

Flexible Integration for Physiological Models

Christopher Landauer, Kirstie L. Bellman

Aerospace Integration Science Center, The Aerospace Corporation, Los Angeles, CA, USA

cal@aero.org, bellman@aero.org

30 March 2003

Contents

1 Introduction	2	case of illness or injury, when there is no possibility of returning them to Earth for treatment in any useful amount of time.
2 Context and Background	2	Part of this problem is to develop appropriate predictive models of human physiological functions, tailored to the individual astronauts, with the expectation that if these models are good enough, ground monitoring crews can understand certain deficiencies or other failure processes long enough in advance to prevent them, or at least soon enough to alleviate them.
3 Challenges of Physiological Model Integration	3	The scope of the problem, which is to model enough physiological functionality for predictive health monitoring and intervention, requires a distributed development, with individual physiological components modeled by different organizations in different places. Distributed component model development means that model integration becomes a fundamentally important part of the project.
4 Our Approach to Model Integration	4	In this paper, we explain how this kind of model integration can be done, and show how our Wrapping approach to intelligent system integration can be used to do it. It should be noted that we are not addressing the (also very difficult) physiological modeling problem, of making the individual component models sufficiently accurate for our purposes. We are only addressing their integration.
4.1 Modeling Systems and Principles . . .	4	
4.2 Physiological Modeling Systems . . .	5	
5 Wrapping for Complex Model Integration	5	
5.1 Problem Posing	5	
5.2 Wrapping Overview	6	
5.3 Wrapping Description	7	
5.4 Advantages of Wrapping for Integration	8	
5.5 Wrapping-Based Modeling System	8	
6 Conclusions and Prospects	9	

Abstract

The application problem we address is fundamental to long duration space flights: to monitor and maintain the health of the astronauts, and to intervene in

Modeling is still hard. Appropriate physiological models must still be developed and described.

Keywords: Integrated Human Function, Integration Science, Long Duration Space Flight, Model Integration, Predictive Physiological Modeling, Wrap-

1 Introduction

The most common difficulty in integration of component models in any subject domain is that the modeling assumptions of the components are inconsistent, rendering the combined model unjustifiable (and often incorrect), even when very good component models are used. In our study of *Integration Science* [5] [7], we have developed some new methods for tracking these inconsistencies, and for helping to manage the integration process. Of course, it remains a difficult modeling problem to identify a component model's assumptions, and to adjust it to change those assumptions, but we can make the integration of component models with differing assumptions explicit and available, so it (the integration process itself) can be studied as a technical aspect of the combined model.

Physiological modeling is especially interesting in this regard, since many of its models are produced using the same differential equation formalism, with the implicit and incorrect assumption that if the models are expressed in the same language, then they must be compatible.

The purpose of these combined models is to help discern what is known about a patient from available measurements, previous history, and observations. This task of information processing and integration is formidable for co-located human physicians, and becomes nearly impossible when they are separated from their patients by either geographic distances here on earth or by the much larger distances (and time delays) of space. The goal of some recent national initiatives has been to complement human physicians with better models of the medical status of specific individuals, including predictive models. The approach presented in this paper provides a framework that makes it easier to integrate a heterogeneous set of models, measurements devices, database results, reports, etc., and hence to better our analysis and understanding of the patient's status.

For this approach to work, the physiological models must be very accurate, both for normal functioning and for anomalous behavior. They must also be tunable to each of the astronauts individually, so that their physiological functions can be measured and compared to the appropriate individual models.

The scope of the problem, which is to model enough physiological functionality for predictive health monitoring and intervention, requires a distributed development, with individual physiological components modeled by different organizations in different places. Distributed component model development means that model integration becomes a fundamentally important part of the project.

2 Context and Background

Among the development projects currently underway in these initiatives, there are many projects that are developing new instruments, and new intervention or intervention assistance devices, intended to make them more informative, more portable, more robust, or more easily controllable from a remote site. There are also several physiology modeling projects, intended to produce more accurate physiological component models that can be used for prediction.

