[MzBe 92] E. Meyer zu Bexten: *Eine Simulationsumgebung für signalverarbeitende Systeme*, Dissertation, Universität Dortmund, Fachbereich Informatik, Shaker Verlag, Aachen, 1992

[MzBH 93] E. Meyer zu Bexten, D. Heinen, C. Moraga: *AUCA: A Tool for Presentation and Analysis of CAD-Simulation Results*, Proceedings of the HCI International 93, Orlando FLA, USA, Elsevier Science Publishers, pp. 267-272, 1993

[MzBM 92] E. Meyer zu Bexten, C. Moraga, J. Büddefeld: *A High-Level Interactive Design Environment for Complex Systems*, in: Computer Aided System Theory - EUROCAST '91, Lecture Notes in Computer Science, Vol. 585, Springer-Verlag, Berlin, pp. 442 - 459, April 1992

[MzBM 93] E. Meyer zu Bexten, C. Moraga: *Simulation für die Systemebene (Elektronik-Cad: Werkzeuge zur Unterstützung der frühen Phasen des Schaltungsentwurfs)*, Elektronik Journal, Europa-Fachpresse-Verlag, München, 28. Jahrgang, Ausgabe 12, S. 16-23, September 1993

[Mokw 95] W. Mokwa: *Mikrosystemtechnik - Entwicklung und Marktumsetzung*, 9. Fachtagung Mikroelektronik, Baden-Baden, März 1995

[PoVM 94] J. Poswig, G. Vrankar, C. Moraga: *VisaVis: A Higher-order Functional Visual Programming Language*, Journal of Visual Languages and Computing, 5, pp. 83-111, 1994

[Ramm 89] F. Rammig: *Systematischer Entwurf digitaler Systeme*, Teubner Verlag, Stuttgart, 1989

[Shne 92] B. Shneiderman: *Designing the User Interface: Strategies for efficient Human-Computer Interaction*, Addison-Wesley Publishing Company, New York, 1992

CAST Extensions to DASME to Support Generalized Information Theory

Cliff Joslyn[1] and Scott Henderson[2]

[1] Code 522.3, NASA Goddard Space Flight Center
Greenbelt, MD 20771, USA
joslyn@kong.gsfc.nasa.gov
http://groucho.gsfc.nasa.gov/joslyn

[2] Senior Scientist, Cambridge Research Associates
1430 Spring Hill Rd. Suite 200, McLean, VA 22102, USA
scott@cambridge.com

Abstract. The Data Analysis and System Modeling Environment (DASME) is a computer-assisted modeling environment, currently under development at NASA's Goddard Space Flight Center, designed to support ground-based mission operations with a mixed discrete/continuous modeling capability. This paper describes planned CAST-based extensions to DASME to support a broader range of systems theoretical computing models, and in particular models utilizing concepts from Generalized Information Theory (GIT) such as fuzzy systems, possibilistic measurement, and possibilistic processes. Support for model-based diagnostics and trend analysis of spacecraft systems is targetted.

1 Introduction

This paper describes a software systems development project ongoing in the Software and Automation Systems Branch (Code 522) of NASA's Goddard Space Flight Center in Greenbelt, Maryland. Code 522's role—within the overall Goddard mission of Earth and space science research with earth-orbiting satellites—is to conduct research and development to support mission operations and data systems. Code 522 provides systems engineering, development, and operation tools, and prototypes demonstrating new technologies and advanced systems architectures. Their areas of technology expertise span from systems modeling and human-computer interaction to software engineering and knowledge-based systems.

The overall thrust of Code 522 research is to move towards increasing autonomy and automation of both spacecraft platforms and their ground-based control and support systems. The foreseen development path moves from rule- and object-based approaches, characteristic of expert systems; through model-based approaches, typical of enhanced knowledge-based systems; and aiming towards sophisticated agent-based AI approaches.

One of the important research areas of Code 522 is the development of methods for trend analysis of spacecraft systems and components. In trend analysis, mathematical representations of spacecraft health are developed from telemetry analysis. These analysis methods are then employed to reveal any long-term trends indicating degradation in system health, or any incipient threat of failure.

The initial focus of the trend analysis program is the battery subsystem for the Small Mission EXplorer (SMEX) family of missions. Batteries provide a particular challenge for trend analysis because of the complexity of their environments (loads and charges) and the complexity and inherent limitations in our knowledge of their electrochemical dynamics. These conditions have resulted in great difficulties in battery quality assurance, and a number of spacecraft platforms are at risk for failure.

Existing Code 522 facilities for trend analysis provide only basic visualization and rudimentary statistical analysis of telemetry. More sophisticated approaches to supplement these methods are under development, including a Model-Based Diagnostic (MBD) approach.

In addition, due to the many forms of uncertainty inherently present in complex engineering systems like spacecraft, and especially in their battery subsystems, Qualitative Modeling (QM) methods for MBD are especially appropriate [10]. To support qualitative MBD, we plan to use the Generalized Information Theory (GIT) computational paradigm, and especially its possibilistic modeling techniques. GIT is the synthesis of modern mathematical theories of uncertainty, including fuzzy systems, random sets, evidence theory, and possibility theory [9, 16, 17]. GIT promises to provide a key generalizing technology for QM methods in systems theory [9, 11].

This paper describes the Data Analysis and Systems Modeling Environment (DASME), which is being used as the development environment for this approach, and our proposed CAST-based extensions to DASME to support a broader range of computational models (for example finite automata, petri nets, or neural nets), and especially a qualitative approach to MBD using possibility theory and possibilistic processes.

