CAS 741 (Development of Scientific Computing Software)

Winter 2024

04 Requirements Continued

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 19, 2024

Requirements

- Administrative details
- Questions?
- Some advice
- Requirements documentation for scientific computing
- A new requirements template
- Advantages of new template and examples
- The new template from a software engineering perspective
- Concluding remarks
- References

Administrative Details

- Course schedule is available as a google calendar
- Name your repo something other than `cas741`
- Assign me an issue to review your problem statements
  - Clearly state that you would like me to review your problem statement
  - Include a link to your problem statement
- Do not put generated files under version control
- Create a `.gitignore` file
- Keep your tex files to 80 character width (easier for change tracking)

Administrative Details: Report Deadlines

Problem Statement	Week 02	Jan 19
System Req. Spec. (SRS)	Week 04	Feb 2
System VnV Plan	Week 06	Feb 16
MG + MIS	Week 09	Mar 15
Drasil Code	Week 09	Mar 15
Final Documentation	Week 13	Apr 12

- The written deliverables will be graded based on the repo contents as of 11:59 pm of the due date
- If you need an extension for a **written** doc, please ask
- When ready, assign issues to your primary and secondary reviewers
- GitHub issues due two days after assignment deadlines
- From Drasil Code onward, Drasil projects no longer need to maintain traditional SRS

Administrative Details: Presentations

SRS	Week 03/04	**Week of Jan 23, 30**
Syst. VnV	Week 06	Week of Feb 13
POC Demo	Week 06, 07	Week of Feb 13, 27
MG + MIS Syntax	Week 09	Week of Mar 13
MIS Semantics	Week 09	Week of Mar 13
Drasil	Week 11	Week of Mar 27
Unit VnV/Implement	Week 12	Week of Apr 3

- Specific schedule depends on final class registration
- Informal presentations with the goal of improving everyone's written deliverables
- Domain experts and secondary reviewers (and others) will ask questions

Presentation Schedule

TBD

Presentation Schedule

- 4 presentations each (please check)
- If you will miss a presentation, please trade with someone else
- Implementation presentation could be used to run a code review, or code walkthrough

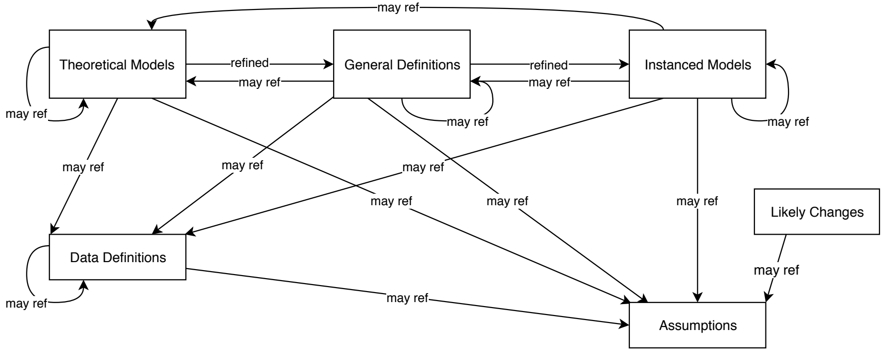
Questions?

- Does anyone need help with cloning capTemplate?
  - "Use this template" button
  - Add `smiths` as a collaborator
  - Remove unneeded folders
  - Write problem statement
  - Assign issue to `smiths`
- Does anyone need help with modifying `Repos.csv`?
  - Access CAS GitLab server
  - Fork GitLab cas741 repo
  - Fresh `git pull`
  - Modify `Repos.csv`, `git add`, `git commit`
  - Check your diff
  - Merge request
- Questions about project choices?
- Questions about software tools?
- Questions about problem statements?

Questions on Qualities for SCS?

- Correctness (Thou shalt not lie)
- Reliability
- Robustness
- Performance
- Verifiability
- Productivity
- Usability
- Maintainability
- Reusability
- Portability
- Reproducibility
- Sustainability

Major Conceptual Parts of SRS/CA



Also Goal Statements and Requirements

First Things To Think About

- Goal statement(s)
- Inputs and outputs
- Units for inputs and outputs (and for all variables)
- Think about the types of your inputs and outputs (not everything is a real number)
- Document your types
- What are some common types in scientific computing?
- How will you write these in your SRS?

