Behavior Pattern - v1.1

Questions?

General questions about the behavior pattern should be asked in the FAQ section of the community page.

You can also use the comments sections at the end of every page to ask more targeted questions or have discussion about things specific to the page.

Synopsis

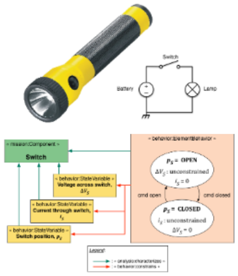
This pattern provides an approach to capture state-based behavior of elements (named `BehavingElements`) and the interactions amongst them. Elements can be components (e.g., sensors, actuators) or environments, and are characterized by `StateVariables` that vary with time. The behavior of a `BehavingElement` is represented through constraints on these `StateVariables`. Interactions among `BehavingElements` are represented in a similar fashion.

Currently, this pattern does not yet encompass all forms of behavior representation, nor causal aspects of behavior constraints. More discussion on extending the pattern is in the open questions section.

Organization of the behavior pattern pages:

This behavior pattern page has three child pages:

- A Conceptual Behavior Ontology v1.1 page that describes in details the *Conceptual* ontology. This page has itself a child page Behavior Pattern: Conceptual Examples v1.1 that describes two examples of application of the behavior pattern in a conceptual fashion;
- A SysML-Embeddable Ontolgy & Implementation v1.1 page that describes in details the *SysML-Embeddable* ontology, its relation to the *Conceptual* ontology and a step-by-step SysML example illustration. This page has itself a child page Behavior Pattern: SysML Example v1.1 that provides a compact view of the SysML example;
- The Community Page of the behavior pattern; this lists meeting notes, frequently asked questions and responses, user-provided discipline-specific examples, and further references (including papers and other work this pattern is based on).
- Previous versions of this pattern are listed here.


Switch
Battery
Lamp
« mission:Component »
Switch
« behavior:ElementBehavior »
p_S = OPEN
ΔV_S : unconstrained
$i_S = 0$
cmd open
cmd closed
p_S = CLOSED
i_S : unconstrained
$\Delta V_S = 0$
« behavior:StateVariable »
Voltage across switch, ΔV_S
« behavior:StateVariable »
Current through switch, i_S
« behavior:StateVariable »
Switch position, p_S
Legend:
: « analysis:characterizes »
: « behavior:constrains »

Pattern Overview

Pattern Status	Tool version
Released	Available with MD17.0.5-02

Line Organization Owner	Submitter	Point of Contact
Primary: 3101 - Engineering Development Office Secondary: 313D - System Architectures & Behaviors	IMCE Behavior Modeling Pattern Working Group	Daniel Dvorak <daniel.l.dvorak@jpl.nasa.gov>

Related Pattern	Relation
Characterization Pattern	For describing several relationships between concepts introduced in this pattern; specifically the characterization of `BehavingElement` by `StateVariables`, `Parameters` and `ElementBehaviors`.

Table of Contents:

Pattern Design Concerns

This is the list of design concerns that were identified at the outset of development of this pattern.

These concerns are presented below in no particular order:

Item #	Concern	Rationale for concern inclusion	Current pattern support	Rationale for limited support
1	The pattern should... Be consistent with other IMCE defined patterns.	To support integrated models, patterns must be consistent.	Fully supported - characterization pattern used in a consistent manner.	N/A
2	The pattern should... Allow for representing types of behavior as follows: • state-based • flow-based (energy, data, matter) • event-based • message-based	These are standard for representing behaviors in complex engineered systems.	• State-based supported explicitly; • Flow-based not explicitly used, but potentially supported using state-based equivalent representation • Minor event-based coverage with state machines, but standards for event specification not included in this pattern • No message-based support	The concept of states of a system was of interest to a majority of stakeholders of this pattern, takes advantage of previous works and simulation tools, and is of direct use for other domains such as control. This is why state-based behavior was tackled first. Other behavioral representation will be investigated in a subsequent iteration of the pattern.