2 The Data Analysis and Systems Modeling Environment (DASME)

DASME is a computer-assisted modeling platform developed by Henderson to support ground-based mission operations. It has the dual capacities of telemetry analysis and systems modeling using a mixture of discrete-event and discrete-time methods. DASME's general and flexible architecture allows it to be easily adapted to fulfill multiple tasks, although it will be first applied in model-based trend analysis of spacecraft battery subsystems.

2.1 DASME Design

DASME is a collection of behavioral components, input components, output components, a graphical model editor, and a run-time executive.

Behavioral components simulate specific elements within the domain (currently satellite power systems) and were implemented as a library of C++ classes. Input components were derived from this same class hierarchy to feed the simulation with file-based data streams or direct user input. Output components (also a part of the class library) produce dynamic plots of their inputs, or transmit those values to a CLIPS environment for processing by expert systems.

The X-Windows based graphical model editor allows the synthesis of complex hierarchical models by connecting the input and output ports of behavioral components, input/output components, or other models. This editor also allows "zooming" into sub-models for inspection or adjustment of parameters. Completed models can be stored to and retrieved from disk through this editor.

The run-time executive provides scheduling and time management for a heterogeneous mix of discrete-time and discrete-event components with an X-Windows interface for starting, pausing, resuming, and stopping execution. Support for other computational paradigms, including causal, state, and qualitative modeling, is planned for the future, and is partially the focus of this paper. Additional future capabilities include a neuro-fuzzy machine learning component for model adaptivity.

2.2 DASME Model Architecture

DASME provides a general black box architecture. Nodes are called "components", and each contains distinct input and output ports and initial state variables called "parameters". It is also strictly hierarchical, with each component containing a number of sub-components whose ports are accessible to the parent and to each other, but not outside the containing component. Atomic components are independently compiled C++ modules, and non-atomic components are aggregates of either atomic or other aggregate components. An aggregate DASME component is shown in Fig. 1. The children `Component A` and `Component B` may either be atomic or themselves aggregates.

Data are passed among components in structures called "polyvalues", which are heterogeneous, hierarchical lists of C++ primitive data elements. For example, `float`, `str`, `(int int)`, and `((char float) int)` are polyvalue types. Type-checking is done at run-time.

Time values are either scalar (a relative duration), or absolute (microseconds from January 1, 1970). Component state transitions (execution of C++ code for atomic components, or propagation to children for aggregate components) can be triggered by means of an elegant mixture of discrete-event and discrete-time methods, as summarized in Table 1.

Events (component executions) are scheduled in either absolute or relative time, and according to one of three timing modifiers. By using the `NotBefore` modifier, components can schedule their own execution at the indicated time and no other. By using the `OrBefore` modifier, components indicate that execution should also occur if one or more of the component's inputs change. By using the `PreferBefore` modifier, components indicate that execution before the indicated time (even in the absence of new inputs) is desirable. The executive is then

Temporal Modifier	Method	When a component updates
`NotBefore`	Discrete Event	At scheduled time
`OrBefore`	Discrete Event	At scheduled time, or if input changes
`PreferBefore`	Discrete Time	At scheduled time, or if input changes, or if desired by executive

Table 1. Temporality in DASME.

allowed to increase the frequency of evaluation to be no greater than a minimal time step, as configured by the user.

This `PreferBefore` method adds a discrete-time modeling capability to DASME. This is crucial to support components whose behaviors are best modeled as continuous functions. For example, a component with piecewise-continuous behavior can guarantee its evaluation at landmark points (such as inflections or discontinuities), while simultaneously allowing the executive to trigger intermediate evaluations. Placing this responsibility in the executive, rather than solely with the component itself, allows the executive to determine an appropriate tradeoff between fidelity of the simulation and its time to compute. The benefit of this approach is that this configuration is achieved without having to modify the model.

2.3 Comparison to DEVS

DASME took substantial initial design inspiration from Zeigler's powerful Discrete EVent Systems (DEVS) modeling formalism. Only a brief summary is provided here; for full details see [24, 25].

Define a DEVS system as

$$\mathcal{D} := \langle X, Y, S, t, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda \rangle , \tag{1}$$

where:

- X is a set of external events;
- Y is a set of outputs;
- S is a set of sequential states;
- $t: S \mapsto \mathbb{R}^+ \cup \{\infty\}$ is the time advance function;
- $\delta_{\text{int}}: S \mapsto S$ is the internal transition function;
- $\delta_{\text{ext}}: Q \times X \mapsto S$ is the external transition function, where

$$Q := \{\langle s, e\rangle : s \in S, 0 \le e \le t(s)\} = \bigcup_{s \in S} [0, t(s)] \tag{2}$$

is the total state set; and

- $\lambda: S \mapsto Y$ is the output function.

These functions are interpreted as follows. Assume that $\mathcal{D}$ is in state $s \in S$. If an event $x \in X$ arrives after a duration $e \le t(s)$, then $\mathcal{D}$ transits to state $\delta_{\text{ext}}(\langle s, e\rangle, x)$. Otherwise $\mathcal{D}$ transits to state $\delta_{\text{int}}(s)$. Finally, at all times, if $\mathcal{D}$ is in state s, then it produces output $\lambda(s)$.