We are focussed in this paper on the software architecture for integration of these models and the available measurements into a predictive whole body model. Our Wrapping approach to intelligent integration can be used to organize and manage the integration of these models, both initially and after deployment, retaining the flexibility in the infrastructure to allow models to be changed as we learn more, and to allow different kinds of models to be incorporated as other kinds of physiological prediction questions arise.

Our Wrapping approach can also be used to improve the physiological component models, since improved flexibility in modeling means:

- more models can be examined,

- alternative hypotheses can be compared,
- individualized models can be developed, and
- models can be used with and compared to diverse real-time measurements.

Our background for this development is as part of The Aerospace Corporation, the FFRDC for military space, and in particular, in the Aerospace Integration Science Center, with many years of experience in developing theoretical methods for practical integration of Constructed Complex Systems.

Historically, space systems are among the most complex engineered systems that human societies build that work (at least they almost always work). They are huge: they involve hundreds of organizations, thousands of people, tens of thousands of components, millions of lines of code. They are distributed around the world, on the ground and in space. They take years to develop, and are expected last for decades. It is remarkable that they can be built at all. The main technical problem for us is how they can be developed systematically and reliably, (relatively) inexpensively and (more) safely.

The Aerospace Integration Science Center has developed a Knowledge-based approach to integration infrastructure for software and model integration problems that is intended to address problems of this scale. We called it the *Wrapping* approach when we first demonstrated the ideas in the late 1980's [14] [16] [3], and it has become a well-studied mechanism for complex or otherwise difficult integration projects [19] [26].

Our experience is that the study of the process of integration as a subject in itself is essential for the success of this kind of integration project [7] [8] [9] [27].

Wrapping is an active integration mechanism (that is, running in parallel with the application domain computations), designed originally to handle large-scale model development systems, such as the *VE-HICLES* system for the Conceptual Design of space systems, with hundreds of computational models, including simulations, evaluations, comparisons, hy-

pothesis testing, optimizations, and graphical user interfaces [4] [5].

Moreover, any integrated modeling system that is deployed has some stringent autonomy requirements, and we have shown that Wrapping-based systems are well-suited to autonomous operation [22] [25].

3 Challenges of Physiological Model Integration

Constructing effective models is hard in any application domain. Making different models interact usefully is also hard.

The modeling process (in general) consists of five steps:

- identify components and functionalities (the phenomena of interest);
- choose a formal or computational space (the *model* space);
- map the phenomena of interest into the model space;
- perform the derivations or computations implied by the analysis questions;
- map the results back into the phenomena;

and presumably, the mapped results imply interesting or important or unexpected properties of the phenomena of interest. It should be noted that we are considering only digital models, that is, we are describing the phenomenon, not constructing a physical analog of it. Other kinds of modeling are possible, but we do not think they will be as useful for this application.

There is enough difficulty in the case at hand, since even the basic physiological components and properties are not always clear, and the interactions are sometimes completely surprising. We would like to make the development and use of integrated models effective enough to help us study the possibilities.

There are also pervasive component interactions. Physiological phenomena do not seem to have any simple boundaries, and this fact is the most important difference to remember between these systems and engineered systems.

Another difficulty is that there are many kinds of behavior modulation (in which small changes cause large effects), and that many of these magnifications are part of complex regulatory mechanisms that reduce or eliminate the possibility of injurious chaotic dynamics.

In the overall development initiative, multiple physiological component models are being developed, with differing assumptions and limitations, somewhat differing modeling mechanisms, and even differing time and space scale (level of detail) and scope (extent). The basic problem is to combine these models usefully.

Of course, some integrated models do already exist, but their integration scaffolding is gone. A well-known example is Guyton's model of the dynamics of the cardiovascular system (see [11] for a discussion of the physiology involved, and references there to various incarnations of the model definition).

All components of this model use the same mathematical formalism: coupled differential equations. There are a few identifiable components, but they have very permeable boundaries, with different kinds of interactions.

Because the physiological behavior of this system depends on chemical activities, this model treats interactions at a very detailed biochemical scale. There do not seem to be any sufficiently useful computational abstractions of these interactions, since none of the interfaces is simple enough for the abstractions to reflect enough of the complexity of behavior. This difficulty is the main one faced by models of biological phenomena, and tends to drive the models to ever-increasing detail. This extra detail is not just unnecessary; it is detrimental to the accuracy of the model unless the model parameters are accurately known.

