Commonality Analysis of Families of Physical Models for use in Scientific Computing

W. Spencer Smith
Computing and Software
Department
McMaster University
1280 Main Street West
Hamilton, Ontario,
CANADA L8S 4K1
smiths@mcmaster.ca

Jacques Carette
Computing and Software
Department
McMaster University
1280 Main Street West
Hamilton, Ontario,
CANADA L8S 4K1
carette@mcmaster.ca

John McCutchan
Computing and Software
Department
McMaster University
1280 Main Street West
Hamilton, Ontario,
CANADA L8S 4K1
john@johnmccutchan.com

ABSTRACT

This paper presents a template for the commonality analysis of a family of models of physical phenomena. The commonality analysis template includes documentation of the potential system context, the variabilities between models and the common aspects shared by all family members, including sections for terminology, goals, assumptions and theoretical models. The commonality analysis document also explicitly shows the dependence between terminology, goals, assumptions, theoretical models and variabilities. The proposed template is demonstrated by presenting the example of a family of constitutive equations, which are equations that model the deformation of a material particle under an applied loading.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification; J.2.0 [**Computer Applications**]: Physical Sciences and Engineering

Keywords

Commonality analysis, program families, scientific computing, models of physical phenomena

1. INTRODUCTION

Scientific computing (SC) problems involve the use of computer tools to analyze and simulate physical models of real world systems so that scientists and engineers can better understand and predict the behaviour of those systems. Although the quality of the computer tools, and associated algorithms and code, is very important in SC, their quality does not matter if the first step of building the physical model is incorrect, or inaccurate. If the wrong model is chosen, then all subsequent steps will yield meaningless results.

Given the importance of the physical model, it is surprising that in many cases little attention is paid to its documentation. Although the governing equations and boundary conditions are usually documented, in many instances not all of the assumptions are made explicit and terminology, sign conventions and the meaning of different symbols can remain ambiguous. For instance, when the Euler-Bernoulli equation is used for modelling the deformation of a beam, the documentation of the model seldom explicitly states the assumption that the beam is long and thin. When there are potential problem like this, where ambiguity and communication between different stakeholders is an issue, the usual solution advocated by software engineers is to systematically gather, analyze and document the requirements. Therefore, requirements documentation for physical models is promoted here as a way to improve the quality of SC software.

Previous work has looked at the documentation of requirements for physical models [17, 18], but this previous work has not considered the case where the requirements are for a family of physical models. With a family approach multiple models are documented concurrently. For instance, using the family approach, the requirements for a thin beam and a thick beam are documented together as two members of a family of beam models. Within the requirements document for the family, sometimes termed a commonality analysis (CA) document [22], the common requirements are highlighted and the variabilities between the models are enumerated and specified. This paper motivates and illustrates the use of a CA for families of physical models for use in SC.

Where applicable, a CA is recommended for families of physical models because of the benefits it provides. To begin with, a CA provides all the benefits usually associated with a requirements document. For instance, a CA can provide improved communication, a basis for estimating costs and schedules, improved opportunities for error detection and a foundation for incremental delivery. In addition, a CA provides the benefits usually associated with the development of software as a program family, such as reduced development time, improved quality, reduced maintenance effort and the ability to cope with complexity. A particularly exciting advantage of a CA is that it can form the starting point for the design of a domain specific language (DSL) [20]. A DSL facilitates the specification of family members using a small language tailored to the problem domain of interest. The

DSL can then be used, often via code generation, to quickly create (generate) individual family members.

A family approach also helps SC because it improves the usability of the software. Usability often suffers in SC because of the current trend to develop physics solvers that are as widely applicable as possible. For instance, modern stress analysis programs allow for intricate three dimensional boundaries, complex constitutive equations and transient analysis. The level of sophistication is overwhelming to the engineer who may only want to find the stress in a rectangular plate made of a linear elastic solid. A program family approach allows generation of family members that have exactly the degree of complexity required for a particular domain. An additional benefit of customized applications is that the generation of the family member can potentially take advantage of potential simplifications to produce more efficient code.