It is clear that DASME components can be generally interpreted as DEVSs, and in general that DASME has much in common with the DEVS implementation in Scheme called DEVS-Scheme [26]. This relation is specified as follows:

- The overall hierarchical structure, including atomic and aggregate components and input and output ports, is essentially identical.
- X and Y are the joint state spaces of the polyvalue types of the input and output ports.
- Depending on whether the component is atomic or aggregate, S is either the states achievable in virtue of the C++ code of the module, or recursively the joint state spaces of the child components.
- Each component has two methods `ClockMeAfter()` and `ClockMeAt()` for scheduling at relative and absolute times respectively. Together these determine t and δ_{int}, which is the maximum elapsed time before the component is updated; in particular, the `ClockMeAfter()` method establishes the *sigma* variable in DEVS-Scheme.
- The `NotBefore` temporal modifier sets

$$\delta_{\text{ext}}(\langle s, e\rangle, x) \equiv s, \qquad \forall s \in S, e \in [0, t(s)], x \in X, \tag{3}$$

  so that external events are essentially ignored.
- The `OrBefore` temporal modifier effectively establishes the component as a regular DEVS. δ_{ext} is fired if an input event occurs by the scheduled time, otherwise δ_{int} is triggered.
- Each component has a `Sleep()` method, which calls `ClockMeAt()` with `EndOfTime` (maximal absolute time) and the `OrBefore` method. This sets $t(s) = \infty$, so that the component will not be updated until an external event occurs, essentially ignoring δ_{int}.

Where DASME departs from the classical DEVS formalism is in the use of the `PreferBefore` modifier. It allows the model executive (basically, the DASME operating environment for the highest level component of the hierarchy) to "reach down" and intervene, triggering component updating before scheduled times, and in the absence of any external events from within the hierarchy. It is this facility which allows the executive to evaluate continuous functions at discrete times. The modeler then has the flexibility to adjust evaluation step size depending on the analytical properties of the continuous functions in question, without modifications to the model itself.

It is this capability which adds a discrete-time component, in order to better handle continuous-time functions, to an otherwise discrete-event modeling environment. Thus it is similar in outlook to the approach of Praehofer and his colleagues [20, 21].

Another difference between DASME and the DEVS approach is at the engineering level. DEVS as a modeling methodology is a completely analytical, mathematical formalism. DEVS platforms have typically been implemented in a highly constrained subset of a very high level artificial intelligence language like Scheme [26].

DASME's reliance on C++ means that it can escape from the strict DEVS formalism. For example, input DASME components can be positioned at any level of the hierarchy. As these rely on external inputs (from disk files or the terminal), there can be no *a priori* knowledge about the possible states of these components. Thus no purely analytical model of a DASME system is possible.

This creates an engineering tradeoff when considering DASME in the context of the wider DEVS world. On the one hand, DASME components have a greater flexibility, and will be generally much more efficient than implementations in higher level languages. But on the other hand, since DASME departs from the formal DEVS model, verification and formal design will be more difficult.

3 Application to Model-Based Trend Analysis

The MBD approach [5] is based on the premise that knowledge about the internal structure of a system can be useful in diagnosing its failure. In MBD, a software model of the system, given inputs from the real system, generates and tests various failure hypotheses.

A typical MBD approach (derived from some of the standard literature [1, 3, 4]) to diagnosing a spacecraft is shown in Fig. 2. The overall MBD system involves two distinct spacecraft models. The Fault Generation Model (FGM) takes inputs from telemetry, alarms (reports of departures from nominal behavior), and errors (reports of departures from predicted behavior), and either produces new, or modifies existing, fault hypotheses. The behavior model takes inputs from telemetry and fault hypotheses, and outputs predictions. These are then corroborated against telemetry to produce errors.

The fault hypotheses act to modify the behavior model so that it predicts system behavior as if the hypothetical system components had actually failed. If the prediction of the behavior model as modified by a particular fault hypothesis produces errors, then that fault hypothesis is not retained. As the system is monitored over time, further observations narrow the class of viable fault hypotheses. Achieving the null set indicates model insufficiency. But if the overall MBD system stabilizes to a non-empty set of fault-hypotheses, then these are advanced as possible causes of the failure.

Model-based trend analysis is similar to MBD, but with some differences:

- Trend analysis is typically done over a longer time-frame.
- In addition to being purely diagnostic, trend analysis attempts to be predictive. By diagnosing certain components to be failing or otherwise trending in a particular direction, potential future failures can be anticipated.
- Trend analysis is especially appropriate for modeling systems where gradual degradation leads to catastrophic failure, as is the case with batteries.

In the context of trend analysis, detection of anomalous component states, even though they may not be in a failed condition, is extremely important. Within the overall model-based approach, a variety of methods are available for signature matching of a time-varying telemetry signal against model predictions, including statistical techniques, neuro-fuzzy adaptive approaches, and possibilistic measurement (discussed below).

4 CAST Extensions to DASME

It is our intention to use DASME as a CAST-based platform for modeling within Code 522. While DASME is generally adherent to the DEVS paradigm, there is a desire for DASME to support a wider range of computational models—for example neural nets, finite automata, or fuzzy reasoning—if possible. DASME as it exists now already has many advantages for this effort, including its generality and flexibility, but some significant enhancements will be required.

The ultimate necessary extension to DASME is the introduction of weights on the links among the children of aggregate components. Similar to the data values passed among components, weights will also be **PolyValues**. These weights can be used generally as elements of a wide class of computational modeling methods, for example, weights in a neural network, conditional possibilities in a possibilistic network (discussed below), or entire fuzzy sets in a more traditional fuzzy system.

In order to facilitate the introduction of weighted aggregates, strict data typing of the components' ports, and thus of the types of the links between components, must be introduced. At model construction time, only links between ports of equivalent types will be allowed, thus eliminating the need for model run-time type checking and error recovery. An example of a DASME component with typed links is shown in Fig. 3.

Two weight data types are introduced for each component. The internal weight type is held in common by all weights on the links between the children of the component (if null, the aggregate is unweighted). The external weight type is the type of weight acceptable to a component on its inputs (if null, the component need not be a child of a weighted aggregate). An example of a DASME component with weighted links is shown in Fig. 4.