Types, Specifying Types, Defining Types

- $\mathbb{R}$
- $\mathbb{R}^2$
- $\mathbb{R}^{3\times 3}$
- $\mathbb{N}$
- $\mathbb{B}$
- $\mathbb{R} \rightarrow \mathbb{R}$
- set of T
- $t \in \{2.5, 2.7, 3.0, ...\}$ (from GlassBR)
- $\text{ElemT} = \{\text{H}, \text{He}, \text{Li}, ...\}$
- sequence of T
- $v : T$
- $\text{time} = \mathbb{R}$
- Refined Projectile uses types

SRS Advice

- Abstraction is your friend
  - You don't say "how" just "what"
  - Input $f : \mathbb{R} \to \mathbb{R}$
  - Output $x^* : \mathbb{R}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}$
- Write your goal(s) first
- Brainstorm theories and assumptions
- Iterate on the relationships between the theories and assumptions
- Remind yourself that the goal is long term productivity
- Reusability - only have to be clever/smart/thorough once
- Think about the information someone would need when joining your project

Examples, Checklist and Template

- Projectile Example
- GlassBR Example
- SWHS Example
- Blank SRS from Template
- Checklist

Problems with Developing Quality Scientific Computing Software

- Need to know requirements to judge reliability
- In many cases the only documentation is the code
- Reuse is not as common as it could be
  - Meshing software survey
  - Public domain finite element programs
  - etc.
- Many people unnecessarily develop "from scratch" [1]
- Cannot easily reproduce the work of others
- Neglect of simple software development technology [12]

Adapt Software Engineering Methods

- Software engineering improves and quantifies quality
- Successfully applied in other domains
  - Business and information systems
  - Embedded real time systems
- Systematic engineering process
- Design through documentation
- Use of mathematics
- Reuse of components
- Warranty rather than a disclaimer

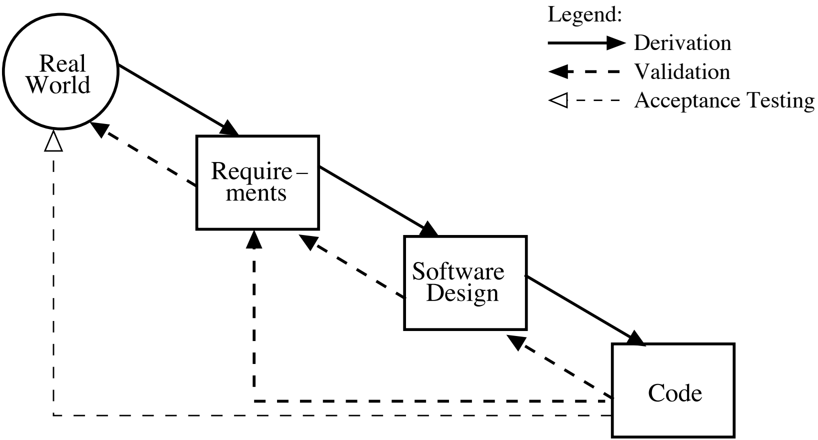
Developing Scientific Computing Software

- Facilitators
  - One user viewpoint for specifying a physical model
  - Assumptions can be used to distinguish models
  - High potential for reuse
  - Libraries
  - Already mathematical
- Challenges
  - Verification and Validation
  - Acceptance of software engineering methodologies
  - No existing templates or examples

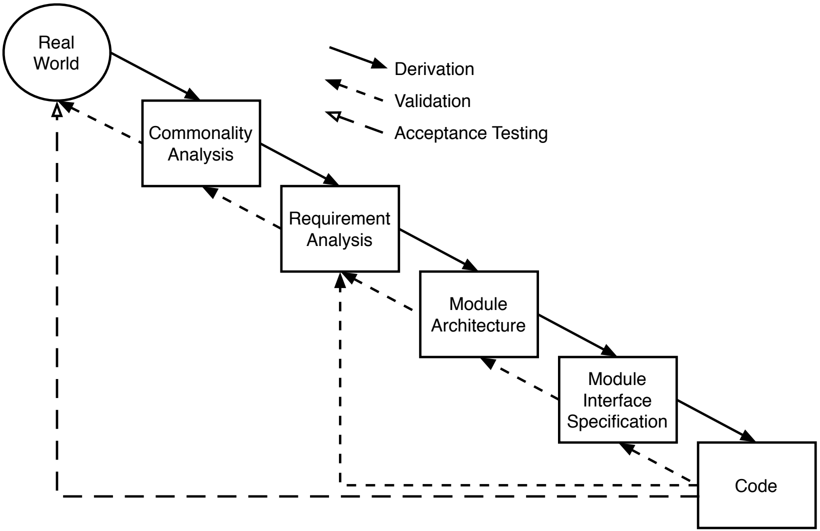
Outline of Discussion of Requirements

- Background on requirements elicitation, analysis and documentation
- Why requirements analysis for engineering computation?
- System Requirements Specification and template for beam analysis software
  - Provides guidelines
  - Eases transition from general to specific
  - Catalyses early consideration of design
  - Reduces ambiguity
  - Identifies range of model applicability
  - Clear documentation of assumptions