3	The pattern should... Use existing SysML capabilities when applicable.	SysML is the language IMCE supports	State-machines used for behavior modeling Activity diagrams envisioned as part of an upcoming scenario pattern (not yet released) Sequence diagrams not supported	N/A As other behavioral representations are included in the patter, they will use the appropriate constructs in SysML.
4	The pattern should... Allow for modeling of individual behaviors of engineered components and environments, as well as behaviors that result from interactions among them.	Allow for decomposition of complex systems into simpler parts	Fully supported	N/A
5	The pattern should... Allow for the separation of the system structure (decomposition or interconnection) from system behavior, but allow for them to be reconciled if desired (especially at later stages in the design cycle).	Simplifies representing the behavior if structure is not yet defined or behavior need not reflect structure	Fully supported	N/A
6	The pattern should... Allow for definition of behavior at different levels of abstraction, and provide hooks for relating the different definitions.	Engineers require different levels of abstraction for performing analyses	Not supported in current version	Issue larger than behavior pattern. Dealing with level of abstractions should be a cross-cutting effort on patterns and elements. See Reconcilation/abstraction pattern.
7	The pattern should... Support the ability for model element re-use.	Efficient modeling	ISO-80000 QUDV constraint re-use through potential constraint block library	N/A
8	The pattern should... Enable behavior-related trade studies.	Engineers perform trade studies by evaluating behaviors against performance metrics	Supported, details deferred to later pattern	We believe that the current pattern supports the performance of trade studies. Specific examples of trade studies to demonstrate that capability were not developed in this iteration, but will be a focus of a subsequent iteration. This is also dependent on the development of a selection capability in the characterization pattern.

Applicability

To help assess the applicability of this pattern to your work (i.e., to the problem you want to solve or your 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 you can or should use the pattern

Content Concerns

This pattern provides a mechanism for modeling the behavior of elements or the behavioral interaction between elements. This includes:

Behavior Description:

- Elements of interest that exhibit internal behavior and/or interact with each other;
- The state variables and parameters (non-time varying) in the scope of the element behaviors or interactions

- Describing internal behavior of these elements
- Describing interaction among these elements:
  - The interaction interfaces that are presented by the interacting elements;
  - The interactions that join elements;
  - The interaction behavior of the elements represented by constraints on state variables presented by the interaction interfaces joined by the relevant interaction

Behavior Execution:

- Scenarios that are relevant for the elements and the described behaviors. The complete specification of scenarios will be described in another pattern.
- Trajectories of state variables resulting from scenarios and behaviors within scope.

Artifact Concerns

This pattern contributes to any artifacts where behavior is relevant.

Generic Reasoning Questions

This pattern supports generic reasoning as every element and relationship are well defined and validation rules (described here) can be applied to ensure complete and well-formed implementation of the pattern.

Various types of reasoning questions will be added here as they are developed through experimentation with the pattern. Behavioral analysis integration is part of the future planned augmentation work.

Assumptions

This pattern assumes that the user is representing their behavior in a state-based form.

No other assumptions are made.

For the Future

The current release of the behavior pattern is the first version. A second iteration is planned through the end of FY14 and into FY15 based on the reviews and commented gathered on this released version.

Some planned augmentation work has already been captured below.

Planned Augmentation Work

Priority scale:

- A = high
- B = medium
- C = low

Item #	Description	Other Notes	Augmentation Priority

1	determination of a recommended way of capturing constraints in SysML, in collaboration wtith Nicolas Rouquette		A
2	integration of some common behavior analyses		A
3	light tooling for the creation of value type (for a flexible approach) in collaboration with Nicolas Rouquette	possibly connected with ISO80000 library work	B
4	tooling to export constraints to solving / simulation tools (e.g. Maple or Mathematica)		B
5	export models to OWL language, in collaboration with Nicolas Rouquette and Steve Jenkins	in work	B
6	remove non-behavior specific concepts from the behavior ontology and re-factor into the other IMCE foundation ontologies (or new foundation ontologies) in collaboration with IMCE	to happen soon	C
7	define how inputs (in control theory sense) and commands are used in the behavior pattern		A
8	investigate observation/measurement integration in the behavior pattern		B
9	investigate what other IMCE elements could be considered for specializing BehavingElement		A
10	add explicit support for referencing time directly in a constraint		A
12	investigate how to handle variables such as "altitude" (allocation to what BehavingElement?)		A