5 Possibilistic Qualitative Modeling

Possibility theory [2] is an alternative information theory to that based on probability. It was originally developed in the context of fuzzy systems theory [23], and was thus related to the kinds of cognitive modeling that fuzzy sets are usually used for. More recently, possibility theory is being developed as a new form of mathematical information theory complementing probability theory [12]. The details of mathematical possibility theory will not be introduced here (see [9, 10]), but will be described only very cursorily, and its role as a QM method briefly outlined.

5.1 Possibility Theory

A possibility distribution π over a given space or set Ω is similar to a probability distribution, but whereas a probability distribution p is additive,

$$\int_{\Omega} dp(\omega) = 1, \qquad \sum_{\omega \in \Omega} p(\omega) = 1, \tag{4}$$

in the continuous and discrete cases respectively, π is "maxitive"

$$\sup_{\Omega} \pi(\omega) = 1, \qquad \bigvee_{\omega \in \Omega} \pi(\omega) = 1, \tag{5}$$

in the continuous and discrete cases, where $\vee$ is the maximum operator. Similarly, a possibility measure Π based on π is maxitive, in that

$$\forall A, B \subseteq \Omega, \quad \Pi(A \cup B) = \Pi(A) \vee \Pi(B). \tag{6}$$

Superficially, possibility theory is broadly similar to probability theory with $\langle +, \times \rangle$ algebra replaced by $\langle \vee, \sqcap \rangle$ algebra, where $\sqcap$ is a t-norm function: a monotonic, associative, commutative operator with identity 1. The minimum operator $\wedge$ and $\times$ are both t-norms, but there are many others. Concepts of marginal, joint, and conditional possibility have all been defined, as have possibilistic correlates to stochastic entropy, called nonspecificities.

Although possibility theory is logically independent of probability theory, they are related: both arise in Dempster-Shafer evidence theory as fuzzy measures defined on random sets; and their distributions are both fuzzy sets. So possibility theory is a component of a GIT, which includes all of these fields [9, 16, 17].

But at a semantic level, probability and possibility are radically different. Probability represents division of knowledge among a distinct set of point outcomes. Possibility, on the other hand, is inherently *non-additive*. Possibility represents *coherence* of knowledge *around* a central core of certainty: the region on which $\pi(\omega) = 1$, which is guaranteed to be non-empty by normalization.

And so while probability is related to *dispersive* concepts like frequency, chance, and likelihood, possibility is related to *ordinal* concepts like capacity, ease of attainment, distance, and similarity. Furthermore, unlike probability, possibility places very weak constraints on the representation of information: the maximum relation is a very weak operator, and there is a choice of many norms to use, some of which are strong, and others of which, like $\vee$, are also weak.

So possibilistic models are appropriate where stochastic concepts and methods are inappropriate, including situations where long-run frequencies are difficult if not impossible to obtain, or where small sample sizes prevail. This is true, for example, in trend-analysis, where even though observations are made over a long time, the trending state variables of concern change only very slowly, and new domains of behavior are only very rarely seen. In these cases the weakness of the possibilistic representation is matched by the weak evidence available.

Mathematical possibility, in both theory and applications, is still in the basic research phase, just out of its infancy. For example, the axiomatic basis for possibility theory and the properties of possibility distributions on continuous spaces are still being defined, and the semantics of possibility in physical systems has been considered only by very few. Joslyn is developing mathematical possibility theory on the basis of consistent random sets, general possibilistic modeling methods (including possibilistic measurement and interpretation procedures), and an empirical semantics for possibility [9].

5.2 Possibility Theory for Qualitative Modeling

There are many reasons why it can be expected that possibility theory can come to play an important role in QM in general, and in the application of QM to MBD in particular. Two of the most important reasons for this are that possibility theory is the appropriate and direct generalization of two of the key methods used in QM: interval analysis and nondeterministic processes.

Interval analysis [19], and in particular interval-valued processes [18], are an important component of general QM methodology. A crisp real interval $I = [a, b] \subseteq \mathbb{R}$ can be represented by its characteristic function $\chi_I\colon \mathbb{R} \mapsto \{0, 1\}$, where

$$\chi_I(x) = \begin{cases} 1, & x \in I \\ 0, & x \notin I \end{cases}. \tag{7}$$

χ_I is a special kind of possibility distribution called "crisp", where $\forall \omega \in \Omega, \pi(\omega) \in \{0, 1\}$. Generalizing (7) from $\{0, 1\}$ to $[0, 1]$, and keeping some simple convexity requirements, results in "fuzzy intervals" or "fuzzy numbers". Fuzzy arithmetic [14] defines mathematical operations such as addition and multiplication on fuzzy intervals and numbers, and directly generalizes interval arithmetic. Probability theory, on the other hand, does not.

A non-deterministic process [6] can be characterized by its transition network. Each state $\omega \in \Omega$ can transit to any number of other states. If the value 1 is assigned to a possible transition, and 0 to one which is not allowed, then a boolean $n \times n$ transition matrix can be constructed, where $n := |\Omega|$. Given an initial state $\omega_i \in \Omega$, then future states are represented as subsets $\sigma(\omega_i) \subseteq \Omega$.

Similarly, nondeterministic processes generalize to possibilistic Markov processes in a way which is much more natural than stochastic Markov processes. Possibilistic processes provide an ideal medium for causal modeling using a graduated, nondeterministic representation of causal relatedness. The transition graph of a possibilistic process can essentially be regarded as a possibilistic network, similar to Bayesian networks in stochastic systems theory.