4 Our Approach to Model Integration

We start with a list of modeling principles that are important for integration in Constructed Complex Systems, which are heterogeneous or distributed systems managed or mediated by software. It should be noted that, while the physiology of humans is not a Constructed Complex System, we expect any kind of integrated modeling system to be one.

4.1 Modeling Systems and Principles

NO one model, modeling method, or even modeling notation suffices for understanding a complex system. There will not be a single model or simulation that can answer all questions. In particular, different modeling mechanisms will be more appropriate for different components, even in the same combined model. We believe that this diversity is necessary for all complex system modeling [10] [2] [33].

Even when the mechanisms are all the same, integration meta-knowledge is extremely useful, for a simple reason. If we learn anything from our models, then we will want to change those models, and if we do not have that meta-knowledge, then changing part of a model is much more difficult. Different component models provide different information in a combined model, or they play different roles in the combined model, and keeping track of those differences is important. We seldom change all components together. We need the integration scaffolding to help understand what to change, and to record the changes and their reasons.

Another point to be emphasized again is that the most detailed model is not always the most appropriate model [17]. In particular, different analysis questions often need different levels of detail in the models. Otherwise, if too detailed a model is used, data needs to be made up to fill in the excess details, and the effects of those inventions is very seldom studied using the models.

4.2 Physiological Modeling Systems

We assume that the models describe component behavior using various mathematical or computational notations. These notations can be compiled together into a code to be run later, or interpreted directly.

Most of the physiological model systems that we have seen use differential equations, and too many of them embed the differential equation solver into the equations, so that changes are therefore expressed as delta time updates. In our opinion, it is much better to leave the equations in their original differential form, since appropriate and accurate solvers can be selected automatically. There is a lot known about criteria for this kind of algorithm selection [30] [32].

Different kinds of analysis need models at different levels of detail. Different hypotheses about the underlying phenomena mean that there will be alternative models of the same component and combination models.

Most of these combined systems are described as monolithic wholes. They have removed the scaffolding of knowledge that was used to put the pieces together. Our experience is that this knowledge is essential for understanding the model and how the pieces fit together, and especially important for understanding how to make changes. In addition, retaining this knowledge explicitly in the system itself is the only way that the system itself can be used to help make integration choices.

5 Wrapping for Complex Model Integration

In this Section, we describe very briefly our Wrapping approach to model integration. There are many references to the details [19] [23] [26] [28]. We are focussed on the software architecture for integration of the physiological component models and the available measurements into a whole body model.

The Wrapping approach is a Knowledge-based integration mechanism that we have developed for in-

tegration in Constructed Complex Systems. It is based on two key, complementary elements: (1) explicit, machine-interpretable descriptions of all software, hardware, and other computational resources in a system, and (2) active integration processes that select, adapt, and combine these resources for particular problems.

We have shown its wide applicability in software and system development [19] [24]. The approach complements existing standards, such as HLA and CORBA, and provides additional context and semantic information.

5.1 Problem Posing

We begin with an important change of attitude in system design and implementation that we have called the “Problem Posing” interpretation of programs [20] [21]. It is a declarative interpretation that can be applied to any programming or design language, and we believe that it affords a clearer way to interpret the expressions of all programs. The basic idea is to consider the code that usually gets written as defining a “resource” that provides some kind of “information service” in response to a “posed problem”, and then keep the problems available in the code along with the solutions. This separation of clients from servers has become interesting and useful in larger units (clients and servers are typically entire programs), but we believe that it is important also for smaller units, as far down as one wants to gain the associated flexibilities.

Thus, programs interpreted in this style do not “call functions”, “issue commands”, “assert constraints”, or “send messages”; they “pose problems” (these are information service requests). Program fragments are not written as “functions”, “modules”, “clauses”, or “objects” that do things; they are written as “resources” that can be “applied” to problems (these are information service providers).

The Problem Posing interpretation is particularly effective in combination with “Wrappings”, our computationally reflective knowledge-based approach to

integration infrastructure and the development, integration, and management of heterogeneous computing systems, which we describe next.

5.2 Wrapping Overview

Wrapping is based on making all the computational processing elements explicit, including all of the information and all of the processing of that information. There is:

- machine-interpretable information (in the *Wrappings* or Wrapping Knowledge Bases) about the uses of all computational resources, and
- a set of active integration processes (the *Problem Managers*) that interpret the Wrapping information to implement the *Intelligent User Support* functions [3]:
 - Discover: which resource(s) are appropriate
 - Select: which resource to use
 - Assemble: syntax (data)
 - Integrate: semantics (information)
 - Adapt: control parameters, switches, etc.
 - Explain: why a resource was or was not used
 - Evaluate: the result of the use of the resource

The computational resources include the domain models, their interpreters and other processing elements, any reference data files, measurement sources, auxiliary or utility functions (such as differential equation solvers or optimizers, databases or graphical user interfaces), interconnection architectures, and every other element that provides an information service, to a user or to other resources in the system.

The Wrappings describe not only how to use a resource (which is about getting the right bits to the

right place in the right format, which is what we mean by *Assembly*), but also more qualitative information about when, whether, and why it is appropriate to use the resource for the current problem in the current context (which is much more about the semantics of the resource use, which is what we call *Integration*). This information is collected into knowledge bases for analysis by the Problem Managers.

The Problem Managers map posed problems into resource applications in a very flexible way, based on context and mediated by the Wrapping Knowledge Bases (*Knowledge-Based Polymorphism*). They use the Wrapping descriptions to determine which resources to use, how to combine them, and how to organize the system's computational resources in response to problems posed to it by users (who can be either computing systems or humans). Wrappings are therefore a very general way of allowing many methods to co-exist, and using specialized methods for planning, integration, monitoring, and analysis in their appropriate context.

Part of the key to this flexibility is the Problem Posing interpretation described above, which allows much more information than is usually available to be taken into account in the computational resource selection process. It goes beyond the usual separation of interface (what to do) and implementation (how to do what is to be done) in the object-oriented paradigm, by separating the posed problem (what needs to be addressed) from the resource that can address it (what to do to provide what is needed). This separation keeps the posed problems in the models, where they can be analyzed by the Problem Managers, so that different resources can be used in different contexts. This is analogous to a kind of problem- and context-dependent invocation, with the Wrapping Knowledge-Bases providing the indirection rules.

This very flexible infrastructure can be used to construct executable codes from model specifications, so that the overhead is entirely contained in compile time.

5.3 Wrapping Description

In this Subsection, we describe Wrappings in a little more detail. The Wrapping theory has four essential properties, that underlie its simplicity and power:

1. ALL parts of a system architecture, at all levels of detail, are *resources* that provide an *information service*, including programs, data, user interfaces, infrastructure services, architecture and interconnection models, and everything else (implementors choose a level of detail below which they do not want to decompose the services).
2. ALL activities in the system are *problem study*, (i.e., all activities *apply* a resource to a *posed problem*), including computations, user interactions, information requests and announcements within the system, service or processing requests, etc. (implementors choose a level of detail below which they do not want to decompose the activities). We therefore specifically separate the problem to be studied from the resources that might study it.
3. *Wrapping Knowledge Bases* (or WKBs) contain *Wrappings*, which are explicit machine-interpretable descriptions of all of the resources and how they can be applied to problems to support what we have called the *Intelligent User Support* (IUS) functions [3] (as described above). Wrappings contain much more than “how” to use a resource. They also provide information to help decide “when” it is appropriate, “why” it might be the right one for the problem, and “whether” it can be used in this current problem and context.
4. *Problem Managers (PMs)*, including the *Study Managers (SMs)* and the *Coordination Manager (CM)*, are algorithms that use the Wrapping descriptions to collect and select resources to apply to problems. Making these infrastructure resources also explicit is one key to the flexibility afforded in Wrapping systems. They use implicit invocation, both context and problem

dependent, to choose and organize resources. The PMs are also resources, and they are also Wrapped. This feature is the other key.

Thus, a system built with Wrappings uses what we have called *Knowledge-Based Polymorphism* to connect the problems to appropriate resources. It is therefore an example of the Problem Posing Paradigm.

A Wrapping is not simply a coded interface “to” a resource, the way the usual “Wrappers” are; it is a conceptual interface to the “use” of a resource, for a particular problem in a particular context (the selection of the appropriate resource to apply for a given problem in a given context is a kind of case-based reasoning [13]). This information is used to generate the appropriate Wrapper interfaces on the fly. We Wrap “uses” of resources instead of resources in and of themselves, since many analysis tools have grown by accretion over the years, and common ways to use them have developed their own style. This non-correspondence is one of the important normalizing features of Wrapping, since it allows the uses of resources to be much more simply described than trying to describe the entire resource at once.

Because all of the resources are Wrapped, including the Wrapping processes, the system is Computationally Reflective [1] [29] [12] [18], which means that it has an interpretable model of itself. This ability of the system to examine its own behavior, including current and expected future behavior, also has many interesting and important consequences. In particular, the system has no privileged resources, since any part of it, including the basic infrastructure resources (such as the Problem Managers), can be selected and replaced using the same processes. It is this ability of the system to analyze and modify its own behavior, at whatever level of detail is important (and that was defined by the implementor’s choice of resources and Wrappings) that provides the power and flexibility of resource use.

The Wrappings are not actually code surrounding a given software resource, as one might imagine various intelligent front-end programs are. Instead, the Wrappings are collected into one or potentially sev-

eral knowledge bases, called Wrapping Knowledge Bases (WKBs), which allows developers to test this meta-knowledge with known static methods for evaluation of knowledge bases [2] [15] [6] [18] [24].

Along with this change of attitude, we have defined a semantically neutral expression notation for Wrappings. The basic expressive notion of Wrapping is the “posed problem”, and the basic computational component is the “resource”. They are connected by the “Wrappings”, which consist of processes and associated knowledge bases that convert a posed problem into coordinated collections of resources that can address the problem. We have developed this notion into the *wrex* notation, which extends the applicability of Wrapping all the way down to the data access and expression evaluation level of detail. We call *wrex* semantically neutral because the semantics of any given expression is completely defined by the collection of resources available to the system when that expression is evaluated. It allows a system builder to write semantically neutral code, thus avoiding language design tyranny [31], and which is then activated by means of the resources provided. Of course, it also leads to some interesting and difficult problems in semantic specification. The problems are not new; they are the “self-modifying code” problem, writ large, but we know that they can be solved in many special cases [28].

5.4 Advantages of Wrapping for Integration

Systems described using Wrappings have explicit meta-knowledge about the uses of the available computational resources, that can be prepared and studied independently of model execution. Other programs can perform other kinds of computations on this meta-knowledge (for example, for model verification and validation as described above).

Model assumptions and conventions can be recorded and retained, and therefore their effects can be more easily identified. In order to build combined models, component models can be automatically matched by level of detail and assumptions, so that multiple al-

ternative hypotheses can be studied.

Overlapping models can be used, with compatibility interface models. This is useful when component models have parts that represent other components (at lower levels of detail). These models can still be integrated, as long as there is an appropriate mapping of activity between the different models of the same components (as monitored and / or enforced by another resource provided by the modelers).

Individualized models can be selected and adapted according to context (adaptation models are needed), and used with or compared to real-time measurements.

The Wrapping infrastructure is inherently distributed, and can use any transport mechanism.

Finally, all of this flexibility can be compiled out so there is essentially no run-time cost [20].

5.5 Wrapping-Based Modeling System

A Wrapping-Based modeling system is a collection of component and interaction models, in various formal or computational notations, a collection of computational utilities and databases that are used in conjunction with those models, a collection of Wrapping Knowledge bases containing Wrappings of those models, and a collection of Problem Managers.

Each model Wrapping includes careful descriptions of assumptions and applicability, limitations and other constraints, styles and conventions of use (many model failures occur when they are used out of their range of applicability, or outside the scope of their assumptions).

Many kinds of analysis programs can be written and used as utilities, once there are resources supplied that interpret the various model notations.

Programs can read interaction models, collect appropriate component models, and compile all of them into a model program to run later. Such model programs can be run with a graphical user interface for

selecting input parameters and output displays, and monitoring the dynamic behavior of the models.

Programs can compare the different levels of detail available, and determine that component models at some levels of detail are missing (that is, the system itself can not only help the developers use it, but also help them develop it in the first place or extend it later on).

Programs can compare two different models of the same component at different levels of detail for compatibility or approximate consistency (according to supplied resources that implement the appropriate criteria).

It is a fundamental tenet of this approach that there are no privileged resources anywhere in the system. The Wrapping Knowledge Base interpreters (and therefore the Knowledge base formats used) can be partially replaced, the Problem Managers (and therefore the notations in which they are written) can be superseded, and specific planning methods that apply only in constrained contexts can be used, alongside of the usual ones supplied in the default Wrapping core (as long as their criteria for use are specified).

The cost of this flexibility and power is that the system developers need to find or write these resources, describe their appropriate context of use in the system, and provide interpreters for the notations used. Of course, this is true whatever kind of integration mechanism is used, and the advantage of using Wrappings is that the use criteria, design decisions, and notation interpreters remain in the system for later perusal and analysis, modification and explanation.

The use of Wrappings also does not solve the problem of inconsistent assumptions in the component models. As above, this inconsistency problem may occur no matter what kind of integration is used, but the Wrapping approach allows those assumptions to be made explicit, where they and their effects can be examined, compared, and changed when necessary.

6 Conclusions and Prospects

We have described Wrappings very briefly in this paper, and argued that it will be a useful approach for developing integrated physiological models. The rest of this Section has a description of our assessment of the prospects for this kind of model integration problem.

The first point to make is that our discussion is about integration of models, not about making individual physiological significant models. The process of (physiological) modeling is hard and will remain hard. What Wrappings offers to the modelers and the modeling community is a framework for simplifying the integration compromises of individual models by making them all explicit and analyzable, and a mechanism for allowing them all to work together in a managed context without requiring all of the interacting models to use exactly the same modeling style (there are, of course, some careful considerations required for the interactions across styles, but again, the use of Wrappings provides a way to make those considerations explicit and the resulting interfaces systematic and repeatable).

Our claim is that many physiological models will benefit from the “Software Disintegration” required to prepare them for a Wrapping-based system. In order to understand the models, their assumptions and limitations, their roles in a combined model and the information services they provide, and the relevant time and space scales, all must be made explicit for analysis, no matter what model integration mechanism is used. Wrappings allows all of this knowledge to be retained in the system for use by run-time or other analysis programs.

References

- [1] Harold Abelson, Gerald Sussman, with Julie Sussman, *The Structure and Interpretation of Computer Programs*, Bradford Books, now MIT (1985)
- [2] Kirstie L. Bellman, “The Modelling Issues In-

- herent in Testing and Evaluating Knowledge-based Systems”, pp. 199-215 in Chris Culbert (ed.), *Special Issue: Verification and Validation of Knowledge Based Systems, Expert Systems With Applications Journal*, Volume 1, Number 3 (1990)
- [3] Kirstie L. Bellman, “An Approach to Integrating and Creating Flexible Software Environments Supporting the Design of Complex Systems”, pp. 1101-1105 in *Proceedings of WSC’91: The 1991 Winter Simulation Conference*, 8-11 December 1991, Phoenix, Arizona (1991); revised version in Kirstie L. Bellman, Christopher Landauer, “Flexible Software Environments Supporting the Design of Complex Systems”, *Proceedings of the Artificial Intelligence in Logistics Meeting*, 8-10 March 1993, Williamsburg, Va., American Defense Preparedness Association (1993)
- [4] Kirstie L. Bellman and April Gillam, “Achieving Openness and Flexibility in VEHICLES”, pp. 255-260 in *Proceedings of EMC’90: The 1990 SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)
- [5] Kirstie L. Bellman, April Gillam, Christopher Landauer, “Challenges for Conceptual Design Environments: The VEHICLES Experience”, *Revue Internationale de CFAO et d’Infographie*, Hermes, Paris (September 1993)
- [6] Kirstie L. Bellman, Christopher Landauer, “Designing Testable, Heterogeneous Software Environments”, pp. 199-217 in Robert Plant (ed.), *Special Issue: Software Quality in Knowledge-Based Systems, Journal of Systems and Software*, Volume 29, No. 3 (June 1995)
- [7] Kirstie L. Bellman, Christopher Landauer, “Integration Science is More Than Putting Pieces Together”, in *Proceedings of the 2000 IEEE Aerospace Conference (CD)*, 18-25 March 2000, Big Sky, Montana (2000)
- [8] Kirstie L. Bellman, Christopher Landauer, “Towards an Integration Science: The Influence of Richard Bellman on our Research”, *Journal of Mathematical Analysis and Applications*, Volume 249, Number 1, pp. 3-31 (2000)
- [9] Kirstie L. Bellman, Christopher Landauer, “Creating a Flexible Integration Framework for Physiological Modeling”, *Proceedings of EMBS’2002’BMES: The Second Joint Meeting of the IEEE Engineering in Medicine and Biology Society and the Biomedical Engineering Society*, 23-26 October 2002, Houston, Texas (2002)
- [10] Richard Bellman, P. Brock, “On the concepts of a problem and problem-solving”, *American Mathematical Monthly*, Volume 67, pp. 119-134 (1960)
- [11] Arthur C. Guyton, M.D., John E. Hall, *Textbook of Medical Physiology (10th Ed.)*, W. B. Saunders (2000)
- [12] Gregor Kiczales, Jim des Rivieres, Daniel G. Bobrow, *The Art of the Meta-Object Protocol*, MIT Press (1991)
- [13] Janet Kolodner, *Case-Based Reasoning*, Morgan Kaufmann (1993)
- [14] Christopher Landauer, “Wrapping Mathematical Tools”, pp. 261-266 in *Proceedings of EMC’90: The 1990 SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990); also pp. 415-419 in *Proceedings of Interface’90: The 22nd Symposium on the Interface (between Computer Science and Statistics)*, 17-19 May 1990, East Lansing, Michigan (1990)
- [15] Christopher Landauer, “Correctness Principles for Rule-Based Expert Systems”, pp. 291-316 in Chris Culbert (ed.), *Special Issue: Verification and Validation of Knowledge Based Systems, Expert Systems With Applications Journal*, Volume 1, Number 3 (1990)
- [16] Christopher Landauer, “Wrapping Mathematical Tools for Design Analysis”, *Proceedings*