A Rational Design Process



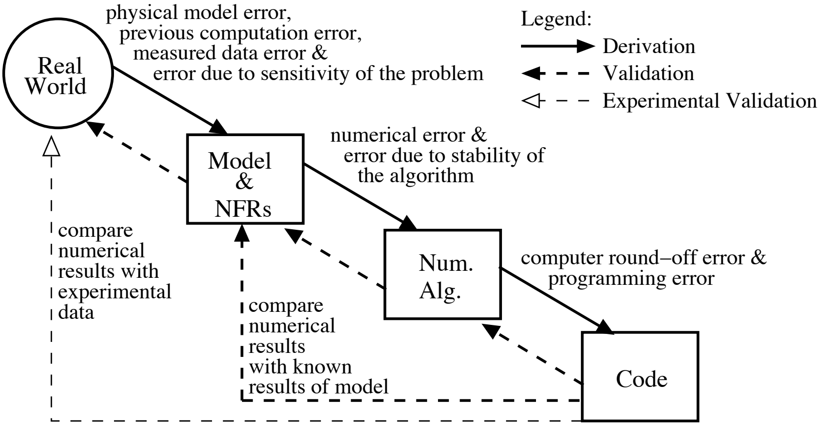
Sometimes Include Commonality Analysis



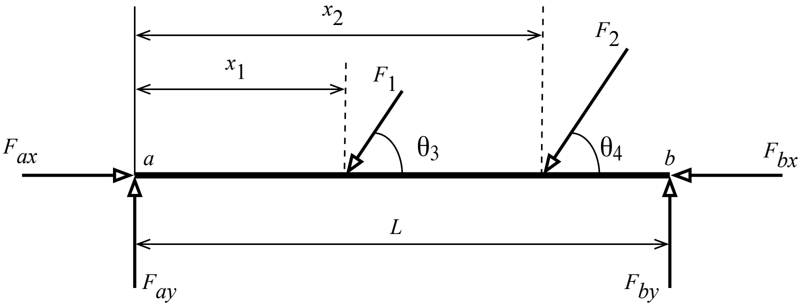
Software Requirements Activities

- A software requirement is a description of how the system should behave, or of a system property or attribute
- Requirements should be abstract, unambiguous, complete, consistent, modifiable, verifiable and traceable
- Requirements should express “What” not “How”
- Formal versus informal specification
- Functional versus nonfunctional requirements
- Software requirements specification (SRS)
- Requirements template

Why Requirements Analysis?



Beam Analysis Software



Proposed Template

From [11]

1. Reference Material: a) Table of Symbols ...
2. Introduction: a) Purpose of the Document; b) Scope of the Software Product; c) Organization of the Document.
3. General System Description: a) System Context; b) User Characteristics; c) System Constraints.
4. Specific System Description:
    4.1 Problem Description: i) Background Overview ...
    4.2 Solution specification: i) Assumptions; ii) Theoretical Models; ...
    4.3 Non-functional Requirements: i) Accuracy of Input Data; ii) Sensitivity ...
5. Traceability Matrix
6. List of Possible Changes in the Requirements
7. Values of Auxiliary Constants

Provides Guidance

- Details will not be overlooked, facilitates multidisciplinary collaboration
- Encourages a systematic process
- Acts as a checklist
- Separation of concerns
  - Discuss purpose separately from organization
  - Functional requirements separate from non-functional
- Labels for cross-referencing
  - Sections, physical system description, goal statements, assumptions, etc.
  - PS1.a "the shape of the beam is long and thin"

