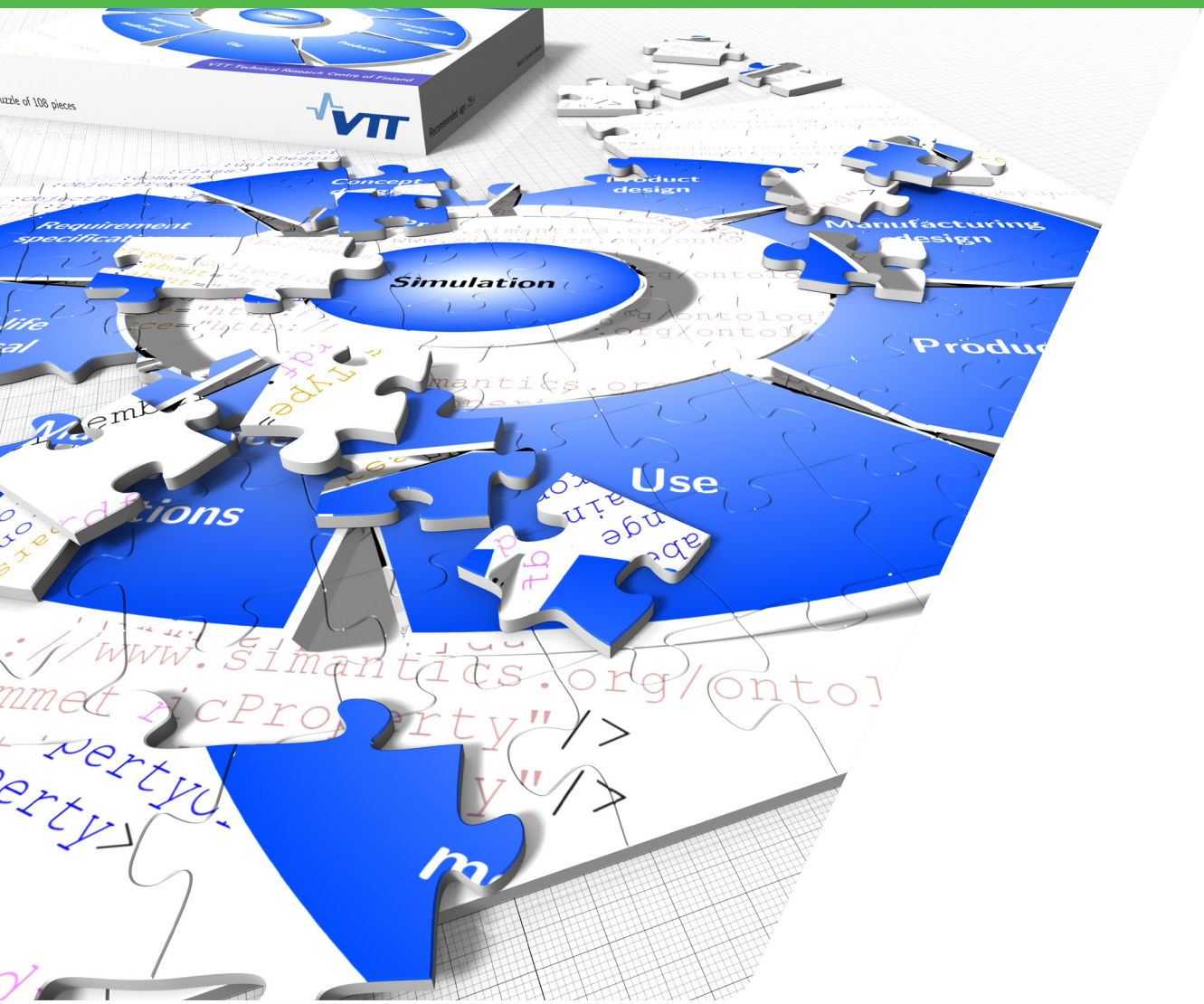
Dissertation

VTT PUBLICATIONS 766



Juha Kortelainen

Semantic Data Model for Multibody System Modelling

VTT

VTT PUBLICATIONS 766

Semantic Data Model for Multibody System Modelling

Juha Kortelainen

Thesis for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium 1383 at Lappeenranta University of Technology, Lappeenranta, Finland on the 19th of August, 2011, at noon.

ISBN 978-951-38-7742-2 (soft back ed.)
ISSN 1235-0621 (soft back ed.)

ISBN 978-951-38-7743-9 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 5, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

Kopijyvä Oy, Kuopio 2011

Juha Kortelainen. Semantic Data Model for Multibody System Modelling [Semanttinen tietomalli monikappaledynamiikan mallitiedon hallinnassa]. Espoo 2011. VTT Publications 766. 119 p. + app. 34 p.

Keywords multibody application, data management, semantic data model, reasoning

Abstract

Multibody system simulation, as one area of system simulation, represents a considerable improvement in predicting machine dynamic performance compared to previous methods that are often based on analytic equations or empirical testing. With multibody system simulation, it is fast and efficient to study the effects of design variables on the dynamic behaviour of the mechanism compared to the experimental approach. Accordingly, the use of multibody system simulation in the design process can decrease the need for physical prototypes, thus accelerating the overall design process and saving resources.

In the product development sense, the interaction between computational tools can be cumbersome. For this reason, multibody system simulation is often omitted in terms of the main stream of product development. Due to the increase of computational resources, the trend is towards extensive usage of simulation, including multibody system simulation, in the product development.

