time related
- Cost
- Benefit
- Probability

User Need captures through Requirements Engineering Step incurs Total Cost of a RE Step

produces (Formalized) Requirement
N N implemented as 1

RTD Rectifying Step

incurs Benefit of Rectifying RTD
eliminates Cost of Rectifying RTD

introduces Benefit of taking RTD

RTD Item
introduces RTD Interest

accrues RTD Interest Probability

incurs RE Costs associated with RTD
considers New Code Cost associated with RTD
considered Rework Cost associated with RTD

Implementation Step introduces Feature
Results (Model development): RTD Quantification Model (RTDQM)

User Need captures through Requirements Engineering Step incurs Total Cost of a RE Step

produces (Formalized) Requirement N N implemented as 1

introduces RTD Rectifying Step incurs

elicitates Benefit of Rectifying RTD

incurs Cost of Rectifying RTD

eliminates Benefit of taking RTD

RTD Item introduces N accrues

benefits

RTD Interest considers RTD Interest Probability

constitutes of

RE Costs associated with RTD
New Code Cost associated with RTD
Rework Cost associated with RTD

Implementation Step introduces Feature
Results (Model development): RTD Quantification Model (RTDQM)
Results (Model development): RTD Quantification Model (RTDQM)
Results (Model development): RTD Quantification Model (RTDQM)
Results (Model development): RTD Quantification Model (RTDQM)

During capturing requirements or formalizing requirements
During capturing requirements or formalizing requirements

Results (Model development): RTD Quantification Model (RTDQM)
Results (Model development): RTD Quantification Model (RTDQM)
Extra costs to do additional customer interviews or to clarify ambiguous requirements
Extra costs to implement changes to a software component to accommodate a missed important user need.
some requirements might not make a difference if unmet, e.g., RTD Items pertaining to an unused feature will not incur Interest.
Results (Model development): RTD Quantification Model (RTDQM)
Results (Model development): RTD Quantification Model (RTDQM)

Total cost to fix a mismatch implementation
Benefit gained by fixing RTD early on
Results (Model development): RTD Quantification Model (RTDQM)

- **User Need** captured through **Requirements Engineering Step** incurs **Total Cost of a RE Step**
- **(Formalized) Requirement** produced by **RTD Rectifying Step** incurs **Benefit of Rectifying RTD** and **Cost of Rectifying RTD**
- **RTD Item** introduces **RTD Interest** incurs **Benefit of taking RTD**
- **RTD Interest Probability** constitutes of **RE Costs associated with RTD**, **New Code Cost associated with RTD**, and **Rework Cost associated with RTD**

Legend:
- Process/Time related
- Cost
- Benefit
- Probability

Benefit in terms of competitive advantage
Results (Model development): RTD Quantification Model (RTDQM)

Feedback loop involving the user

Legend
- Process/Time related
- Cost
- Benefit
- Probability

User Need → Requirements Engineering Step (N:1) incurs Total Cost of a RE Step

Implementation Step (1:1)

Feature

RTD Rectifying Step (N:1)
- Benefit of Rectifying RTD
- Cost of Rectifying RTD
- Benefit of taking RTD

RTD Item (1:N)
- RTD Interest
- RTD Interest Probability

RTD Interest
- RE Costs associated with RTD
- New Code Cost associated with RTD
- Rework Cost associated with RTD
Although **RTD** is similar to code-related TD (i.e., Code TD, Design TD, Architecture TD) in some aspects, it also **has its own components**

- Different from code-related TD, RTD **has a feedback loop involving the user**

- **RTD Interest** can incur **extra costs associated with Requirements Engineering as well as Implementation**

- **RTD Items** can be introduced during Requirements Engineering or Implementation activities, regardless of the presence of code-related TD

- Similar to benefits accrued by refactoring code, **rectifying RTD can also accrue benefits**
RTD applies for all types of software requirements. We are interested in understanding how the equation might change for quantifying RTD for software requirements concerning veracity.

All software systems have veracity requirements, in most cases as a non-functional requirement similar to `security' or `reliability'. An example is veracity of financial data in banking systems.

Some software systems may require specific veracity requirements to be met as functional requirements. For example, an Organic Products Certification System may have specific veracity requirements related to the organic regulation to be implemented as functional requirements.

What would be the impact of Requirements Technical Debt (RTD) for software systems implementing software requirements concerning veracity unless RTD is managed? Will stakeholders be convinced to manage RTD if the interest or the cascading impact, e.g., on software implementation or the customer, can be quantified?
This research is funded by the New Zealand Ministry of Business, Innovation and Employment via The Science for Technological Innovation (SfTI) National Science Challenge’s Veracity Technology Spearhead Research Project.

The Veracity Spearhead team for their valuable insights and support throughout.

The University of Auckland.

You can find me on LinkedIn ‘Judith Perera’ or Twitter @Perera_Judith or email me at: jper120@aucklanduni.ac.nz

Research Lab: https://hasel.auckland.ac.nz/people/

Veracity Lab: https://veracity.wgtn.ac.nz/