An example is shown in Fig. 5 for $\Omega = \{x, y, z\}$. The arcs indicate possible state transitions, each of which is non-additively weighted with the conditional possibility of the transition. This results in the possibilistic transition matrix

$$\boldsymbol{\Pi} = \begin{bmatrix} 0.0 & 0.8 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.2 & 1.0 & 1.0 \end{bmatrix}. \tag{8}$$

Each column of Π plays the role of $\sigma(\omega)$ in a nondeterministic process. Given an initial state possibility distribution, future possibilistic states are calculated by $\langle \vee, \sqcap \rangle$ matrix composition. Joslyn has also developed possibilistic Monte Carlo methods [9], which are required to select a specific final outcome given a possibility distribution on the state variables of the process at any given time.

6 Possibilistic Modeling with CAST-Extended DASME

We intend to use CAST-extended DASME in a number of different ways to support qualitative model-based trend analysis of spacecraft.

6.1 Data Analysis

The most immediate application of DASME to possibilistic modeling, even before the addition of weighted links, is the development of possibilistic representations of telemetry through possibilistic measurement procedures [7, 8]. The essential requirement is the collection of the frequency of occurrence $m(A_j)$ of subsets or intervals $A_j \subseteq \Omega$, yielding what is called an "empirical random set" [15]. If any of the observed intervals are overlapping, then such a representation cannot be reduced to a traditional point-valued random variable, and thus no traditional point-valued probability distribution can be forthcoming [13]. If the core of the observed intervals is nonempty, so that $\bigcap_j A_j \neq \emptyset$, then

$$\pi(\omega) := \sum_{A_j \ni \omega} m(A_j). \tag{9}$$

is an empirical possibility distribution, called a possibilistic histogram, with the same core.

An example is shown in Fig. 6. On the left, four observed intervals are shown. The bottom two occur with frequency 1/2, while each of the upper two have frequency 1/4. Together they determine an empirical random set. The step function on the right is the possibilistic histogram derived from (9).

There are a variety of well-justified continuous approximations to a possibilistic histogram. Two examples are shown in the figure. The rising diagonal on the left is common to both. The two falling continuous curves on the right are distinct to each. The trapezoidal form marked π^* is one of the most commonly used continuous approximations, but it must be noted that this is only one possibility among many, including smooth curves. This approach to possibilistic measurement generalizes to n intervals and to the continuous case.

So possibilistic measurement is distinguished from measurement in probability theory by its reliance on interval-based observations. There are a number of different ways in which intervals result from measurement [9]. The method of most immediate interest for spacecraft modeling derives from local extrema of a telemetry stream.

A simple example is shown in Fig. 7. Given a time-varying telemetry signal, each time the curve turns marks a local extrema. The segments of the ordinate between these local extrema produce a statistical collection of intervals, and thus an empirical random set and possibilistic histogram. It should be noted that this method is sensitive to noise, since each non-signal fluctuation generates two spurious local extrema, and thus interrupts the "real" interval being observed.

In the context of trend analysis, possibilistic histograms of telemetry promise to provide a novel and significant new representation of the long-term trending of the data. The core is the central region of purely nominal behavior, while the support (the larger region on which the possibility distribution is positive at all) represents the domain of observed behavior, and is thus similar to a concept of a "yellow limit". Comparison, through possibilistic distance measures [22], between possibilistic histograms at different times reveals trending information.

The data path for possibilistic measurement in DASME is shown in Fig. 8. Each node indicates a different DASME component. In the context of spacecraft systems modeling, measurements are typically desired of such variables as voltages and temperatures. The noisy telemetry signal requires smoothing. Intervals are typically observed between orbital periods, yielding one data interval approximately every 45 minutes. Data sets on the order of months are required.

6.2 Systems Modeling

Finally, CAST-extended DASME provides a rich modeling environment within which to implement possibilistic causal network models. Fig. 9 shows how the three-state possibilistic process shown in Fig. 5 would be implemented in a CAST-extended aggregate DASME component.

The three inputs to the component represent the initial possibility values of the three automata states, each of which in turn is represented by the three child components. These are atomic components, which take the initial possibility value as input, and calculate the current possibility value as output. They also take as input the values of any other state which can transition to them, that is, which have a non-zero conditional possibility.

The internal data type of the weights on the links between the components are called `fits`, for "fuzzy digit", a `float` in $[0, 1]$. The weights represent the conditional possibility of transition between states, and together comprise Π. The external weight type is null, indicating that the whole possibilistic process does not participate in a higher level systems model. Finally, the aggregate process component outputs the current possibility value of each state.

References

1. Davis, R and Hamscher, W: (1992) "Model-Based Reasoning: Troubleshooting", in: *Readings in Model-Based Diagnosis*, ed. W Hamscher *et al.*, pp. 3-24, Morgan Kaufman, San Mateo CA
2. Dubois, Didier and Prade, Henri: (1988) *Possibility Theory*, Plenum Press, New York
3. Dvorak, D and Kuipers, B: (1992) "Model-Based Monitoring of Dynamic Systems", in: *Readings in Model-Based Diagnosis*, ed. W Hamscher *et al.*, pp. 249-254, Morgan-Kaufmann, San Mateo CA

Hall, Gardiner A; Schuetzle, James; and La Vallee, D *et al.*: (1992) "Architectural Development of Real-Time Fault Diagnositc Systems Using Model-Based Reasoning", in: *Proc. 1992 Goddard Conf. on Space Applications of AI*, ed. Steve Rash, pp. 77-86, NASA Goddard, Greenbelt MD

Hamscher, W; Console, Luca; and Kleer, Johan de, eds.: (1992) *Readings in Model-Based Diagnosis*, Morgan-Kaufman

Hopcroft, John E and Ullman, Jeffery D: (1979) *Introduction to Automata Theory Languages and Computation*, Addison-Wesley, Reading MA