The research emphasis in the field of multibody system dynamics has been on the areas of improved multibody system formulations, such as recursive methods, representation of flexible structures, application of multibody simulation in new areas such as biomechanics, and including multibody simulation into multitechnical and multiphysics simulation. The research on modelling data management and integration approaches concerning tools applied in product design and development has not been in the main stream.

The growth of the World Wide Web and the evolution of Web technologies have multiplied the amount of data and information available on the Internet. In this expansion of information, it has become cumbersome to find and distil the useful information from the data mass. In order to utilise the information available on the Internet and to sort meaningful information out of the data mass, the World Wide Web Consortium began a development project for the next generation Web, the Semantic Web. In this development effort, methods and technologies based on semantic data representation are developed and utilised. These methods and technologies are general enough to be applied to other application areas as well.

This work concentrates on the modelling data management of multibody

system simulation within a simulation-based product process. The main objectives of this work can be summarised as follows: introduce a procedure for managing multibody system modelling data using semantic data model and ontology-based modelling approach, demonstrate that the semantic data model allows application-based reasoning on the model data, and show that ontology-based modelling is able to capture domain knowledge by using semantic data and constraint- and rule-based reasoning.

In this work, the semantic data representation approach is used for describing modelling data of a multibody system. This is accomplished by developing a multibody system modelling ontology, i.e. a semantic data model for multibody system model description, and then applying this ontology to model an example multibody system.

Juha Kortelainen. Semantic Data Model for Multibody System Modelling [Semanttinen tietomalli monikappaledynamiikan mallitiedon hallinnassa]. Espoo 2011. VTT Publications 766. 119 s. + liitt. 34 s.

Avainsanat multibody application, data management, semantic data model, reasoning

Tiivistelmä

Monikappaledynamiikka, yhtenä systeemisimuloinnin erikoisalueena, edustaa merkittävää parannusta koneiden ja mekaanisten järjestelmien suorituskyvyn arvioinnissa verrattuna muihin menetelmiin, jotka usein perustuvat joko analyyttisiin laskelmiin tai kokeellisiin menetelmiin. Verrattuna kokeellisiin menetelmiin monikappaledynaamiikkaa soveltamalla on nopeaa ja tehokasta selvittää eri muuttujien vaikutus mekanismin dynaamisiin ominaisuuksiin. Täten monikappaledynamiikan soveltaminen suunnitteluprosessissa voi vähentää tarvetta todellisten prototyyppien käytölle ja siten nopeuttaa suunnitteluprosessia kokonaisuutena sekä säästää resursseja.

Laskennallisten ohjelmistojen yhteiskäyttö voi olla hankalaa tuotekehityksen näkökulmasta. Tästä syystä tuotesuunnittelun valtavirrassa usein vältetään monikappaledynamiikan käyttöä. Laskennallisten resurssien kasvusta johtuen suuntaus on kuitenkin kohti enenevää simuloinnin käyttöä tuotekehityksessä, mukaan lukien monikappaledynamiikka.

Monikappaledynamiikan tutkimus on painottunut monikappaledynamiikan formuloinnin, kuten esimerkiksi rekursiivisten menetelmien, kehittämiseen, rakenteellisen joustavuuden huomioimiseen, monikappaledynamiikan soveltamiseen uusilla tutkimusalueilla, kuten biomekaniikassa, sekä monikappaledynamiikan soveltamiseen moniteknisessä ja monifysikaalisessa simuloinnissa. Monikappaledynamiikan tiedonhallinta sekä tuotekehitykseen käytettyjen laskennallisten ohjelmistojen ja menetelmien integrointi eivät ole kuuluneet keskeisiin tutkimusalueisiin monikappaledynamiikassa.

Internetin kasvu sekä Web-teknologioiden kehitys ovat moninkertaistaneet Internetissä tarjolla olevan tiedon määrän. Tämän tiedon määrän kasvun myötä oikean tiedon löytämisestä Internetin tietomassasta on tullut hankalaa. Internetissä tarjolla olevan tiedon paremman hyödynnettävyyden mahdollistamiseksi World Wide Web Consortium on käynnistänyt kehityshankkeen nimeltä Semanttinen Web, jossa kehitetään seuraavan sukupolven verkkoa. Kehityshankkeessa kehitetään ja sovelletaan tiedon semanttiseen kuvaukseen perustuvia menetelmiä ja tekniikoita, jotka ovat riittävän yleisiä sovellettaviksi myös muille alueille, kuten laskennallisen tiedon hallintaan.

Tässä työssä keskitytään tuotekehitykseen liittyvän monikappaledynamii-

kan mallitiedonhallintaan. Työn tavoitteet voidaan tiivistää seuraavalla tavalla: esitellä periaate monikappaledynamiikan mallitiedon hallintaan käyttäen semanttista tietomallia sekä ontologiapohjaista mallinnusmenetelmää; osoittaa, että semanttisen tietomallin soveltaminen mahdollistaa sovelluspohjaisen päättelyn monikappaledynamiikan mallitiedosta; sekä osoittaa, että ontologiapohjainen mallintaminen mahdollistaa myös tietämyksen tallentamisen yhdessä mallitiedon kanssa soveltaen semanttista tietomallia sekä rajoite- ja sääntöpohjaista päättelyä. Tämä osoitetaan kehittämällä mallinnusontologia monikappaledynamiikan mallitiedon hallintaan ja soveltamalla tätä ontologiaa monikappaledynaamisen esimerkkitapauksen mallin kuvaukseen.

