CAS 741, CES 741 (Development of Scientific Computing Software)

Fall 2017

02 Requirements

Dr. Spencer Smith

Faculty of Engineering, McMaster University

September 7, 2017

Requirements

- LiCS overview by Dan
- Administrative details
- Project choices
- Software tools
- Motivation
  - Scientific Computing Software Qualities
  - Challenges to Developing Quality Scientific Software
- 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

- Benches and white boards
- Use folder structure given in repo
- Problem statement due Friday, Sept 15 by 11:59 pm

Administrative Details: Grade Assessment

1. Presentations and class discussion 10%
2. Quality of GitHub issues provided to classmates 5%
3. Problem Statement 0%
4. System Requirements Specification (SRS) 20%
5. Verification and Validation Plan 10%
6. Module Guide (MG) 10%
7. Module Interface Specification (MIS) 10%
8. Final Documentation (including revised versions of previous documents, plus the source code and a testing report) 35%

Administrative Details: Report Deadlines

Problem Statement	Week 02	Sept 15
System Requirements Specification (SRS)	Week 05	Oct 4
Verification and Validation Plan	Week 07	Oct 25
Module Guide (MG)	Week 09	Nov 8
Module Interface Specification (MIS)	Week 11	Nov 22
Final Documentation	Week 13	Dec 6

- 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, please ask
- Two days after each major deliverable, your GitHub issues will be due

Administrative Details: Presentations

SRS Present	Week 04	Week of Sept 25
V&V Present	Week 06	Week of Oct 16
MG Present	Week 08	Week of Oct 30
MIS Present	Week 10	Week of Nov 13
Implementation Present	Week 12	Week of Nov 27

- Tentative dates
- Specific schedule depends on final class registration and need
- Informal presentations with the goal of improving everyone's written deliverables

Project Selection: Desired Qualities

- Related to scientific computing
- Simple, but not trivial
- If feasible, select a project related to your research
- Ideally, re-implement existing software
- Each student project needs to be unique
- Possibly a specific physical problem
- Possibly a (family of) general purpose tool(s)
- Some examples follow, the links are just places to get started

Project Selection: Specific Physical Problem

- Heated rod
- Heated plate
- Double pendulum
- Rigid body dynamics
- Column buckling
- Damped harmonic oscillator
- Stoichiometric calculations (chemical balance)
- Predator prey dynamics
- Imaging: filters, edge detection etc.
- etc.

Project Selection: Family of General Purpose Tools

- Solution of ODEs
- Solution of $Ax = b$
- Regression
- Interpolation
- Numerical integration
- FFT
- Mesh generation
- Finite element method
- Any chapter from a standard numerical methods textbook
- etc.

Tool Tutorials

- point to repo
- Learn by doing

Git and GitLab

- point to repo
- Learn by doing

LaTeX

- point to repo
- Learn by doing

Important Qualities for Scientific Computing Software

- External qualities
  - Correctness (Thou shalt not lie)
  - Reliability
  - Robustness
  - Performance
    - Time efficiency
    - Space efficiency
- Internal qualities
  - Verifiability
  - Usability
  - Maintainability
  - Reusability
  - Portability

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 develop "from scratch"
- Cannot easily reproduce the work of others
- Neglect of simple software development technology [Wilson(2006)]