- of SOAR'90: *The 1990 Conference on Space Operations, Automation, and Robotics*, 26-28 June 1990, Albuquerque, New Mexico (1990)
- [17] Christopher Landauer, Kirstie L. Bellman, "Integrated Simulation Environments" (invited paper), *Proceedings of DARPA Variable Resolution Modeling Conference*, 5-6 May 1992, Herndon, Virginia, Conference Proceedings CF-103-DARPA, published by RAND (March 1993); shortened version in Christopher Landauer, Kirstie L. Bellman, "Integrated Simulation Environments", *Proceedings of the Artificial Intelligence in Logistics Meeting*, 8-10 March 1993, Williamsburg, Va., American Defense Preparedness Association (1993)
- [18] Christopher Landauer, Kirstie L. Bellman, "Knowledge-Based Integration Infrastructure for Complex Systems", *International Journal of Intelligent Control and Systems*, Volume 1, No. 1, pp. 133-153 (1996)
- [19] Christopher Landauer, Kirstie L. Bellman, "Wrappings for Software Development", pp. 420-429 in *Proceedings of HICSS'98: The 31st Hawaii Conference on System Sciences (also on CD), Volume III: Emerging Technologies, Software Process Improvement Mini-Track*, 6-9 January 1998, Kona, Hawaii (1998)
- [20] Christopher Landauer, Kirstie L. Bellman, "Generic Programming, Partial Evaluation, and a New Programming Paradigm", Paper ETSPI02 in *Proceedings of HICSS'99: The 32nd Hawaii Conference on System Sciences (CD), Track III: Emerging Technologies, Software Process Improvement Mini-Track*, 5-8 January 1999, Maui, Hawaii (1999); revised and extended version in Christopher Landauer, Kirstie L. Bellman, "Generic Programming, Partial Evaluation, and a New Programming Paradigm", Chapter 8, pp. 108-154 in Gene McGuire (ed.), *Software Process Improvement*, Idea Group Publishing (1999)
- [21] Christopher Landauer, Kirstie L. Bellman, "Problem Posing Interpretation of Programming Languages", Paper ETECC07 in *Proceedings of HICSS'99: The 32nd Hawaii Conference on System Sciences (CD), Track III: Emerging Technologies, Engineering Complex Computing Systems Mini-Track*, 5-8 January 1999, Maui, Hawaii (1999)
- [22] Christopher Landauer, Kirstie L. Bellman, "New Architectures for Constructed Complex Systems", in *The 7th Bellman Continuum, International Workshop on Computation, Optimization and Control*, 24-25 May 1999, Santa Fe, NM (1999)
- [23] Christopher Landauer, Kirstie L. Bellman, "Lessons Learned with Wrapping Systems", pp. 132-142 in *Proceedings of ICECCS'99: The 5th IEEE International Conference on Engineering Complex Computing Systems*, 18-22 October 1999, Las Vegas, Nevada (1999)
- [24] Christopher Landauer, Kirstie L. Bellman, "Detecting Anomalies in Constructed Complex Systems", Paper ETSSV01 in *Proceedings of HICSS'00: The 33rd Hawaii International Conference on System Sciences (CD), Track IV: Emerging Technologies, Advances in Software Specification and Verification Mini-Track*, 4-7 January 2000, Maui, Hawaii (2000)
- [25] Christopher Landauer, Kirstie L. Bellman, "Reflective Infrastructure for Autonomous Systems", pp. 671-676, Volume 2 in *Proceedings of EMCSR'2000: The 15th European Meeting on Cybernetics and Systems Research, Symposium on Autonomy Control: Lessons from the Emotional*, 25-28 April 2000, Vienna (April 2000)
- [26] Christopher Landauer, Kirstie L. Bellman, "Wrappings for One-of-a-Kind System Development", Paper STSSV04 in *Proceedings of HICSS'02: The 35th Hawaii International Conference on System Sciences (CD), Track IX: Software Technology, Advances in Software Specification and Verification Mini-*

Track, 7-10 January 2002, Waikoloa, Hawaii (Big Island) (2002)

- [27] Christopher Landauer, Kirstie L. Bellman, “Integration Science and Distributed Networks”, *Proceedings of SPIE AeroSense 2002: SPIE’s 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, 01-05 April 2002, Orlando, Florida (2002)
- [28] Christopher Landauer, Kirstie L. Bellman, “Self-Modeling Systems”, (to appear) in R. Laddaga, H. Shrobe (eds.), “Self-Adaptive Software”, Springer Lecture Notes in Computer Science (2002)
- [29] Pattie Maes, D. Nardi (eds.), *Meta-Level Architectures and Reflection, Proceedings of the Workshop on Meta-Level Architectures and Reflection*, 27-30 October 1986, Alghero, Italy, North-Holland (1988)
- [30] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press (1986)
- [31] Mary Shaw, William A. Wulf, “Tyrannical Languages *still* Preempt System Design”, pp. 200-211 in *Proceedings of ICCL’92: The 1992 International Conference on Computer Languages*, 20-23 April 1992, Oakland, California (1992); includes and comments on Mary Shaw, William A. Wulf, “Toward Relaxing Assumptions in Languages and their Implementations”, *ACM SIGPLAN Notices*, Volume 15, No. 3, pp. 45-51 (March 1980)
- [32] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., Translated by R. Bartels, W. Gautschi, C. Witzgall, Springer (1993)
- [33] Donald O. Walter, Kirstie L. Bellman, “Some Issues in Model Integration”, pp. 249-254 in *Proceedings of EMC’90: The 1990 SCS Eastern MultiConference*, 23-26 April 1990, Nashville, Tennessee, Simulation Series, Volume 22(3), SCS (1990)