Preface

In a short period of technological evolution, the world has seen enormous changes. We are now heading into a new era of information and knowledge. Humankind is producing information at a continuously increasing speed – with the wealth of information available, the challenge now facing us is how to distil the desired information and knowledge out of the huge data masses. The development of computation, including progress in algorithms, computational methods, and computer hardware, has provided us with new tools to search and analyse the data masses. On the other hand, new methods and innovations in computational modelling and simulation have drastically increased the speed of producing new data, e.g. for product development.

This doctoral thesis concentrates on combining new methods in knowledge management and system simulation. The former seeks to solve the challenges rising from the latter in order to ensure further improvements in the overall efficiency of applying computational methods for purposes such as product development.

Acknowledgements

I would like to thank several people for helping me with this work. First, I would like to thank Professor Aki Mikkola for providing me with a great environment for my scientific work. In this tranquil but inspiring environment, in the company of the skilled and bright members of the research group, I could concentrate on the subject. I was very lucky to be a member of the research group while writing my thesis. The whole group gave me great support and the positive and excited atmosphere helped me in my work. I would especially like to thank doctoral student Adam Kłodowski with whom I shared an office while writing my thesis. We had countless discussions about modelling and simulation, and some other topics, too. You were a great source of inspiration for my work. I acknowledge Petri Kärkkäinen for his help with manually constructing the OWL model of the Modelica MultiBody Library based on the library documentation.

I would like to thank all my colleagues and supervisors at VTT Technical

Research Centre of Finland for encouraging me to take a break from my hectic work at VTT and concentrate on academic basic research. Professor Tommi Karhela and Doctor Hannu Niemistö from VTT gave me inspiration and excellent advice for my studies, for which I am very grateful. The whole Simantics platform development team has been a source of both great support and motivation for me. Thank you for that. Professor Kari Tammi at VTT is one of my role models for his scientific research work and attitude. All our conversations and debates, both about the content and the process, have helped me during this effort. Thank you for that. In addition, I acknowledge Technology Manager Pekka Koskinen and Chief Research Scientist, Doctor Olli Ventä from VTT for encouraging me in this subject, which is difficult and abstract – at least for a mechanical engineer. You helped me with the arrangements for my research period and provided me with a safety network for this project. Thank you.

Graduate School Concurrent Engineering (CE) Tampere, the Tekes-funded research project Computational Models in Product Life Cycle, and VTT were the major financiers for the period of research and writing the doctoral thesis. The possibility to concentrate on research and writing was crucial for the work. The hectic pace of modern working life is not optimal for academic research and especially for writing a thesis. The funding and Lappeenranta University of Technology provided me with an excellent environment for my work, for which I am very grateful.

I want to express my great gratitude to my entire family for your continuous support and encouragement during my whole life and especially during this work. Without your support, example, and encouragement, ever since I was a little boy in comprehensive school, this would not have happened. And special thanks to my godson Elmo. Due to your young age, you probably do not know yet how you have helped me in this process, but your straightforward and frank child's attitude has shown me what is actually important in life. You all are very dear to me and I am grateful for the support all of you have given me for this work.

Finally, I would like to thank all the developers and contributors behind great open source tools such as TEX, LATEX, AMS-LATEX, TEXlipse, Protégé, NeOn Toolkit, Pellet, Inkscape, Gimp, Blender, and Linux. You have provided scientists with great tools for research work and high-quality publishing. I highly appreciate your efforts and attitude.

Espoo, June 2011

Juha Kortelainen

Contents

List of Tables

List of Figures

Nomenclature

Symbols

$\bar{\alpha}_1$, $\bar{\alpha}_2$, $\bar{\alpha}_3$	The components of the angular acceleration vector, expressed in the body local frame of reference.
$\bar{\boldsymbol{\alpha}}$	Angular acceleration vector, expressed in the body local frame of reference.
$\tilde{\bar{\boldsymbol{\alpha}}}$	The skew-symmetric matrix of angular acceleration.
$\boldsymbol{\theta}$	The vector of generalised orientation coordinates.
$\boldsymbol{\lambda}$	The vector of Lagrange multipliers.
ρ	Density.
Φ_{im_i}	The mth component of the ith atomic formula, where $m_i, i > 0$.
Ψ_i	The ith atomic formula, where $i > 0$.
Ω	A string of quantifiers.
$\bar{\omega}_1$, $\bar{\omega}_2$, $\bar{\omega}_3$	The components of the angular velocity vector, expressed in the body local frame of reference.
$\bar{\boldsymbol{\omega}}$	The angular velocity vector, expressed in the body local frame of reference.
$\tilde{\bar{\boldsymbol{\omega}}}$	Skew-symmetric matrix of angular velocity.
$\mathbf{A}$	Rotation matrix.
a	An instance of an ontology class.
a_i	The ith rule atom in a SWRL rule body or head.
b	An instance of an ontology class.
C_i	A component of the constraint vector.
$\boldsymbol{C}$	A constraint equation in vector form.