The first section below reviews the literature on program families and on documenting requirements for SC. In the section following this, a CA analysis template suitable for families of physical models is presented. The next section shows excerpts from an example CA for a family of material behaviour models, where each member model consists of equations that characterize the response of a material to applied loads.

2. BACKGROUND

The idea that software should be developed using a program family concept has a long history in software engineering. The idea of program families was first introduced in the early 1970s by Dijkstra [5] and later investigated by Parnas [9, 10]. More recently, Weiss [1, 23] has considered the concept of a program family in the context of what is termed Family oriented Abstraction, Specification and Translation (FAST) [22]. Other approaches to developing program families, also known as software product lines, can be found in [3] and [12].

Many of the ideas associated with program families are not new to researchers in SC. For instance, Carette [2] shows how to create a family of efficient, type-safe Gaussian elimination algorithms by fixing different design decisions and then using code generation. Code generation is also used in ATLAS [24] and Blitz++ [21] to produce efficient and portable SC code for linear algebra and array processing. Other ideas related to program families, in particular software reuse and capturing domain knowledge, figure prominently in SC research on Problem Solving Environments (PSEs). A PSE is "a computer system that provides all the computational facilities necessary to solve a target class of problems efficiently" [13]. Although the contributions listed above can improve quality and productivity of SC software, these examples have typically underemphasized the requirements stage and instead focused on algorithms and design.

A requirements template for SC have been presented in [17] and [18]. This template focuses on engineering mechanics problems, which provide an example of the class of SC software that is based on physical models. This class of SC software involves solving governing equations for such dependent variables as displacement, velocity, pressure, temperature, and concentration. Another class of SC software is multi-purpose tools. SC programs from this class of programs are not tied to a specific physical model, but rather they provide tools that can be used in many different con-

texts, for instance to solve different physical models. The members of the set of multi-purpose SC programs roughly correspond to the chapters in a typical introductory SC text. For example, typical domains include ordinary differential equation solvers, linear solvers, numerical integrators, mesh generators etc. A template for documenting the requirements for families of multi-purpose tools is presented in [15]. Unlike the template presented for a single physical model, the template for multi-purpose tools uses a CA and a family approach. The current paper combines these two previous approaches to create a CA template for a family of physical models.

3. COMMONALITY ANALYSIS TEMPLATE

The first step in documenting a family of physical models it to determine the family of interest. The second step consists of a Commonality Analysis (CA) [4, 22] on this identified family. The CA can be seen as a method for summarizing the requirements for all potential models that are considered to be within the scope of the project. The CA includes documentation of the potential system context, commonalities (including terminology, goals and theoretical models) and variabilities.

A potential template for use when documenting a CA for physical models is shown in Figure 1. The template includes a section listing potential system contexts, user characteristics and system constraints. The CA covers all the potential physical models in the family, so it is not possible to know exactly what information to place in these sections; however, there will often be information that can be recorded on typical uses of the family members. Although the information in this section cannot be presented as variabilities, because it does not represent requirements, the hints provided in the CA can later be refined during the process of application engineering.

---

1. Reference Material: a) Table of Contents b) Table of Units c) Table of Symbols d) Abbreviations and Acronyms e) Types

2. Introduction: a) Purpose of the Document b) Scope of the Family c) Organization of the Document

3. General System Description: a) Potential System Contexts b) Potential User Characteristics c) Potential System Constraints

4. Commonalities a) Background Overview b) Terminology Definition c) Goal Statements d) Assumptions e) Theoretical Models f) Derived Quantities

5. Variabilities

6. Dependence Graphs

7. Sample Family Members

8. References

Figure 1: Commonality Analysis Template

A "Terminology Definition" section is commonplace in requirements documentation. In the case of the CA its inclusion is motivated by a need to clarify the domain con-

cepts and to serve as a reference aid. The contents of this section consist of a list of mathematical concepts and their exact meaning, along with associated symbols and sign conventions. This section should provide enough information to allow understanding of the later sections "Goal Statement," "Assumptions," "Theoretical Models," and "Variabilities." The terminology section is necessary to make the physical model unambiguous because terminology often has subtly different meanings, even in very similar contexts.