Joslyn, Cliff: (1992) "Possibilistic Measurement and Set Statistics", in: *Proc. NAFIPS 1992*, v. **2**, pp. 458-467, Puerto Vallerta

Joslyn, Cliff: (1993) "Some New Results on Possibilistic Measurement", in: *Proc. NAFIPS 1993*, pp. 227-231, Allentown PA

Joslyn, Cliff: (1994) *Possibilistic Processes for Complex Systems Modeling*, PhD Dissertation, Binghamton University, UMI Disseration Services, Ann Arbor MI

Joslyn, Cliff: (1994) "Possibilistic Approach to Qualitative Model-Based Diagnosis", *Telematics and Informatics*, v. **11**:4, pp. 365-384

Joslyn, Cliff: (1995) "An Object-Oriented Architecture for Possibilistic Models", in: *Proc. 1994 Conf. Computer-Aided Systems Technology*, to appear

Joslyn, Cliff: (1995) "Towards an Independent Possibility Theory with an Objective Semantics", in: *Proc. 1995 Int. Workshop on Foundations and Applications of Possibility Theory*, to appear

Joslyn, Cliff: (1995) "Strong Probabilistic Compatibility of Possibilistic Histograms", in: *Proc. 1995 Int. Symposium on Uncertainty Modeling and Analysis*, to appear

Kaufmann, A. and Gupta, M.M.: (1985) *Introduction to Fuzzy Arithmetic*, Reinhold, New York

Kendall, DG: (1974) "Foundations of a Theory of Random Sets", in: *Stochastic Geometry*, ed. EF Harding and DG Kendall, pp. 322-376, Wiley, New York

Klir, George: (1993) "Developments in Uncertainty Based Information", in: *Advances in Computers*, v. **36**, ed. M. Yovitz, pp. 255-332, Academic Press

Klir, George and Yuan, Bo: (1995) *Fuzzy Sets and Fuzzy Logic*, Prentice-Hall, New York

Kuipers, BJ: (1994) *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, MIT Press, Cambridge MA

Moore, RM: (1979) *Methods and Applications of Interval Analysis*, in: *SIAM Studies in Applied Mathematics*, SIAM, Philadelphia

Praehofer, Herbert: (1991) "Systems Theoretic Formalisms for Combined Discrete-Continuous Systems Simulation", *Int. J. of General Systems*, v. **19**, pp. 219-240

Praehofer, Herbert and Zeigler, Bernard P: (1995) *Automatic Abstraction of Event-Based Control Models*, in preparation

Ramer, Arthur: (1990) "Structure of Possibilistic Information Metrics and Distances: Properties", *Int. J. of General Systems*, v. **17**, pp. 21-32

Zadeh, Lotfi A: (1978) "Fuzzy Sets as the Basis for a Theory of Possibility", *Fuzzy Sets and Systems*, v. **1**, pp. 3-28

Zeigler, BP: (1976) *Theory of Modeling and Simulation*, Wiley, New York

Zeigler, BP: (1985) *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, San Diego

Zeigler, BP: (1990) *Object-Oriented Simulation with Hierarchical Modular Models*, Academic Press, San Diego

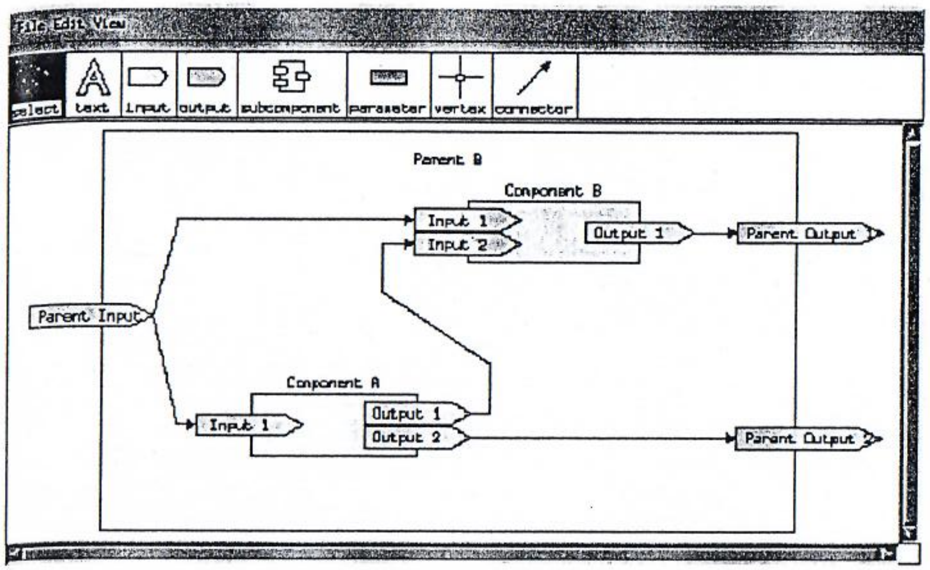


Fig. 1. An aggregate DASME component.