$\boldsymbol{C}_t$	The vector of the first partial time derivative of the constraint equations.
$\boldsymbol{C}_{tt}$	The vector of the second partial time derivative of the constraint equations.
$\mathbf{C}_{\boldsymbol{q}t}$	The matrix of the constraint equations differentiated with respect to generalised coordinates and time.
$\mathbf{C}_{\boldsymbol{q}}$	A matrix presenting constraint equations expressed in generalised coordinates. The Jacobian matrix of a multibody system.
c	An instance of an ontology class.
D	An object property of an instance.
E	An object property of an instance.
$\boldsymbol{F}_a$	The vector of applied forces.
$\boldsymbol{F}_e$	The vector of external forces.
$\boldsymbol{F}_i$	The vector of inertial forces.
$\mathbf{G}$	Transformation matrix between time derivative of the orientation coordinates and angular velocities.
$\bar{\mathbf{G}}$	Transformation matrix between time derivative of the orientation coordinates and angular velocities, expressed in the body local frame of reference.
$\mathbf{I}$	Identity matrix.
$\mathbf{I}_{ij}$	Inertia tensor, where indices i and j denote to coordinate directions X, Y, and Z.
L	A positive literal in Horn's clause.
$\mathbf{M}$	Mass matrix.
M	Number of dimensions in dimensional space in mathematics.
m	Mass.
N	Number of dimensions in dimensional space in mathematics.
n	Number of generalised coordinates.
n_c	Number of constraint equations.
O	Number of dimensions in dimensional space in mathematics. Origin of a reference frame.
P	Particle of a body. Point of a body. A property in an ontology.
$\boldsymbol{Q}_c$	The vector of generalised constraint forces.
$\boldsymbol{Q}_e$	The vector of generalised external forces.

$\boldsymbol{Q}_i$	The vector of generalised inertial forces.
$\boldsymbol{Q}_v$	Quadratic velocity vector.
q	Generalised coordinate.
$\boldsymbol{q}$	The vector of generalised coordinates.
$\mathbb{R}$	Dimensional space in mathematics.
$\boldsymbol{R}$	Location of the body local frame of reference presented in the global frame of reference.
$\boldsymbol{r}$	Position vector in the global frame of reference.
$\delta\boldsymbol{r}$	Virtual displacement.
S	A string of quantifiers.
t	Time.
$\bar{u}_1$, $\bar{u}_2$, $\bar{u}_3$	The components of the location vector, expressed in the body local frame of reference.
$\bar{\boldsymbol{u}}$	Location of a point, expressed in the body local frame of reference.
$\tilde{\bar{\mathbf{u}}}$	The skew-symmetric matrix notation of the location of a point.
V	Volume.
δW	Virtual work.
δW_c	Virtual work done by the constraint forces.
δW_e	Virtual work done by the external forces.
δW_i	Virtual work done by the inertial forces.
X	Coordinate direction in Cartesian coordinate system.
x	Coordinate value in X direction in Cartesian coordinate system.
Y	Coordinate direction in Cartesian coordinate system.
y	Coordinate value in Y direction in Cartesian coordinate system.
Z	Coordinate direction in Cartesian coordinate system.
z	Coordinate value in Z direction in Cartesian coordinate system.

Abbreviations

AI	Artificial intelligence.
CFD	Computational fluid dynamics.

CWA	Closed world assumption.
CAD	Computer-aided design.
CAE	Computer-aided engineering.
DAMOS-C	Data model for multibody systems implemented in C.
DBMS	Database management system.
DL	Description Logic.
DSL	Dynamic System Language.
EL	An OWL 2 profile that refers to E++, a description logic derived from OWL 1.1 DL language.
FEM	Finite element method.
IGES	Initial Graphics Exchange Specification, which defines the digital representation and exchange of product definition data among computer-aided design and computer-aided manufacturing (CAD/CAM) systems.
IP	Internet Protocol.
ISO	International Organization for Standardization.
MBS	Multibody system.
MECHAMOS	An MBS analysis tool based on the object-relational database system AMOS II. AMOS stands for Active Mediators Object System.
MbsML	Multibody systems Markup Language, a XML-based neutral data format for multibody system model data representation and exchange.
OWA	Open world assumption.
OWL	Web Ontology Language; OWL 2 is the second version of the language specification.
PDM	Product data management.
PLM	Product lifecycle management.
QL	An OWL 2 profile that refers to Query Language.
RDF	Resource Description Framework.
RDF-S	Resource Description Framework Schema.
RL	An OWL 2 profile that refers to Rule Language.
SEED	Simulation environment for engineering design.
SPARQL	SPARQL Protocol and RDF Query Language.
SQL	Structured Query Language.

STEP	Standard for the exchange of product model data. ISO 10303 standard for product data representation and exchange.
SWRL	Semantic Web Rule Language.
SysML	System Modelling Language.
TCP	Transmission Control Protocol.
UML	Unified Modelling Language.
UNA	Unique name assumption.
URI	Unified resource identifier.
VHDL-AMS	VHDL analog and mixed-signal extensions; VHDL stands for VHSIC Hardware Description Language; VHSIC stands for a U.S. government program to develop very-high-speed integrated circuits.
W3C	The World Wide Web Consortium.
WWW	World Wide Web.
XML	Extensible Markup Language.

Typographical Conventions

In this thesis, the following typographical conventions are used for expressing data model components, the command line input for software applications, and the output of software applications:

Slated text: Ontology names and class and property names in ontologies.

`Typed text`: Software code and software run listings.

Sans serif: Modelling component names, e.g. instances of the ontology in a semantic model.

Roman bold face: In mathematical expressions, the symbol denotes a matrix or a tensor.

Italics bold face: In mathematical expressions, the symbol denotes a vector.

Otherwise, in mathematical expression, symbols are typed in italics text and functions in roman text.