The motivation of the goal statement section of the CA is to capture the goals in the requirements process. A goal, in this context, is a functional objective that each member of the physical model is expected to achieve. The goal statements do not include nonfunctional objectives because nonfunctional requirements are not commonalities between family members. Goals provide criteria for sufficient completeness of a requirements specification and for requirements pertinence. Goals will be refined in the Section "Theoretical Models." The goal statements are intended to be written at a level that is easy to understand, which usually means that goals are written using natural language. The goal statements should briefly summarize the commonalities shared by all models of the physical phenomenon.

The "Theoretical Models" section specifies the theory that all members of the physical model share. The model is presented as it would be presented in a mathematics or physics textbook. That is, the model is specified as the ideal mathematical case, without reference to the limitations that an actual computer implementation will have to overcome. This is done so that there is a relatively uncomplicated reference model that all stakeholders can agree on and understand.

The "Assumptions" section emphasizes the importance of assumptions to SC for making the theoretical model something that can be solved. In many cases, if no constraints are placed on the theoretical model, then it cannot be solved numerically for all possible inputs. The assumptions are used as a means to transform the goal statement into a formal theoretical model. The assumptions are also potentially used by some of the terminology definitions, when the associated concept depends on an assumption. The assumptions are shared by all family members. If an assumption is potentially not shared between all family members, then it should be documented as a variability.

For each of the variabilities in the CA there is a parameter of variation and a binding time. The parameter of variation specifies the type of the possible values for the variability. The binding time is the time in the software lifecycle when the variability is fixed. The binding time could be during specification of the requirements (specification time), or during building of the system (build time), or during execution of the system (run time). It is possible to have a mixture of binding times. For instance, a parameter of variation could have a binding time of "specification or build" to represent that the parameter could be set at specification time, or it could be postponed until the given family member is built. The presence of a DSL allows postponing the binding until build time. The DSL itself could have the facilities to specify that the binding will be postponed until run time.

The dependence graphs perform the role of showing the relationship between the terminology, goals, theories, assumptions and variabilities. The dependence relation represented by the graph is important so that change can be tracked through the CA document. A traceability matrix with the same purpose was introduced in [16, 17]. The dependence graph summarizes the same information as the related commonality field that is suggested for each variability in the Weiss approach [22].

4. EXAMPLE OF A FAMILY OF MATERIAL MODELS

The example CA presented here, which is partially based on [8, 19], is for a family of material models for predicting the deformation of a material particle in response to some applied loading. This family is of interest because the modelling of deformation is necessary to solve many engineering problems, such as for determining the deflection of a structure, or the stresses in an airplane wing, or the thickness of a manufactured sheet of plastic film. For problems like these, where the material body cannot be assumed to be rigid, the conservation equations of mechanics (conservation of mass and conservation of momentum) do not provide enough information to solve for changes in the body's configuration. To determine deformation, another equation has to be introduced, the so-called closure or constitutive equation, which relates the deformation history of the body and the current stress field. A wide range of constitutive equations, or material models, are used in engineering applications. For instance, materials may be modelled as elastic, viscous, viscoelastic, plastic or viscoplastic. Although the behaviour of these different types of materials can be very different, the mathematics used to describe them is similar. This allows us to find an abstraction that makes it possible to consider the above range of material behaviours as a family of material models, as described in this section.

Although the theory, terminology and equations presented are not new, the manner in which the information is presented in the CA is apparently unique. A reasonably complex theory is presented, but at the same time the document remains self-contained. To accomplish this goal and at the same time keep the size of the document small enough to be practical, no attempt is made to cover the breadth of continuum mechanics, but within the scope of the constitutive equations in the family, all necessary details are presented. An example of this approach is illustrated by the presentation of the definition of deformation that is given within the CA. Although there are many potential measures of deformation, which are usually documented within continuum mechanics textbooks, the only one necessary in the current context, and thus the only one presented in the CA, is the rate of natural strain tensor.

One way in which the example CA adds value, over what could be achieved by simply reading the various sources on which it is based, is that it employs a consistent notation and terminology. Moreover, potential ambiguities that would exist when combining different documents are removed. For instance, in continuum mechanics many different measures of stress and strain are used, but a specific equation is only valid for the specific measures for which it was derived. It is not possible to simply swap one stress or strain measure for another, especially in the case of large deformations. To remove this potential ambiguity, the stress and strain measures associated with the presented equations are clearly defined (see Section 4.1). Given that irrelevant details are left out of the documentation, one may be concerned that implicit assumptions about relevance were made during prepa-

ration of the document. To guard against this problem, an effort is made to clearly document all assumptions. Moreover, the documented assumptions provide the important role of assisting with delimiting the scope of the family.

4.1 Terminology Definition

This section consists of definitions of terminology that are necessary to understand the modelling of material behaviour. These definitions will be used later to explain the assumptions, goal statement, theoretical model and variabilities. Although in some cases the definitions will look like some of the variables that are introduced later, this section is describing concepts, not specific variables. For instance, it is the concepts of stress that is presented here, not a variable; the concept of stress will later be used to understand the input variable of initial stress and the output variable of stress history.

The terminology definitions for the family of material models have the following labels: D_Stress, D_StrainRate, D_YieldFunction, D_HardeningParameter, D_PlasticPotential, D_DescriptionOfMotion. For reasons of space, the full details of each definition are not provided here. In the actual CA each definition is presented using a tabular structure with the following rows:

Label: The label is a short identifying phrase, each with the prefix "D_." This label provides a mnemonic that helps with quickly remembering which definition is being presented.

Symbol: This field shows the symbol that is used to represent variables related to this concept.

Type: Each variable designated by a symbol has a type associated with it, which is listed in this field of the data definition template. The type information helps to clarify the meaning of the symbol and the variables.

Units: Where applicable the units associated with the symbol are given. These units are given in terms of the mass (M), length (L), time (t) and temperature (T). For convenience the units are also given in SI.

Related Items: A related item is a data definition or assumption that is used by the current definition. That is, if the used data definition or assumption should change, then the current data definition will also need to be modified.

Sources: This field lists references that can be consulted for additional information on the concept in question.

Description: The actual definition is given here. In some cases where the description is lengthy, some of the details are moved to a section following the table. When appropriate the description will reference the related definitions and assumptions.

History: Each data definition ends with a history of the definition, including the creation date and any subsequent modifications.

An example data definition for D_YieldFunction is provided below. In this definition, the type tensor2DT is used to represent two-dimensional tensors [7] and Q is known as the plastic potential function [6, pages 356–377].
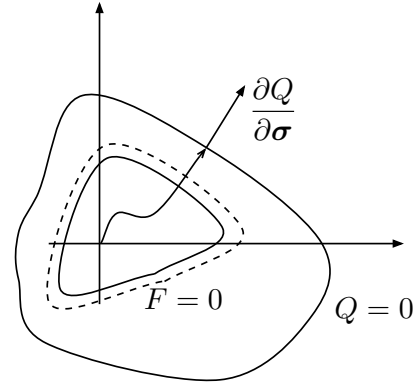

Figure 2: Yield function, hardening and the plastic potential in stress space

Label:	D_YieldFunction
Symbol:	$F = F(\boldsymbol{\sigma}, \kappa)$
Type:	$(\text{tensor2DT} \times \mathbb{R}) \to \mathbb{R}$
Units:	–
Related Items:	D_Stress, D_HardeningParameter
Sources:	[11]; [6, pages 327–356]; [7, pages 175–181]; [25, pages 74–78]
Description:	The yield function defines a surface $F = 0$ in the six dimensional stress space, which can be visualized by looking at the sketch in Figure 2. Within this surface the material behaves as an elastic solid. Outside this surface the material is assumed to have yielded and thus must obey a different constitutive equation. When the material has yielded, which occurs when the stress path reaches the yield surface, as shown in Figure 2, the yield surface may change shape. This change in shape is caused by the strain hardening (or softening) of the material. The new yield surface is shown in Figure 2 as a dashed line. This behaviour is mathematically represented in the yield function by its dependence on the instantaneous values of the hardening parameter κ, as described in D_HardeningParameter.
History:	Created – June 15, 2007

4.2 Goal Statement

The family of material models has one common goal, as shown in the table below. As for the data definition tables, the goal is assigned a unique label and the table shows fields for the description of the goal and its history. Goals also have a field for "Refinement," which shows which theoretical model refines the goal under consideration. Changes in the theoretical model, potentially caused by changes in the assumptions it is based on, may cause the theoretical model to no longer satisfy the goal statement.

Label:	G_StressDetermination
Description:	Given the initial stress and the deformation history of a material particle, determine the stress within the material particle.
Refinement:	T_ConstitEquation
History:	Created – June 8, 2007

4.3 Assumptions

The assumptions for the family of material models have the following labels: A_ContinuumHypothesis, A_CauchyStress, A_DeformationHistory, A_NoDistribMoments, A_SmallDefGradients, A_CartesianCoord, A_Isotropic, A_Isothermal, A_AdditivityPostulate, A_ElasticConstit, A_PerzynaConstit, and A_DescriptionOfMotion. Each of the assumptions is documented following a template similar to that adopted for the data definitions in Section 4.1. As for the data definitions, the following fields are used: label, related items, description, source and history. The new fields introduced and an example assumption are as follows:

Equation: Some of the assumptions include an equation, when this makes the description more precise. For each equation the types of each of the terms is listed.

Rationale: This field justifies the appropriateness of the assumption within the context of the current family. If changes in the assumptions are made in the future, it will be because the rationale is inadequate in some sense.

Label:	A_AdditivityPostulate
Related Items:	D_StrainRate
Equation:	$\dot{\epsilon} = \dot{\epsilon}^e + \dot{\epsilon}^{vp}$ with the following types and units $\dot{\epsilon}$: tensor2DT (1/t) (1/s) $\dot{\epsilon}^e$: tensor2DT (1/t) (1/s) $\dot{\epsilon}^{vp}$: tensor2DT (1/t) (1/s)
Description:	The total strain rate ($\dot{\epsilon}$) is assumed to decompose into elastic ($\dot{\epsilon}^e$) and viscoplastic ($\dot{\epsilon}^{vp}$) strain rates.
Rationale:	This is a standard assumption for elastoplastic and elastoviscoplastic materials. The appropriateness of this assumption is born out by the success of theories built upon it.
Source:	[6, page 339]; [7, page 181]
History:	Created – June 11, 2007

4.4 Theoretical Model

The template for the table describing the theoretical model uses the fields of label, related items, description and history, as introduced in Section 4.1. In addition the table introduces the following fields:

Input: The input field consists of a list of the input variables and their types. Where appropriate, the units of the variables are listed as well.

Output: This field lists the output variable and its type. The units of the output variable is listed as well.

Derivation: The derivation explains how the theoretical model is derived from the assumptions on which it is based.

The theoretical model for the constitutive equation is presented below. This model uses the type $\mathbb{R}^+$ for positive reals and the type poissonT for a real number between 0 and 0.5. In addition, the model uses symbols not defined elsewhere in this paper, including γ, which is the fluidity parameter, and ϕ, which is a function of F. Finally, the units used in this example include StressU, which is defined as $L^{-1}Mt^{-2}$ or in SI as Pascal (Pa), where Pa = N/m^2, where N is Newtons.

Label:	T_ConstitEquation
Related Items:	A_CauchyStress, A_DeformationHistory, A_PerzynaConstit, A_AdditivityPostulate, A_ElasticConstit, A_DescriptionOfMotion, V_MaterialProperties
Input:	σ_0 : tensor2DT (StressU) (Pa) t_{begin} : $\mathbb{R}$ (t) (s) t_{end} : $\mathbb{R}$ (t) (s) $\dot{\epsilon}(t)$: $\{t : \mathbb{R} \| t_{begin} \leq t \leq t_{end} : t\} \rightarrow$ tensor2DT (1/t) (1/s) mat_prop_val : string $\rightarrow \mathbb{R}$ E : $\mathbb{R}^+$ (StressU) (Pa) ν : poissonT (dimensionless)
Output:	$\sigma(t)$: $\{t : \mathbb{R} \| t_{begin} \leq t \leq t_{end} : t\} \rightarrow$ tensor2DT such that $$\dot{\sigma} = \mathbf{D}\left(\dot{\epsilon} - \gamma < \phi(F(\sigma, \kappa)) > \frac{\partial Q(\sigma)}{\partial \sigma}\right)$$ and $\sigma(t_{begin}) = \sigma_0$, the components of σ have the units of StressU (Pa)
Derivation:	The governing differential equation is found by first solving for $\dot{\epsilon}^e$ in A_AdditivityPostulate and then substituting the resulting expression into the elastic constitutive equation A_ElasticConstit. The final form is found by substituting in the expression for $\dot{\epsilon}^{vp}$ from A_PerzynaConstit.
Description:	The theoretical model is only completely defined once the associated variabilities (V_MaterialProperties) that define the material have been set. Given the material properties, which always include E and ν, and the input parameters describing the deformation $\dot{\epsilon}(t)$ of the material particle over the relevant history from t_{begin} to t_{end} solve the governing differential equation for the stress history ($\sigma(t)$) with the initial condition that the stress tensor $\sigma(t_{begin}) = \sigma_0$.
History:	Created – June 14, 2007

4.5 Variabilities

The labels for the variabilities in the family of material models are as follows: V_MatName, V_PlasticPotential, V_HardeningParameter, V_Phi, V_FluidityParameter, V_MaterialProperties, V_Description, V_StressState, and V_StrainState. The tables used to present the variabilities borrow the following fields from the terminology definition template (Section 4.1): Label, Related Items, Symbol, Type, Description and History. The variabilities also have an additional field for the binding time. An example variability follows.
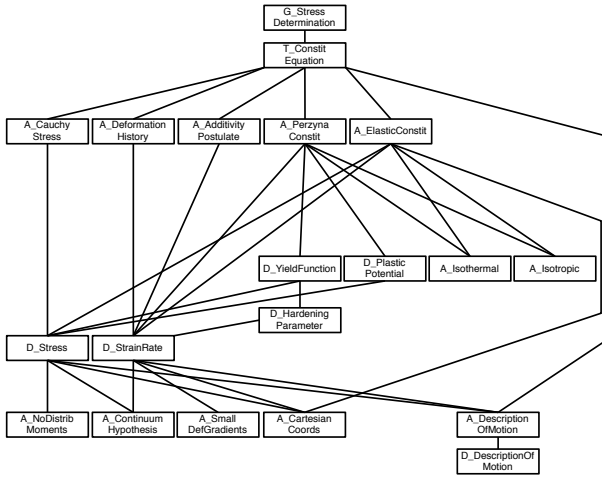
Figure 3: Dependence graph within the physical model (commonalities) of the family of material models

Figure 4: Dependence between the variabilities and the commonalities

Label:	V_PlasticPotential
Related Items:	T_ConstitEquation
Symbol:	$Q = Q(\boldsymbol{\sigma})$
Type:	tensor2DT $\rightarrow \mathbb{R}$
Description:	The plastic potential function (see Figure 2) is one of the characteristics that distinguished one member of the family of materials from another. In the case of associative materials, such as metals, the yield function and the plastic potential function will be the same. This is not the case for nonassociative flow materials, such as soils. The units of the plastic potential function depend on the particular function. In many cases the units will be StressU (Pa) or StressU^2 (Pa^2).
Binding Time:	Specification or Build
History:	Created – Aug 24, 2007

4.6 Dependence Graphs

Figure 3 shows the relationship between the common parts of the family of material models: the goal, the theoretical model, the data definitions and the assumptions. If an entry has a line between it and a higher entry, then this means that a change in the lower entry will likely require a change in the higher entry. This dependence graph summarizes the "Related Item" field in the tables given in the Commonalities Section of the CA. An example of a potential change would be to remove the assumption that the material is isothermal (A_Isothermal). As the graph shows, removal of this assumption would mean changing the assumed form for the Perzyna constitutive equation (A_PerzynaConstit) and the elastic constitutive equation (A_ElasticConstit). The change would involve making the material properties for the constitutive equation temperature dependent.

Figure 4 illustrates the dependence between the variabilities and the associated commonalities, and potentially between variabilities. As for the previous dependence graph
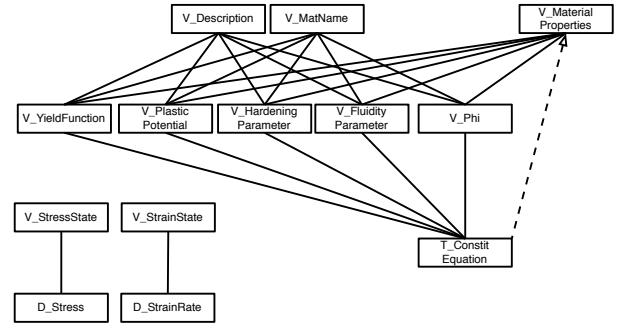
(Figure 3), a line between a higher and a lower item shows that the higher item depends on the lower item. In this case if the lower item changes, then the associated variability might need to be modified. Several of the variabilities (V_Description, V_MatName and V_MaterialProperties) depend on other variabilities. This is to reflect the fact that the other variabilities need to be set before it makes sense to set the variabilities that describe the material and its material properties. Figure 4 shows a dashed arrow between T_ConstitEquation and V_MaterialProperties to indicate that expressing the theoretical model depends on the choice of material properties. This dependency occurs because the input to the theoretical model includes the values of the material properties, and the specific material properties are only known once the material model has been determined.

4.7 Sample Family Members

To clarify how the CA can be used to specify specific family members, a section is provided listing potential family members. The family members are specified by setting the value of their variabilities. In all cases for this example the binding time for the variabilities is specification or build time. The example of a viscoplastic strain hardening material is provided below. In this description, q is the effective stress [6, page 364] and η is the viscosity.

Label:	E_StrainHardening
V_MatName	$name =$"Strain-Hardening Viscoelastic"
V_YieldFunct	$F = q\kappa^{\frac{n-1}{m}}$ (StressU) (Pa)
V_PlasticPot	$Q = q$ (StressU) (Pa)
V_HardParam	$\kappa = \epsilon_q^{vp}$, which is the second invariant of the deviatoric viscoplastic strain tensor. (L/L) (m/m)
V_Phi	$\phi = F^{\frac{m}{n}}$ (StressU$^{\frac{m}{n}}$) (Pa$^{\frac{m}{n}}$)
V_FluParam	$\gamma = nA^{\frac{1}{n}}$ (StressU^{-m}t^{-1}) (Pa^{-m}s^{-1})
V_MatProps	$mat\_prop\_names = \{$"A", "m", "n" $\}$, where the type of the material properties are as follows: $$A : \mathbb{R}^+, m : \mathbb{R}^+, n : \mathbb{R}^+ \qquad (1)$$ There is likely an upper limit on the values for m and n, but at this time the value of these limits is unclear. With respect to units, m and n do not have units and A has units of StressU^{-m}t^{-1} (Pa^{-m}s^{-1})
V_Descrip	$descript = $ "This constitutive equation combines a power-law viscoelastic material with a strain hardening (softening) material. A strain-hardening (softening) material is one where accumulated viscoplastic strain causes the material to be more difficult to deform (easier to deform). The strain hardening material will behave the same as the power-law viscoelastic material if $n = 1$ and it will behave like the linear viscoelastic material if $n = 1$, $m = 1$ and $A = \frac{1}{2\eta}$. In the case where $n < 1$ the material is strain hardening and when $n > 1$ the material is strain softening."
Source	[14]
History	Created − Sept 21, 2007

5. CONCLUSIONS

This document presented a CA template for families of models of physical phenomena. This template adds to the already existing set of templates tailored to documenting requirements for SC, which includes a template for a single model of a physical phenomenon [17, 18] and for a family of multi-purpose SC tools [15]. The new template's structure is similar to the structure of the previous SC templates. As for the other templates, the structure is essentially top down, with details added as one proceeds through it.

The first section of the proposed CA template serves the purpose of introducing the family of models, while the second section provides a general description of the physical problem and the different contexts where the physical model might be used. This second section includes subsections listing potential system contexts, user characteristics and system constraints. The third section presents the common terminology and requirements, including assumptions, the goal statement and the theoretical model. The fourth section describes the variabilities that distinguish the family members and the fifth section includes dependence graphs that show the relationship that exists between data definitions, goal statements, assumptions, theoretical models and the variabilities. The final section provides examples of potential family members, by explicitly assigning values to their variabilities.

The example CA presented in this paper summarizes a family of material models, where each member model consists of equations, often called constitutive equations, that characterize the material's response to applied loads. The CA for the material model family can be refined into part of a requirements document for a specific physical problem that requires a constitutive equation, or it can be used as the basis for a DSL. In the case of a DSL, a specification for a constitutive equation is written using the DSL and then the necessary code can be generated from this specification. To judge the effectiveness of the proposed CA, future work will be performed using it to verify and enhance an existing DSL that was designed to specify material models for use within a virtual material testing laboratory [8].

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] M. Ardis and D. M. Weiss. Defining families: The commonality analysis. In *Proceedings of the Nineteenth International Conference on Software Engineering*, pages 649–650. ACM, Inc., 1997.

[2] J. Carette. Gaussian elimination: A case study in efficient genericity with MetaOCaml. *Science of Computer Programming*, 62(1):3–24, 2006.

[3] P. Clements and L. M. Northrop. *Software product lines: practices and patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[4] D. A. Cuka and D. M. Weiss. Specifying executable commands: An example of FAST domain engineering. *Submitted to IEEE Transactions on Software Engineering*, pages 1 – 12, 1997.

[5] E. W. Dijkstra. *Structured Programming*, chapter Notes on Structured Programming. Academic Press, London, 1972.

[6] L. E. Malvern. *Introduction to the Mechanics of Continuous Medium*. Prentice Hall, 1969.

[7] G. E. Mase. *Schaum's Outline of Theory and Problems of Continuum Mechanics*. McGraw-Hill Publishing Company, 1970.

[8] J. McCutchan. A generative approach to a virtual material testing laboratory. Master's thesis, McMaster University, 2007.

[9] D. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, SE-2(1):1–9, 1976.

[10] D. L. Parnas. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, pages 128–138, March 1979.

[11] P. Perzyna. Fundamental problems in viscoplasticity. *Advances in Applied Mechanics*, pages 243–377, 1966.

[12] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer-Verlag, 2005.

[13] J. R. Rice and R. F. Boisvert. From scientific software libraries to problem-solving environments. *IEEE*

Computational Science & Engineering, 3(3):44–53, Fall 1996.

[14] W. S. Smith. *Simulating the Cast Film Process Using an Updated Lagrangian Finite Element Algorithm.* PhD thesis, McMaster University, Hamilton, ON, Canada, 2001.

[15] W. S. 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.

[16] W. S. Smith, L. Lai, and R. 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.

[17] W. S. Smith, L. Lai, and R. Khedri. 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.

[18] W. S. Smith, L. Lai, and R. Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13:83–107, 2007.

[19] W. S. Smith, J. McCutchan, and F. Cao. Program families in scientific computing. In J. Sprinkle, J. Gray, M. Rossi, and J.-P. Tolvanen, editors, 7th *OOPSLA Workshop on Domain Specific Modelling (DSM'07)*, pages 39–47, Montréal, Québec, 2007.

[20] A. van Deursen, P. Klint, and J. Visser. Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Notice*, 35(6):26–36, June 2000.

[21] T. L. Veldhuizen. Arrays in Blitz++. In *Proceedings of the 2nd International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'98), Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[22] D. Weiss and C. Lai. *Software Product Line Engineering*. Addison-Wesley, 1999.

[23] D. M. Weiss. Commonality analysis: A systematic process for defining families. *Lecture Notes in Computer Science*, 1429:214–222, 1998.

[24] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, 2001.

[25] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann, 6th edition, 2005.