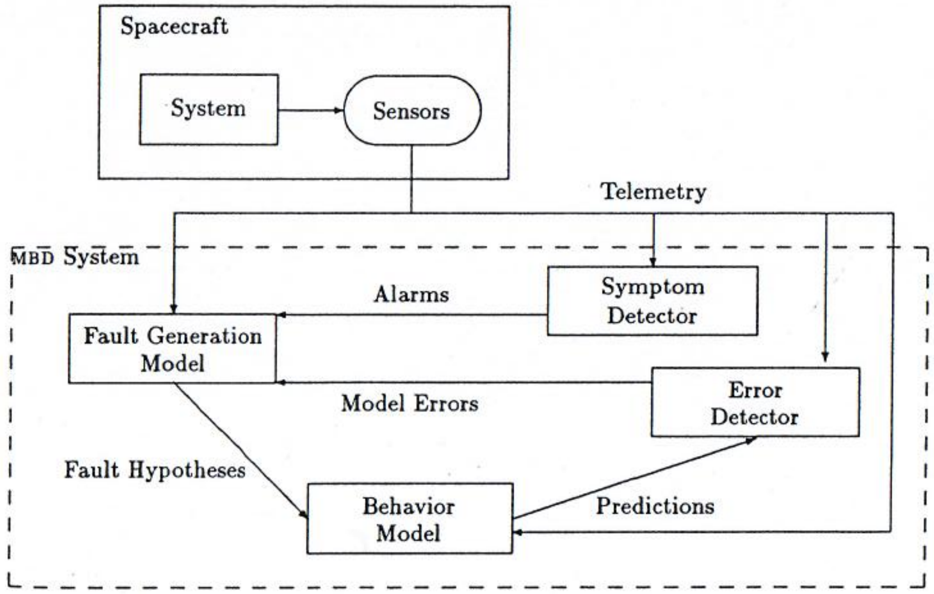


Fig. 2. A typical model-based diagnostic system.

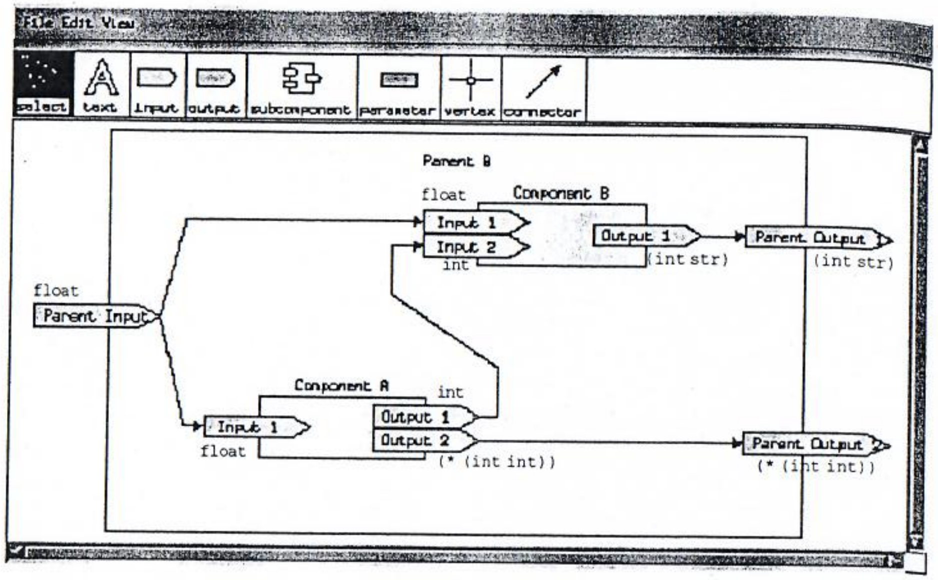


Fig. 3. A DASME component with typed links.

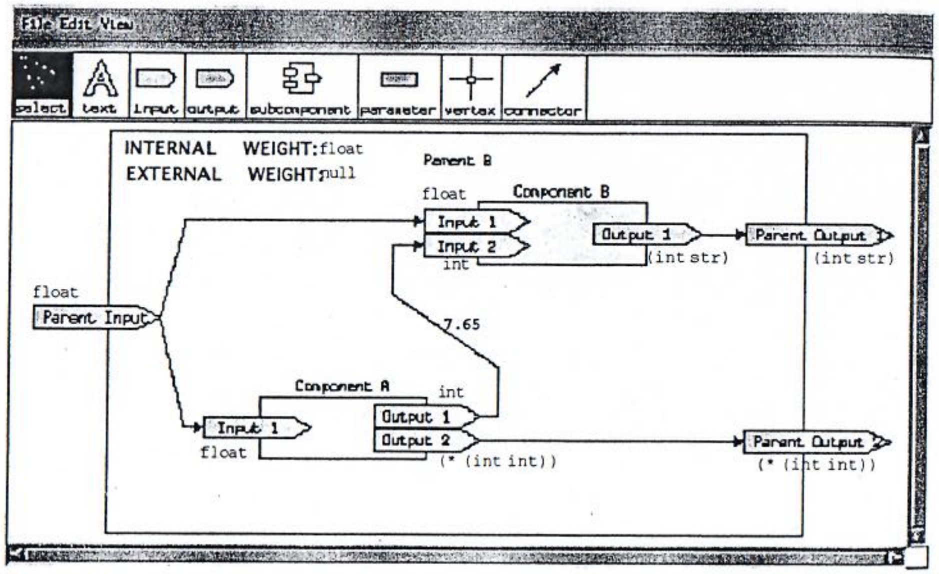


Fig. 4. A DASME component with weighted links.

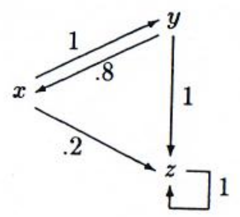


Fig. 5. Weighted state transition diagram for a possibilistic Markov process.