Chapter 1

Introduction

The usage of computational tools for product development has become a standard approach in mechanical engineering design. The development of a complex high technology product, such as a modern passenger car, an aeroplane, or a diesel engine (Figure 1.1), is cumbersome without computer-aided design (CAD) systems, computational analyses, and system simulation. They provide valuable feedback to the designers about the function and performance of the product under development. In addition to the product process, modern research and science rely heavily on computer simulation [3, 4]. The use of modelling and simulation enables researchers to study phenomena that are experimentally difficult to examine, such as material properties at the atomic level [5] or an actively controlled magnetic flux inside an electric motor [6]. It is said in the President's Information Technology Advisory Committee report that computational science has become "the third pillar of scientific inquiry" together with theory and experimentation [7].

1.1 Multibody System Simulation

Multibody system (MBS) simulation, as one area of system simulation, represents a considerable improvement in predicting machine dynamic performance compared to previous methods that are often based on analytic equations or empirical testing. With multibody system simulation, it is fast and efficient to study the effects of design variables on the dynamic behaviour of the mechanism compared to the experimental approach. Accordingly, the use of multibody system simulation in the design process can decrease the need for physical prototypes, thus accelerating the overall design process and saving resources.

The term multibody system simulation refers to the analysis concepts of a system that is constructed of two or more bodies. A system is a composition of elements that together form a whole, and have behaviour, structure

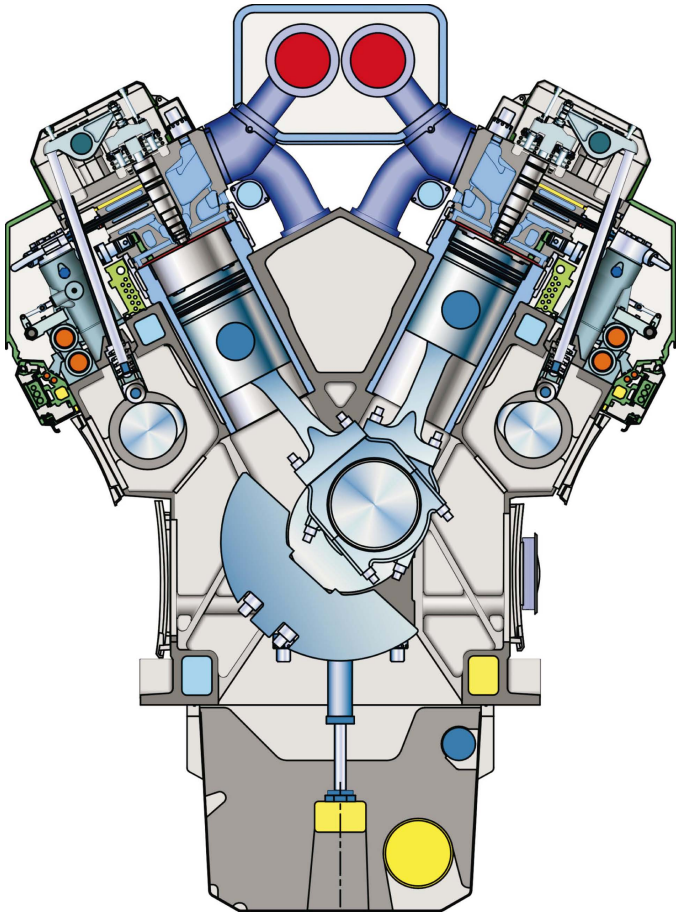


Figure 1.1: Cross-section of the Wärtsilä V32 diesel engine. Image courtesy of Wärtsilä Corporation.

or topology, and function. A multibody system is generally defined as an assembly of bodies which interact with each other through joints that limit the motion possibilities of the interconnected bodies relative to each other [8, 9]. Simulation[1], in turn, refers to the prediction or imitation of the behaviour of a system using a typically simplified model of the original system. Based on the definitions above, it can be concluded that multibody system simulation is an attempt to predict the behaviour of a mechanical system, consisting of a collection of interconnected bodies that can move relative to each other.

[1]The Free On-line Dictionary of Computing: `http://foldoc.org/simulation`

1.2 Simulation in Product Process

The driving force for applying computational methods in an industrial product process is fundamentally economical. The need to improve the quality and functionality of the product requires the implementation of novel and often more complex subsystems. The pressure from the competitors in the markets sets demands on getting products more rapidly to the market while keeping product costs at a reasonable level. The increasing complexity of the product often compounds the interference of the subsystems and may lead to unfavourable side effects, such as noise and vibration problems or unsatisfactory performance of the system. For these challenges, the comprehensive application of simulation, in other words the use of virtual prototypes, offers a remedy by enabling designers of different disciplines to obtain feedback on their work already in the conceptual design phase. Potential problems that are discovered in the early phases of the product development process are usually easier to solve than those found later.

The added value of the usage of modelling and simulation comes in several aspects of the product process. Solely, the modelling of a system helps the designer to structure the system and understand the relationships of the different parts of it. In addition, modelling, particularly when applied based on the strict modelling domain principles such as those of the multibody system domain, is a method to produce compact but detailed documentation of the structure of the system, its components, and functionality. Furthermore, the modelling data combined with the simulation results describe both the system structure and its performance. The understanding of the system relations gained from applying system modelling and the understanding of the dynamics of the system achieved from analysing the results of the simulations together increase the overall product understanding of the designers.

The evolution of computational methods has influenced the product development process and the overall design paradigm. The use of computational models instead of physical prototypes, i.e. using virtual prototypes, has increased the interaction between different design disciplines and has shortened the design iteration turnover time. One of the advances of using computational methods in product development is enabling the interaction and design data exchange between different disciplines from the early phase of the product process and thus providing better premises for concurrent engineering in product design [10]. Figure 1.2 illustrates an idealised process of applying simulation and analysis in machine design. In the Figure 1.2 a, a modification is made to the geometry of a part that radiates a request for update in the multibody system model to compute the overall behaviour of the mechanism, and especially, the loads for the structural analysis. In the Figure, the update process to the virtual product model is represented with a dash line arrow. Applying a struc-

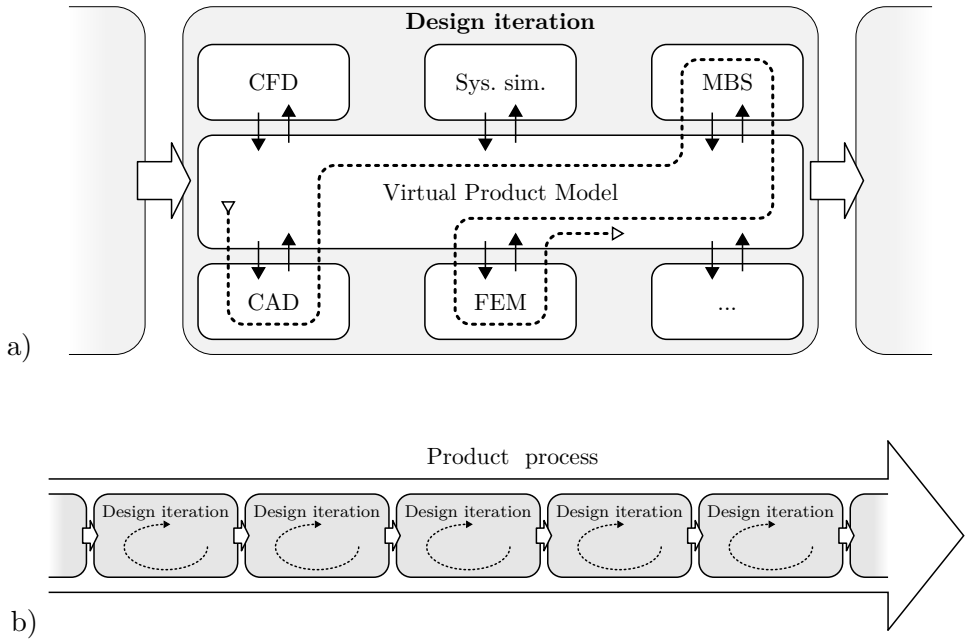


Figure 1.2: a) An example of a design iteration using computational tools. b) The concept of design iterations in a digital product process.

tural analysis for given parts, new design strains and stresses for the part can be computed. Figure 1.2 b illustrates the design process when a virtual product model is applied as the tool to manage modelling and simulation through the product development. For one design iteration, several simulation models and the results of several simulations and analyses are produced. While there may be a large number of design iterations during an overall design process, the amount of both model and results data may become large.

The increasing application of modelling and simulation in the product process has given rise to a new set of challenges for computational systems, namely data management and integration of data sources. To take full advantage of the computational methods, the data required for model composition and the data produced as a result of simulations and analyses need to be fed into the product process seamlessly and efficiently. This has become a critical issue due to a simultaneous increase in the application of simulation, progress in the development of computational methods and software, and increase in computational power that together have enhanced the speed at which the data is produced.

Product data management (PDM) systems have been used for managing distributed product development data in the product process. This data can be e.g. design data, a bill of materials, or product documentation. For enterprise

level and product life cycle length product data management, product life-cycle management (PLM) systems are used. PDM systems have evolved from document management systems to general data management systems with interfaces to other information systems involved in the product process. The scope of PDM systems is to manage the overall information flow related to the product. Thus, the granularity of the data details is evidently high compared to the requirements for simulation model data management. Characteristic to modelling data is the importance of small details and exactness. For design and product data, the existence and data source linking is important. In addition, it is noteworthy that the application of PDM systems, as all document-based systems, often fails to capture the engineering knowledge together with the data [11]. The unstructured knowledge has to be stored as documents, which again separates the knowledge from the model data.

1.2.1 Multibody System Simulation Process

Figure 1.3 represents the typical phases of a multibody system simulation process. For multibody system simulation software, the most common architecture for the application package is to have separate software applications for:

- *Pre-processing*: includes the creation of the model topology, definition of the model components and their parameters, and often setting the solving parameters for the numerical solver.

- *Solving*: the actual computation of numerical results based on the mathematical model of the system.

- *Post-processing*: may include the calculation of additional dependent variables, visualisation of the simulation results, plotting of the results components, and analysis of the results.

The pre-processing phase of the simulation process usually requires the most manual work and is thus the most prone to human errors. In addition, the pre-processing is usually the most time consuming phase, and due to this fact, it is the natural choice as the starting point for the improvement of higher simulation process efficiency. In this work, the emphasis is on the pre-processing phase and especially on the management of the pre-processing model description data.