Eases Transition from General to Specific

- "Big picture" first followed by details
- Facilitates reuse
- "Introduction" to "General System Description" to "Specific System Description"
- Refinement of abstract goals to theoretical model to instanced model
  - **G1**. Solve for the unknown external forces applied to the beam
  - **T1** $\sum F_{xi} = 0$, $\sum F_{yi} = 0$, $\sum M_i = 0$
  - **M1** $F_{ax} - F_1 \cdot \cos\theta_3 - F_2 \cdot \cos\theta_4 - F_{bx} = 0$

Ensures Special Cases are Considered

H_1

		$S_{GET} = S_{sym} - S_{unkF}$	$S_{GET} \neq (S_{sym} - S_{unkF})$
$S_{unkF} \notin \mathbb{P}_3$	-	$(ErrorMsg' = InvalidUnknown) \wedge ChangeOnly(ErrorMsg)$	
$S_{unkF} = \{@F_{ax}, @F_{bx}, @F_{ay}\}$	-	$ErrorMsg' = NoSolution \wedge ChangeOnly(ErrorMsg)$	
$S_{unkF} = \{@F_{ax}, @F_{ay}, @F_1\}$	$x_1 \neq 0 \wedge \theta_3 \neq 0 \wedge \theta_3 \neq 180$	$F'_{ax} = \frac{-\cos\theta_3 F_2 x_2 \sin\theta_4 + \cos\theta_3 F_{by} L + F_2 \cos\theta_4 x_1 \sin\theta_3 + F_{bx} x_1 \sin\theta_3}{x_1 \sin\theta_3} \wedge F'_{ay} = -\frac{F_2 x_2 \sin\theta_4 - F_{by} L - F_2 \sin\theta_4 x_1 + F_{by} x_1}{x_1} \wedge F'_1 = \frac{-F_2 x_2 \sin\theta_4 + F_{by} L}{x_1 \sin\theta_3} \wedge ChangeOnly(S_{unkF})$	*FALSE*
	otherwise	$(ErrorMsg' = Indeterminant) \wedge ChangeOnly(ErrorMsg)$	

H_2 $\qquad$ G

Catalyses Early Consideration of Design

- Identification of significant issues early will improve the design
- Section for considering sensitivity
  - Conditioning?
  - Buckling of beam
- Non-functional requirements
  - Tradeoffs in design
  - Speed efficiency versus accuracy
- Tolerance allowed for solution: $|\sum F_{xi}|/\sqrt{\sum {F_{xi}}^2} \leq \epsilon$
- Solution validation strategies (now in a separate document)
- List of possible changes in requirements

Reduces Ambiguity

- Unambiguous requirements allow communication between experts, requirements review, designers do not have to make arbitrary decisions
- Tabular expressions allow automatic verification of completeness
- Table of symbols
- Abbreviations and acronyms
- Scope of software product and system context
- User characteristics
- Terminology definition and data definition
- Ends arguments about the relative merits of different designs

Identifies Range of Model Applicability

- Clear documentation as to when model applies
- Can make the design specific to the problem
- Input data constraints are identified
  - Physically meaningful: $0 \leq x_1 \leq L$
  - Maintain physical description: PS1.a, $0 < h \leq 0.1L$
  - Reasonable requirements: $0 \leq \theta_3 \leq 180$
- The constraints for each variable are documented by tables, which are later composed together
- $(min_f \leq |F_{ax}| \leq max_f) \wedge (|F_{ax}| \neq 0) \Rightarrow$ $\forall(FF|@FF \in S_F \cdot FF \neq 0 \wedge \frac{max\{|F_{ax}|,|FF|\}}{min\{|F_{ax}|,|FF|\}} \leq 10^{r_f})$
- Typical values

Summary of Variables

Var	Type	Physical Constraints	System Constraints	Prop
x	*Real*	$x \geq 0 \wedge x \leq L$	$min_d \leq x \leq max_d$	NIV
x_1	*Real*	$x_1 \geq 0 \wedge x_1 \leq L$	$min_d \leq x_1 \leq max_d$	IN
x_2	*Real*	$x_2 \geq 0 \wedge x_2 \leq L$	$min_d \leq x_2 \leq max_d$	IN
e	*Real*	$e > 0 \wedge e \leq h$	$min_e \leq e \leq max_e$	IN
h	*Real*	$h > 0 \wedge h \leq 0.1L$	$min_h \leq h \leq max_h$	IN
L	*Real*	$L > 0$	$min_d \leq L \leq max_d$	IN
E	*Real*	$E > 0$	$min_E \leq E \leq max_E$	IN
θ_3	*Real*	$-\infty < \theta_3 < +\infty$	$0 \leq \theta_3 \leq 180$	IN
θ_4	*Real*	$-\infty < \theta_4 < +\infty$	$0 \leq \theta_4 \leq 180$	IN
V	*Real*	$-\infty < V < +\infty$	-	OUT
M	*Real*	$-\infty < M < +\infty$	-	OUT
y	*Real*	$-\infty < y < +\infty$	-	OUT
...	...	...	...	...

Clear Documentation of Assumptions

Phy. Sys. /Goal	Data /Model	Assumption										Model	
		A1	A2	...	A4	...	A8	A9	A10	...	A14	**M1**	...
G1	**T1**	√		...		...	√	√		...		√	...
G2	**T2**	√		...		...	√	√		...			...
G3	**T3**	√		...		...		√	√	...			...
	M1		√	...		...				...		√	...
PS1.a	L			...		...			√	...		...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...

A10. The deflection of the beam is caused by bending moment only, the shear does not contribute.

More on the Template

- Why a new template?
- The new template
  - Overview of changes from existing templates
  - Goal $\rightarrow$ Theoretical Model $\rightarrow$ Instanced Model hierarchy
  - Traceability matrix
  - System behaviour, including input constraints

Why a New Template?

From [10, 3]

1. One user viewpoint for the physical model
2. Assumptions distinguish models
3. High potential for reuse of functional requirements
4. Characteristic hierarchical nature facilitates change
5. Continuous mathematics presents a challenge

Overview of the New Template

- Reference Material
- Introduction: a) Purpose of the Document b) Scope of the Software Product c) Organization of the Document
- General System Description: a) System Context b) User Characteristics c) System Constraints
- Specific System Description: a) Problem Description b) Solution Characteristics Specification c) Non-functional Requirements
- Other System Issues
- Traceability Matrix
- List of Possible Changes in the Requirements
- Values of Auxiliary Constants
- References

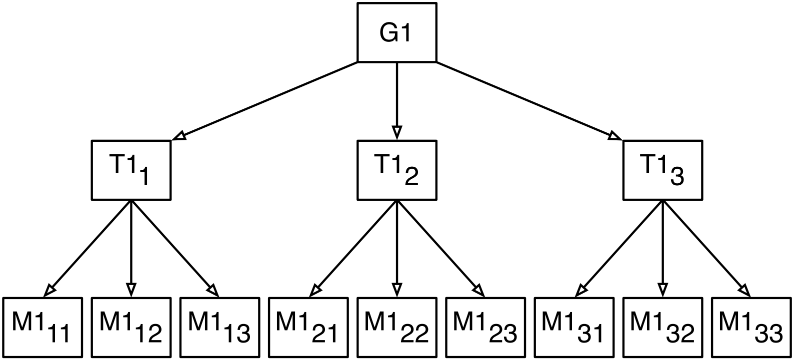
Overview of the New Template

- Reference Material
- Introduction: a) Purpose of the Document b) Scope of the Software Product c) Organization of the Document
- General System Description: a) System Context b) User Characteristics c) System Constraints
- Specific System Description: a) Problem Description b) Solution Characteristics Specification c) Non-functional Requirements
- Other System Issues
- Traceability Matrix
- List of Possible Changes in the Requirements
- Values of Auxiliary Constants
- References

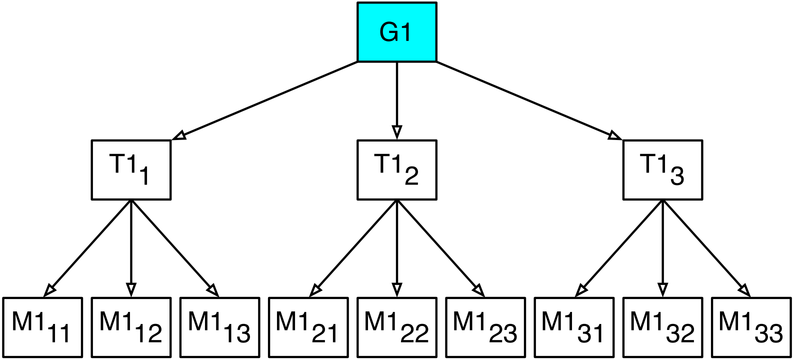
Excerpts from Specific System Description

- Problem Description
  - Physical system description (**PS**)
  - Goals (**G**)
- Solution Characteristics Specification
  - Assumptions (**A**)
  - Theoretical models (**T**)
  - Data definitions
  - Instanced models (**M**)
  - Data constraints
  - System behaviour
- Non-functional Requirements
  - Accuracy of input data
  - Sensitivity of the model
  - Tolerance of the solution
  - Solution validation strategies (now moved to a separate document)

Refinement from Abstract to Concrete

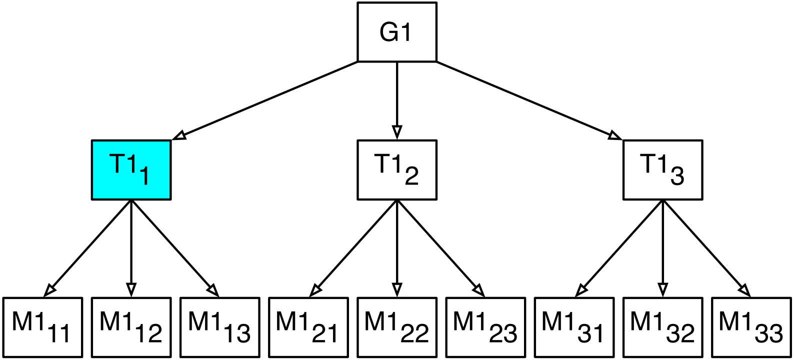


Refinement from Abstract to Concrete



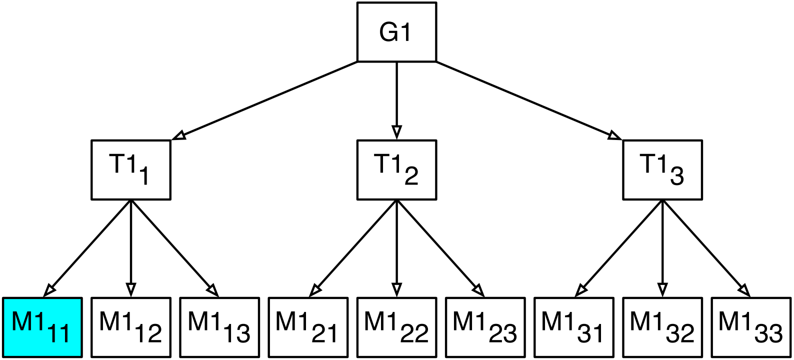
G1: Solve for unknown forces

Refinement from Abstract to Concrete



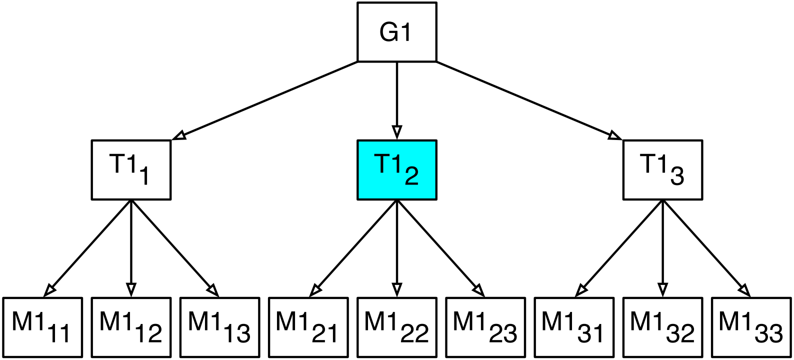
$$(\mathbf{T1_1}) \begin{cases} \sum F_{xi} = 0 \\ \sum F_{yi} = 0 \\ \sum M_i = 0 \end{cases}$$

Refinement from Abstract to Concrete



$$\textbf{(M1)}\ \begin{cases} F_{ax} - F_1 \cdot \cos\theta_3 - F_2 \cdot \cos\theta_4 - F_{bx} = 0 \\ F_{ay} - F_1 \cdot \sin\theta_3 - F_2 \cdot \sin\theta_4 + F_{by} = 0 \\ -F_1 \cdot x_1 \sin\theta_3 - F_2 \cdot x_2 \sin\theta_4 + F_{by} \cdot L = 0 \end{cases}$$

Refinement from Abstract to Concrete



The virtual work done by all the external forces and couples acting on the system is zero for each independent virtual displacement of the system, or mathematically $\delta U = 0$

Other goals and models

- **G2**: Solve for the functions of shear force and bending moment along the beam
- **G3**: Solve for the function of deflection along the beam
- $\mathbf{T3_1}$: $\frac{d^2y}{dx^2} = \frac{M}{EI}$, $y(0) = y(L) = 0$
- $\mathbf{T3_2}$: y determined by moment area method
- $\mathbf{T3_3}$: y determined using Castigliano's theorem
- $\mathbf{M3_{11}}$: $y = \frac{12 \int_0^L (\int_0^L M dx) dx}{Eeh^3}$, $y(0) = y(L) = 0$

Kreyman and Parnas Five Variable Model

- See [2]
- An alternative approach
- Unfortunately the numerical algorithm is not hidden in the requirements specification
- The analogy with real-time systems leads to some confusion

Links to Papers

- SmithAndChen2004 [8] Commonality Analysis overview
- SmithAndChen2004b [7] CA example MG
- SmithAndLai2005 [10] New requirements template
- Smith2006 [5] General purpose tool template
- SmithEtAl2007 [11] Template for SC audience
- SmithAndKoothoor2016 [9] Nuclear fuelpin example
- SmithEtAl2019_arXiv [4] Debunk upfront myth
- Smith2016 [6] Overview of artifacts
- KreymanAndParnas2002 [2] 4 variable method

Summary of Template

- Quality is a concern for scientific computing software
- Software engineering methodologies can help
- Motivated, justified and illustrated a method of writing requirements specification for engineering computation to improve reliability
- Also improve quality with respect to usability, verifiability, maintainability, reusability and portability
- Tabular expressions to reduce ambiguity, encourage systematic approach
- Conclusions can be generalized because other computation problems follow the same pattern of *Input* then *Calculate* then *Output*
- Benefits of approach should increase as the number of details and the number of people involved increase

Summary of Template (Continued)

- A new template for scientific computing has been developed
- Characteristics of scientific software guided the design
- Designed for reuse
- Functional requirements split into "Problem Description" and "Solution Characteristics Specification"
- Traceability matrix
- Addresses nonfunctional requirements (but room for improvement)

References I

Paul F. Dubois.
Designing scientific components.
Computing in Science and Engineering, 4(5):84–90, September 2002.

K. Kreyman and D. L. Parnas.
On documenting the requirements for computer programs based on models of physical phenomena.
SQRL Report 1, Software Quality Research Laboratory, McMaster University, January 2002.

References II

Lei Lai.
Requirements documentation for engineering mechanics software: Guidelines, template and a case study.
Master's thesis, McMaster University, Hamilton, Ontario, Canada, 2004.

Spencer Smith, Malavika Srinivasan, and Sumanth Shankar.
Debunking the myth that upfront requirements are infeasible for scientific computing software debunking the myth that upfront requirements are infeasible for scientific computing software.
https://arxiv.org/abs/1906.07812, June 2019.

References III

W. Spencer Smith.
Systematic development of requirements documentation for general purpose scientific computing software.
In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006.

W. Spencer Smith.
A rational document driven design process for scientific computing software.
In Jeffrey C. Carver, Neil Chue Hong, and George Thiruvathukal, editors, *Software Engineering for Science*, Chapman & Hall/CRC Computational Science, chapter Examples of the Application of Traditional Software

References IV

Engineering Practices to Science, pages 33–63. Chapman and Hall/CRC, Boca Raton, FL, 2016.

W. Spencer Smith and Chien-Hsien Chen.
Commonality analysis for mesh generating systems.
Technical Report CAS-04-10-SS, McMaster University, Department of Computing and Software, 2004.
45 pp.

References V

W. Spencer Smith and Chien-Hsien Chen.
Commonality and requirements analysis for mesh generating software.
In F. Maurer and G. Ruhe, editors, *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2004)*, pages 384–387, Banff, Alberta, 2004.

W. Spencer Smith and Nirmitha Koothoor.
A document-driven method for certifying scientific computing software for use in nuclear safety analysis.
Nuclear Engineering and Technology, 48(2):404–418, April 2016.

References VI

W. Spencer Smith and Lei Lai.
A new requirements template for scientific computing.
In J. Ralyté, P. Ȧgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

References VII

W. Spencer Smith, Lei Lai, and Ridha Khedri.
Requirements analysis for engineering computation: A systematic approach for improving software reliability.
Reliable Computing, Special Issue on Reliable Engineering Computation, 13(1):83–107, February 2007.

Gregory V. Wilson.
Where's the real bottleneck in scientific computing? Scientists would do well to pick some tools widely used in the software industry.
American Scientist, 94(1), 2006.