Open Questions

Notes

Question	Status	Discussion

How to handle universal constants?	OPEN	Several possibilities are considered: • capture universal constants as `Parameters` and recapture them for each `BehavingElement` that requires it. However, this solution leads to the replication of the same information and the possible introduction of value inconsistency; • capture universal constants once as `Parameters` in a `PropertyGroup` that does not characterizes any `BehavingElement`. This solution is inconsistent wit the rule that states that `ElementBehavior uses Parameters` of the `BehavingElement` it characterizes; • create a new abstract class for "static property", that is specialized into `Parameter` and universal constant.
How can we set different values of a `Parameter` for different `ElementBehavior` characterizations?	OPEN	A possible solution is to create an additional property in the `ElementBehaviorConstraint` / `InteractionBehaviorConstraint` that is bound to the appropriate `ParameterConstraintParticipant` and set the value of the `Parameter` to that additional property.
Is there a recommended hierarchy structure for the SysML model?	OPEN	None is recommended at the moment, but some hierarchy could be recommended for some concepts (i.e., `PropertyGroups` should be owned by the `BehavingElement` they characterize).
How should the scope or limitations of the behavior model be specified? (e.g. operational scope of `ElementBehaviors` or `InteractionBehaviors`)	OPEN	The behavior models described in this ontology are valid within a scope. For example, Ohm's law not longer applies if the component loses its ohmic properties or if a massive current is applied to it... No specification of such scope is currently handled by the ontology.
What is the relation between `InteractionTerminal` and `mission:Interface`? Same question for `Interaction and mission:Junction`?	OPEN	None is specified for now. It might be of interest to provide a mechanism to relate them if needed. Idea: `Interaction` could **generalize** `mission:Junction` just like `BehavingElement` generalizes `mission:Component`
How do we indicate whether a constraint applies to all states of a state machine?	OPEN	For now, an `analysis:characterizes` relationship can be pointed at state(s) or region(s), or no relationship exists. Some conventions have been discussed in the SysML-embeddable ontology page in the State Machines section. Some more agreement/discussion might be necessary.
Should we allow several `TimeDomain` definition for a same `StateVariable`? How to define the same `TimeDomain` for several `StateVariables`?	OPEN	First question: not allowed for now Second question: can be done in SysML by typing the `TimeDomains` by the same `TimeDomainValueType`. No support on the ontology side.
Do we want to define compound `StateVariables`?	OPEN	No explicit relationship between `StateVariables` at different levels in the current ontology.

Can `Scenario` specifications also include `Parameters`?	OPEN	Current ontology does not support this.
How do we represent field effects on multiple elements using `InteractionBehaviors`? (such as gravitational effect of a singular large body on many smaller bodies; or electromagnetic force on independent charges, etc)	OPEN	haven't worked this out yet
Do we want to develop additional patterns (or extensions to this pattern?) that efficiently cover the other major forms of behavior representation: event-based, message-based, and flow-based?	OPEN	The answer to this question will be somewhat determined by the response of the modeling community to the state-based pattern, including indication of modeling inefficiencies that could be alleviated through other behavior representations
How to treat the specification of partial `Trajectories` (those that do not span the entire `TimeDomain`)?	OPEN	none yet
Interpretation of under-, over-, or fully-constrained systems with respect to "cardinality" of the `FamilyofTrajectories` set?	OPEN	TBD
Additional validation/well-formedness rules? E.g.: • check for at least one `ElementBehavior` or `InteractionBehavior` related to a `BehavingElement`? • Should `ElementBehavior` and `InteractionBehavior` constrain/use at least one `Property`?	OPEN	TBD
How to handle constraint parameters bound to several state variables/ behavior parameters?	OPEN	This is not explicitly forbidden by the pattern, but its support is not acknowledged either. This case would occur for example for multiple competing `ElementBehavior` characterizations (you must choose one for a specific simulation) linked to the same interaction constraint. A selection mechanism should be designed first (characterization pattern open question).

Behavior Page Navigation - continue reading:

(0) Community Page

(1) Main Behavior Pattern Page

(2) Conceptual Behavior Ontology v1.1 (3) Behavior Pattern: Conceptual Examples v1.1

(4) SysML-Embeddable Ontology & Implementation v1.1 (5) Behavior Pattern: SysML Example v1.1