Spring 2009

On the Role of Assertions for Conceptual Modeling as Enablers of Composable Simulation Solutions

Robert Dennis King
Old Dominion University

ON THE ROLE OF ASSERTIONS FOR CONCEPTUAL MODELING AS ENABLERS OF COMPOSABLE SIMULATION SOLUTIONS

by

Robert Dennis King

B.S. June 1973, Marquette University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

MODELING AND SIMULATION

OLD DOMINION UNIVERSITY
May 2009

Approved by:

_____
Andreas Tolk (Director)

_____
Ghaith Rabadi (Member)

_____
John Sokolowski (Member)

_____
Levent Yilmaz (Member)

ABSTRACT

ON THE ROLE OF ASSERTIONS FOR CONCEPTUAL MODELING
AS ENABLERS OF COMPOSABLE SIMULATION SOLUTIONS

Robert Dennis King
Old Dominion University, 2009
Director: Dr. Andreas Tolk

This research provides a much needed systematic review of the roles that assertions play in model composability and simulation interoperability. In doing so, this research contributes a partial solution to one of the problems of model composability and simulation interoperability—namely, why do simulation systems fail to achieve the maximum level of interoperability possible? It demonstrates the importance of the assertions that are made during model development and simulation implementation, particularly as they reflect the unique viewpoint of each developer or user. It hypothesizes that it is possible to detect composability conflicts by means of a four-step process developed by the author for capturing and comparing assertions. It demonstrates the process using a well understood example problem—the Falling Body Problem—developing a formal model of assertion, a strategy for assertion comparison, an inventory of forces, and a catalog of significant assertions that might be made for each term in the solution to the problem. Finally, it develops a software application to implement the strategy for comparing sets of assertions. The software successfully detects potential conflicts between ontologies that were otherwise determined to be ontologically consistent, thus proving the hypothesis.

ACKNOWLEDGEMENTS

There are many people who have contributed to the successful completion of this dissertation. I extend many, many thanks to my committee members for their patience and hours of guidance on my research and editing of this manuscript. The concept of alignment of model-domain viewpoint can be attributed to conversations with James Muguira—the idea is his as much as it is mine. My colleague Chuck Turnitsa has been a resonant sounding board for ideas throughout the effort. The research grant support of Dr. Roland Mielke, Graduate Program Director for Modeling and Simulation is especially appreciated. Finally, the untiring efforts of my advisor, Dr. Tolk, deserve special recognition.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The Modeling and Simulation community generally recognizes that the issues of composability (models that fail to compose) and interoperability (simulations that fail to interoperate) represent unsolved problems, but there is not a consensus in how best to solve them. Among the specific problems is that undetected conflicts can exist between components that result in hidden, unintended behaviors. There is a need, then, for a system or method to detect these conflicts. Furthermore, interoperability of systems requires composability not only to ensure correctness but more importantly to permit software agents to reason about model concepts in an unambiguous, machine understandable form. Therefore, one of the goals of this research is to contribute a method for standardized representation and use of assertions so that a conceptual model can be annotated with a list of critical assertions that the system relies upon. If this is achievable, then it becomes possible to create software agents that detect mismatches in conceptual models (at least with respect to the listed assertions).

This dissertation documents the author's research into the causes of interoperability problems and into the requirements for achieving real world model composability. The remainder of this section frames the research question. Section 2 reviews the pertinent literature and demonstrates that this research is a logical extension of accepted work. Section 3 provides the theoretical context for reasoning about assertions. Section 4 presents a framework developed by the author to capture and compare sets of assertions. Section 5 applies the framework process to a well understood example problem. Section 6 demonstrates conflict detection using the process and presents experimental results for validation. Section 7 discusses the results, implications with respect to the science of modeling and simulation, and topics for future research.

1.1 Background

In many instances, the conflicts that prevent interoperation can be traced to a failure to capture and communicate the details of assertions (modeling decisions) made at all stages of development. Surprisingly, comparatively little research has focused

Citation format for this manuscript is taken from the journal SIMULATION: Transactions of the Society for Modeling and Simulation International.

specifically on assertions and the roles they play in the development of models and simulations.

Among the reasons for little research is that assertions are so much a fundamental part of formulating a system solution that they are used in many different ways. To illustrate, consider various ways authors treat assumptions, which are a type of assertion:

- Some authors treat a list of assumptions as though it were a theory of the world. A list of assumptions does not constitute a theory; this way of listing assumptions is very likely to present an incomplete view.
- Some authors treat a list of assumptions as if it were a conceptual model. It is not, but a conceptual model should include a list of its assumptions.
- Some authors take a shotgun approach in constructing a list of assumptions, listing each one that occurs in their mind. This approach generally lacks organization and focus.
- Some authors correctly use a list of assumptions to specify restrictions on (and characteristics of) a problem solution.
- Assumptions are a fundamental part of every problem solution: a first step in problem solving is for the analyst to identify the problem's assumptions. Many assumptions remain hidden and unrecognized until a deliberate effort is made to identify them. Often it is the unrecognized assumption that prevents a good solution.

The term *assumption* is often used interchangeably with *assertion*. However, strictly speaking assertion is the more general concept. Assertions include not only assumptions, but also constraints, considerations, implemented considerations, and required computational competencies. Therefore, except during this introductory section, assertion will be the term that is used. Assertions are necessary for several reasons:

- Assertions reflect desired values that should be maintained throughout the solution.
- Assertions set limits to the problem and thus provide a framework within which to work. These limits might include constraints of possibility, economics, or some other desired narrowing.

- Assertions simplify the problem and make it more manageable by providing fewer things to consider and solve. A problem with no assertions is usually too general to handle.

All problems involve the interaction of domains that exist in the world. Domains exist either physically (e.g. people, vehicles, structures) or logically (e.g. data, processes). Every domain has a collection of assertions that define various aspects of that domain. One way to define a *domain expert* is to say he is a person who understands the implicit assertions in a domain.

A conceptual model always involves a particular viewpoint. This may be the viewpoint of the model developer, system integrator, federation member, verification team member, model results user, and so on. Some may argue that a conceptual model represents an intersection of viewpoints—a kind of common ground that practitioners can agree upon. This supports a notion that only the common elements between views should be incorporated into a conceptual model. Others take the opposite view—a conceptual model is the union of elements. The difficulty with the first position is that significant elements may be left out of the conceptual model because they lack common interest. The problem with the latter position is the amount of conceptual baggage that must be carried by all parties.

This research takes a third position—that different sets of assertions stand behind each practitioner's viewpoint and must be taken into account. To provide conceptual alignment between models is to align the assumptions and model constraints, thereby mediating between the multiple domain views. The difficulty of doing so varies. Within a small, specialized community of practitioners, the sharing of a common viewpoint is easier than between a large, diverse group.

Choosing specific modeling methods and parameters involves making many assertions, both explicit and implicit. These derive initially from the viewpoint and are later refined in a number of model development processes.

1.2 The Problem, Hypothesis and Solution Approach

To the extent that conflicting assertions are at the root of problems in creating composable models and interoperable systems, a partial solution is achieved by

developing a methodology for detecting conflicts. The author suggests a novel idea namely, if the assertions are adequately captured and listed for each component, then comparing the assertion lists can reveal potential conflicts between the model components. Accordingly, the author hypothesizes that:

It is possible to identify model component conflicts by comparison of lists of assertions made about the components, the system, the environment, and the stated problem that the model is to address.

To test the hypothesis, a sample problem (The Falling Body Problem) is analyzed to determine the assertions (assumptions, constraints, implemented considerations, and required computational competencies) made about each component. Assertions are captured using a formalism developed for the purpose by the author. Assertion lists for the basic problem solution are encoded in an ontology to enable reasoning about them. Alternative solutions to the problem are similarly analyzed, and those assertions are captured and encoded. Finally, the assertion lists between the alternatives are compared to determine where potential conflicts might arise if they were to be combined. A successful test result is the identification of a potential conflict by an automated process. To achieve this latter objective, a custom software application is required to perform the comparison.

2. RELEVANT RESEARCH

Several bodies of research are germane to this investigation. Literature in Composability, Conceptual Modeling, and Interoperability establish general problem context.

The literature on the topic of assumptions and, in particular, the roles that assumptions play in modeling and simulation, is relatively sparse. Perhaps this is because of the ubiquity of assumptions in problem solving—there is a temptation to take them for granted.

The Levels of Conceptual Interoperability Model (LCIM) provided the inspiration for development of Conceptual Linkage—thus it is one of the foundations for this work. The thorny part of conceptual linkage is handling assumptions and model constraints, and that is largely what set the research direction.

There has been a recent explosion of research into ontology, largely the result of development of the Semantic Web. Ontology is critical to this research because reasoning requires unambiguous definition of concepts, relations, functions, axioms and instances.

Finally, the well known frame problem in artificial intelligence has a direct bearing on this research.

2.1 General Context

There have been many efforts aimed at defining composability. Davis and Anderson define composability as the capability to select and assemble components in various combinations to satisfy specific user requirements meaningfully[1]. Achieving a composable system is not easy: in our imperfect world, when designing and creating models analysts decide what to ignore and what to include (as well as how to model what is included). Occasionally and often unpredictably, this process produces incompatibilities between models. Davis and Anderson discuss many factors governing why this is so and explain why complete elimination of conflicts may possibly be unachievable. They present many suggestions to enhance prospects for composability—among them is recognition that models are different from general software components,

and model composability needs to be based on the science of modeling and simulation. Petty, Weisel and Mielke [2;3] defined composability in a similar manner, excepting that they required that to be composable, only *valid* simulation systems result[1].

Page, Briggs and Tufarolo elaborated the definition suggested by Petty et al, noting that composability is more than just the ability to put simulations together from parts; it is the ability to combine and recombine, to configure and reconfigure, sets of parts from those available into different simulation systems to meet different needs[4]. They propose a framework for the broader simulation interconnection problem and suggest roles for composability, interoperability and integratability within that framework. They view these as three separate dimensions in the general simulation interconnection problem. They address objectives and assumptions in the proposed framework, suggesting that assumptions need to be studied and an algebra or calculus for composing models needs to be developed.

Robinson and others highlight the importance of capturing assumptions in the conceptual modeling process. Robinson's presents an analysis of the issues and research requirements for conceptual modeling for simulation [5]. Robinson notes that conceptual modeling is probably the most important aspect of a simulation study, and it is the most difficult and least understood [6]. There are several conceptual modeling guides that the analyst may choose to draw upon—Robinson outlines a framework for conceptual modeling [7], a practical example is offered by Borah [8], and detailed discussions may be found in [9] and [10;11].

2.2 The Roles and Importance of Assertions

Assertions have a potentially tremendous impact on alignment of model domain viewpoint, particularly those implicit assertions that are part of every domain of discourse. Assertions provide a framework for interpretation of the model domain viewpoint. Consideration of assertions is very often an afterthought or side issue in modeling, yet it *should be* at the foundation of model or system development. Several authors have considered the topic, but few have focused on the subject exclusively.

---

[1] Italics added.

Garlan, Allen and Ockerbloom used their experience building a family of software design environments from existing parts to illustrate a variety of types of mismatch that center around the assumptions a reusable part makes about the structure of the application in which is to appear [12]. They observed that the creators of the reused subsystems that were studied were neither lazy, stupid, nor malicious. Nor were the system integrators using the pieces in ways inappropriate to their advertised scope of applicability. Therefore, the root causes must lie at a deeper systemic level. Each of the packages that were used to construct the studied system made assumptions about the structure of the system and, in particular, the nature of the environment in which they were to operate. Virtually all of the serious problems were traced back to places where these assumptions were in conflict. They introduced the term *architectural mismatch* to describe the problem that stems from the mismatched assumptions a reusable part makes about the structure of the system it is to be part of. They note these assumptions often conflict with the assumptions of other parts and are almost always implicit, making them extremely difficult to analyze before building a system. Garlan et al shows how an architectural view of the mismatch problem exposes several fundamental challenges for software composition and suggests possible research avenues needed to solve them. The four main categories of architectural mismatch are:

- Assumptions about the nature of the components, including (1) infrastructure—assumptions about the substrate on which the component is built; (2) control model—assumptions about which component(s) (if any) control overall the sequencing of computations; (3) data model—assumptions about the way the environment will manipulate data managed by a component
- Assumptions about the nature of the connectors, including (1) protocols—assumptions about the patterns of interaction characterized by a connector; and (2) data model—assumptions about the kind of data that is communicated
- Assumptions about the global architectural structure, including assumptions about the topology of the system communications and about the presence or absence of particular components and connectors
- Assumptions about the construction process

Assumption-based Planning (ABP) is a concept developed at the Rand Corporation by Dewar, et al. [13] that provides an extensive framework for dealing with assumptions in decision making. ABP defines an *assumption* as an assertion about some characteristic of the future that underlies the current operations or plans of an organization. In ABP, the task is to identify those assumptions that are vulnerable to failure in the period of planning interest. An assumption is *load-bearing* if its negation would lead to significant changes in operations or plans. ABP identifies *signposts*—indicators of when assumptions are violated—and uses them as triggers for initiating alternative actions. ABP also provides a framework for planning actions that (a) protect or maintain the state of vulnerable assumptions, or (b) are contingencies in the case that vulnerable assumptions fail.

Hofmann [14] offered a definition of *assumption* that supports the formal framework of modeling and simulation presented in detail by Zeigler, Praehofer and Kim [15]. Hofmann discusses the critical influence of assumptions in reaching interoperability on the pragmatic and conceptual level. He considers models as epistemological tools for gaining knowledge about reality—many of which are based on simplifying and completing assumptions. He further notes assumptions are not empirically proven and within different epistemological paradigms assumptions play different roles. He concludes this leads to a rather pessimistic view on the possibilities of a priori validation of assumption based models.

The HLA 'Federation Development and Execution Process' (FEDEP) describes a structured, systems engineering approach to federation development and execution [16]. As a 'guide to best practices' the FEDEP falls short in that it mentions assumptions only twice, almost *en passant* in its manner, and in the most general terms. Describing only what the federation conceptual model must represent, it fails to address the constraints implied by user assumptions.

The NATO Code of Best Practices for C2 Assessment is the product of international collaboration among leading experts to capture the best practices in conducting operational assessments—that includes the use of modeling and simulation [17]. The importance of assumptions is recognized in several ways. Assumption providers are identified among the list of key assessment participants. The importance of capturing

assumptions is stressed repeatedly: defining assumptions is a key activity in Problem Formulation; the assessment team leader is advised to keep a journal of assumptions and decisions; assumptions are specific elements to be documented in associated supporting plans, and so forth.

Assumptions were examined by Spiegel et al [18], who conducted a small case study in order to clarify the role that model context plays in simulation composability and reusability. The research employed an example problem: compute the position and velocity of a falling body, which was described in detail by Davis and Anderson [1] in their monograph on modeling and composability. Spiegel and his colleagues found that a reasonable formulation of a solution included a surprising number of implicit assumptions—their non-exhaustive list included twenty-nine constraints. They observed that failure to appreciate the importance of various constraints when selecting a model can lead to unacceptable results. Several assumptions, such as *special relativity (assumed not to be significant)* and *Coriolis Effect (can be ignored)*, were not obvious[2]. Moreover, Spiegel et al. caveat their work, noting that while it may be that their formulation for the falling body is a suitable approximation for a golf ball[3] or cannon ball in flight, the decision should be made knowledgeably by a domain expert. Such a decision can only be made if the assumptions associated with each model are identified and understood.

King and Turnitsa [19] examined how assumptions are used in modeling and simulation and presented:

- A taxonomy of assumption characteristics
- An ontology of assumption
- A formalism for expressing assumptions in logic
- A strategy for comparing assumptions lists between system components

---

[2] Even so, given any one of Spiegel's listed assumptions, a competent engineer or physicist should be able to construct an example where taking it into account is critical.

[3] In fact, the falling body formulation is ***not*** suitable for golf ball trajectory prediction due to an assumption of perfect smoothness—the United States Golf Association publishes a Conforming Golf Ball list that specifies which balls are legal for tournament play based on the number and size of surface dimples on golf balls precisely because of the significant aerodynamic effects that these characteristics have on trajectory and distance.

Careful consideration of comments in [19] revealed that whilst capturing and comparing modeling assumptions is important, it does not provide all that is needed to align system components. Accordingly, this research extends the ideas behind the author's work on assumptions to encompass the more general case of modeling *assertions*.

2.3 The Levels of Conceptual Interoperability Model (LCIM)

Tolk and Muguira [20] describe the Levels of Conceptual Interoperability Model (LCIM) to identify various levels of interoperability between two systems ranging from no interoperability to full interoperability. Hofmann [14] and Turnitsa [21] extend the LCIM to its current form. The LCIM is a maturity model for interoperation—the higher the level achieved the greater the expectation of successful interoperation between elements. The hierarchical nature of the LCIM facilitates the process of aligning models by organizing concepts into dependent layers. The lower LCIM levels, *Technical* and *Syntactic* interoperability, deal with communication infrastructure and data protocols. Having a common term definition that results in unambiguously exchanging data largely satisfies the *Semantic* level. Reaching the *Pragmatic* level requires exchange of data context. At the *Dynamic* level, interoperating systems comprehend state changes that occur in the assumptions and constraints that each other are making over time—essentially allowing the unambiguous exchange of information. To accomplish the highest level, *Conceptual* interoperability, interoperating systems must not only understand the concepts, assumptions, and relations that are particular to each other, but must align their models and processes as well. This requires that conceptual models be fully documented based on engineering methods enabling their interpretation and evaluation by other engineers. In other words, a "fully specified but implementation independent model" as stipulated in Davis and Anderson [1] is needed, and not just a text describing the conceptual idea.

Within each layer, the interoperability concept addressed can be further broken down in terms of its definitions, sub-concepts, processes and requirements. In this manner, the necessary elements for achieving a particular LCIM level can be listed. Figure 1 is adapted from a recent evaluation by Tolk et al of the state of the art for the contributions of selected simulation protocols and knowledge representation languages towards

satisfying the levels of the LCIM [22]. For each protocol, the density of the square indicates the relative degree of support for the indicated level. As can be seen, the study reported a general lack of support for achieving the highest LCIM level, Conceptual Interoperability. Using the same evaluation criteria as the study, Tolk et al [23] first adds evaluations of the potential contributions of model-based data engineering (MBDE) and process engineering (PE). Even with these there is difficulty in reaching conceptual interoperability. The final column represents the addition of conceptual linkage (CL).

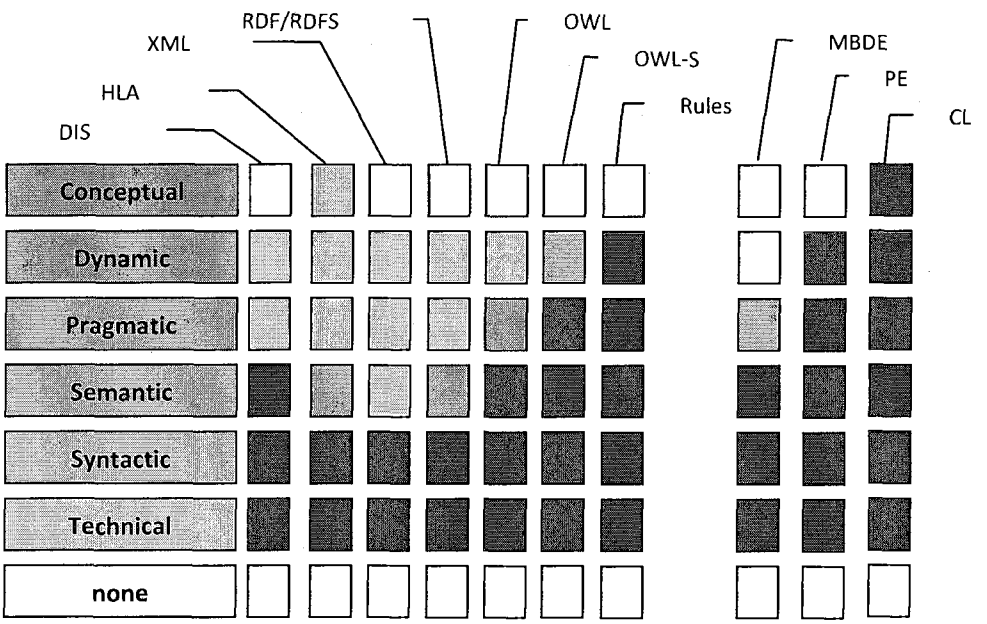


Figure 1. LCIM Contributions of Various Protocols

2.4 The Elusiveness of LCIM Level 6

King, et al [24] identify a failure to capture and communicate the details of conceptual modeling decisions to be the root of model interoperation conflicts. For example, design decisions made during implementation can become undocumented changes to the conceptual model. As a result, not all aspects of the conceptual model, its specified model, and modeling artifacts of the implementation are captured. Thus, when it becomes time to integrate models at the very least, there will be some conflicts between them—owing to the failure to capture conceptual model details fully. The effects can

range from very benign (and unnoticed) to catastrophic. See Pace [11] for discussions of the consequences of failures in conceptual modeling as it relates to system architecture.

The author also documented the requirements for achieving conceptual interoperability [25]. He demonstrated that when linking models or simulations, even the most complete description of the data exchanged between systems does not permit composition that guarantees the absence of emergent behaviors or structural variances. To define the problem better, the author coined the term ***functional composability*** to denote the situation wherein the outputs of one model become the inputs to another *without ambiguity or unintended effect*. This satisfies the requirement for validity advocated by Petty and Weisel.

Components can be functionally composed as long as they result in an engineering model as defined by Foo [26]. Foo discusses the frame problem that has occupied the attention of AI researchers in the logic of action. The frame problem is the challenge of representing the effects of action without having to represent explicitly a large number of intuitively obvious non-effects. To many philosophers, the AI researchers' frame problem represents a wider epistemological issue, namely whether it is possible, in principle, to limit the scope of the reasoning required to derive the consequences of an action (for more on the topic, see [27]). Engineers who model dynamic systems often consider the frame problem to be an artifact of logic. Foo clarifies the main issues: an engineering model does not (generally) suffer from the frame problem because of implicit assumptions, generally known as the inertia rule[4], made as a fundamental component of the problem statement and solution. The inertia rule is the assumption that effects are *local*[5] unless otherwise stated. A composition fails to produce an engineering model when effects are *not* local as assumed. Put another way, to show that a situation prevents a valid composition, it is sufficient to show the possibility of unintended effects. This is the basis for the arguments presented in the next subsection.

---

[4] Sometimes referred to in the philosophical literature as the *common sense inertia rule*.
[5] In the sense that most actions only have local effects—e.g. moving a cup does not normally change its color. *(NB—but moving it into a pot of paint does!)*

2.4.1 Barriers to Functional Composability

To date, the author has identified five activities (interaction, evolution, infinity, transformation, and conceptual model misalignment) that can act as barriers to prevent functional composition. The first three cases derive from work by Eberbach et al [28], who discuss new models of computation that are more appropriate for today's interactive, networked, and embedded computing systems. The latter two cases derive from the author's analysis, and discussions between him and his colleagues at the Virginia Modeling Analysis and Simulation Center. The discussion that follows presents these, along with arguments why each case can result in an inertia rule-related failure. Regrettably, detailed proofs require considerably more space than is available to this summary, but the arguments can be viewed as outlines of proofs. Furthermore, the list is preliminary and may be added to by future research.

Interaction. Interaction can involve either human input or decisions by agents, *during* the process of executing a model or simulation, rather than *before* or *after* it. Examples include the various Semi-Automated Force (SAF) simulations (e.g. JSAF, ONESAF), in which a human operator can interact directly with the running simulation to alter command and control behaviors, sensors, logistics, weapons effects, and entities' reactions to various combat stimuli. The Joint Forces Command (JFCOM) Experimentation Directorate, J9, makes extensive use of JSAF for Human-in-the-Loop, virtual experiments.

Proof outline: When human interaction is involved, it is impossible to enumerate all states and state transitions possible by the human mind. It is equally impossible to enumerate all possible actions of a person in a system[6], or the number of variables that affect the person's action. Finally, it is impossible to state all assumptions related to the infinite number of states, transitions, actions and variables[7]. Thus, there are infinitely many possible inertia rule failures.

---

[6] Although the number of actions that are *valid* may be limited by the system.

[7] It may be argued whether this capability should be extended to decision making by software agents or not—much depends on the agents' sophistication. Strong arguments can be made that interactive agents that learn or evolve can produce an infinity of responses.

Evolution of System. Evolution of system involves cases where the *architecture* of the system can be altered during the process of executing a model or simulation. Examples include the use of genetic algorithms, neural networks, run-time selection of services and components, and learning systems.

Proof outline: During system evolution, the many possible connections between system components are subject to change—meaning that the system's behavior is not fixed. Thus, the system can generate new states and paths. Consequently, it is possible that the system enters unknown states (and additionally via unknown paths); therefore, it is impossible to enumerate all of the assumptions necessary to achieve an engineering model. This produces infinitely many possible inertia rule failures.

Infinity of states. An infinity of states results from having infinite memory, using infinite precision, or having infinite time to solve. Examples include using massively parallel scalable computers or the Internet, or computing problems that are expected never to halt.

Proof outline: Each of these results in an extension by infinity to functional computation and means that it is impossible to enumerate all possible states. Therefore, it is impossible to list all of the assumptions needed, which in turn produces infinitely many possible inertia rule failures.

Transformation. Transformation occurs whenever the context of information is altered because of translation, filtering, aggregation or modification for transmission. Examples include federations that use a simulation protocol (such as HLA, DIS, ALSP) that requires aggregation or discretization of data spatially, temporally, logically, or in some other dimension[8].

Proof outline: Many transformations produce a loss of contextual information that ultimately leads to a **loss of frame assumptions**. Thus, even when the original contributing systems are completely defined with no implicit assumptions, the transformation process can destroy functional composability. The key issue here is that transformation processes can (and often do) cause a loss of information, not whether it is

---

[8] A specific example is the well-known problem of aggregating individual entities into a single military unit by one simulation, and disaggregating the unit into components by another.

possible to make lossless translations. Each assumption veiled by transformation processes becomes a possible inertia rule failure.

Conceptual Model Misalignment. Conceptual model misalignment arises when, despite best efforts, not all aspects of the conceptual model, its specified model, and modeling artifacts of the implementation are captured, whether any problems caused are detected or not. Misalignment can result from the use of legacy applications as well as from incomplete specification. Note that application of iterative development paradigms to modeling and simulation can exacerbate these problems. Iterative development paradigms vary considerably in the number and type of iterations involved. Some paradigms, such as Incremental, Spiral and Evolutionary Development, *depend* on conceptual model refinement. Even so, there is no guarantee that the conceptual model is faithfully updated.

Proof outline: saying that conceptual models are misaligned is akin to **defining** that components will fail to compose. The proof is in the definition. The reality is that this is one of the major sources of conflicts between model components. When integrating sophisticated systems, there are infinitely many permutations and subtleties of contextual meaning that potentially confound composability.

To summarize, each of these cases limit functional composition because they require invoking an inertia rule (e.g. making implicit assumptions) that *can* fail. They result in systems with states that are potentially unknowable, uncountable, unpredictable[9], or unaccountable[10].

No assertion is made as to the completeness of the list—it is doubtful whether an exhaustive list of cases can be constructed. Furthermore, it would be extremely difficult, if not impossible, to prove the completeness of such a list. Nevertheless, all that is necessary to join the list is to show that a situation has the potential to cause a composition not to be functional.

The consequence of these is that when combining models or simulations even the most complete description of the data exchanged between models does not necessarily permit composition that guarantees the absence of problems (e.g. as emergent behaviors

---

[9] i.e., the next state cannot be predicted
[10] i.e., all paths leading to a state cannot be predicted

or structural variances.) Rather, metadata that conveys important details of *how* the models accomplish their functions must be exchanged, parsed, and understood. This is the purpose of conceptual linkage.

2.4.2 Conceptual Linkage

Having identified that barriers to functional composability exist, the next step was to ask, "Is there a way to combine models to achieve conceptual interoperability to overcome the five barriers?" The author proposed ***conceptual linkage*** as a candidate solution and presented an initial list of conceptual linkage requirements [25]:

- Unambiguous meaning of terms and concepts—that is, a basis in ontology is vital
- Use of a supportive framework
- Functional composability of parts
- *Alignment of model domain viewpoint*[11]—the intention of the simulation developer, stated explicitly or derived implicitly, that objects and processes be represented in a certain way.

Model domain viewpoint is not the same as conceptual modeling—the concept builds upon conceptual modeling and extends it in a number of ways. Chiefly, alignment of model domain viewpoint is concerned in *identifying and resolving differences between the conceptual models of systems or components.* Alignment of viewpoint requires addressing assumptions about the model, system, and environment. To the extent that each model participant (e.g. user, developer, stakeholder, reviewer, and so on) has a unique world view, each also has a unique viewpoint of the conceptual model. Alignment is about mediating between these similar, but different, model domain viewpoints.

2.5 Ontology

To reason unambiguously about objects, characteristics, and processes the concepts used to describe them require unambiguous meaning. That is, a basis in ontology is critical. Ontology captures knowledge about a domain of interest. Beyond the terms used to describe and represent an area of knowledge (subject matter), ontology is the model (set of concepts) for the meaning of those terms. Ontology thus defines the vocabulary

---

[11] In previous papers, this concept was referred to as *Alignment of modeler's intent.*

and the meaning of that vocabulary within a domain. By encoding domain knowledge (i.e. its properties, values and concepts), ontology makes it possible to share and to reason about it.

There has been considerable research into ontology in the past decade, largely because of the role that ontology plays in providing formal descriptions for the Semantic Web of concepts, terms, and relationships within a given knowledge domain. Several important issues need to be reviewed with respect to how ontology is used in this research. These include the overall approach to ontology and the degree of modularization, the choice of knowledge representation language, the availability of tools for creating ontologies, and the suitability of existing ontologies to serve as a basis for constructing an experimental system.

2.5.1 Approach to Ontology

Wache, et al. [29], provide a survey of existing approaches to information integration. Figure 2 is adapted from this work and summarizes the three alternative approaches for employing ontology in complex systems:

Single ontology approaches use one global ontology to provide a specification of the semantics (see Figure 2a). The systems using the ontology must align themselves with it—which is not difficult if, for example, they are interoperating instances of the same model.

In *multiple ontology* approaches, each model is described by its own ontology (Figure 2b). It cannot be assumed that the different model ontologies share the same vocabulary. The advantage of multiple ontology approaches is that no common and minimal ontology commitment to a single, global ontology is needed—each model's ontology can be developed without respect to other models. While this architecture can simplify integration and supports adding and removing models, the lack of a common vocabulary makes it difficult to compare different ontologies and an additional representation defining the inter-ontology mappings is needed. Dealing with these mappings can be problematic as a domain that integrates n ontologies requires $n(n-1)$ mappings.

The drawbacks of the single or multiple ontology approaches can be overcome by the use of *hybrid ontology* approaches (Figure 2c). Similar to multiple ontology approaches the semantics of each model is described by its own ontology. However, to make the

local ontologies comparable to each other, they are built from a global shared vocabulary, which can itself be an ontology.

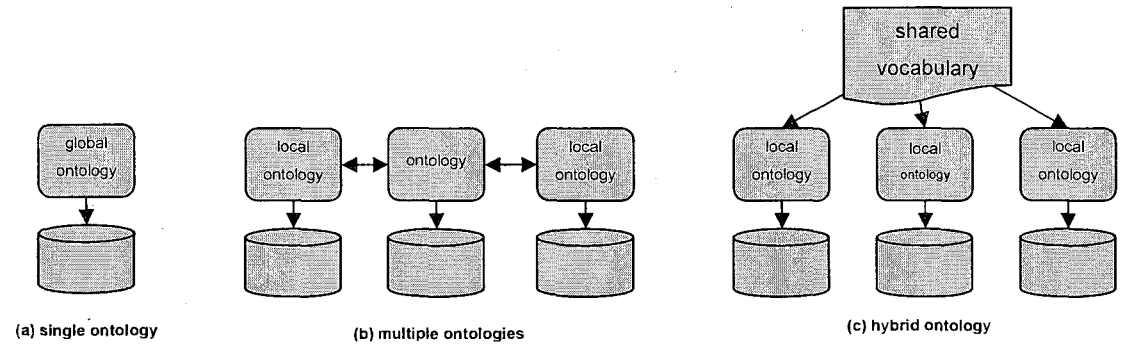


Figure 2. Ontology Alternatives

Wache also addresses the use of rule-based mediators to map knowledge sources to an integrated view through transformation rules [30]. One of the central issues presented in this work is identifying the context of the integrated view. Providing an ontological basis for context leads to consideration of another important issue in the approach to ontology—whether modularization is supported. A modular ontology is a set of individual descriptions of the same domain (e.g., Food) that represent correlated but not identical points of view of multiple observers or agents. Thus, each ontology module can be seen as describing a point of view held by an agent with respect to the entities (objects) and their relations in the domain. Bao, Caragea and Honavar offer precise definitions of semantic soundness such as localized semantics and exact reasoning and present expressivity requirements for modular ontology languages [31]. The importance of the semantic soundness and expressive power of several ontology languages is discussed in the following subsection.

2.5.2 Knowledge Representation Language, Tool, and Reasoners

Ontology languages are formal languages used to construct ontologies. They allow the encoding of knowledge about specific domains and often include reasoning rules that support the processing of that knowledge. Ontology languages are usually declarative languages, are usually generalizations of frame languages, and are commonly based on either first-order logic or on description logic. Examples of traditional ontology

languages—those based on first-order logic—are CycL [32], F-Logic (Frame Logic)[33], and KIF (Knowledge Interchange Format)[34].

Markup ontology languages use a markup scheme—most commonly XML—to encode knowledge. The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies that is endorsed by the World Wide Web Consortium [35]. This family of languages is comprised of three largely, but not entirely compatible sublanguages: OWL-Full, OWL-DL, and OWL-Lite. The OWL-DL and OWL-Lite sub-languages are based on description logic. Description logics (DLs) are a family of logics that are decidable fragments of first-order logic. DL has become a cornerstone of the Semantic Web for its use in the design of ontologies. See [36] for additional information on the OWL language.

The choice of knowledge engineering tool is important because in choosing a tool one chooses the particular language (or languages) supported by that tool. By choosing a tool, the user is also making a decision to use the particular logic system(s) supported by the tool. Over several decades a number of ontology authoring tools have been developed in both research laboratories and commercial endeavors. Ding [37] discusses ontology requirements in the context of the Web, compares several languages with existing knowledge representation formalisms, and surveys tools for managing and applying ontologies. An extensive review and summary of the characteristics and features of the tools available was conducted by Denny [38] who observed that the choice of tool/language depends on whether it affords the scalability necessary to implement the required ontologies and the degree to which it possesses the needed representational power or expressiveness.

Protégé [39] was chosen for this research for several reasons. To begin, it is free and supported by a large user community who ensure it is constantly being updated and improved. Perhaps most importantly, it has support for an embedded description logic reasoner.

A Description Logics reasoner is a software implementation of an inference engine whose purpose is to reason with a knowledge base expressed in OWL-DL. There are principally two reasoners available that have been integrated with knowledge engineering environments in general and Protégé in particular: Pellet and Fact++. Pellet [40] is an

OWL-DL reasoner originally created at the University of Maryland MIND Lab to support reasoning with individuals (including nominal support and conjunctive query), with user-defined data types and support for OWL/Rule hybrid reasoning. Fact++ [41] is a product of the University of Manchester School of Computer Science that is similar in capability to Pellet.

2.5.3 Evaluation of Existing Approaches

The prudent problem solver looks for previous solutions—even a partial answer to a problem may save time and effort. Additionally, a search often yields insights that result in a more efficient solution. This section reviews several noteworthy ontology development efforts. Ontology development efforts generally fall into one of two categories: top down or bottom up. Top down approaches seek to generate a more or less complete reference ontology that can be adapted to a particular problem. They seek to capture common information in one domain that can be used across several others. Three top-level development efforts are noteworthy: SUMO, MILO, and DOLCE.

WordNet. WordNet is a large lexical database of English, developed and now maintained by the Cognitive Science Laboratory of Princeton University [42]. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called *synsets* (that are uniquely numbered), each expressing a distinct concept. The lexicalized noun/verb concepts are organized hierarchically by means of hypernymy /hyponymy. As the most basic semantic relation, this organization serves to construct a hierarchy of the concepts in the domain and also provides a common way of reasoning for natural language processing researchers. However, there are several kinds of inappropriate hierarchy in WordNet [43], and its definitions can result in a degenerate structure. This prevents straightforward reasoning and eventually leads to errors (e.g. circular reasoning, a consequence of natural language processing). A second problem is that even though a particular installation of WordNet produces uniquely numbered synsets, the indices between different versions and even installations of the same version are different.

SUMO. The Suggested Upper Merged Ontology (SUMO) was developed within the IEEE Standard Upper Ontology Working Group [44]. First released in December 2000, SUMO was arbitrarily capped at around 1,000 concepts. SUMO originally concerned itself with meta-level concepts (general entities that do not belong to a specific problem

domain), and thereby would lead naturally to a categorization scheme for encyclopedias. Today, SUMO is at the apex of a collection of formal ontologies that include the MILO (MId-Level Ontology) and various domain ontologies. Together, these define a hierarchy of classes and related rules and relationships that is the largest formal public ontology in existence and is being used for research and applications in search, linguistics and reasoning. The difficulty with SUMO and MILO is the ad hoc manner in which the decisions to include individual concepts were made.

DOLCE. The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is a formal upper level ontology that aims to capture the ontological categories underlying natural language and human commonsense. Essentially, DOLCE is an ontology about concepts. Like the other top level ontologies, one can question whether DOLCE is a practical starting point.

A bottom up approach to developing an ontology seeks to capture the knowledge in a limited domain. There are many domain specific ontologies available on the World Wide Web, and several web sites are devoted to maintaining libraries of those accessible. A search through the libraries yielded several promising candidates discussed in the following paragraphs.

TSONT. Durak et al provide the trajectory simulation ontology (TSONT) that implements specific models for various aspects of missile trajectory [45]. The ontology is a thorough, detailed documentation of a narrowly defined problem. The work endeavors to capture most of the used missile trajectory algorithms and also includes models of seeking sensors and control methods. Despite the fact that is complete, the result it is highly dependent on 'standard' modeling approaches and, therefore, contains many implicit assumptions.

PHYSICS-PRIMITIVE. An ontology of physics concepts was prepared by the Dumontier Lab, but included only four forces: CentrifugalForce, Frictionalforce, GraitationalForce, and MagneticForce [46].

ONTOSENSOR. Russomanno, Kothari and Thomas devised, an ontology for sensor networks [47] that is noteworthy in that it extends the SUMO concepts.

Two final ontology-related works are worth mentioning. Collins and Clark [48] advocated an ontology of Physics as being necessary to achieve meaningful

interoperability between physics-based models. Although no specific ontology was generated by the work, the authors argued for standardized description of the physical laws governing physical objects. Finally, a series of articles by Hestenes et al [49;50] was aimed at capturing the conceptual structure of physics as part of organizing subject matter to be taught in physics courses. Their Force Concept Inventory proved invaluable in organizing the information in the Force Inventory that is presented in Section 5.3 below.

When considering whether to make use of an existing ontology, the analyst needs to be wary of introducing unwanted concepts. The conclusion to the search for existing ontologies was that it was necessary to develop the needed ontologies specifically for this research.

2.6 Assumptions and the Frame Problem in Artificial Intelligence

It should be clear to the reader that providing a consistent context for data is critically important. The Frame Problem was briefly introduced in the discussion in Section 2.4 to help explain functional composability and the barriers to functional composition. There are some additional aspects of the frame problem that are of interest to this research.

The frame problem originated as a narrowly defined technical problem of artificial intelligence (AI) researchers in the logic of action [51;52] . There are two forms of frame problem: computational and philosophical (epistemological). Both are germane to the use of assumptions in models. The computational frame problem is the challenge of representing the effects of actions on the properties of domain objects without having to represent explicitly a large number of intuitively obvious non-effects. Computational complexity is not the root of the genuine philosophical puzzle. The epistemological question is not so much how the computational challenge can be met, but rather how one could ever be sure they had sufficiently thought through the consequences of an action to know that nothing important has been missed.

Shanahan maintains that the frame problem, in its computational form, is more-or-less solved [53]. For simple systems, the most obvious way to address the problem is to add *frame axioms* that explicitly describe the non-effects of each action. However, in a domain comprising M actions and N properties this requires, in general, writing out almost MN frame axioms—clearly, the problem quickly becomes computationally

intractable. Approaches to the computational problem include fluent occlusion (circumspection or predicate completion), use of calculi specifically designed for solving the frame problem (fluent calculus, event calculus, successor state axioms, and so on), and default logic. The default logic solution relies on the assumption known as the *common sense law of inertia,* which declares the general rule-of-thumb that an action can be assumed ***not*** to change a given property of a situation *unless* there is evidence to the contrary. However, each of the solutions to the computational problem exists in a restrictive context:

- Domain knowledge must be expressed in a representational formalism that is computationally decidable, such as propositional logic.
- The formalisms may tolerate incompleteness but not ambiguity.
- Note that if the solution requires the representational power of first-order predicate logic, then the epistemological issue remains separate.

The general composability problem in modeling and simulation must consider the epistemological frame problem. The method developed in the following sections rests on the fact that only certain properties of a situation are relevant in the context of any given action, and consideration of the action's consequences can be conveniently confined to those. This topic will be revisited at the end of the dissertation.

3. FOUNDATIONS FOR ALIGNING ASSERTIONS

This section discusses important concepts that must be defined before a process for capturing and aligning assertions can be presented. To begin, there are many ways to define and describe a system. Texts dealing with system analysis often take the view of a system performing a service that meets a need [54;55]. In this view, analysis begins with defining the customer's need or problem. Requirements analysis follows and is aimed at characterizing the performance of the system that consists of various interacting components to ensure it meets the customer's needs.

This is the context of this research. A *problem statement* represents a collection of requirements which are potentially satisfied by a *system solution* whose components are interacting objects and processes.

3.1 Assertion Concepts

In the course of defining a system solution to a problem, *assertions* are made that describe the properties of the components, connectors, architecture, or construction process of the system solution. Four kinds of assertions are made:

- **Assumptions** are statements taken for granted about the framework of a system solution. Assumptions can belong to the problem statement or to a solution alternative.
- **Constraints** are part of the problem statement and specify *required* properties of the solution.
- **Considerations** define *desired* or *alternative* properties and belong to candidate solutions to the problem[12]. **Implemented considerations** are specified properties of a particular solution to the problem. In this context, an implemented consideration is a fixed part of a particular solution under consideration.

---

[12] The distinction between a consideration and a constraint is subtle, but important. Inconsistencies between considerations affect the optimality of a solution; inconsistent constraints invalidate a solution.

- **Competencies** are proficiencies in performing specific tasks. Examples include mathematical operations such as vector sums, differentiation, and integration. A competency indicates the quality of being able to accomplish the described task that may itself be a combination of other tasks.

Figure 3 is a view of the process of aligning models that highlights the roles played by assertions. As shown, the components involved in the process are a model, a description of its inputs and output, a system that the model is being integrated into, and a description of the system input. To simplify the discussion we stipulate the model output has been aligned syntactically and semantically with system input. To determine compatibility, the process creates and compares lists of assertions between model and system.

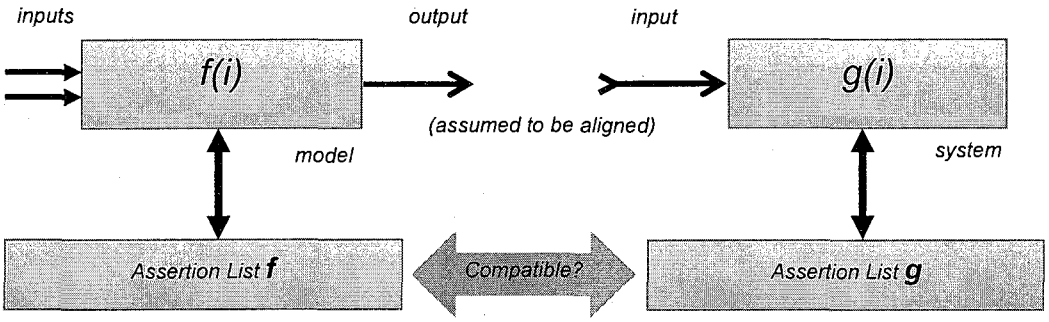


Figure 3. Overview of Comparing Assertions

To build and manipulate these lists requires an ontology to express the assertions, and a formalism for representing them. Figure 4 presents a formal model of assertion in terms of its components (*use function, referent, proposition, and scope).* The parts are described in the following paragraphs.

Use function. The use function describes the role of the assertion in potentially modifying system behavior. The value will be one of the terms in the following list: [uses | does not use | ignores | requires | denies]. The use function plays an important role in processing assertion lists. It establishes the relevance of the proposition with respect to the model and with respect to the role of the proposition when integrating the model with a system.

Referent. The referent of an assertion is the entity to which it refers. A referent can be an object, a model, a process, a data entity, a system, or a property of one of these. When an assertion acts as a constraint, the referent is what is being limited by the proposition.

Proposition. The proposition of an assertion is what it is saying – the statement that it is making. Propositions are not restricted to simple concepts—they may encompass the content expressed by theories, books, and even whole libraries.

Scope. Scope is an optional description that extends the portions of the overall system to which the assertion applies. A system is a collection of components (objects and processes), assembled for a purpose. The system components exist within an environment. Scope can limit consideration to a component, the environment, the system, or to combinations of these (e.g. *component-environment* scope means that the scope of assertion is the relationship between the component and its environment.) If scope is not specified, then the assertion has *component* scope. Finally, note that scope can be stated explicitly or implicitly.

Assertion <=> (referent useFN Proposition <scope>)

where:

referent is the model or system component that the assertion is about

useFN describes how the assertion is used by the model or system

(uses | does_not_use | requires | ignores | denies)

Proposition is a statement about the referent's existence, relations, or quality

scope is an optional description of which parts of the overall system that the assertion refers to

(component, system, environment, component-system, etc.)

Figure 4. Assertion Formalism

The reader will note that the formalism does depend on the *kind* of assertion (assumption, constraint, consideration or competency) introduced at the start of this section. The taxonomy was presented for the purpose of providing a complete as possible definition for assertion.

3.2 System Concepts

The concept of a system meeting a stated need is a common one in engineering, science, and modeling and simulation. These additional definitions help to refine the following system concepts:

Problem Statement. The problem statement describes a need that can be satisfied by a valid system solution subject to the restriction that the assertions of the system are compatible with the assertions of the problem statement.

System. A system is a group of independent but interrelated elements comprising a unified whole.

System Element. A system element is an independent component or process that is part of a system. A ***component*** is an artifact that is one of the individual parts of which a composite entity is made up, especially a part that can be separated from or attached to a system. A ***process*** is a particular course of action intended to achieve a result. Components and processes are interrelated: processes act on components, and components participate in and are changed by processes.

Environment. The environment is that which a system operates within.

Assertion Set. An assertion set is a collection of assertions about a system element, the system, the environment, or the problem statement.

Solution. A solution is a system that attempts to meet the requirements of a problem statement. Included in a solution are the sets of assertions that apply to the construction and use of its components and processes.

Valid solution. A solution is valid if, in addition to satisfying the need identified in the problem statement, the assertions sets of the system, system elements, environment and the problem statement are consistent. Conversely, an ***invalid solution*** is one where the need is not satisfied or where conflicts exist between the assertions contained in the assertion sets.

Having done the necessary groundwork, the concepts can be encoded and the relationships between them can be captured using an ontology development tool as shown in Figure 5 below.

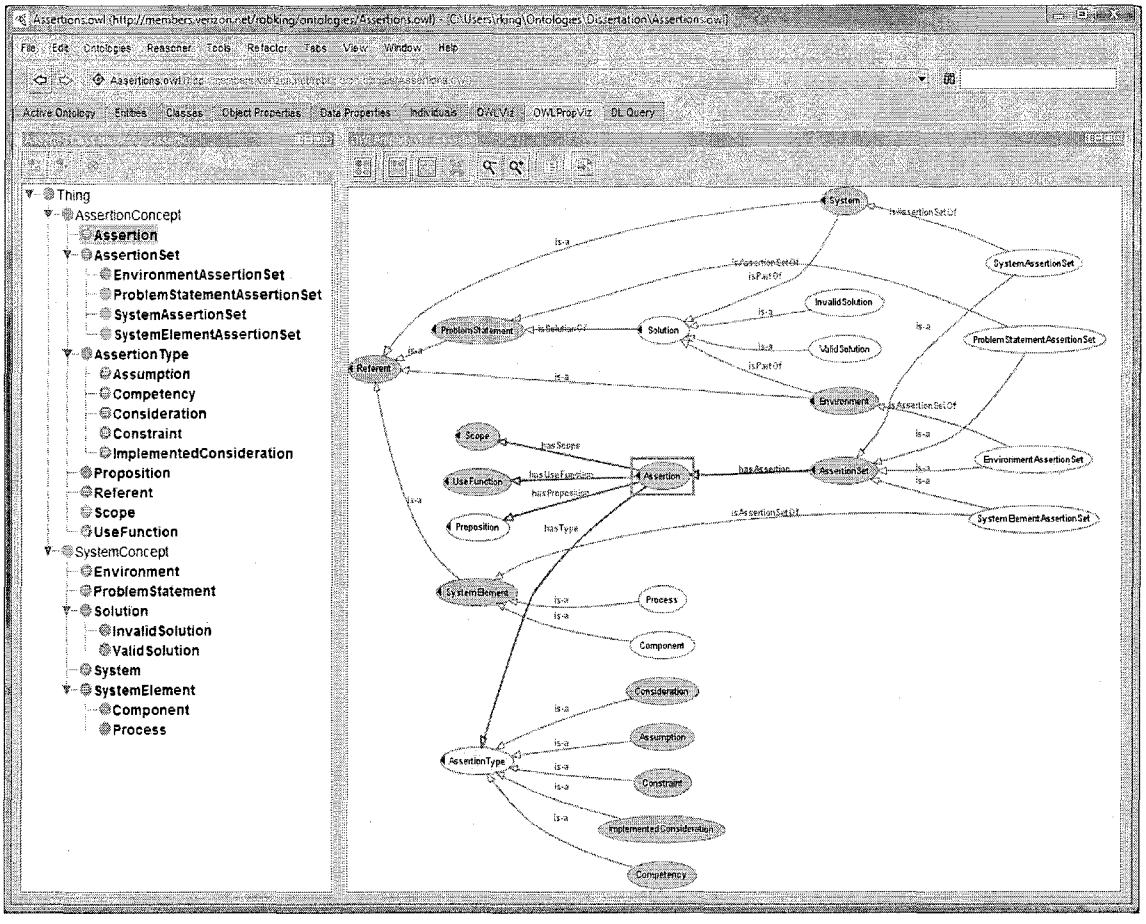


Figure 5. Ontology of Assertion and System Concepts in Protégé

3.3 Taxonomy of Assertion Properties

It is helpful, at this point, to identify some of common properties by which assertions can be classified. This is not intended to be an exhaustive list, and by its nature is not hierarchical.

Intension vs. Extension. Definition by intension is definition by giving all the criteria by which something is satisfied, thus defining a set. Extension is defining something by listing all of its examples[56]. The first (intension) yields a system under which something can be evaluated as being part of the set; the second (extension) yields a list of constituent members of the set. "The body will bounce off of anything solid. The body will bounce off the wall, the floor, and the obstacle. The first of these is an assertion defined by intension, and the second is defined by extension.

Primary vs. Derivative. Derivative assertions are derived from primary assertions. "There are no bodies, other than the ground, in the environment that the falling body is falling within". The derived assertion from this is that the effects of gravity are only applied by the ground as there are no other bodies to consider.

Load bearing vs. Non-load bearing. This refers to the importance of the correctness of an assertion with respect to the problem solution of which it is a part. If the assertion affects the overall behavior of the model or system in such a way that it cannot be ignored or accounted for, then it is considered to be load-bearing. If, on the other hand, the effects of the assertion (perhaps on the behavior of a single component, or on one relation between two components) can be accounted for or corrected within the final results, then it is considered to be non-load bearing.

Joint vs. Disjoint with others. This refers to whether the assertion acts independently or whether it operates in conjunction with other assertions. This raises the question of how they are joined. Boolean combinations are suitable if the assertion can be evaluated with respect to its truth. Other methods of combining, such as descriptive logics, may apply.

Exogenous vs. Endogenous. This refers to the source of the assertion. An exogenous assertion is one that comes from outside the system and is unaffected by the model. An endogenous assertion is one that originates from within the model and possibly affects it.

Dynamic vs. Static. This categorization addresses the question, "is the assertion fixed over the course of a solution?" The course of the solution may refer to temporal, spatial, or behavioral stability. This property of the assertion can most likely be described by the type categorization characteristics.

Deterministic vs. Probabilistic. It is likely that some assertions will always have the same, repeatable and measurable, effect on the model or system. These assertions are deterministic. On the other hand, there may be assertions that have some random factor, either intrinsic or representational (due to not knowing all of the affecting variables).

Controllable vs. Non-controllable. Some assertions might be able to be controlled but are worth being evaluated in a controlled state vs. a non-controlled state. Other assertions might not be able to be controlled. Consider, "The surface of the body is

covered in non-reflective paint. The falling body will not pass through a physical object." The first of these assertions is controllable; the second is not.

Explicit vs. Implicit. Explicit assertions are often stated in the description of the model or system. They are presented in such a way that they can be directly referred to. Implicit assertions often concern knowledge about the context or world that the model or system is expected to operate in. "The falling body starts at an at-rest condition. The laws of thermodynamics are in effect throughout the process of the body falling." The first of these assertions is most likely explicit and is stated in the description of the model. The second of these is implied, by the fact that it describes a physical process existing in a 3D world with normal laws of physics.

Like the taxonomy of the *kinds* of assertion, the taxonomy of *properties* is presented primarily to provide a more complete understanding of the topic of assertions.

4. A PROCESS TO CAPTURE AND COMPARE ASSERTION SETS

This section outlines a process for capturing and comparing sets of assertions. It is based on the formalism developed in the previous section. With this process, the assertions that define and shape each system component are examined to determine their compatibility with each other. To ensure success, the procedure described below should be performed in collaboration with both a domain expert and an ontology engineer. The process is greatly simplified if the analyst has a well documented conceptual model in hand. If one is not available, it is probably worth the effort to create one although strictly speaking a conceptual model is not absolutely necessary. The benefits to having a conceptual model to use as a guide are increased accuracy and completeness of the assertion sets and less time spent in capturing them. These benefits should outweigh the cost of capturing and documenting a conceptual model.

4.1 Preliminary Work – Capture the Conceptual Model

Much valuable insight into the process of developing a conceptual model can be gained from Robinson's framework for conceptual modeling [7]. The framework consists of five iterative activities: understanding the problem situation, determining the modeling and general project objectives, identifying the model outputs, identifying the model inputs, and determining the model content. The importance of assumptions is recognized as is the need for capturing them. Note that Robinson distinguishes between assumptions (about the problem) and simplifications (to the problem). In the context of this research, however, simplifications can be thought of as implemented considerations. The product of Robinson's framework is a well-documented conceptual model.

4.2 Step 1 – Capture Assertions

The first step is to capture the assertion propositions (assumptions, constraints, implemented considerations and competencies) for the model, system and environment. Each proposition represents a concept that is expressed as a natural language statement about the problem, one or more of its components, or a particular solution. The objective

is to write down what the main concepts are as this will form the basis of the ontology content. It is not necessary to document absolutely everything, just the things that are known to be within scope or that are important.

Several factors will determine how much information is gathered in this step. In the initial stages of system development, few assertions may have been made and "place holders" may represent model components whose characteristics are nebulous at the start. In later stages of development, and in particular when capturing the assertions of legacy models and systems, a wide variety of data may be available in the form of design documents—sifting through these to extract only important information may prove challenging.

The analyst will need to pay attention to the difference between core and secondary concepts. Core concepts are those terms (usually nouns) that are central to the model or system—their absence would result in an incomplete description of the domain. Secondary concepts are those that are not central to the domain but are required to complete the definition of core concepts. Obviously, core concepts should be documented thoroughly whilst secondary concepts need receive only limited detail.

When evaluating a collection of alternatives, the analyst will find some concepts that are common to all solutions, some that are shared among several and some that are unique to a particular solution. The analyst benefits from taking a close look at any algorithms used—there are likely to be several assertions for each term or factor in an equation. The analyst should also write down any competencies that appear in the problem.

4.3 Step 2 – Encode Propositions

The output of the first step is a list of propositions expressed in natural language statements. These must be encoded in a knowledge representation language (e.g., OWL-DL or KIF) before they can be used by a software agent such as a description logic reasoner.

Each proposition will consist of its axioms and logical assertions that relate it to other concepts and propositions. The astute reader will recognize this process as being identical with building an ontology. This will require, as a minimum, use of an ontology editor or ontology-building tool such as Protégé.

This step of the process reduces the likelihood of ambiguity by converting a natural language model into a formal model. A formal model provides strict interpretations of what the relations between items are meant to be. In a formal model, relationships are spelled out explicitly; even the fact there might be several words for the same thing should be represented explicitly. Note also that precision comes at a cost, and the overall product is only as good as this process step.

The analysis may benefit from a search for existing ontologies to determine if core concepts have already been defined. If an ontology is found, special care should be taken to ensure the semantic description in the existing ontology matches that desired of the core concepts. The structure of the ontology is potentially critical to the subsequent processing steps. As long as there are no conflicting statements, the ontology engineer can consider reusing the existing ontology.

It is possible (indeed likely) some propositions are found that encapsulate others. For example, the use of Newton's second law encapsulates the concepts of force, mass and acceleration (which depend on the concepts of position and velocity). At this point, the domain expert and ontology engineer have an important trade off to consider, that of complexity vs. computability.

The second process step also consists of assigning the use function, referent and scope to each proposition in both the model and the system lists. This establishes the relevance and use of each proposition. The analyst should be prepared to make several iterations through this process step as the assertion lists are refined.

The output of this step is list of statements encoded in a knowledge representation language—the list of assertions for the both component and system.

4.4 Step 3 – Compare Assertion Lists

The third process step is to perform a comparison of the model and system lists. The task of comparing lists requires a multi-level strategy to be effective.

4.4.1 Scan Strategy

The multi-level comparison strategy can be illustrated by examining alternative models of a phenomena—wind provides a good example. Consider the integration of a

model for wind effects on a body ***f*** into an existing model system ***g***. Before addressing questions of *how* wind is modeled, it must first be asked if wind *is* a factor.

Table 1 shows the strategy for comparing assertions about ***f*** with assertions about ***g***. The first objective is to determine whether or not ***g*** uses the assertion in model ***f*** that wind affects the body. The upper half of the table addresses the first strategy level (use, or non-use, of each proposition in ***f***). This is accomplished by a straightforward search for the assertion's proposition in the assertion sets. Note that **does_not_use** is the result of not finding the proposition during the search for it. The right hand table column lists the action to be taken as a consequence of having or not finding a match. Examining the table, the reader sees that two cases satisfy the first level comparison; either both model and system use the proposition or neither of them do. The other two cases—where one component uses the proposition and the other does not—represent potential conflicts. This situation is signaled by raising an alert. An alert indicates an abnormal condition—in a real-world implementation of the strategy, raising one would typically initiate another layer of detailed processing to determine the extent of the conflict.

The second objective is to take appropriate action based on *how* both ***g*** and ***f*** make use of the proposition. The lower half of the table addresses this level and represents situations where the model is interested in how the system behaves with respect to the proposition. Returning to the example, consider the case where the model ***f*** **requires** the use of wind. The comparison is successful if ***g*** either **uses** or **requires** wind. The other three situations, ***g*** **does_not_use**, ***g*** **ignores,** and ***g*** **denies**, are clearly in conflict. As above, the raising of an alert signals the need for further examination of the conflict. Similar reasoning applies to the cases where ***f*** **ignores** or **denies** the use of wind.

Note that the table lists all of the possible permutations of (0,1,i,r,d) between ***f*** and ***g*** and thus is complete. Also, note the table addresses the strategy only with respect to a search for propositions that match; it does not consider how the decision is made whether or not the propositions match. This is discussed in the next section.

Table 1. Strategy for Assertion List Comparisons, Based on useFunction Values

useFN of model *f*	Example assertion ***Interpretation***	useFn of the matching assertion in system **g**
uses = 1	body uses (wind affects) ***The body is affected by wind***	f g 0 0 – don't care: **OK** 1 0 – not found in g: **raise alert** 0 1 – not found in f: **raise alert** 1 1 – found in both: **OK (aligned)**
does_not_use = 0	body does_not_use (wind affects) ***The model does not use the effects of wind on the body***	
ignores = i	body ignores (wind *) ***The model is not affected by whether wind is considered or not***	f g i 0, 1, i, d – don't care: **OK** i r – ignores-requires conflict: **raise alert**
requires = r	body requires (wind *) ***The model requires that wind on the body to be taken into account***	r 1, r – uses or requires: **OK (aligned)** r 0 – requires & not found: **raise alert** r i – requires & ignores: **raise alert** r d– requires & denies: **raise alert**
denies = d	body denies (wind *) ***The model requires that the wind on the body not be taken into account***	d 0, i, d – not found, ignores or denies: **OK** d 1 – denies & found: **raise alert** d r – denies & requires: **raise alert**

4.4.2 Types of Matches

When comparing assertion propositions, it is important to note each proposition represents a concept and there are different ways that concepts can match. The topic of semantic similarity—deciding if, and how closely concepts match—is the subject of much current study, particularly with respect to research into the Semantic Web. The issue is a complex process influenced by many different factors or characteristics. In a recent analysis, Kokla observed that category comparison consists of the identification of similarities and heterogeneities between similar categories [57]. This process relies on available elements, which describe categories' semantics, such as terms and definitions. Different combinations of terms lead to four possible comparison cases:

- equivalence, when the categories are identical in meaning
- subsumption (partial equivalence), when one category has broader meaning than the other
- overlap (inexact equivalence), when categories have similar, but not precisely identical meanings.

- difference (non-equivalence), when the categories have different meanings

Equivalence. The first level, equivalence, is the result of a search for concepts that match exactly. This is most likely to happen when the concepts are described in a limited, controlled vocabulary. For the general case rarely are the *descriptions* of two concepts exactly the same—especially if the descriptions are written by different authors. Thus, whilst exact matches are the most powerful in terms of stating that concepts are identical, it is more likely two concepts are similar but slightly different. In this case more sophisticated comparison methods are needed.

Subsumption. The next level of concept matching depends on logical inference. For example, if the model has a particular assertion whose parts are individually asserted by the system, then the assertion is satisfied. The question here is, given two propositions A and B, what are the cases where both A and B can pertain? To illustrate, consider two propositions, A and B. Table 2 lists the cases where both can pertain. The right hand column indicates whether the information in the relationship is sufficient to permit a descriptive logic reasoner to infer the equivalency of assertions.

Table 2. Logical Cases for Matching Propositions

Relationship	symbolic logic	Usable?
A and B are equivalent	$A \equiv B$	Yes
A and B are independent, with no common cause, both pertain	A B	No
Causal relationship: A causes B and A pertains, or B causes A and B pertains	$((A \supset B) \wedge A)$, or $((B \supset A) \wedge B)$	Yes
Common cause: A and B are independent with common cause C that pertains	$(C \supset A) \wedge (C \supset B) \wedge C$	Yes
Set theory: A and B are both elements or subsets of C, and C pertains	$(A \in C) \wedge (B \in C) \wedge C$, or $(A \subset C) \wedge (B \subset C) \wedge C$	Yes

Subsumption can be a useful and powerful method for determining if propositions match. Consider the situation where a model f has implemented part of Classical Mechanics—Newton's second law and universal gravitation—and the model is being

integrated with a system ***g*** that uses all of Classical Mechanics. Table 3 presents examples from the point of view of matching propositions about ***f*** with ***g***.

Table 3. Examples of Subsumption Matches

Example proposition in model ***f***	**uses** is satisfied by these propositions from Classical Mechanics in system ***g***
the body has mass	the object has mass (equivalent) the object uses Newton's 2nd Law (mass ∈ Newton 2nd Law) the object uses the Universal Gravitation (mass ⊂ Universal Gravitation) the object uses the rules in Classical Mechanics ((mass ⊂ Universal Gravitation) ^ (Universal Gravitation ⊂ Classical Mechanics))
the body accelerates	the object uses Newton's 2nd Law (acceleration ⊂ Newton 2nd Law)

The scan strategy presented in the previous section remains sound using subsumption comparisons to determine if propositions match. However, attention must be paid to the details of the comparison. To begin, subsumption comparisons are asymmetric and transitive but not associative (equivalence comparisons are associative).

Another of the problems with using subsumption is the comparison depends on the structure of the ontology used to encode the propositions. To illustrate, consider the following propositions.

a. Each component consists of several elements.

b. Each of the elements can be comprised of several components.

The circular logic that is created by including these propositions in a system would make it impossible to perform a subsumption test. Another problem would surface when attempting to combine a system whose propositions rely on *a* and one whose propositions rely on *b*.

In summary, testing by subsumption can lead to complex issues that are beyond the scope of this dissertation.

Overlap. The third level of concept matching depends on semantic distance. Semantic distance[13] is a method that assigns a metric to a set of terms or concepts based on the likeness of their having similar meaning or semantic content. In essence, it asks, "How much does term *A* have to do with term *B*?" If the semantic distance is zero, it is said that propositions match, and no further processing is needed. For other cases, where there is either a partial match, a match in metalevel mappings, or possibly no match at all, a different kind of processing is needed. As with subsumption, determining the degree of overlap leads to complex issues beyond the scope of the dissertation. See [31] for an overview of methods, [58-60] for examples of ontology modularization (i.e. resolving overlap issues) and [61;62] for discussion of semantic matchmaking and intelligent brokering mechanisms.

Difference. Just as there can be varying degrees of overlap between concepts, there are also varying degrees of difference. This can be the consequence of any of several different issues. Two concepts can be antithetical in which case one denies the other, or each can exist in a non-intersecting domain of discourse. In the latter case, they simply have no effect on one another.

As used in this research, the term "conflict" refers to a logical contradiction in an axiom space. Logically speaking, "conflict" refers to situations in which two concepts seem to imply contradictory statements that result in confusion when aligning models. Furthermore, in logic a theory is consistent if it does not contain a contradiction. The lack of contradiction can be defined in either semantic or syntactic terms. The semantic definition states that a theory is consistent if it has a model; this is the sense used in traditional Aristotelian logic although in contemporary mathematical logic, the term satisfiable is used instead. The syntactic definition states that a theory is consistent if there is no formula P such that both P and its negation are provable from the axioms of the theory under its associated deductive system. Note that if these semantic and syntactic definitions are equivalent for a particular logic, the logic is complete. As above, determining a degree of difference leads the research to consider complex issues beyond the scope of the dissertation.

---

[13] The literature generally considers semantic similarity, semantic distance, and semantic relatedness to refer to the same idea.

The impact of this discussion is that because of the complexity of the issues involved, only an equivalence comparison will be used in the research. Moreover, because the ontologies developed comprise a controlled vocabulary, the concepts that underlie each proposition are not subject to misinterpretation. Hence, the research asserts that a proposition is adequately described by its label[14].

4.5 Step 4 – Adjudication and Resolution of Conflicts

The final step in the process is examination of each conflict and judgment as to its significance. Of course, some process for resolving conflicts is needed as part of the assertion alignment process, but its details are not important. The primary focus of this research is on the process for detecting conflicts.

The kind of adjudication depends upon the use function associated with each proposition and whether the assertion is *load bearing* (i.e. its negation would lead to significant changes in system operation) or not. The intention is that for load bearing assertions that are relied upon, or cared about, the adjudication is performed by a human or an agent that supports a human. The results of the adjudication can be stored and recalled at future times to provide precedents to human operators. The results can also be used to train agent-based systems.

---

[14] If required, a comparison can be supplemented by a verbatim comparison of all descriptive comments attached to each proposition.

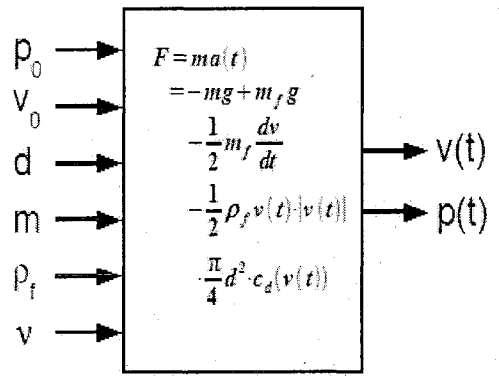
5. THE FALLING BODY EXAMPLE PROBLEM

The goal of this section is to present an example of the process by capturing assertions derived by detailed analysis of a problem and its solution. The falling body problem is chosen for that example because it is well defined (e.g. it is limited, widely understood and used) and thus it is amenable to rigorous analysis in terms of its components and their underlying assertions. Whilst it cannot be *guaranteed* that all of the assertions that could possibly pertain to the problem will be captured, there is a considerable likelihood that all of the *significant* assertions will be accounted for. This analysis benefits from previous consideration of the problem by several authors. The falling body problem was used as a general example by Davis and Anderson [1], and as a specific example for investigating the complexity of validation constraints by Spiegel [18]. Lastly, the falling body problem adopted in Spiegel *et al* appeared originally as an example in the first chapter of an engineering textbook by Chow [63].

Spiegel et al collected modeling assertions through expert review of a particular system solution to the problem. The completed model was presented in the form of equations and definitions shown in Figure 6 to the experiment participants who were then asked to list any validation constraints that they could think of[15]. The study subjects were engineering professors and graduate students, and so it is reasonable to conclude each possessed at least the minimum level of expertise necessary to understand and address the problem. Nevertheless, each subject had a particular point of view that influenced his or her approach to the problem. This is borne out by the fact that on average each respondent came up with less than half of the total number of constraints collected in the survey. The approach was a good one for the purposes of that study, but not necessarily the best one for studying modeling assertions in detail. An analogy can be made between the method by Spiegel *et al* and the process of documenting a legacy model as part of integrating it into a system—some aspects of the model were not (or will not) be

[15] Although the idea of compiling a list of validation constraints is very similar to compiling a list of assertions, the two are not precisely the same. However, they will be considered equivalent for the purposes of this research.

captured. To illustrate, none of the respondents in the Spiegel et al study identified assertions dealing with electromagnetic or electrostatic forces.



Gravity: The sphere experiences constant acceleration, $g \approx 9.8\ m/s^2$.

Buoyancy: mf = (1/6) πd3ρf against gravity.
Inertial drag: (1/2) mf v′ (t)

Viscous drag: (1/2)ρf • v(t) • | v(t) | • π/4 • d^2 •cd(v(t))

Wave drag: Wave drag is negligible at subsonic speeds.

Figure 6. Chow's Falling Body Problem as Presented by Spiegel et al

To capture its assertions as completely as possible, a thorough analysis of the falling body problem will be made. The problem is open to solution using a number of different methods. Classical mechanics is one approach that is familiar to most engineers and physicists, and it is chosen for that reason. Equivalent methods include Lagrangian mechanics and Hamiltonian mechanics[16]. Generally, these alternative formulations provide deeper insights into the general structure of classical mechanics and its connection to quantum mechanics as well as its connection to other areas of science.

5.1 Conceptual Model (Preliminary Work)

As previously mentioned, development of a conceptual model is an important activity—one was developed for this research. Table 4 presents the first iteration of Robinson's framework for conceptual modeling for the falling body problem.

---

[16] Lagrangian mechanics was introduced by Joseph Louis Lagrange in 1788. It combines conservation of momentum with conservation of energy. The trajectory of a system of particles is derived by solving Lagrange's equation for each of the system's generalized coordinates. Hamiltonian mechanics was discovered in 1833 by Irish mathematician William Rowan Hamilton. The Hamiltonian method differs from the Lagrangian in that instead of expressing second-order differential constraints on an n-dimensional coordinate space, it expresses first-order constraints on a 2n-dimensional phase space.

Table 4. Conceptual Model for the Falling Body Problem: First Iteration

Problem situation	Classical mechanics has been chosen for the form of the solution. Classical mechanics can be used for describing the motion of macroscopic objects, from projectiles to parts of machinery, as well as astronomical objects, such as spacecraft, planets, stars, and asteroids. It produces very accurate results within these domains, and is one of the oldest and largest subjects in science and technology. While the terms classical mechanics and Newtonian mechanics are usually considered equivalent (if relativity is excluded), much of the content of classical mechanics was created in the 18th and 19th centuries and extends considerably beyond (particularly in its use of analytical mathematics) the work of Newton. Newton's laws were verified by experiment and observation for over 200 years, and they are excellent approximations at the scales and speeds most often encountered. However, Newton's laws are inappropriate for use in certain circumstances, most notably at very small scales, very high speeds (in special relativity, the Lorentz factor must be included in the expression for momentum along with rest mass and velocity) or very strong gravitational fields.
Objectives	The primary research objective is to foster understanding of the processes that formulate and utilize modeling assertions. The consequence of this objective is in the way that it shapes the modeling process. Typically, the modeler focuses on the important aspects of the problem. If a concept is not in the modeler's thoughts or is not particularly important in the current worldview, then it is likely that it, and its assertions, will not be captured at all. In contrast, this research aims to capture as much detail as possible, and so emphasis is placed on examining the fundamental theories that lie behind Classical Mechanics.
Inputs	The inputs include the initial position and velocity of the body, its mass, dimensions, shape, and surface composition. Environmental inputs include the type of fluid (such as air, water, or hydraulic fluid), its density, temperature, pressure, and viscosity.
Outputs	The output of the model is the position and velocity of the body over time. Usually, an inventory of the assertions made would be considered an optional output product, but in this case, it is a primary one.
Content	The framework of classical mechanics includes a number of concepts, laws and theories. scope: The system consists of the falling body and the earth, the forces acting on each object, and the environment. level of detail: the solution seeks to capture only the most important effects. Secondary effects, such as determined through complicated aerodynamic calculations will be avoided. assumptions: assumptions about each system component will be discussed

Now that a conceptual model is in hand, we can proceed with the process of capturing assertions. It is good practice to begin by writing down the top-level assertions that deal with the definition of the problem and the general approach to a solution.

5.2 Initial Assertions (Step 1a)

The first step is to capture the assertion propositions for the model, system and environment in natural language statements. The content of the problem can be summarized by several initial assertions:

Initial Assertion IA_1: The system consists of the falling body and the earth, the forces acting on each object, and the environment.

Initial Assertion IA_2: The solution uses Classical Mechanics.

Initial Assertion IA_3: Classical Mechanics consists of the principle of superposition of forces, Newton's three laws of motion, the Law of Universal Gravitation, and conservation of energy, momentum, and angular momentum. [64]

Initial Assertion IA_4: The solution uses the principle of force superposition; i.e. the resultant force on the body is the vector sum of individual forces.

This assertion demonstrates an important point, namely that a proposition may subsume or depend on another proposition. The principle of force superposition permits summing the forces that act on a body and treating the resultant as a single force, if the forces act independently. In other words using the principle of force superposition implicitly invokes the assumption of independence of the forces involved. Use of the principle of force superposition also involves a competency in being able to perform vector sums.

Initial Assertion IA_5: The acceleration on the body derives from Newton's 2nd law of motion.

Newton's 2nd law is commonly written $\vec{F} = m\vec{a}$, or if mass is not constant[17] $\vec{F} = \frac{d}{dt}(m\vec{v})$. By examining the equation, it can be seen that the expression can be decomposed into a collection of individual propositions as listed in Table 5.

---

[17] For example, in rocketry $dm/dt \neq 0$.

Table 5. Propositions for Newton's 2nd Law

Proposition	Comments
N2_1: The body has mass	
N2_2: The body has position	
N2_3: Position changes when velocity is nonzero. Equivalently, it is the time integral of velocity $\int v\, dt$	This requires a competency in differential calculus, C_CALCULUS (includes both differential and integral calculus)
N2_4: Velocity is the rate of change of position with respect to time, defined as the time derivative of position (*d***p***/dt*), or equivalently as the time integral of acceleration $\int a\, dt$	This requires a competency in calculus, C_CALCULUS
N2_5: Acceleration is the rate of change of velocity with respect to time, defined as the time derivative of the velocity (*d***v***/dt*).	This requires a competency in calculus, C_CALCULUS
N2_6: The acceleration of a body is the result of force applied and is equal to that force divided by the body's mass. (***a*** *= F/m)*	This requires a competency in being able to use algebraic manipulation to obtain an equivalent representation, C_ALGEBRA
N2_7: Newton's 2nd law requires the caveat that it holds only in an *inertial frame*.	This is defined to be a frame in which a free particle with $\dot{m}$= 0 travels in a straight line, e.g. $\boldsymbol{r} = \boldsymbol{r}_0 + \boldsymbol{v}t$. Note that Newton's 1st law is the statement that such frames exist.

Note that proposition N2_6 is the statement of Newton's 2nd law and the other propositions operate in support of the law. This assertion illustrates another important point, namely a proposition may have several equivalent expressions—and one of those is the one that is needed. The alternative equivalent expressions may be listed explicitly or may be the result of the algebraic manipulation of the asserted algorithm.

A note on ontology engineering is also in order. The sequence with which the ontology engineer steps through the engineering analysis can have potentially great impact on the structure of the ontology. This can have important consequences during the comparison phase when concept matching can depend on ontology structure.

Lastly, some assertions are usually found together—although not dependent on one another, using one from the group *very often* means the others pertain even when not explicitly mentioned.

5.3 Force Inventory

Having specified the use of Newton's second law and the principle of force superposition, the next thing to be done is to list the forces that are involved. This section creates an inventory of possible forces. Once the inventory has been created, it is rather straightforward to indicate whether each force is utilized in the problem or not. Table 6 is a modest catalog of possible forces, organized by force type (how it is applied). A non-contact force is one that acts over a distance—there are four known non-contact forces in the universe. Solid-contact forces require that objects touch. Fluid-contact forces apply to motion through liquids and gasses. A fictitious force is an apparent force that acts in a non inertial frame of reference—it does not arise from any physical interaction but rather from the acceleration of the reference frame itself. Finally, several phenomena are labeled or thought of as forces, yet are not.

Table 6. Force Inventory

Force	Alternative representations	Comments (assertions in natural language)
Non-Contact Forces		
Gravitational	$\mathbf{F}_g = \sum F_{eb} = \sum_e -G\frac{m_e m_b}{\lvert\mathbf{r}_{eb}\rvert^2}\hat{\mathbf{r}}_{eb}$ $F_g = G\frac{(m_1 m_2)}{r^2}$ $F_g = -mg$	The first equation is the vector form: (the net force on body *b* is sum of contributions by mass elements *e*). *G* is the universal gravitational constant The second equation is the point mass scalar form The third equation is the Earth gravitational field form (force is down)
Strong nuclear	F_{strong} = (not presented)*	Force that holds the holds quarks and gluons together to form protons and neutrons: negligible at distances > 10^{-15} m
Weak nuclear	F_{weak} = (not presented)*	10^{-6} times weaker than strong nuclear force: negligible at distances > 10^{-18} m

Force	Alternative representations	Comments (assertions in natural language)
Electromagnetic Magnetic Electrostatic	$\mathbf{F}_{Lorentz} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$ $\mathbf{F}_{mag} = q(\mathbf{v} \times \mathbf{B})$ $\mathbf{F}_{el} = q\mathbf{E}$ $\mathbf{F}_{el} = \frac{q_1 q_2}{4\pi\varepsilon\lvert\mathbf{r}_{12}\rvert^2}\hat{\mathbf{r}}_{12}$	The first equation is the Lorentz (electromagnetic) force which is the combination of magnetic and electrostatic force effects q is the electric charge on the object, v is the object's velocity and B is the magnetic field The second equation is the electric field form: E is the electric field strength. The third equation is the vector form between charged particles: ε is the electrical permittivity of the medium and q_n is the electric charge on particle n.
Solid-Contact Forces		
Reaction	$\mathbf{F}_{contact} = -\mathbf{F}_{applied}$	Reaction forces arise from application of Newton's 3rd law.
Thrust	$\mathbf{F}_{thrust} = -\frac{d}{dt}(m\mathbf{v})$	Thrust is a reaction force that results when a system expels or accelerates mass (such as air) in one direction and experiences a force in the opposite direction.
Sliding Friction	$F_{friction\,s} = \begin{cases} \mu_s F_n \\ \mu_k F_n \end{cases}$	μ_s is for surfaces at rest relative to each other, μ_k is for surfaces in relative motion. Generally, $\mu_k < \mu_s$. F_n is the force normal to the surface. The direction of the force is directly opposite to the direction of motion.
Rolling Friction	$F_{friction\,r} = \frac{Wa}{r}$	W is the weight, a is the coefficient of rolling friction, r is the radius. The direction of the force is directly opposite to the direction of motion.
Spring	$F_{spring} = -k\Delta x$	This is a reaction force to the elastic deformation (change in length by Δx) of a spring. k is the spring constant. The direction of the force is directly opposite to the displacement.

Force	Alternative representations	Comments (assertions in natural language)
Damping	$F_{damper} = -cv$	c is the viscous damping coefficient. The direction of the force is directly opposite to the motion.
Deformation	F_{deform} = (not presented)*	Forces that resist the plastic deformation of objects.
Fluid-Contact Forces		
Buoyancy	$F_{buoy} = m_f g \quad m_f = \frac{4}{3}\pi r^3 \rho_f$	The formula is for a spherical object where m_f is the mass of the fluid displaced, ρ_f is the fluid density. The direction of the force is opposite to the direction of the gravity field.
Aerodynamic Lift Lift Induced Drag	$\boldsymbol{F}_{aero} = \boldsymbol{F}_{lift} + \boldsymbol{F}_{lift\_drag}$ $F_{lift\_sphere} = \frac{4}{3}(\pi^2 b^3 \omega \rho v)$ F_{lift_drag} = (not presented)*	Aerodynamic force is the resultant force on a body by a fluid (e.g. air) that is due to the relative motion between the body and the fluid. It is commonly resolved into two components: lift and lift-induced drag Lift is a mechanical force generated by the interaction and contact of a solid body with a fluid (liquid or gas). Lift calculations based on first principles are extremely complicated, except for simple shapes. Usually, mathematical expressions for lift approximate empirical data. The expression shown is the theoretical lift on a spinning sphere, where b is the radius of the ball, ω is the speed of rotation measured in revolutions per second, ρ is the density of air and v is its velocity.
Inertial Drag	$F_{inertial} = -\frac{1}{2} m_f \frac{dv}{dt}$	The force represents the change in inertia of m_f (the mass of the fluid displaced) that is the result of acceleration of a body immersed in it. The direction of the force is directly opposite to the acceleration.

Force	Alternative representations	Comments (assertions in natural language)
Parasitic Drag	$\boldsymbol{F}_{drag} = \begin{cases} -6\pi\eta r v\hat{\mathbf{v}} & R_e < 1 \\ -\frac{1}{2}\rho v^2 A C_d \hat{\mathbf{v}} & 1 \leq R_e \end{cases}$ (not presented for very large R_e)*	Drag calculations based on first principles are extremely complicated. Usually, mathematical expressions for drag approximate empirical data. R_e is the Reynolds number that indicates the degree to which flow around the object is laminar or turbulent. $\hat{\mathbf{v}}$ is a unit vector in the direction of motion. η is the coefficient of viscosity
Wave Drag	F_{wave} = (not presented)*	Caused by the formation of supersonic shock waves that radiate away considerable energy—experienced by the object as drag. Associated with supersonic flight, but can be seen at speeds of about Mach 0.8
Fictitious Forces (d'Alembert Forces)		
Rectilinear	$\mathbf{F}_{rect} = m\mathbf{a}_{rect}$	The apparent force due to an acceleration, a_{rect} ,of the reference frame origin in a straight line
Centrifugal	$\mathbf{F}_{centrifugal} = m\omega^2\mathbf{r}$	The apparent force acting outward from the axis of a rotating reference frame. The vector r is perpendicular to the center of rotation and points outward to the location of the rotating object.
Coriolis	$\mathbf{F}_{Coriolis} = -2m\boldsymbol{\Omega} \times \mathbf{v}$	Ω is the angular velocity vector which has magnitude equal to the rotation rate ω and is directed along the axis of rotation of the rotating reference frame, and v is the velocity of the particle in the rotating system.
Euler	$\mathbf{F}_{Euler} = -m\frac{d\boldsymbol{\omega}}{dt} \times \mathbf{r}$	Euler forces arise from a change in angular velocity ω in a rotating reference frame
Other Forces		
Electromotive	n/a	The term Electromotive force is a misnomer. It has SI units of volts, not Newtons. Accordingly, electromotive

Force	Alternative representations	Comments (assertions in natural language)
		force is not a force.
Torque	n/a	Torque is not a force, but rather a moment about an axis
* Calculations of these forces from first principles are extremely complicated and beyond the scope of this dissertation		

The force inventory can be encoded in an ontology. Figure 7 shows the ontology that has been developed, organized by force type (how it is applied)[18]. A non-contact force is one that acts over a distance—there are four known non-contact forces in the universe. Solid-contact forces require that objects touch. Fluid-contact forces apply to motion through liquids and gasses. A fictitious force is an apparent force that acts in a non inertial frame of reference—it does not arise from any physical interaction but rather from the acceleration of the reference frame itself. Finally, several phenomena are labeled or thought of as forces, yet are not.

[18] Note that there are other possible ways to organize the taxonomy that could impact how it is processed.

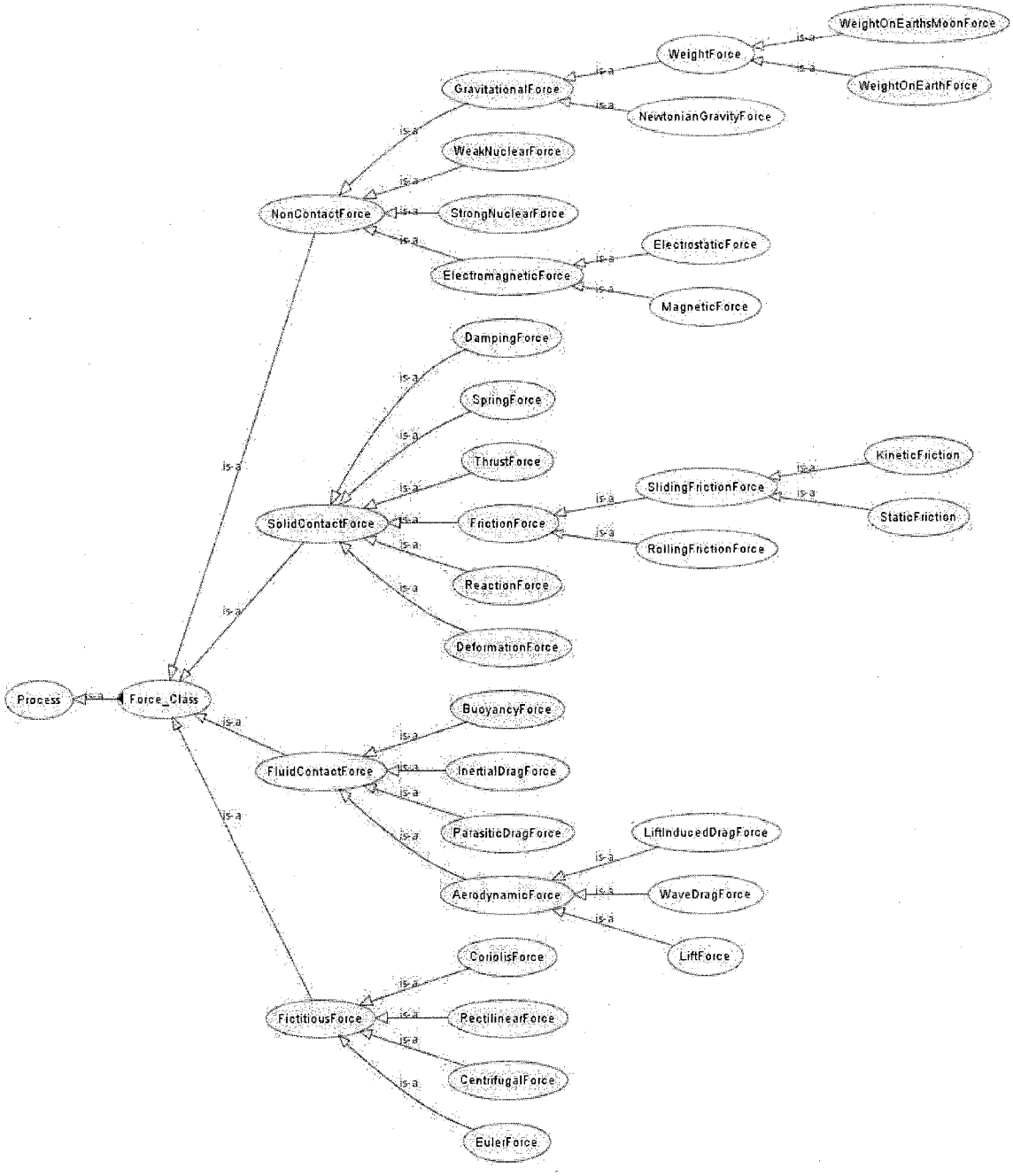


Figure 7. Ontology of Forces in Protégé

5.4 Assertions for Forces in the Falling Body Problem (Step 1b)

Equation 1 is a statement that the net force on the body is the sum of all of the forces possible. The equation is derived by application of IA_4 to the force inventory. Note that

the resultant force and each component force are shown as vector quantities. Thus, it is noted that a competency in vector mathematics, C_VECTORMATH, is needed.

$$\sum \boldsymbol{F} = net\ force = \boldsymbol{F}_g + \boldsymbol{F}_{Lorentz} + \boldsymbol{F}_{strong} + \boldsymbol{F}_{weak} + \boldsymbol{F}_{contact} + \boldsymbol{F}_{thrust} + \boldsymbol{F}_{friction\,s} + \boldsymbol{F}_{friction\,r} + \boldsymbol{F}_{spring} + \boldsymbol{F}_{damper} + \boldsymbol{F}_{b\acute{u}oy} + \boldsymbol{F}_{inertial} + \boldsymbol{F}_{viscous} + \boldsymbol{F}_{wave} + \boldsymbol{F}_{lift_{sphere}} + \boldsymbol{F}_{lift_{drag}} + \boldsymbol{F}_{rect} + \boldsymbol{F}_{centrifugal} + \boldsymbol{F}_{Coriolis} + \boldsymbol{F}_{Euler} \quad (1)$$

At this point in the problem formulation, the competent analyst simplifies the problem by selecting the appropriate components from the force inventory. His choices for which ones to include or exclude depend on a number of factors, but in making each choice he is making one or more assertions about the problem solution.

Note that each force can have a number of different, but similar, expressions. Choosing which representation to use introduces a complication to the problem. Each alternative algorithm has a potentially different set of assertions that accompanies it. To illustrate, the earth gravitational field representation of gravitational force is a simplification of the point mass scalar representation that rests on several assumptions:

- The mass of the earth is much greater than the mass of the body.
- The distance over which the gravitational force is acting is much less than the radius of the earth.
- The size of the earth is so much greater than the dimensions of the problem that the force generated acts directly downward with respect to the surface of the earth.
- Moreover, the point mass scalar form of the gravitational force is a simplification of the vector form that rests on its own set of assumptions:
- The distribution of the earth's mass is uniform.
- For the purposes of the problem, both the body and the earth can be considered point masses.

Using knowledge of physics and following sound engineering practice, the force inventory can be surveyed to refine the level of detail and to simplify the problem. As the analyst considers each item, he records the assertions that justify his choices or that constrain the solution. Each simplification rests on one or more propositions that are listed in the following table.

Table 7. Simplification Assertions

Force	Representation	Assumptions, Constraints, Considerations, Competencies
Non-Contact Forces		
Gravitational	$F_g = -mg$	g1: gravity is provided by the earth g2: the mass of the body is much less than the mass of the earth. g3: the distance over which F_g is acting is much less than the radius of the earth. g4: the dimensions of the body are much less than the radius of the earth g5: $g \approx G\frac{m_{earth}}{r_{earth}{}^2}$ and is represented by the value 9.8 m/s^2 g6: the distribution of the earth's mass is uniform g7: the distribution of the body's mass is uniform
Electromagnetic	$\mathbf{F}_{Lorentz} = 0$	Lorentz1: there are no electromagnetic forces $\therefore$ Lorentz2: there are no magnetic forces $\therefore$ Lorentz3: there are no electrostatic forces
Strong nuclear	$\boldsymbol{F}_{strong} = 0$	strong1: distances involved are > 10^{-15} m
Weak nuclear	$\boldsymbol{F}_{weak} = 0$	weak1: distances involved are > 10^{-18} m
Solid-Contact Forces		
Thrust	$\mathbf{F}_{thrust} = 0$	thrust1: there is no change in mass of the object
Reaction		r1: (logic) contact forces require contact between ≥ 2 objects there is only 1 object $\therefore$ no contact between objects $\therefore$ there are no contact forces, alternatively $\therefore$ all contact forces are 0
Sliding Friction	$F_{friction\_s} = 0$	friction_s1: sliding friction is a contact force friction_s1 $\wedge$ r1 $\Rightarrow$ force is 0

Force	Representation	Assumptions, Constraints, Considerations, Competencies
Rolling Friction	$F_{friction\_r} = 0$	friction_r1: rolling friction is a contact force friction_r1 ∧ r1 ⇒ force is 0
Spring	$F_{spring} = 0$	spring1: spring force is a contact force spring1 ∧ r1 ⇒ force is 0
Damping	$F_{damper} = 0$	damper1: damping force is a contact force damper1 ∧ r1 ⇒ force is 0
Fluid-Contact Forces		
Buoyancy	$F_{buoy} = m_f g \quad m_f = \frac{4}{3}\pi r^3$	buoy1: the model requires the object be a sphere buoy2: the model requires uniform fluid density buoy3: g1..g7 apply
Aerodynamic Lift Lift-Induced Drag	$\boldsymbol{F}_{aero} = \boldsymbol{F}_{lift} + \boldsymbol{F}_{lift\_drag}$ $F_{lift\_sphere} = 0$ $F_{lift\_drag} = 0$	aero1: the object is not an airfoil aero2: (analyst decision) the object does not spin ∴ ω=0. aero3: wind does not affect the body lift_sphere1: (algebraic substitution of ω=0) into expression $\frac{4}{3}(\pi^2 b^3 \omega \rho v)$ ⇒no lift HOWEVER: if the sphere is allowed to spin: lift_sphere2: the model ignores viscosity. lift_sphere3: the model requires that the axis of spin be perpendicular to the velocity. lift_sphere4: the model requires that the object be a sphere. lift_sphere5: the model requires that the object be smooth. lift_drag1: with no lift, there is no lift-induced drag
Wave Drag	$F_{wave} = 0$	wave1: the velocities involved are < .8 mach1

Force	Representation	Assumptions, Constraints, Considerations, Competencies
Inertial Drag	$F_{inertial} = -\frac{1}{2} m_f \frac{dv}{dt}$	inertial1: the model requires the object be a sphere inertial2: the model requires uniform fluid density
Viscous Drag	$\boldsymbol{F}_{viscous} = \begin{cases} -6\pi\eta r v\hat{\mathbf{v}} & R_e < \\ -\frac{1}{2}\rho v^2 A C_d \hat{\mathbf{v}} & 1 \leq R \end{cases}$	viscous1: the model requires the object be a sphere viscous2: no heat transfer occurs viscous3: viscosity is independent of temperature viscous4: the specific heat of the fluid is independent of temperature viscous5: the use of the Cd form of the equation is valid viscous6: the atmospheric conditions for the problem are reasonably close to those for which the data used in calculating Cd were collected
Fictitious Forces (d'Alembert Forces)		
Rectilinear	$\mathbf{F}_{rect} = m\mathbf{a}_{rect}$	rect1: there are no rectilinear forces
Centrifugal	$\mathbf{F}_{centrifugal} = 0$	centrifugal1: there are no centrifugal forces
Coriolis	$\mathbf{F}_{Coriolis} = 0$	Coriolis1: there are no Coriolis forces
Euler	$\mathbf{F}_{Euler} = 0$	Euler1: there are no Euler forces

Note that the listed assertions do not constitute a complete list. To illustrate, the topic of calculating aerodynamic forces is exceedingly complex. Only a few, high level, exemplary assertions have been made. Consider the assertions aero1-aero3. The combination of denying that the object is an airfoil, requiring that it does not spin and denying any wind affects should be sufficient to raise assertion conflicts in situations that most users would consider important. The result of applying the assertions listed in Table 7 to equation (1) is shown in equation (1a). Removing the terms set to zero yields

equation (2). The astute reader will recognize this as being the form of the problem as stated by Chow and used by Spiegel et al that was presented in Figure 6.

$$\begin{aligned}\sum \boldsymbol{F} = net\ force \\ = \boldsymbol{F}_g + \underbrace{\boldsymbol{F}_{Lorentz}}_{0} + \underbrace{\boldsymbol{F}_{strong}}_{0} + \underbrace{\boldsymbol{F}_{weak}}_{0} + \underbrace{\boldsymbol{F}_{contact}}_{0} + \underbrace{\boldsymbol{F}_{thrust}}_{0} \\ + \underbrace{\boldsymbol{F}_{friction\,s}}_{0} + \underbrace{\boldsymbol{F}_{friction\,r}}_{0} + \underbrace{\boldsymbol{F}_{spring}}_{0} + \underbrace{\boldsymbol{F}_{damper}}_{0} + \boldsymbol{F}_{buoy} + \boldsymbol{F}_{inertial} \\ + \boldsymbol{F}_{viscous} + \underbrace{\boldsymbol{F}_{wave}}_{0} + \underbrace{\boldsymbol{F}_{lift\_sphere}}_{0} + \underbrace{\boldsymbol{F}_{lift\_drag}}_{0} \\ + \underbrace{\boldsymbol{F}_{rect} + \boldsymbol{F}_{centrifugal} + \boldsymbol{F}_{Coriolis} + \boldsymbol{F}_{Euler}}_{0}\end{aligned} \tag{1a}$$

$$\begin{aligned}\sum \boldsymbol{F} = \boldsymbol{F}_g + \boldsymbol{F}_{buoy} + \boldsymbol{F}_{inertial} + \boldsymbol{F}_{viscous} = \\ -mg + m_f g - \frac{1}{2} m_f \frac{dv}{dt} - \frac{1}{2}\rho v^2 A C_d \hat{v}\end{aligned} \tag{2}$$

There is a distinction between (2) and Chow's problem solution shown in Figure 6. The applicability of the equation to the falling body problem rests on Initial Assertions IA_1 through IA_5, and propositions N2_1 to N2_7. The model domain viewpoint is captured by the assertions g1-7, Lorentz1, strong1, weak1, friction_s1, friction_r1, spring1, damper1, buoy1-3, inertial 1 to 2, viscous1-6, aero1-3, lift_sphere1, lift_drag1, wave1, rect1, centrifugal1, Coriolis1, and Euler1. Finally, the solution requires competencies C_ALGEBRA, C_CALCULUS, and C_VECTORMATH.

Note that slight variations on the problem can affect the model domain viewpoint. To illustrate, consider the propositions for lift on a sphere. Aero1 is the analyst's choice to simplify the problem by denying spin on the body. The proposition lift_sphere1 is the conclusion that the force F_{lift_sphere} can be ignored because there is no spin (aero2). However, what if circumstance causes the analyst to change his mind and use spin (perhaps as part of re-using the falling body model)? In this case, the assertions lift_sphere2 to 5 apply to the previously removed term for lifting force on a sphere F_{lift_sphere}.

6. THE DEMONSTRATION

As stated earlier, the potential for conflict between model components arises when choices made by the modeler introduces inconsistencies between components. The objective of this section is to demonstrate conflict detection using an automated process. Before that can occur, issues relating to ontology organization and comparison software must be reviewed.

6.1 Ontology Organization

One of the characteristics of the hybrid ontology architecture recommended by Wache [29] is the need to provide a common reference language for the concepts. This is accomplished by layering ontologies as shown in Figure 8. The force ontology imports the ontology of assertion and system concepts. The solution to the Falling Body Problem on Earth is a separate ontology that imports the concepts of the force ontology and makes the top level assertion that gravity is provided by the Earth. An alternative solution, perhaps for a problem formulated to take place on the Moon would import the same force concepts and assert that gravity is provided by the Moon instead. This has the effect of eliminating the need to resolve ambiguity.

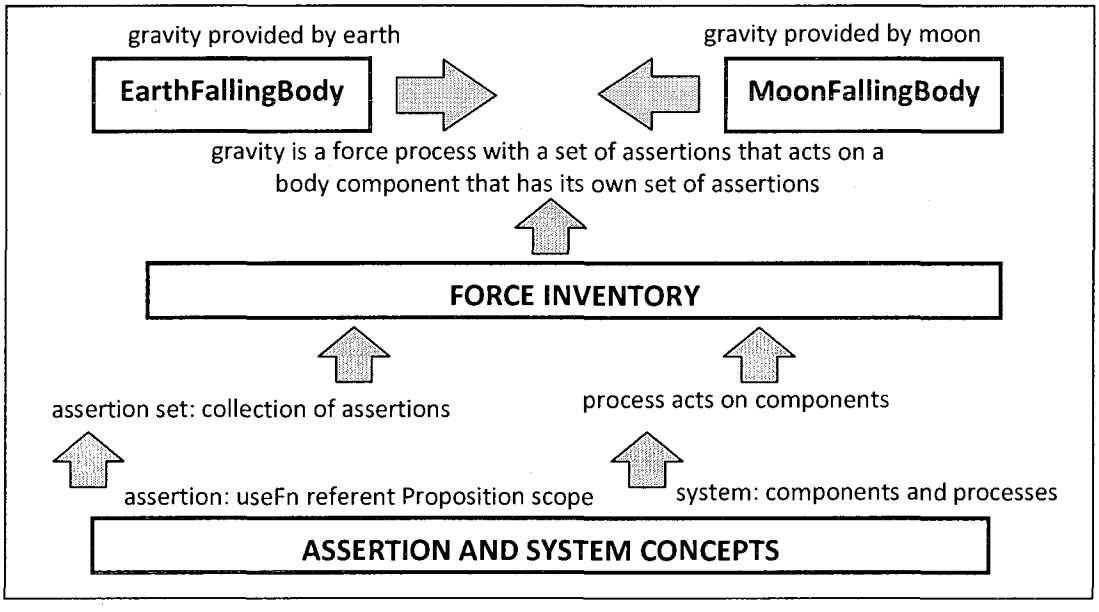


Figure 8. Ontology Layering to Achieve Common Reference

6.2 Ontology Comparison Software Application

For various reasons a custom software application needed to be developed. The most significant reason relates to the current state of the art in ontology engineering and automated reasoning. As previously discussed, a Description Logics reasoner is a software implementation of an inference engine whose purpose is to reason with a knowledge base expressed in OWL-DL. DL reasoners cannot operate with OWL-Full ontologies because logic expressed in OWL-Full cannot be guaranteed to be decidable. For example, in OWL-Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right; this is not permitted in OWL DL. The significance to this research is that the query to decide if two assertions are compatible necessarily treats the assertions being compared as both individuals and classes. An attempt to include such a query in an ontology would cause the ontology to be classified as OWL-Full, rendering it incompatible with either of the available DL reasoners. The solution was to develop the software described in the following subsection.

6.2.1 Comparison Software

Development was greatly facilitated by use of OWL API [65]—a Java interface and implementation for OWL languages. The OWL API provides support for parsing ontologies, queries, integration with reasoners such as Pellet and FaCT++, and writing OWL files.

Figure 9 is a flowchart of the software developed to perform the comparison. Generally speaking, it reads in an OWL ontology and checks it for consistency using the Pellet reasoner. If the ontology is consistent, the software builds a list of assertions found in the file. To do this, it scans for assertions defined using the author's formalism—therefore, any ontology being compared must include the assertion and system concepts ontology in its imports closure. Once the list is built, every assertion is compared against the others in the list for conflicts previously listed in Table 1. If any conflicts are detected, the program ends with a diagnostic error message. If a second ontology has been named for comparison, the program performs the same checks and list-building activities. Finally, every assertion in the first ontology is compared to each one in the second to test for conflicts. As above, conflicts result in an error message; otherwise the program notifies the user of success.

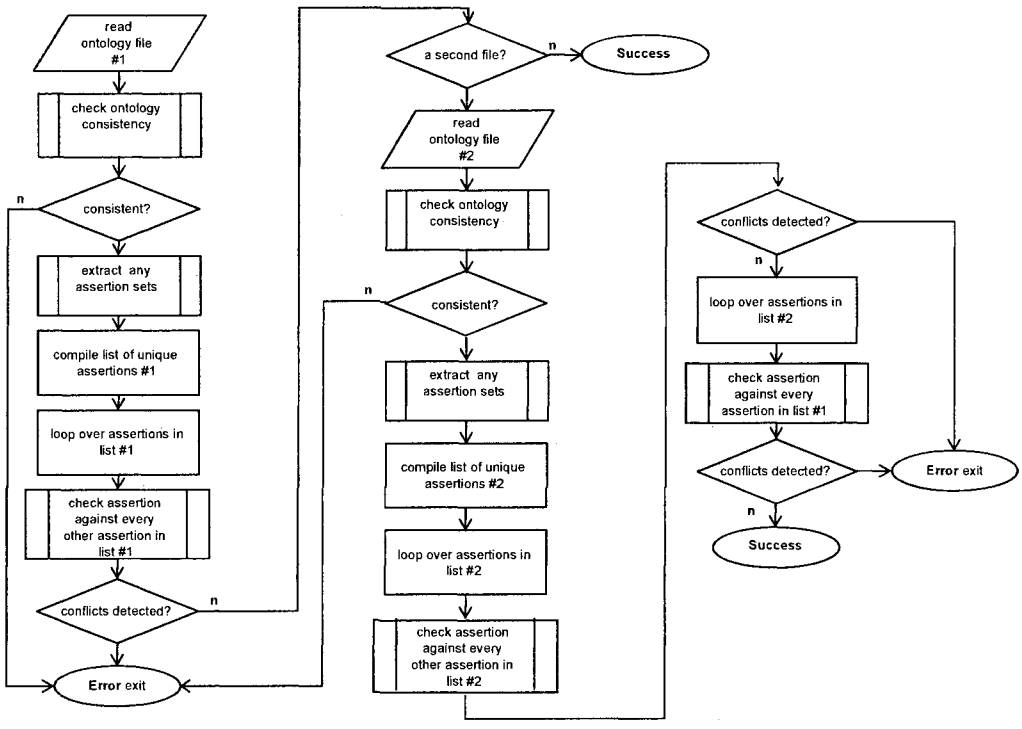


Figure 9. Comparison Software Flowchart

6.2.2 Test Cases

To ensure that the software is performing correctly, a series of simple test cases was constructed. Several straightforward ontologies were prepared that implemented one of the propositions shown in Table 8. In a structure similar to that discussed in section 6.1, a middle ontology was coded to provide assertions regarding color. This ontology is shown in Figure 10. Note that to encode the proposition that something is not colored blue, it is necessary to say that it ***is*** a color ***other than blue***. One might be tempted simply to assert the property **hasColor blue** is false, but this is not an option as this almost always will produce an inconsistent ontology. This highlights one of the shortcomings in the use of the description logics reasoner namely, the *open world assumption.* Under this assumption, if a statement cannot be proved to be true using current knowledge, one cannot draw the conclusion the statement is false. Thus, negating the property **hasColor blue** is equivalent to stating something belongs to the class of everything that does not have the color blue—even things with no color at all.

Table 8. Proposition Details Used in Validation Tests

Proposition	Class membership	Class Properties
p_colored_red	colored_red class	hasColor red
p_colored_blue	colored_blue class	hasColor blue
p_colored_not_red	not_colored_red class	hasColor some (orange yellow green blue violet black white)
p_colored_not_blue	not_colored_blue class	hasColor some (red orange yellow green violet black white)

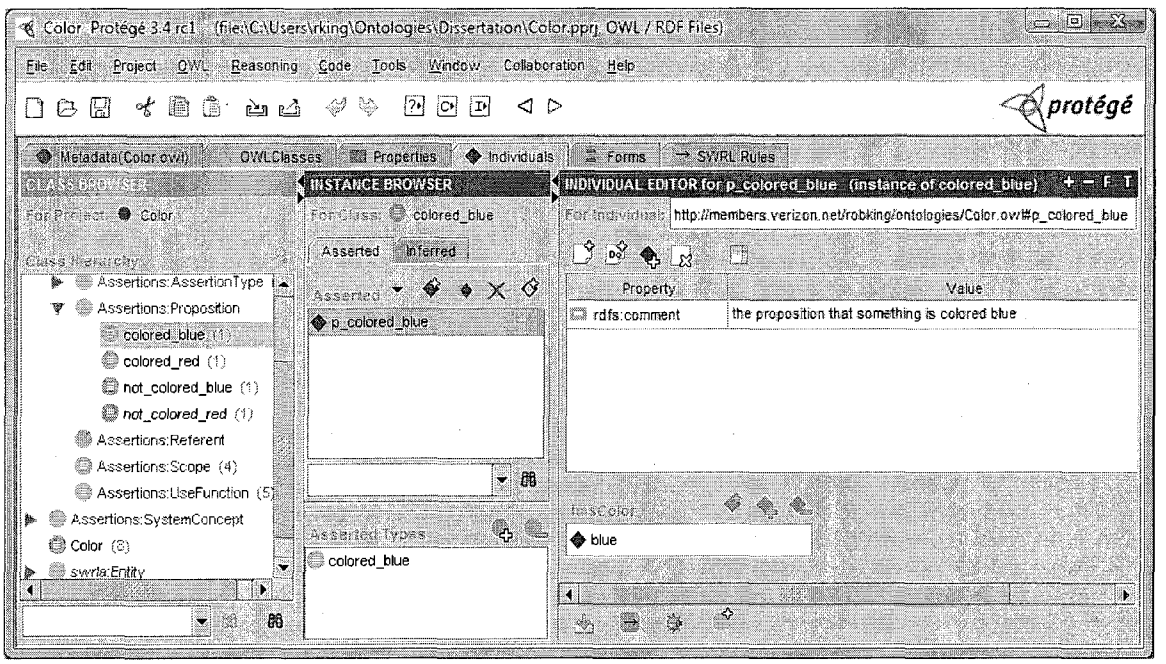


Figure 10. Color Ontology with Proposition p_colored_blue

presents the results of the validation tests conducted on the comparison software. The ontologies involved formed a hierarchical layering of definitions similar to that depicted in Figure 8 above. An ontology file, Color.OWL, provided the common reference language for color. Eight colors were enumerated as individuals, (black, blue, green, orange, red, violet, white and yellow). As can be seen, each test produced a successful result. The test exercised each of the potential conflicts listed in Table 1 and therefore verified the program functioned properly.

Table 9. Validation Test Matrix with Results

File (s)	Elements	Assertion set (s)	Assertions	Propositions Involved default = used	Test Purpose / Results
redObject.owl	theRedObject	SysElemAssertSet_RedObject	a_theObjectIsRed	p_colored_red	Check ontology consistency **SUCCESS: CONSISTENT**
blueObject.owl	theBlueObject	SysElemAssertSet_BlueObject	a_theObjectIsBlue	p_colored_blue	Check ontology consistency **SUCCESS: CONSISTENT**
redObject.owl & blueObject.owl	theRedObject theBlueObject	SysElemAssertSet_RedObject SysElemAssertSet_BlueObject	a_theObjectIsRed a_theObjectIsBlue	p_colored_red p_colored_blue	Check found/not found conflict **SUCCESS: DETECTED CONFLICT** fail_possible_conflict_not_found
redObject.owl & redObject.owl	theRedObject theRedObject	SysElemAssertSet_RedObject SysElemAssertSet_RedObject	a_theObjectIsRed a_theObjectIsRed	p_colored_red p_colored_red	Check found/found alignment **SUCCESS: CONSISTENT** (succesful trivial comparison)
blueObject.owl & notBlueObject.owl	theBlueObject, theNotBlueObject	SysElemAssertSet_BlueObject, SysElemAssertSet_CannotBeBlueObject	a_theObjectIsBlue a_theObjectIsNotBlue	p_colored_blue p_colored_not_blue	Check found/not found conflict **SUCCESS: DETECTED CONFLICT** fail_possible_conflict_not_found
blueObject.owl & blueObjectDenied. owl	theBlueObject, theCannotBeBlueObject	SysElemAssertSet_BlueObject, SysElemAssertSet_CannotBeBlueObject	a_theObjectIsBlue a_theObjectCannotBeBlue	p_colored_blue p_colored_blue (denied)	Check use/denied conflict **SUCCESS: DETECTED CONFLICT** fail_denial_conflict
blueObject.owl & blueObjectRequired. owl	theBlueObject, theMustBeBlueObject	SysElemAssertSet_BlueObject, SysElemAssertSet_MustBeBlueObject	a_theObjectIsBlue a_theObjectMustBeBlue	p_colored_blue p_colored_blue (required)	Check use/required conflict **SUCCESS: CONSISTENT**
blueObjectDenied. owl & blueObjectRequired. owl	theCannotBeBlueObject, theMustBeBlueObject	SysElemAssertSet_CannotBeBlueObject, SysElemAssertSet_MustBeBlueObject	a_theObjectCannotBeBlue a_theObjectMustBeBlue	p_colored_blue (denied) p_colored_blue (required)	Check denied/required conflict **SUCCESS: DETECTED CONFLICT** fail_denial_conflict
blueObjectRequired. owl & blueObjectDenied. owl	theMustBeBlueObject, theCannotBeBlueObject	SysElemAssertSet_MustBeBlueObject, SysElemAssertSet_CannotBeBlueObject,	a_theObjectMustBeBlue a_theObjectCannotBeBlue	p_colored_blue (required) p_colored_blue (denied)	Check denied/required conflict **SUCCESS: DETECTED CONFLICT** fail_requirement_conflict

6.3 Encoding the Assertions (Step 2)

The next step in the example is to encode the assertions in a knowledge engineering tool. Because of the complexity of the overall model, the demonstration will focus on assertions relating to the use of gravity. The following figures illustrate how the assertions that support use of the term *–mg* for gravitational force are encoded in the Protégé knowledge engineering tool—expressing gravitational force as weight on the surface of the Earth, **WeightOnEarthForce**. As shown in Figure 11, weight on the Earth means the force acts on **some** component, namely A_Physical_Object. The statement that it acts on **only** components is a *closure axiom* and is necessary because forces cannot act on anything but physical objects. Also, all forces are represented as processes and a process is a kind of system element. Therefore, the property **hasAssertionSet** can be applied to any force, and the assertion set is of the type **SystemElementAssertionSet**.

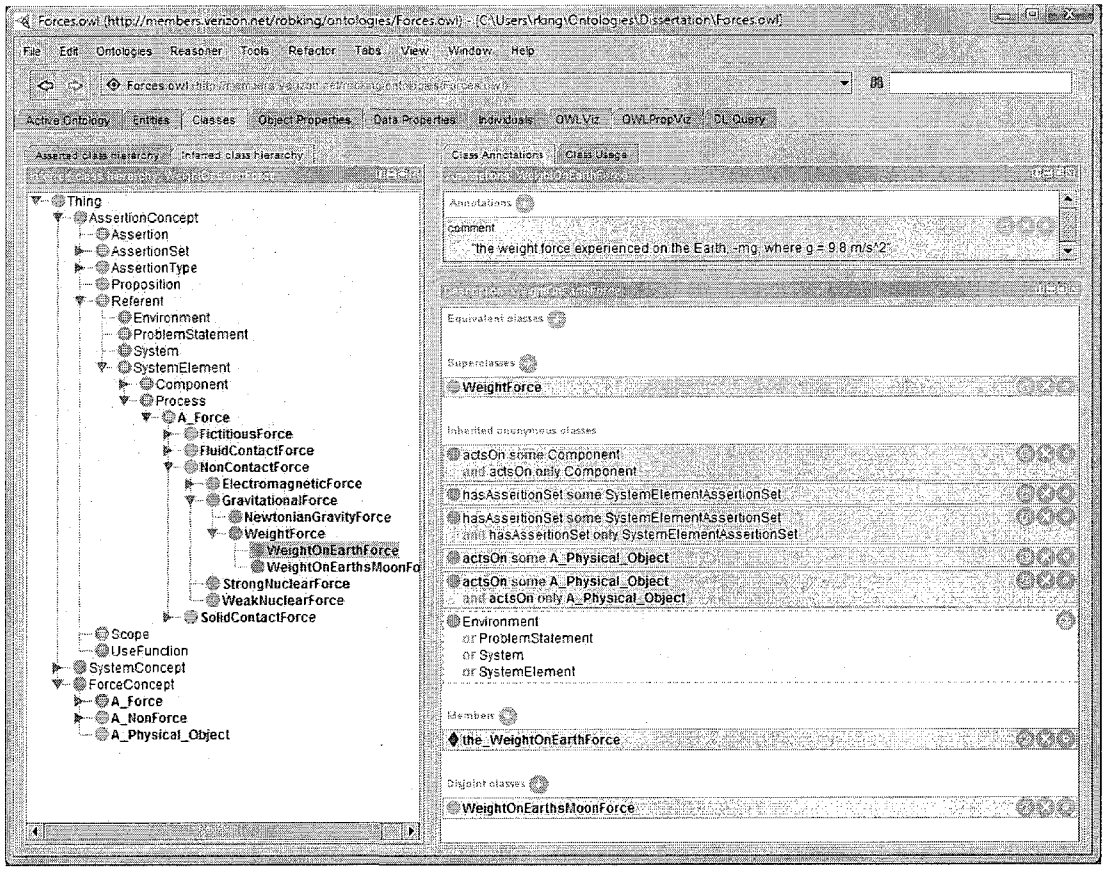


Figure 11. Weight on Earth Force Encoded in Protégé

Figure 12 shows the **SystemElementAssertionSet** that is assigned to the force, **WeightOnEarthForce**. As can be seen, it makes six assertions that correspond to g1 through g6 described in the previous section.

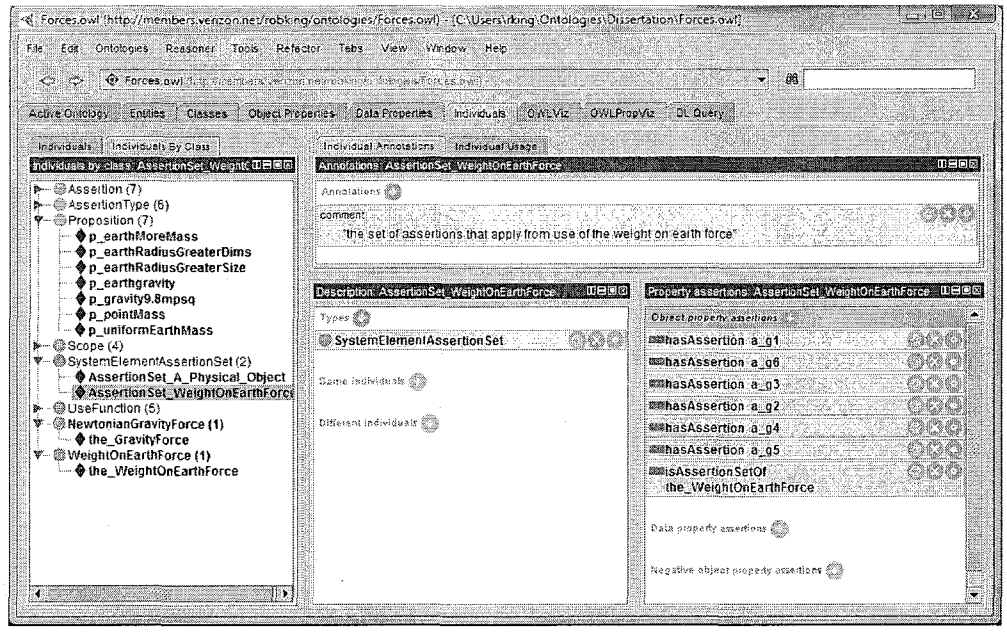


Figure 12. Assertions in the Weight on Earth System Element Assertion Set

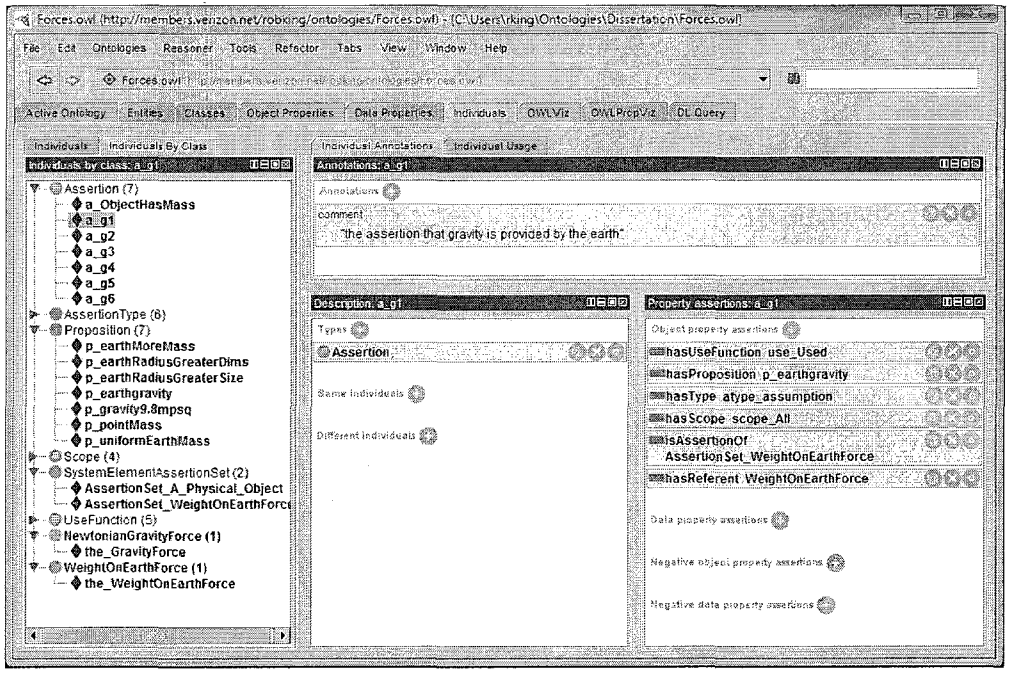


Figure 13. Asserting Earth Gravity (g1)

6.4 Detecting Conflicts (Step 3)

The final demonstration was almost anti climactic. The ontology for the falling body problem was compared with ontologies for variations on the problem such as FallingMoonBody (the source of gravity was not the earth), and MoonOrbit (the size of the body was not much less than the earth's radius). The ontologies may be found on the compact disk that accompanies this dissertation.

6.5 Resolving Conflicts (Step 4)

As previously stated, primary focus of this research is on demonstrating that conflicts can be detected. For this reason, little will be said concerning the resolution of each conflict. The diagnostic information available will include the system components or processes involved, the proposition that is in conflict, and, of course, the reason why the assertions fail to match.

7. CONCLUSION

The objective of this research is to contribute a partial solution to one of the problems of model composability and simulation interoperability. It does so by demonstrating the importance of the assertions that are made during model development and simulation implementation, particularly as they reflect the unique viewpoint of each developer. It hypothesized it would be possible to detect conflicts by means of the four-step process for capturing and comparing assertions. It demonstrated the process using a well understood example problem—the Falling Body Problem—developing an inventory of forces and cataloging the significant assertions that might be made about each force in the context of the problem. Finally, it developed a software application that employs the assertion formalism and the comparison strategy to compare ontologies. The software was validated using straightforward test cases. The software successfully detected potential conflicts between ontologies that were otherwise determined to be ontologically consistent, thus proving the hypothesis.

7.1 Contributions Made by This Research

This research has demonstrated the importance of assertions in composing and integrating model components. It has provided an analysis of the roles that assertions play that has been ***previously lacking in the Modeling and Simulation literature***. In this respect, it provides an additional insight into interoperability that is not captured by current literature in Conceptual Modeling. Whilst some current writers advise it is important to capture assumptions[19], virtually nothing is said about ***what to do with them***.

This research has provided ***a new formal model of assertion*** suitable for capturing the assertions made about a system and its components. The general topic of assertions was examined in detail, including a taxonomy of assertion characteristics, thereby providing a definitional basis for follow-on research.

---

[19] Not to mention the more general class of assertions.

Solving the problem of reaching LCIM Level 6 has been the research goal from the start. The analysis of why models and simulations fail to reach the highest level of interoperability yielded a new concept: ***Barriers to Functional Composition explain why some conflicts emerge only during implementation***. The proposed solution to the barrier problem, Conceptual Linkage, focuses attention on the need to consider model attributes that have not been generally accounted for. Exploring requirements for aligning model domain viewpoint brings new understanding to the role that viewpoint plays in interoperability. Each of these research contributions ***extends the applicability of the LCIM***, giving it increased ability to explain interoperability issues.

Additionally, the research ***developed and demonstrated a process for capturing and comparing assertions*** that makes use of the formal model of assertion. The comparison strategy provides a new framework for comparing systems. Ultimately, ***the research contributes a key element for the handling of assertions by autonomous agents.***

As part of addressing the Falling Body Problem, the research contributes a new, Physics-based analysis that extends work of Spiegel et al, and is a step towards the ontology advocated by Collins and Clark. The Force Inventory extends the work by Hestenes et al in that it ***establishes an inventory of forces with ties to the assertions that support its use.***

Also, the research ***establishes a new class of metadata for describing models***. It has shown that consistency extends beyond comparing inputs and outputs or even methods.

7.2 Relationship to Other Research

The research ***contributes to the composability and interoperability framework*** envisioned in [23]. The research can be viewed in the context of the three engineering methods of data engineering, process engineering, and constraints engineering described in that chapter. This research provides the foundation for the third pillar, constraint engineering, by better defining the context of Level 6—Conceptual Interoperability—of the LCIM. In this respect it provides a framework for ongoing work by the author's colleagues at the Virginia Modeling Analysis and Simulation Center, and also benefits from their work as well.

The research addresses issues regarding composability and interoperability. Page et al [66] state models should be considered composable if they share compatible objectives

and assumptions. They offer no suggestions as how to accomplish this, except to suggest that research in quantifying and reasoning about the "compatibility" of objectives and assumptions is needed[20]. This research ***has made progress towards that goal***. Note also they emphasize a separation between composability and interoperability. However, this research ***demonstrates the opposite effect***—that problems for both non-composability and non-interoperation can be traced in part to a common cause, namely, conflicts in assertions.

7.3 Caveats and Future Research Suggestions

It is not uncommon that the solution to an engineering problem depends on the structure of the solution approach, and this research is no exception. The use of ontology-based reasoning depends on how the knowledge has been structured. In the case of this research, the falling body problem solution depends on the structure and completeness of the force inventory created by the author. The representation of assertions in the force ontology depends, in turn, on the definitions and axioms in the ontology of assertion and system concepts. The practical effect is the creation of a common reference ontology, similar to the hybrid ontology approach suggested by Wache et al [29]. Note that the author's ontologies have not been independently vetted by experts in the appropriate domains. Thus, although they may be substantially complete and accurately represent their domains, validation is deferred to follow-on research. Therefore, it is appropriate to add a caveat that the solution to the example problem depends on the author's ontologies.

Another caveat needs to be made regarding reusability of the software application for comparing assertion sets. Use of the software requires propositions be encoded in OWL-DL, the author's ASSERTIONS.OWL ontology be imported, and the assertions in the assertion sets be assigned to system components or processes, problem statements, or solutions using the author's formalism.

The author's decision to use only exact matching for proposition concept comparison is a two edged sword. On the one hand, this guarantees a Boolean answer to the question. On the other hand, it does not address the question, "what if the concepts are close?"

---

[20] See discussion in section 2.1

Certainly, this is an extremely interesting question, but addressing it would have drawn the research focus away from answering the more basic question it has addressed. Therefore, it is appropriate to add a caveat that the method has been shown to work only for the case of exact comparison of propositions.

The comparison strategy presented in section 4.4.1 remains valid under subsumption because subsumption ultimately rests on finding a subsuming proposition that is equivalent to the one under consideration. The strategy will likely remain valid for overlap, largely because the response to potential conflicts is to raise an alert—essentially passing questionable match decisions to a human for adjudication. However, these have not been demonstrated. Therefore, it is appropriate to add a caveat that although it is suggested the method works for subsumption and overlap, this has not been demonstrated.

Another issue is the question, "what if the referent is a part of a complex system, or if there are multiple referents? Where are the practical limits imposed when addressing real-world systems?" The criticism is valid, and it certainly marks an avenue for follow-on research. However, the research needs to establish the method is valid for a simple system first before delving into complexity issues. Any statement regarding complexity would be speculation at this point. Therefore, it is appropriate to add a caveat that the method has been shown to work only for simple problems at this time.

7.4 Summary

To summarize, this research is the first of its kind that contributes to achieving Level 6 of the LCIM in machine understandable form. The assertion formalism developed is a first step to make assumptions, content, and other elements identified by Robinson [5] accessible to machine implementations, such as web services or software agents. The process for comparing assertion sets captures aspects that cannot be derived from the implementation or from data specification. Without this contribution, services may be composed that are conceptually not aligned. This work is the initial step to avoid this.

In conclusion, perhaps the most worthy contribution of this research is the path it has lighted for others to tread.

REFERENCES

[1] P. K. Davis and R. A. Anderson, "Improving the Composability of Department of Defense Models and Simulations," RAND National Defense Research Institute, 2004.

[2] M. D. Petty and E. W. Weisel, "A Composability Lexicon," in *Proceedings of the Spring Simulation Interoperability Workshop,* 2003.

[3] M. D. Petty, E. W. Weisel, and R. R. Mielke, "Computational Complexity of Selecting Components for Composition," in *Proceedings of the Fall Simulation Interoperability Workshop,* 2003.

[4] E. H. Page, R. Briggs, and J. A. Tufarolo, "Toward a Family of Maturity Models for the Simulation Interconnection Problem," in *Proceedings of the Spring Simulation Interoperability Workshop,* 2004.

[5] S. Robinson, "Conceptual Modeling for Simulation: Issues and Research Requirements," in *Proceedings of the Winter Simulation Conference,* IEEE, 2006, pp. 792-800.

[6] S. Robinson, "Conceptual modelling for simulation Part I: definition and requirements," *Journal of the Operational Research Society*, vol. 59, no. 3, pp. 278-290, 2007.

[7] S. Robinson, "Conceptual modelling for simulation Part II: a framework for conceptual modeling," *Journal of the Operational Research Society*, vol. 59, no. 3, pp. 291-304, 2007.

[8] J. Borah, "Conceptual Modeling-How do we do it?-A practical Example," in *Proceedings of the Spring Simulation Interoperability Workshop,* 2003.

[9] "Conceptual Model Development and Validation." VV&A Recommended Practices Guide. Available from http://vva.dmso.mil accessed: 7-29-2007

[10] D. K. Pace, "Development and Documentation of a Simulation Conceptual Model," in *Proceedings of the Fall Simulation Interoperability Workshop,* 1999.

[11] D. K. Pace, "Simulation Conceptual Model Development Issues and Implications for Reuse of Simulation Components," in *Proceedings of the Fall Simulation Interoperability Workshop,* 2000.

[12] Garlan, David, Allen, Robert, and Ockerbloom, John,"Architectural Mismatch: Why Reuse Is So Hard." IEEE SOFTWARE 1995 12:6 17-25

[13] J. A. Dewar, C. H. Builder, W. M. Hix, and M. H. Levin, "Assumption-Based Planning: A Planning Tool for Very Uncertain Times," The Rand Corporation, Santa Monica, CA, 1993.

[14] M. A. Hofmann, "Modeling Assumptions: How they affect Validation and Interoperabilitiy," in *Proceedings of the European Simulation Interoperability Workshop,* 2005.

[15] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation, 2nd Ed.* John Wiley, 2000.

[16] IEEE, "Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP) (STD 1516.3)," Institute of Electrical and Electronics Engineers, 2003.

[17] NATO RTO Studies Analysis and Simulation Panel, "NATO Code of Best Practice for Command and Control Assessment," NATO Research and Technology Organization (RTO),RTO Technical Report TR-081, 2004.

[18] M. Spiegel, P. F. Reynolds, and D. C. Brogan, "A Case Study of Model Context for Simulation Composability and Reusability," in *Proceedings of the Winter Simulation Conference,* 2005.

[19] R. D. King and C. D. Turnitsa, "The Landscape of Assumptions," in *Proceedings of the Agent Directed Simulation Conference,* 2008.

[20] A. Tolk and J. Muguira, "The Levels of Conceptual Interoperability Model," in *Proceedings of the Fall Simulation Interoperability Workshop,* 2003.

[21] C. Turnitsa, "Extending the Levels of Conceptual Interoperability Model," in *Proceedings of the Summer Computer Simulation Conference,* 2005.

[22] A. Tolk, S. Y. Diallo, and C. Turnitsa, "Ontology Driven Interoperability – M&S Applications," Old Dominion University,*Whitepaper in support of the I/ITSEC Tutorial, VMASC Report 2548*, 2006.

[23] A. Tolk, S. Y. Diallo, R. D. King, and C. D. Turnitsa, "A Layered Approach to Composition and Interoperation in Complex Systems," in *Complex Systems in Knowledge-based Environments: Theory, Models and Applications*. A. Tolk and L. C. Jain, Eds. Springer-Verlag, 2009, pp. 41-74.

[24] R. D. King, S. Y. Diallo, and A. Tolk, "How to Play Fairly: Agents and Web Services Can Help," in *Proceedings of the Spring Simulation Interoperability Workshop,* 2007.

[25] R. D. King, "Towards Conceptual Linkage of Models and Simulations," in *Proceedings of the Fall Simulation Interoperability Workshop,* 2007.

[26] N. Y. Foo, "Why Engineering Models Do Not Have A Frame Problem," in *Discrete Event Modeling And Simulation Technologies*. H. S. Sarjouhian and F. E. Cellier, Eds. New York: Springer-Verlag, 2001, pp. 15-26.

[27] "Stanford Encyclopedia of Philosophy: The Frame Problem." Available from http://plato.stanford.edu/entries/frame-problem/ Accessed: 7-24-2007

[28] E. Eberbach, D. Q. Goldin, and P. Wegner, "Turing's Ideas and Models of Computation," in *Alan Turing: Life and Legacy of a Great Thinker*. C. Teuscher, Ed. New York: Springer-Verlag, 2004, pp. 159-194.

[29] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner, "Ontology-based Integration of Information -- a Survey of Existing Approaches," in *Proceedings of the IJCAI-Workshop Ontologies and Information Sharing,* Seattle, WA: 2001, pp. 108-117.

[30] H. Wache, "Towards Rule-Based Context Transformation in Mediators," in *Proceedings of the International Workship on Engineering Federated Information Systems (EFIS),* 1999, pp. 107-122.

[31] J. Bao, D. Caragea, and V. G. Honavar, "Modular Ontologies - A Formal Investigation of Semantics and Expressivity," in *The Semantic Web - ASWC 2006,* Springer Berlin / Heidelberg, 2006, pp. 616-631.

[32] D. B. Lenat and R. V. Guha, "The evolution of CycL, the Cyc representation language," *ACM SIGART Bulletin*, vol. 2, no. 3, pp. 84-87, 1991.

[33] M. Kifer and G. Lausen, "F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme," *SIGMOD Record*, vol. 18, no. 2 1989.

[34] M. R. Genesereth and R. E. Fikes, "Knowldege Interchange Format Version 3.0 Reference manual," Logic Group, Computer Science Department, Stanford University,Logic-92-1, 1992.

[35] "OWL Web Ontology Language Guide." World Wide Web Consortium (W3C). Available from http://www.w3.org/TR/owl-guide/ Accessed: 3-8-2009

[36] "Web Ontology Language." WIKIPEDIA. Available from http://en.wikipedia.org/wiki/Web_Ontology_Language Accessed: 3-8-2009

[37] L. Ding, P. Kolari, Z. Ding, and S. Avancha, "Using Ontologies in the Semantic Web: A Survey," UMBC Technical Report CS-05-07, 2005.

[38] "Ontology Tools Survey, Revisited." XML.com. Available from http://www.xml.com/pub/a/2004/07/14/onto.html Accessed: 3-8-2009

[39] Stanford Center for Biomedical Informatics Research, "Protégé ," Available from http://protege.stanford.edu Accessed: 3-8-2009

[40] E. Sirin, B. Parsia, B. C. Grau, A. Kkalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51-53, 2007.

[41] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: System description," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* Berlin: Springer-Verlag, 2007, pp. 292-297.

[42] C. Fellbaum, *WordNet An Electronic Lexical Database* MIT Press, 1998.

[43] L. Yang, "A Comprehensive Study of Inappropriate Hierarchy in WordNet," in *The Semantic Web - ASWC 2006,* Springer Berlin / Heidelberg, 2006, pp. 639-645.

[44] I. Niles and A. Pease, "Towards a Standard Upper Ontology," in *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS),* Ogunquit, Maine: ACM, 2001.

[45] U. Durak, H. Oguztuzun, and S. K. Ider, "An Ontology for Trajectory Simulation," in *Proceedings of the Winter Simulation Conference,* 2006, pp. 1160-1167.

[46] "physics ontology." Dumontier Lab. Available from http://ontology.dumontierlab.com/physics-primitive Accessed: 4-1-2009

[47] D. J. Russomanno, C. R. Kothari, and O. A. Thomas, "Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models," in *Proceedings of the International Conference on Artificial Intelligence,* 2005, pp. 637-643.

[48] J. B. Collins and D. Clark, "Towards an Ontology of Physics," in *Proceedings of the European Simulation Interoperability Workshop,* 2004.

[49] Hestenes, David, Wells, Malcom, and Swackhamer, Gregg,"Force Concept Inventory." The Physics Teacher 1992 30:March 141-158

[50] Hestenes, David and Wells, Malcom,"A Mechanics Baseline Test." The Physics Teacher 1992 30:March 159-166

[51] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in *Machine Intelligence 4.* B.Meltzer and D.Michie, Eds. Edinburgh University Press, 1969, pp. 463-502.

[52] "Concepts of Logical AI." Computer Science Department, Stanford University. Available from http://www-formal.stanford.edu/jmc/concepts-ai.html Accessed: 3-8-2009

[53] M. Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* Cambridge, MA: MIT Press, 1997.

[54] M. Dorfman, "Requirements Engineering," in *Software Requirements Engineering, Second Edition.* R. H. Thayer and M. Dorfman, Eds. IEEE Computer Society Press, 1997, pp. 7-22.

[55] J. D. McCabe, "Introduction," in *Network Analysis, Architecture and Design* Amsterdam: Morgan Kaufman, 2003, pp. 1-48.

[56] A. Church, "Extension," in *Dictionary of Philosophy.* D. Runes, Ed. Totowa, NJ: Littlefield, Adams & Company, 1972, pp. 147-148.

[57] M. Kokla, "Guidelines on Geographic Ontology Integration," in *Proceedings of the ISPRS Technical Commission II Symposium,* 2006.

[58] J. Bao, D. Caragea, and V. G. Honavar, "On the Semantics of Linking and Importing in Modular Ontologies," in *Proceedings 5th International Semantic Web Conference (ISWC 2006),* Springer, 2006.

[59] O. Corcho, M. Fernández-López, and A. Gómez-Pérez, "Methodologies, tools and languages for building ontologies. Where is their meeting point?" *Data & Knowledge Engineering*, vol. 46, pp. 41-64, 2003.

[60] P. Doran, "Ontology Reuse via Ontology Modularisation," Department of Computer Science Report, University of Liverpool, 2006.

[61] L. Yilmaz and S. Paspuleti, "Toward a Meta-Level Framework for Agent-Supported Interoperation of Defense Simulations," *The Journal of Defense Modeling and Simulation*, vol. 2, no. 3, pp. 161-175, 2005.

[62] L. Yilmaz, "On Improving Dynamic Composability via Ontology-driven Introspective Agent Architectures," in *Proceedings of the World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI,* 2006.

[63] C.-Y. Chow, *An Introduction to Computational Fluid Mechanics.* New York: John Wiley & Sons, 1979.

[64] "Classical Mechanics." WIKIPEDIA. Available from http://en.wikipedia.org/wiki/Classical_mechanics Accessed: 1-9-2009

[65] M. Horridge, S. Bechofer, and O. Noppens, "Igniting the OWL 1.1 Touch Paper: The OWL API," 2007.

[66] E. H. Page, R. Briggs, and J. A. Tufarolo, "Toward a Family of Maturity Models for the Simulation Interconnection Problem," in *Proceedings of the Spring Simulation Interoperability Workshop,* 2004.

APPENDIX A. FILES ON THE ACCOMPANYING DISK

A compact disk will accompany the completed dissertion with the following files:

Name	Description
Assertions.owl	The ontology of assertions and systems concepts.
blueObject.owl	Test ontology of a blue object
blueObjectDenied.owl	Test ontology of an object that is denied to be blue
blueObjectRequired.owl	Test ontology of an object that is required to be blue
Color.owl	Ontology of common color definitions
FallingBodyProblem.owl	The ontology of the falling body problem
FBPMerge.owl	The merged falling body problem with Earth gravity asserted
FBPMoonMerge.owl	The merged falling body problem with Moon gravity asserted
Forces.owl	The ontology of force concepts
NotBlueObject.owl	Test ontology of an object that is not blue
RedObject.owl	Test ontology of a red object
AssertionTest.java	Java source code of the ontology comparison program

VITA

Robert Dennis King
Lieutenant Commander, United States Navy (Retired)

LCDR King was born in Chicago, Illinois, and received a Bachelor of Science (cum laude) in Electrical Engineering at Marquette University in Milwaukee Wisconsin in 1973. He has held a variety of research positions in science and engineering:

Senior Modeling and Simulation Analyst, Chipton-Ross: working onsite at Northrop-Grumman Shipbuilding-Newport News, primary developer of a simulation-based production control system for the Steel Production Facility.

Researcher, NATO Consultation, Command and Control Agency (NC3A), The Hague: improved tracking algorithms for a passive bistatic radar system by developing air traffic and radar detection simulations, and a track association module, and integrated them with the NC3A real time air traffic tracking system.

Senior Research Scientist, ITT, Advanced Engineering Systems: Supported the Director of Virtual Reality Systems and Research in the Information Technology Division at the Naval Research Laboratory (NRL), by leading the VR Lab software development team, creating software architectures for interoperable, collaborative virtual reality systems, and in developing multi-modal (voice and gesture) input techniques for human computer interaction.

Associate, Daniel H. Wagner Associates: Provided technical expertise to the NRL Advanced Integrated Technology Branch in decision support systems, conducting detailed testing and evaluation of tracking and correlation algorithms. Developed the Surveillance Operational Concept Model for the Applied Physics Laboratory of Johns Hopkins University.

Senior Systems Engineer, Physical Dynamics: Managed operations research and systems analysis support for the DARPA Hypersonic Weapons Technology Program.

His naval career included qualifications as Electronics Technician, Reactor Operator, Naval Flight Officer, and ASW Tactical Coordinator. Responsibilities included, branch officer, division officer, nuclear weapons instructor, Nuclear Weapons Technical Supervisor, COMNAVAIRPAC Force Information Systems Officer for Ashore Activities, and CNO Special Projects Assistant Program Manager. He retired in 1988.

Past and present membership in associations includes: ACM, IEEE, Mensa, Eta Kappa Nu, Tau Beta Pi, Phi Theta Kappa