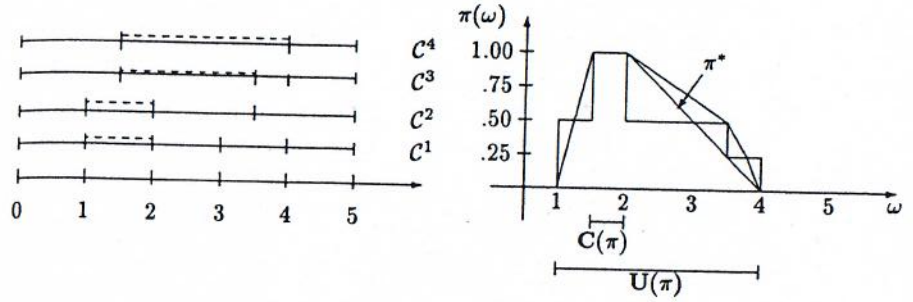


Fig. 6. (Left) Four example observed intervals. (Right) The possibilistic histogram and two continuous approximations.

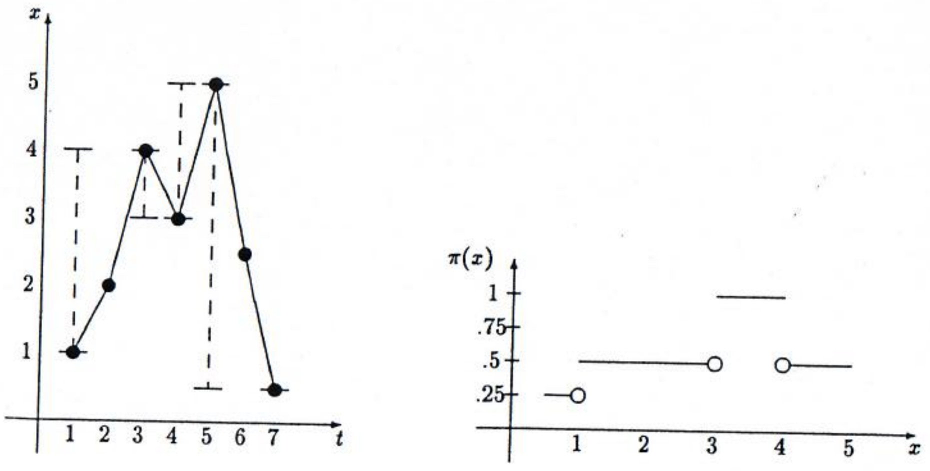
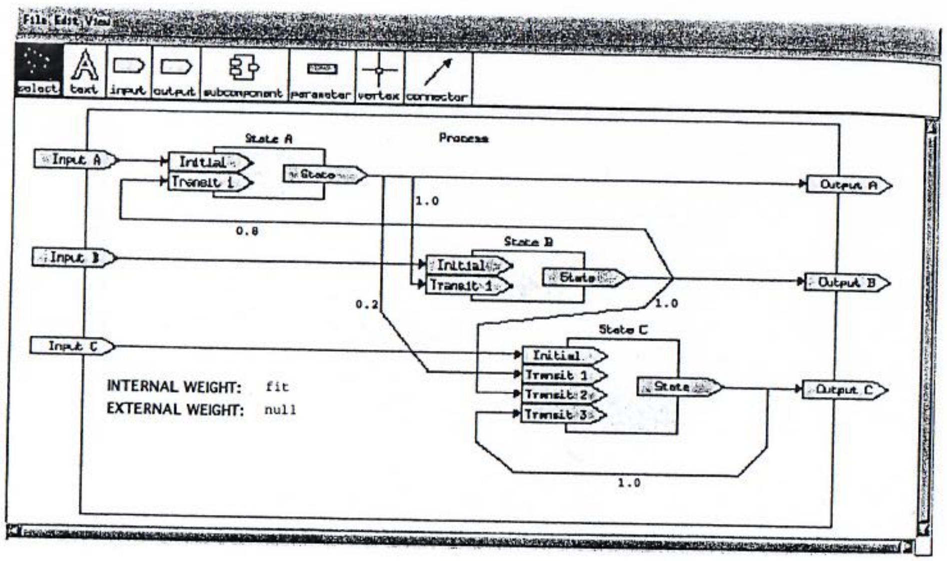


Fig. 7. Observed intervals and resulting possibilistic histogram from local extrema.

$$\begin{tikzcd}
\text{Telemetry Acquisition} \arrow[d] \\
\text{Smoothing} \arrow[d] \\
\text{Stream of Intervals (Local Extrema)} \arrow[d] \\
\text{Discrete Random Set on } \mathbb{R} \arrow[d] \\
\text{Possibilistic Histogram} \arrow[d] \\
\text{Continuous Envelope}
\end{tikzcd}$$

Fig. 8. Possibilistic measurement in DASME.



INTERNAL WEIGHT: fit
EXTERNAL WEIGHT: null

Fig. 9. A possibilistic process implemented in a CAST-extended DASME component.

Symbolic Computing Aided Design of Nonlinear PID Controllers

Jesús Rodríguez-Millán and Juan Cardillo

Universidad de Los Andes - Facultad de Ingenieria

Escuela de Sistemas - Dept. de Sistemas de Control

Apartado 11 - La Hechicera, Mérida 5251-A, VENEZUELA

Fax: 58-74-402846, E-mail: jrmillan@ing.ula.ve

Abstract. In this paper we introduce a symbolic computing tool, denoted by NLPID in the sequel, for the automatic design of linear and nonlinear PID controllers for nth order nonlinear control systems. The nonlinear design algorithm is based upon Rugh's Extended Linearization Technique, and it was implemented using Mathematica® as symbolic computing platform. At its present stage of development NLPID uses Ziegler-Nichols tables to synthesize linear PID controllers, and therefore its ability to deal with first and second order plants could be limited.

Keywords: Nonlinear PID Controllers, Jacobian and Extended Linearization, Symbolic Computing.

Contents

1 Introduction

NonLinear Control Systems (NLCS) are dynamical systems defined through (i) a *state equation*, i.e., an ordinary differential equation: