

Characterization Pattern

The characterization pattern describes a way to annotate functional or behavioral models of a system by attaching additional properties to existing elements. This pattern has the desirable feature that characterizing an element does not require modification of the characterized element. This separation allows for more than one distinct characterization of a target element. It also keeps the identity of the target components distinct from the characterizations.

Pattern Overview

Pattern Status	Tool Version
In Work	SSCAE-17.0.2sp3-02.1 ODI Package or newer

Line Organization Owner	Submitter	Point of Contact
3101	IMCE Pattern Consolidation Working Group	Dan Dvorak

Related Patterns
<ul style="list-style-type: none">• Reconciliation/Abstraction Pattern: For constructing and reasoning across multiple levels of abstraction (logical/physical, conveyance of data across networks, etc.)• Analysis Explanation Pattern

[Go directly to SysML Examples...](#)

Table of Contents:

- [Pattern Overview](#)
- [Applicability](#)
 - [Content Concerns](#)
 - [Artifact Concerns](#)
 - [Generic Reasoning Questions](#)
 - [Assumptions](#)
- [Pattern Implementation](#)
 - [Generic/Ontology Implementation](#)
 - [SysML Implementation](#)
- [Advanced Questions](#)
- [References and Pattern Resources](#)
- [Further Examples](#)
- [Community Page](#)
- [Open Questions](#)

Applicability

To help users assess the applicability of this pattern to their work (i.e., to the problem they want to solve or their area of interest), we describe the way in which this pattern addresses a few kinds of common concerns. In particular, we address:

- **Content concerns:** the kind of content users can capture in this pattern
- **Artifact concerns:** the kinds of artifacts (documents and views) that can come from this pattern
- **Reasoning concerns:** the kind of reasoning (analysis) that this pattern is meant to support
- **Assumptions:** what we expect to be true about the user's situation that is relevant to whether they can or should use the pattern.

Content Concerns

The Characterization Pattern is intended to help engineers describe the properties of their systems in a way that allows for separation of concerns according to different viewpoints.

Summary

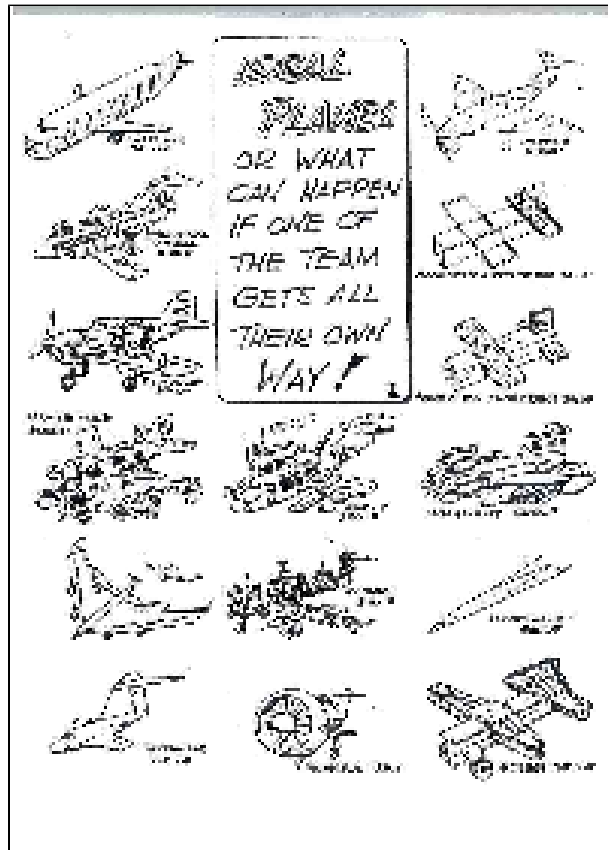
What can the Characterization Pattern do for you?

- Give each stakeholder a place to describe elements in the system according to the properties they care about.
- Allow many people to add to the description of an element without changing the element itself
- Describe elements while keeping their base definitions minimal and reusable
- Create libraries of useful descriptions for various kinds of elements
- Provides a place for users to input data for analyses, such as mass at certain times in the project lifecycle for MEL analyses

To get started thinking about characterization, take a look at the cartoon at right. While amusing and slightly insulting to most disciplines, it also shows how the features of a system that are of interest to one domain may be totally unnecessary in another. The cartoon is definitely absurd (all the different planes at right can't actually be perspectives of the same design solution), but to take advantage of Characterization, we need to be thinking in this multifaceted way.

Characterization is simply the ability to describe the characteristics (properties, constraints, behaviors, etc.) of an element in a space that is external to the element, but tightly and formally bound to it. The element itself only contains the set of properties that are necessary for it to exist in a recognizable way; intrinsic properties that contribute to its identity (by their existence, not their value) and distinguish it from other classes of element. The link between the element and all of its characterizations means that we can analyze all of the characterizations for inconsistencies, and address the kind of tensions depicted in the cartoon before they become an implementation problem. Characterizing a system from different viewpoints may result in attributes of the same name (as an example, "mass" for a car could refer to the situation where the car has a full tank or an empty tank) that actually refer to different attributes, and our engineering work then becomes resolving the conflicts and obtaining a solution where all properties across the different characterizations are consistent.

In the Characterization Pattern we will describe the tools available to build characterizations of the system. We will address how characterizations will be analyzed, reconciled, and checked for consistency in a later pattern.



source

If you find yourself thinking that characterization is unnecessary, that you would have no issue putting all the properties of your system directly into the system elements, we offer a few points to consider:

1. Leaving only the intrinsic properties of an element in the definition of the element allows for re-use (across projects, organizations, and within projects).
2. Using a library of characterizations makes analysis and reasoning re-usable.
3. You can take advantage of the separation of concerns. See SysML examples.

This pattern gives engineers the tools necessary to manage that separation of concerns.

Artifact Concerns

Characterizations contribute to many artifacts - the following are a few examples:

- MEL (in this case, a Mass Equipment List) - mass characterizations can be traversed and the MEL computed from the values in the leaf elements.
- PEL (Powered Equipment List)
- Link budgets and margin
- Requirements metrics
- other budgets (outside of link budgets): science error budgets, GNC pointing budgets, etc...
- behavioral scenarios and traces of power, mode, system state
- FMEA (failure modes and effects analysis)
- executable models

Generic Reasoning Questions

Like the [Structural Decomposition Pattern](#), the Characterization Pattern supports generic reasoning. Characterizations are tied to the elements they characterize and characterization can stand in for the element in parametric analysis, or other reasoning, and that link between a characterization and an element is formal.

Assumptions

The only assumption is that the user requires the separation of concerns provided by separating the characterization from the element it characterizes.

Pattern Implementation

Here we describe the elements that make up the Characterization Pattern, their relationships, responsibilities, and collaborations. The solution does not describe a particular concrete design or implementation. Instead, the pattern provides an abstract description of the problem of describing attributes of elements and systems that are related and of concern, but not intrinsic properties of the system, and how a general arrangement of elements solves it. The solution is presented first in modeling language independent terms (in the [Ontology Description Section](#)) and then as a SysML embedding (in the [SysML Implementation Section](#)).

Generic/Ontology Implementation

Some context for the word "characterization" – for those who find such things interesting. [Click to expand.](#)

We wish to provide the ability to describe the characteristics of components in our systems from a variety of viewpoints and according to different concerns. Let's take the characteristics of a person as an example. The following could all be considered characteristics of a person: red hair, seven feet tall, honest, dignified, friendly, wearing an orange t-shirt. If you are trying to meet them at the airport, the physical characteristics will be much more helpful in achieving that goal than knowing if they are dignified or friendly. On the other hand, if you are thinking of having them as a roommate, you are probably more concerned with their honesty and other "personal" traits. All the attributes may be part of what distinguishes or defines that person, but your viewpoint dictates which attributes are of interest.

The same is true of elements with which we work in systems engineering. Descriptions of systems according to different concerns are all (and must be) "true" and consistent, but working simultaneously with all possible qualities and attributes is difficult both practically and intellectually. This pattern provides tools for creating "sketches" of systems according to any number of different viewpoints or groupings.

Summary

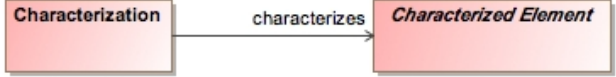
characterization is a mechanism allowing engineers to describe components in ways that are not intrinsic to the definition of those components but are necessary to work within their problem space.

To effectively use characterization:

- a. Create and re-use the definition of a thing by identifying it in terms of its intrinsic properties;
- b. Extend the description of thing as needed with characterizations; and
- c. Define characterization libraries and re-use them.

Ontological Elements

The Characterization Pattern identifies three ontological elements necessary to perform characterization: an element to characterize (*CharacterizedElement*), the Characterization element itself, and the *characterize* relationship, which formally links the Characterization to the *CharacterizedElement*. Note that *CharacterizedElement* is abstract and is in fact an ancestor of every other concept and relationship.

Image	Description
 <pre>graph LR; C[Characterization] -- characterizes --> CE[Characterized Element]</pre>	The <i>Characterization</i> is a distinct object. It is a separate element from the <i>CharacterizedElement</i> , and the properties it describes exist in the namespace of the <i>Characterization</i> .

When a *Characterization* owns properties and is bound to a *CharacterizedElement* with the *characterizes* relationship, the formal meaning is that those properties describe the *CharacterizedElement*. The *Characterization* can stand in for the *CharacterizedElement* in analysis.

The characterization augments the *CharacterizedElement* in the sense that the attributes of the *Characterization* are considered as attributes of the *CharacterizedElement* (e.g., a Component) but is itself not modified.

Features of Characterization:

- When you apply a Characterization to a *CharacterizedElement*, it is also assumed to apply to all specializations of the *CharacterizedElement*. You may use redefinition to override some or all of the general characterization.
- A *Characterization* can characterize many *CharacterizedElements*, assuming that they do in fact share an identical set of attributes and values as defined in the Characterization.

The "Characterization" concept is extremely generic - any set of properties can be put into a Characterization, as long as they truly describe the *CharacterizedElement*. This is both helpful in its flexibility, and hindering in its lack of formality. Thus, we expect that engineers will specialize the Characterization concept to create sets of re-usable Characterizations that come with specific property sets. For example, a mass-margin-tracking characterization might be its own specialization, and may define properties related to current best estimates, contingency, etcetera at the ontological level.

Authorities define what characterizations should exist for a discipline and what properties go in those characterizations. That work is in the scope of the Authorization Pattern. Maintaining consistent sets of cross-cutting information is addressed in the [Reconciliation/Abstraction Pattern](#).

When a characterization characterizes multiple elements, it means that the characterization applies fully to each element (i.e., not the union of those elements).

SysML Implementation

The concepts we described in the last section are mapped to concrete implementation in SysML so that they can be used to encode system information in a model. In this section we describe first the embedding of the ontology into SysML (so that the user can understand how the concepts are made concrete) and then provide examples.

As a reminder, the Characterization Pattern is for capturing characteristics of a characterized element. In order to capture intrinsic properties (such as a unique identifier), the convention is to place that property inside the element itself. It's not wrong to put other properties inside the element, but then you are not utilizing the Characterization Pattern and its benefits.

For the purpose of analysis, views may be constructed (by hand or via automation) where the properties in the characterization appear as properties in the element. (See the [Advanced Questions](#) section.)

Embedding

The following table describes how the elements in the ontology appear in SysML. Note: the "analysis:" prefix only indicates that these concepts are part of an "analysis" package in the ontology. It does not mean that you must be doing analysis in order to use characterization.

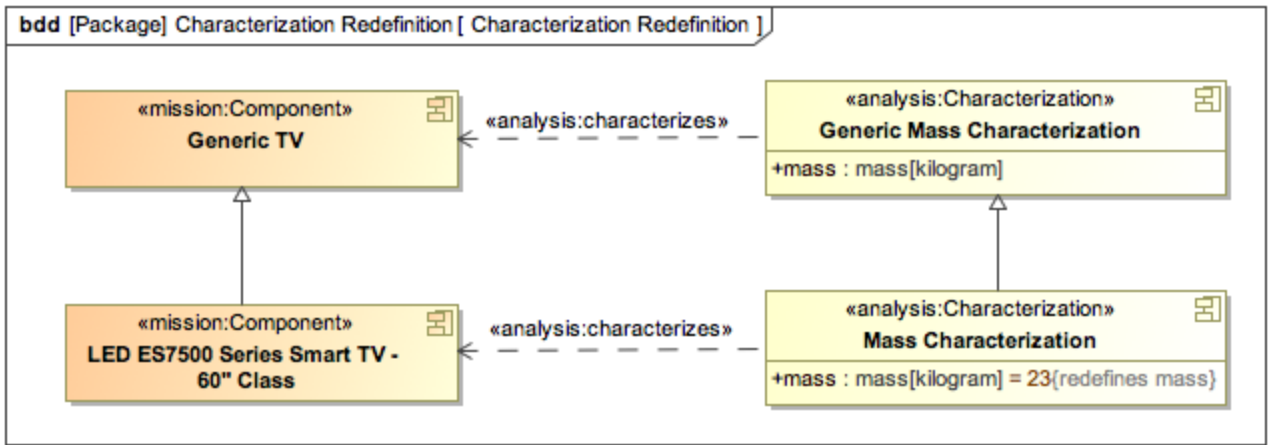
Ontology	Classification	SysML	Metaclass	Stereotype
<i>analysis:Characterization</i>	Concept	Component	UML::Component	«analysis:Characterization»
<i>analysis:CharacterizedElement</i>	Concept	<i>analysis:CharacterizedElement</i> does not map precisely to one SysML metaclass. Anything in the model can be characterized, so <i>CharacterizedElement</i> can be anything.	UML::NamedElement	N/A
<i>analysis:characterizes</i>	Relationship	Dependency	UML::Dependency	«analysis:characterizes»

Concept Mapping: *analysis:Characterization* is embedded as a UML Component. Every concept and relationship in the ontology can be characterized (which we map to all UML::NamedElements in the model), so the abstract *analysis:CharacterizedElement* is in fact an ancestor of every other concept and relationship. There is no stereotype that needs to be applied to *analysis:CharacterizedElements*, as every other element is a *CharacterizedElement* by nature.

Relationship Mapping: the *analysis:characterizes* relationships is embedded as a UML::Dependency with the «analysis:characterizes» stereotype applied. Its source can only be a component stereotyped with «analysis:Characterization» and its target can be any other element.

Attributes of the Characterizations are embedded as SysML «ValueProperty»s. Other properties may be used, but we don't define any conventions for what they mean.

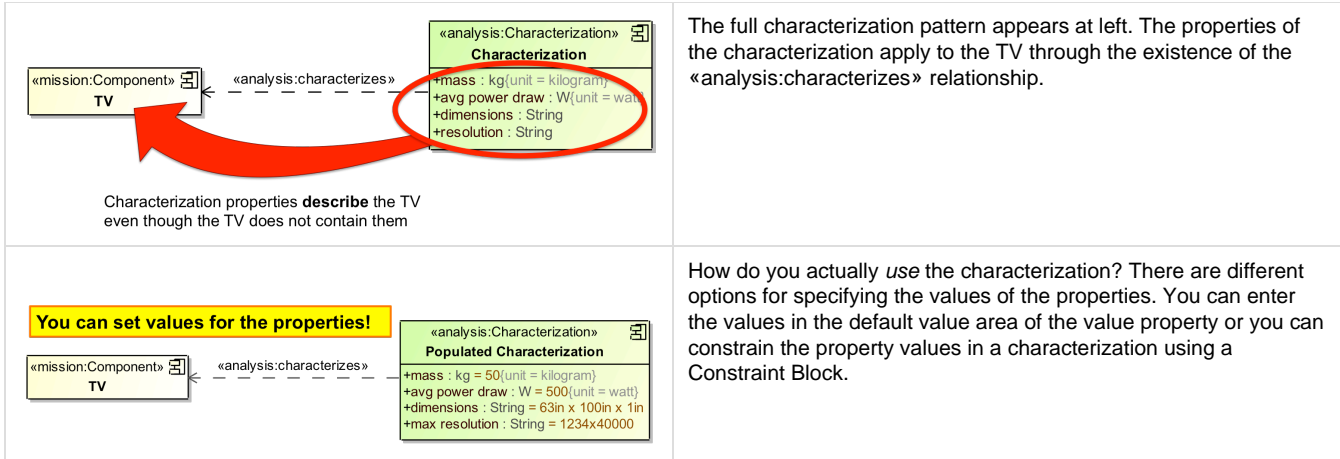
Characterizations may specialize each other and redefine properties (because they are embedded as UML:Components).



SysML Examples

Let's actually characterize something in a model - a TV, for example. We'll step through this simple example, and then follow up with some more complicated examples after.

<p>What do we do with Characterization?</p>	<p>Characterization is about describing elements of the system by adding properties that you care about. Characterization is different than just adding properties to the elements directly because you can define and edit the properties you care about without changing the definition of the element the properties describe.</p> <p>You can think of a characterization as your space to say what's important about an element without affecting the element. It's your take on the element, and you can use that for reasoning, analysis, or whatever you want.</p>
<p>At the nuts and bolts level, the complete characterization pattern looks like the image at the right. There's the characterization element, marked «analysis:Characterization», and it is linked to the element it characterizes (the TV) by a UML::Dependency with the «analysis:characterizes» stereotype applied. Although we have characterized a «mission:Component» here, remember that a Characterization can characterize any Named Element in the model.</p>	<p>What does Characterization actually look like in the model?</p>
<p>So how do we actually make this useful? The most common thing to do with Characterizations is to add properties. Here, we have added properties for the mass, average power draw, dimensions, and resolution of the TV. We have also added QuantityKind to the properties to better describe the attributes of interest.</p>	<p>How might you characterize a TV?</p>



SysML Example: Characterized Elements

As we stated previously, a Characterization can characterize any kind of element, not just a «mission:Component». You can even characterize a «analysis:characterizes» relationship. That can be useful in describing when certain characterizations apply. Open this section for an example!

[? Expand to continue reading...](#)

SysML Example: A different kind of mass rollup...

What if we want to calculate the mass of the launching Apollo 11 mission... at different times as it heads toward the moon?

[? Expand to continue reading...](#)

Fun with Reasoning Questions

Here we will revisit the reasoning questions from the applicability section and show how they can be addressed in our example.

[? Expand to continue reading...](#)

Rules/Axioms/Invariants

If you specialize a CharacterizedElement, you inherit its characterizations.
Chains of characterizations are allowed, but they must be acyclic.
A characterization may not characterize itself.

There are no restrictions on the use of characterizations.

Model Implementation Concerns

Validation/Well-Formedness Reasoning

Rules:

- all specializing characterizations have to redefine the inherited attributes

Supporting Scripts/Tooling

- Pattern Factory (<http://sscae-build.jpl.nasa.gov/jenkins/job/SSCAE-MD17.0.2SP3-Packagel.4/>)
- TemplateElement creation/sync wizard ([contact M Jackson for more details](#))
- Prototype characterization chooser included in MDK

Tooling Tricks

Trick	How to do it	Why?

Advanced Questions

How should characterizations appear in parametrics?

References and Pattern Resources

Currently no Working Group approved references. See the [Community Page: Characterization Pattern](#) references area for unofficial references.

Modeling Guide (somewhat outdated): [SysML Modeling Guide](#) 

Further Examples

Currently no Working Group approved examples. See the [Community Page: Characterization Pattern](#) examples area for further examples.

Community Page

The [Community Page: Characterization Pattern](#) has been set up to collect Frequently Asked Questions, Discipline Specific (and extended) examples and reasoning, and References. Everyone should have write access and are free to discuss and contribute.

Open Questions

Question	Discussion
Proposed rule: "There shall be only one authority which characterizes the same component for a characteristic"	<p>This rule was found in Robert's reply to Maddalena's email of 4/29/14. We discussed it on 5/6/14 and determined...</p> <ol style="list-style-type: none"> 1. This is outside the scope of this pattern. There are contexts where one authority makes sense, and contexts where it doesn't. Elaboration: the rules for this pattern are about MEANING - conflicting authorities asserting stuff about an element doesn't make the meaning of the element unclear, we know what the element is we just disagree about its characteristics. 2. Team's response was "fair enough, but then where does this go?" 3. Response: mentioning it in the pattern is OK. But it's up to model architect to add whatever additional methodology or rules are necessary for disambiguation. Some mention of this to go in model implementation concerns? Why? <ol style="list-style-type: none"> a. Also, by the way, the same authority CAN characterize the same thing twice in the same way. That might be an observation you make. b. How do you know which authority asserted a characterization? <ol style="list-style-type: none"> i. OWL model of a mission, every statement including "characterization characterizes characterizedelement" belongs to a context. Context is innermost containing authority. <p>So this didn't really get to an answer for where it goes.</p>
At some point we need to show how we use constraints.	This is out of scope for this pattern; it will be addressed in Analysis Pattern.

<p>how do you say an allocation for mass is related / also constrains a realized mass?</p>	<p>when we talk about "mass property" we want to impose a constraint (like an allocation or a budget), and then on the realization/assembly side, we want to characterize the mass that is there/designed/wahtever. we want to have a mechanism where we say that those two things talk about The Mass (singular).</p> <p>this is not part of THIS pattern, but it builds off of it.</p> <p>approaches: characterize functional performer and characterize realization, link characterizations/properties with analysis. OR have a characterization where there's ONE mass property which characterizes both sides, and somehow the constraints from both sides get attached to the one.</p>
<p>How should we identify a particular kind of characterization, such as a mass characterization?</p>	<p>Option 1: The characterization is a specialization of a characterization with the name mass.</p> <p>Option 2: Extend the ontology to introduce a concept of mass characterization and associated stereotype and therefore establish constraints on what a mass characterization can characterize.</p>
<p>How do you model an array as a value type?</p>	

Copyright

Copyright 2014 California Institute of Technology. Government sponsorship acknowledged.