Adapt Software Engineering Methodologies

- 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

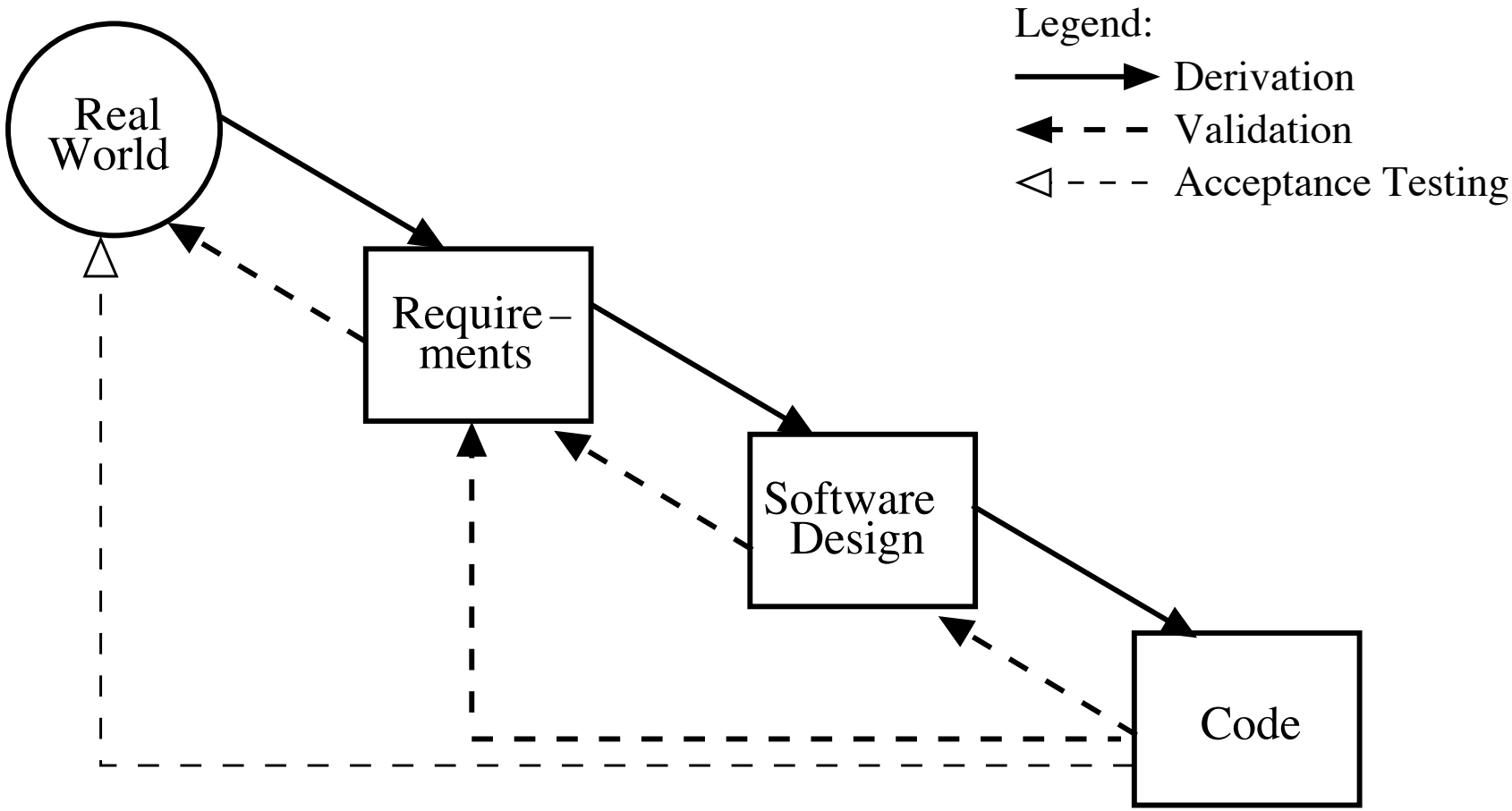
Developing Scientific Computing Software

- 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

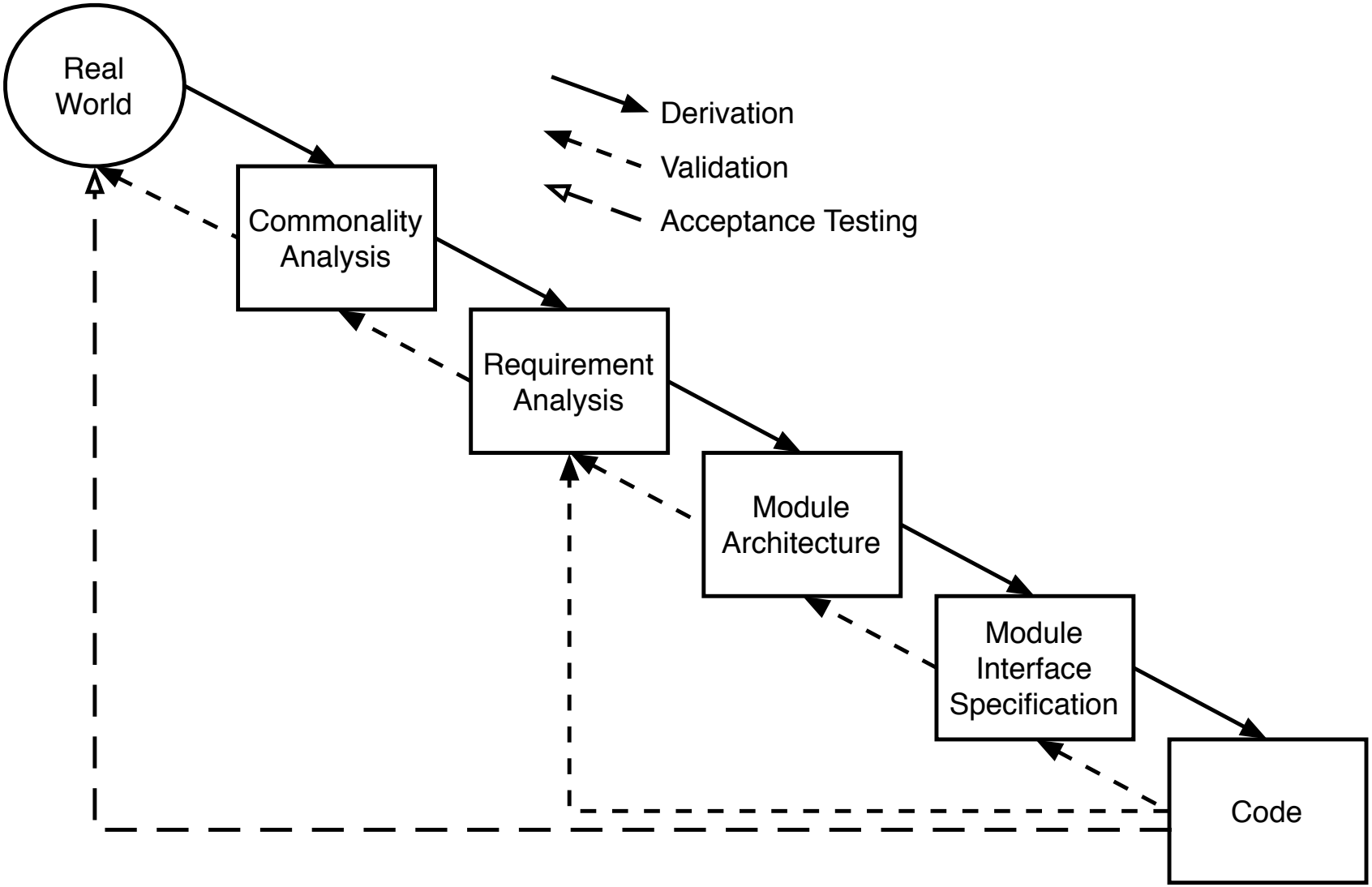
Outline of Discussion of Requirements

- Background on requirements elicitation, analysis and documentation
- Tabular expressions
- 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 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

Tabular Expressions

Composition rule	$\cup_{i=1}^{4} H_2[i] \cap (\cap_{j=1}^{2} H_1[j] \ ; \ G[i,j])$

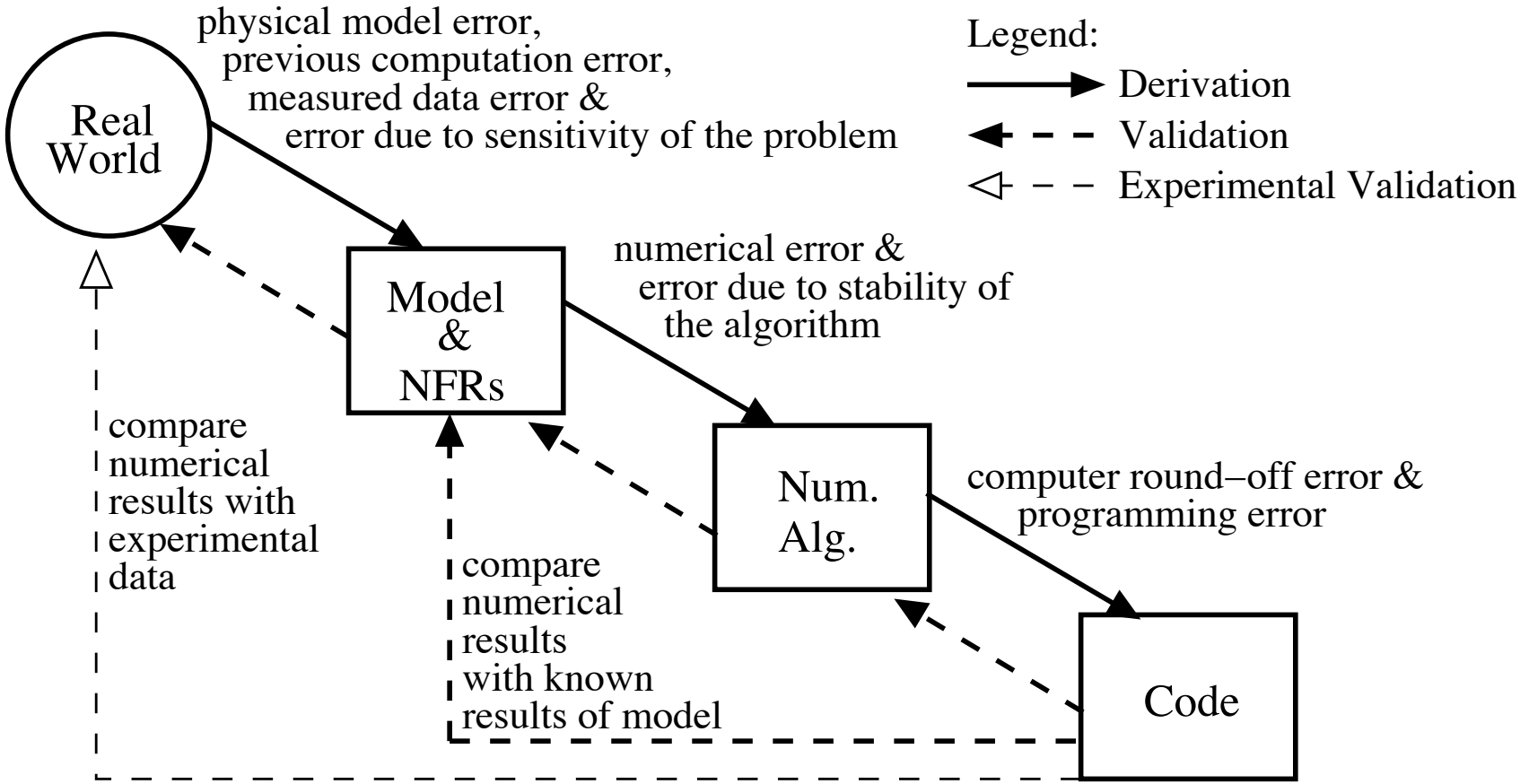
H_1

$S'_{GET} \cup =$	$ErrorMsg' + =$

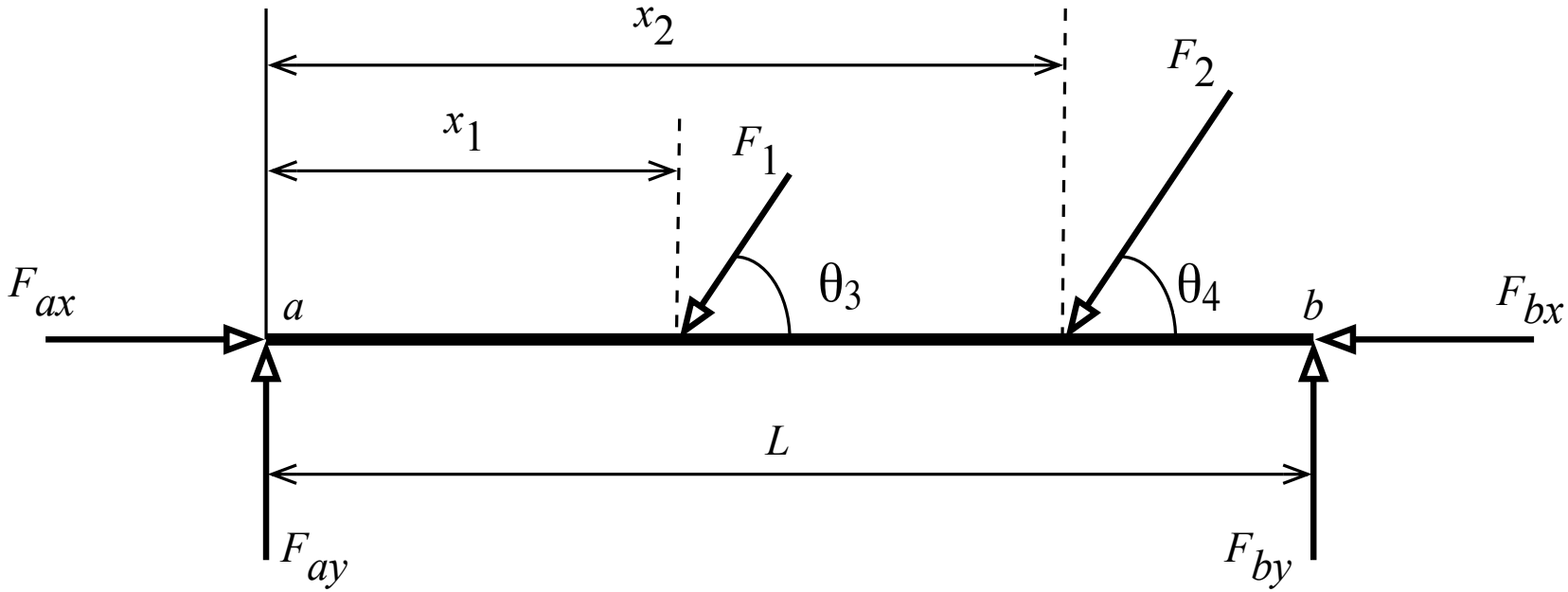
H_2	G
$x_1 < 0$	$\emptyset$, $InvalidInput\_x_1$
$0 \leq x_1 < min_d$	$\emptyset$, $x_1\_TooSmall$
$x_1 > max_d$	$\emptyset$, $x_1\_TooLarge$
$min_d \leq x_1 \leq max_d$	$\{@x_1\}$, $NULL$

$\wedge ChangeOnly(S_{GET}, ErrorMsg)$

Why Requirements Analysis?



Beam Analysis Software



Proposed Template

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_2

$S_{unkF} \notin \mathbb{P}_3$	-
$S_{unkF} = \{@F_{ax}, @F_{bx}, @F_{ay}\}$	-
$S_{unkF} = \{@F_{ax}, @F_{ay}, @F_1\}$	$x_1 \neq 0 \wedge \theta_3 \neq 0 \wedge \theta_3 \neq 180$
	otherwise

H_1

G

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

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
- 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
- $(min_f \leq |F_{ax}| \leq max_f) \wedge (|F_{ax}| \neq 0) \Rightarrow$
  $\forall(FF|@FF \in S_F \cdot FF \neq 0 \wedge \frac{max\{|F_{ax}|,|FF|\}}{min\{|F_{ax}|,|FF|\}} \leq 10^{r_f})$

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 New 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?

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

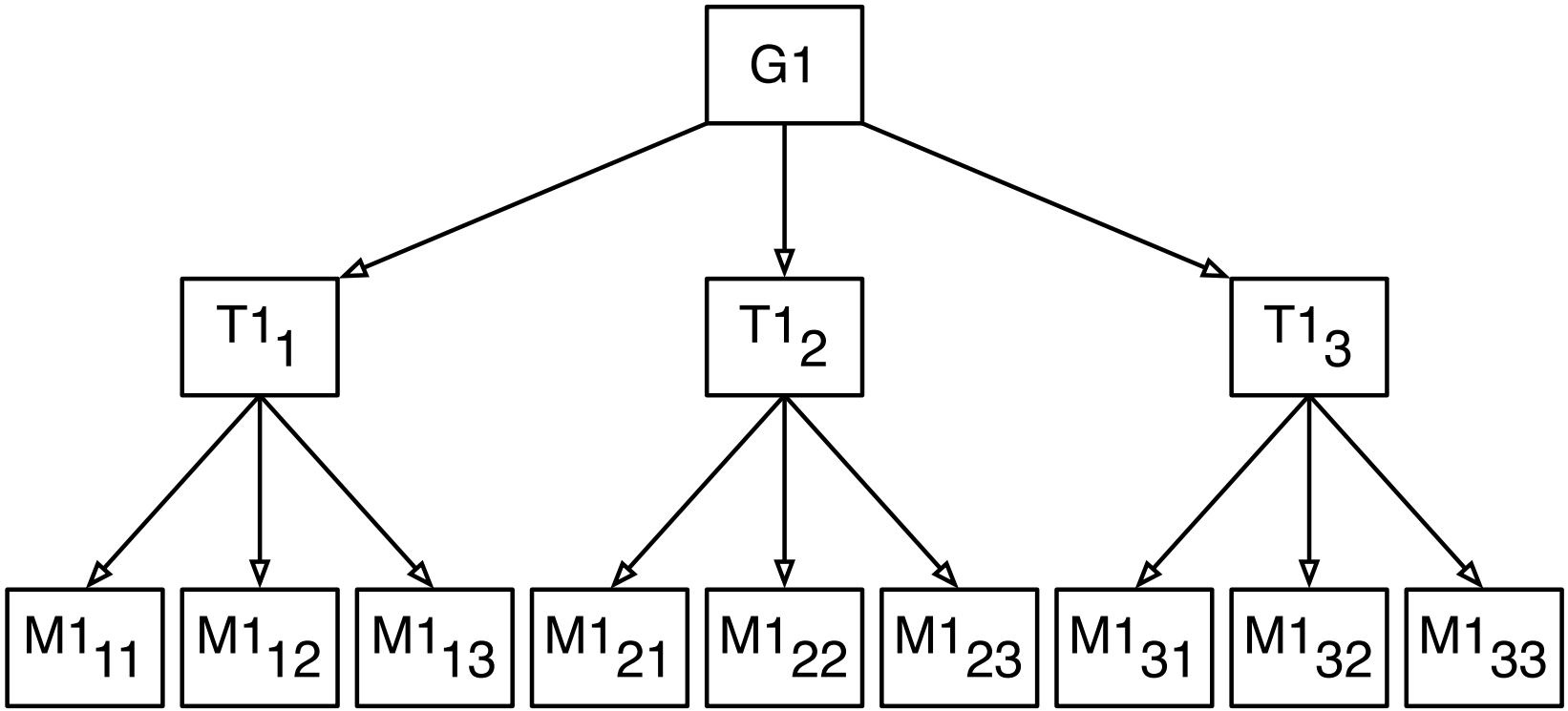
Overview of the New Template

- 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

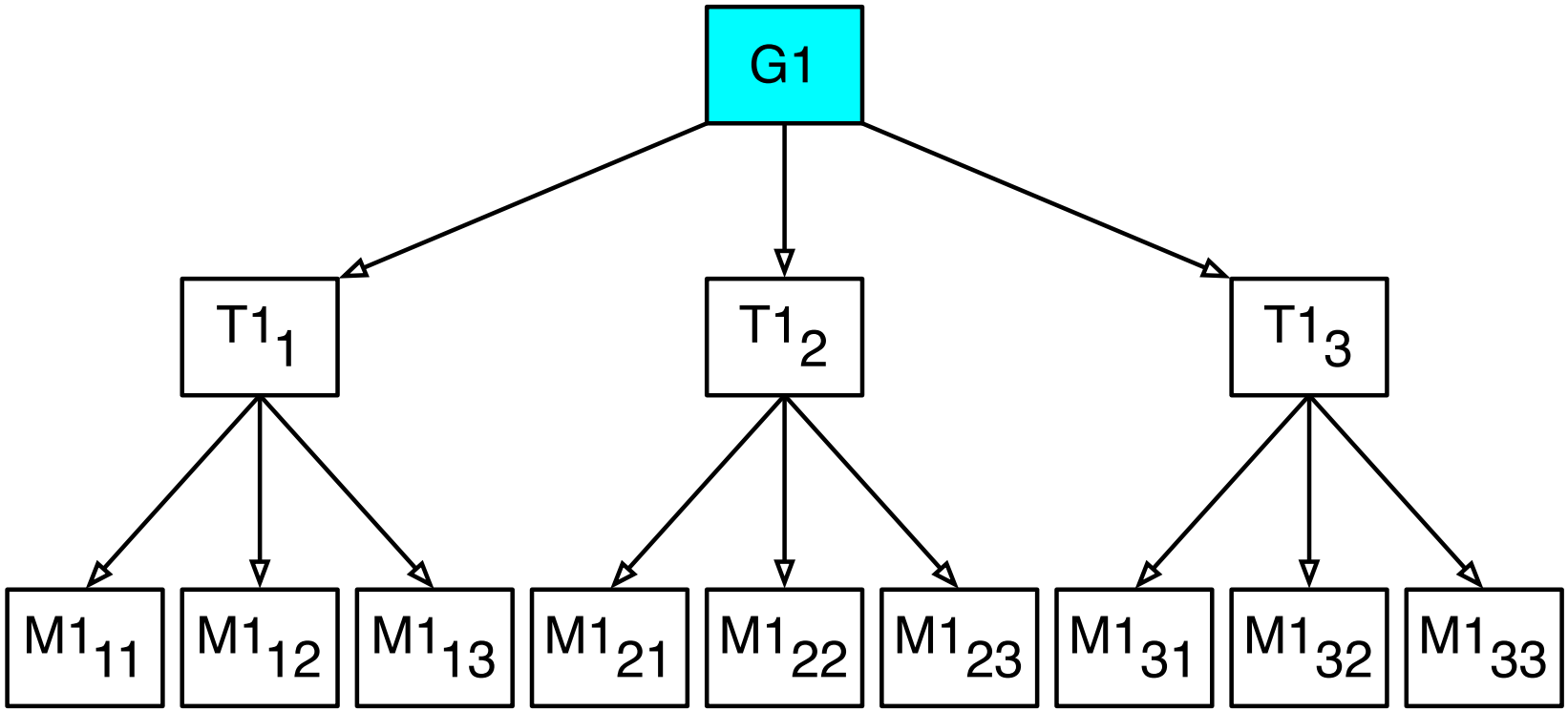
Excerpts from Specific System Description

- 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

Refinement from Abstract to Concrete

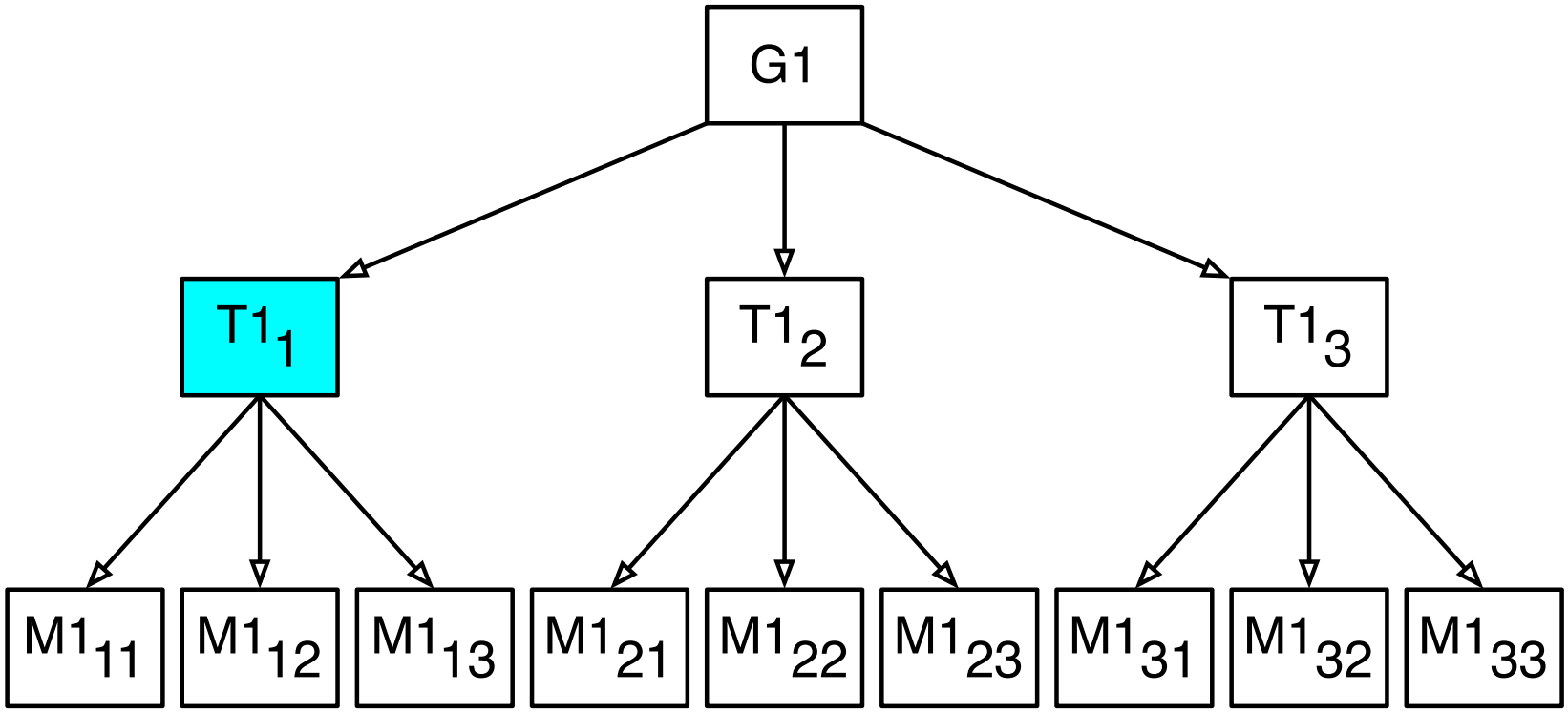


Refinement from Abstract to Concrete



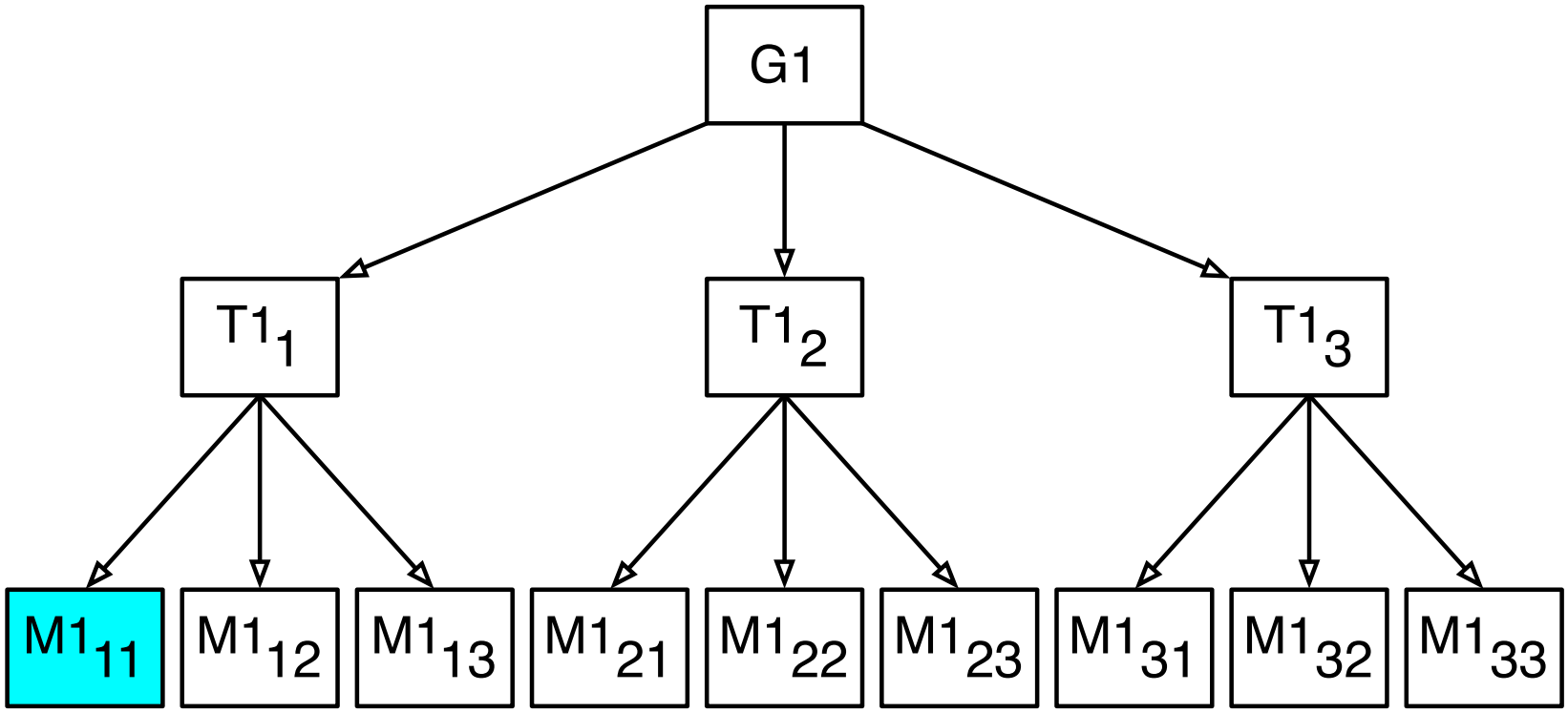
G1: Solve for unknown forces

Refinement from Abstract to Concrete



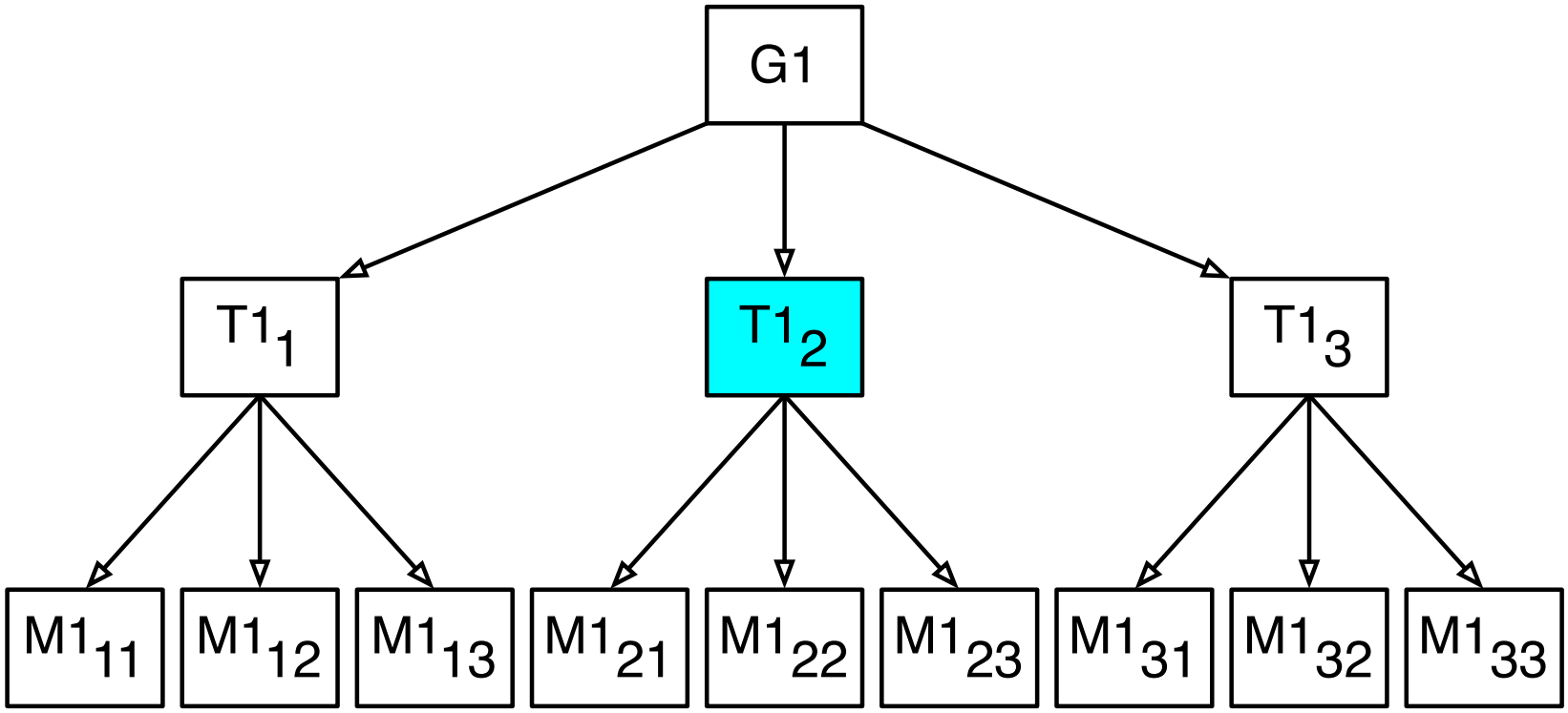
$$(\mathbf{T1_1}) \begin{cases} \sum F_{xi} = 0 \\ \sum F_{yi} = 0 \\ \sum M_i = 0 \end{cases}$$

Refinement from Abstract to Concrete



(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

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

Concluding Remarks

- 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

Concluding Remarks (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


Jules Desharnais, Ridha Khedri, and Ali Mili.
Representation, validation and integration of scenarios using tabular expressions.
Formal Methods in System Design, page 40, 2004.
To appear.


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


Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli.
Fundamentals of Software Engineering.
Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

References II

IEEE.
Recommended practice for software requirements specifications.
IEEE Std 830-1998, pages 1–40, Oct 1998.
doi: 10.1109/IEEESTD.1998.88286.

R. Janicki and R. Khedri.
On a formal semantics of tabular expression.
Science of Computer Programming, 39(2-3):189–213, 2001.

References III

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.

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

References IV


David L. Parnas and P.C. Clements.
A rational design process: How and why to fake it.
IEEE Transactions on Software Engineering, 12(2): 251–257, February 1986.


Suzanne Robertson and James Robertson.
Mastering the Requirements Process, chapter Volere Requirements Specification Template, pages 353–391.
ACM Press/Addison-Wesley Publishing Co, New York, NY, USA, 1999.

References V

Judith Segal.
When software engineers met research scientists: A case study.
Empirical Software Engineering, 10(4):517–536, October 2005.
ISSN 1382-3256.
doi: 10.1007/s10664-005-3865-y.
URL http://dx.doi.org/10.1007/s10664-005-3865-y.

References VI

Judith Segal.
Some problems of professional end user developers.
In *VLHCC '07: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 111–118, Washington, DC, USA, 2007a. IEEE Computer Society.
ISBN 0-7695-2987-9.
doi: http://dx.doi.org/10.1109/VLHCC.2007.50.

Judith Segal.
End-user software engineering and professional end-user developers.
In *Dagstuhl Seminar Proceedings 07081, End-User Software Engineering*, 2007b.
URL http://drops.dagstuhl.de/opus/volltexte/2007/1095.

References VII

Judith Segal.
Models of scientific software development.
In *Proceedings of the First International Workshop on Software Engineering for Computational Science and Engineering (SECSE 2008)*, pages 1–6, Leipzig, Germany, 2008. In conjunction with the 30th International Conference on Software Engineering (ICSE).
URL
http://www.cse.msstate.edu/~SECSE08/schedule.htm.

References VIII


W. Spencer Smith and Lei Lai.
A new requirements template for scientific computing.
In J. Ralyté, P. Ågerfalk, 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 IX


W. Spencer Smith, Lei Lai, and Ridha Khedri.
Requirements analysis for engineering computation.
In R. Muhanna and R. Mullen, editors, *Proceedings of the NSF Workshop on Reliable Engineering Computing*, pages 29–51, Savannah, Georgia, 2004.


R. H. Thayer and M. Dorfman, editors.
IEEE Recommended Practice for Software Requirements Specifications.
IEEE Computer Society, Washington, DC, USA, 2nd edition, 2000.


The Institute of Electrical and Electronics Engineers, Inc.
Software Requirements Engineering.
IEEE Computer Society Press, 2nd edition, 2000.

References X

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.
URL http://www.americanscientist.org/issues/pub/wheres-the-real-bottleneck-in-scientific-computing.