Different modelling and design disciplines have their own methods of managing the product data while a large number of different design and modelling parameters are involved in the process. It is important to note, however, that all of the simulation models in the process describe features of the same target, i.e. the product. This means that instead of scattering the data and inform-

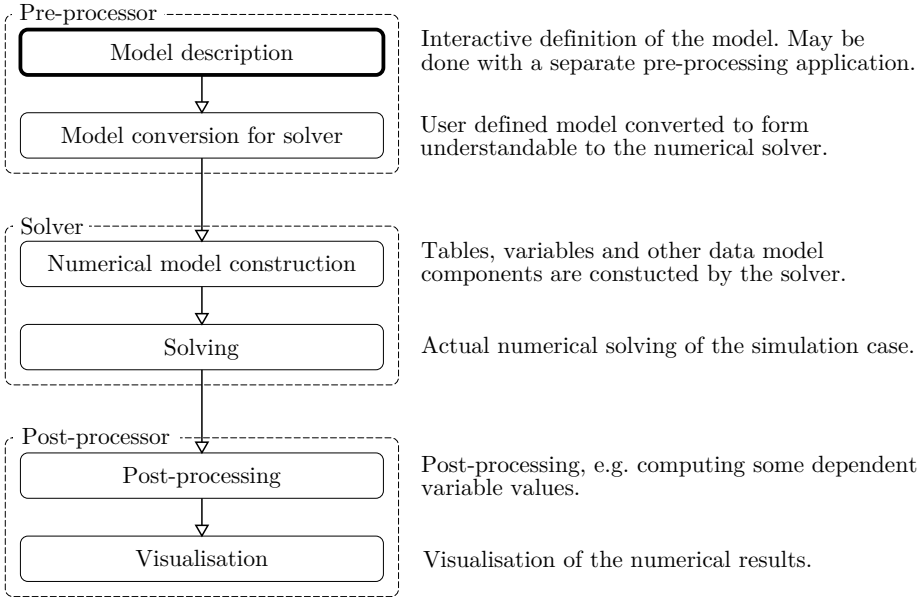


Figure 1.3: Typical phases of a multibody system simulation process.

ation into several sources in different formats, the attempt should be towards one product model for each product.

Figure 1.4 presents the evolution of the application of simulation in the product process. In many cases, the application of simulation in an industrial company begins with solving specific problems in some product detail. This means that simulation is used as a special tool in some detail of the product development process and the development process is driven by other factors. The second phase is to simulate the product by applying virtual prototypes. A virtual prototype contains all major systems of the product, and it can simulate the overall functionality of the product. This phase increases the requirements of the simulation technology compared to the first phase. The third phase does not require much improvement on the technology but requires more changes in the development process itself. Instead of using simulation to validate the development, it is used for specifying requirements for the design process, i.e. modelling and simulation are driving the process instead of being in a minor role. The fourth phase is to use simulation to predict the influence of design decisions on the product life cycle, including technical factors, but also aspects such as the economical and environmental impact. All this remarkably increases requirements for data management in the product process. In this progress, the increase in applying simulation in the product process typically also requires changes to the process itself to take full advantage of the change.

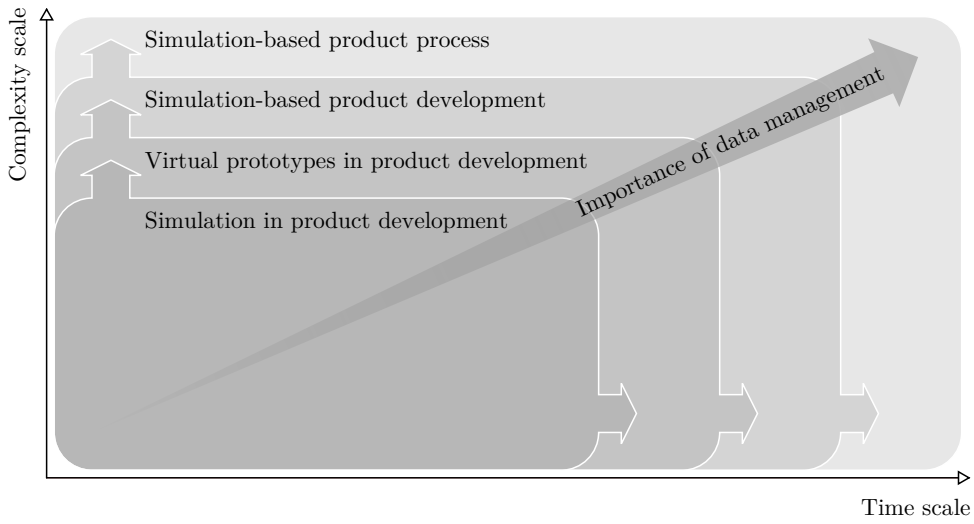


Figure 1.4: Evolution of application of simulation in product process and the increase of the importance of data management.

In the product development sense, the interaction between computational tools can be cumbersome. For this reason, multibody system simulation is often omitted in terms of the main stream of product development. Due to the increase of computational resources, the trend is towards extensive usage of simulation, including multibody system simulation, in the product development. This trend follows the use of finite element method (FEM) tools that are already integrated into many CAD systems. In this trend, communication between a software application of multibody system simulation and a design system, and the data management in particular, becomes crucial. It is important to note that in the current implementations of software application integration, a user is limited to apply the certain existing set of software application. This may be a problem in a networked design model where for example designers in the subcontracting chain need to exchange modelling and design data.

Independent of the technology used, the following requirements hold for modelling data management:

- *knowledge representation*: explicitly capture the data, information, and knowledge of the domain.
- *Knowledge mapping*: link the domain modelling data with data from other domains.
- *Reasoning*: be able to do reasoning on the data to reveal the possibly hidden implicit information.

In addition, for general modelling and simulation data management, the following requirements can be set for data representation:

- *All the data is preserved*: whatever conversion is done to the data it can always be restored to its original form and content.

- *The data representation is flexible and extensible*: domain descriptions evolve and data representation must follow.

- *The form and content of the data does not restrict its use*: using the data should be as easy as possible. This includes retrieving, storing, modifying, mapping, reasoning, and manipulating the data.

There are several methods for doing this, and none of them have proven to be superior to another. However, on the other hand, none of the methods widely used so far have proven to meet all of these requirements equally. Because of the expansion of the use of modelling and simulation in product processes, the emphasis is moving from the first requirements to the balance of all of the requirements.

The research emphasis in the field of multibody system dynamics has been on the areas of improved multibody system formulations, such as recursive methods, representation of flexible structures, application of multibody simulation in new areas such as biomechanics, and including multibody simulation into multitechnical and multiphysics simulation [12, 13, 14]. The research on modelling data management and integration approaches concerning tools applied in product design and development has not been in the main stream.

Figure 1.5 illustrates the evolution of the communication of engineering software applications from early-phase stand-alone analysis tools (Figure 1.5 a) to an integration of analysis tools (Figure 1.5 b and c). Although multibody system simulation is a well-established approach, it can still be seen as a tool for experts [15]. Software applications of multibody system simulation operate under the stand-alone principle, according to which interaction with other tools may be difficult. In practise, the interaction between multibody system simulation software and other engineering applications can be accomplished by employing data exchange in which several exchange formats and many data conversion tools are used. This scenario of the communication of software applications is depicted in Figure 1.5 a, in which the arrows between applications represent the data flow and circles represent intermediate exchange formats and data conversion tools.

Communication barriers between computational tools can be removed by employing standardisation and open exchange formats. To this end, modelling and simulation tools can be coupled using simulation tool interfaces in CAD

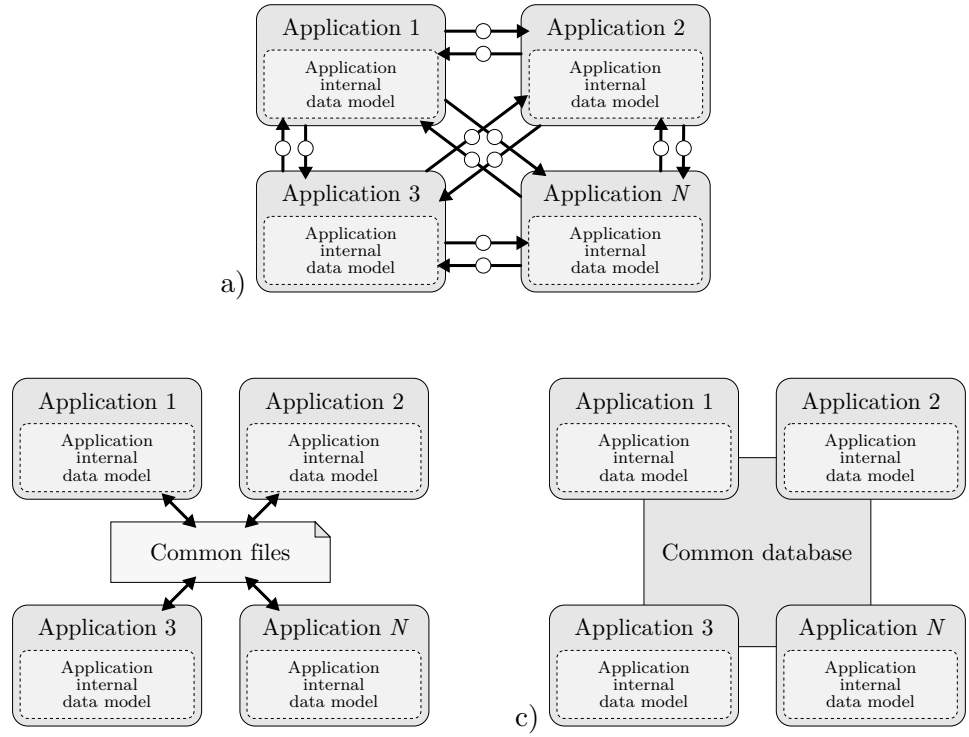


Figure 1.5: Evolution in communication of engineering software applications. a) Independent software applications with specific data formats. b) Unified document-based data exchange using common data format, for example XML. c) Unified dynamic data exchange with common modelling database.

systems and geometry exchange with either proprietary formats, such as ACIS[2] and Parasolid[3], or existing open exchange formats such as ISO 10303 STEP [16, 17] and IGES [18]. Schielen [12] discusses the data transfer strategies between software applications and the requirement of a neutral data model for modelling data transfer. His proposal is based on the use of a document-based approach and the standardisation of the file formats, such as ISO 10303 STEP. However, it is important to note that the use of these formats does not solve the main problem, i.e. seamless exchange of modelling and results data between a set of software applications in the product design process. The rapidly increasing amount of design data and the need to exchange data in a networked operating environment require new approaches to enable the progress in virtual product development.

[2]Website of Spatial Corporation, ACIS geometry kernel:
`http://doc.spatial.com/index.php/Portal:ACIS`

[3]Website of Siemens PLM Software, Parasolid geometry kernel:
`http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml`

The first attempt to create an open specification for an exchange file format for multibody system simulation models was the design of the MbsML language, an Extensible Markup Language (XML) format specification for describing multibody system simulation models, analysis, and solving methods [19, 20, 21] (Figure 1.5 b). Instead of specific solutions for each software application communication need, a common intermediate file format is used, which decreases the number of different format and conversion tools needed. This is depicted in Figure 1.5 b with bidirectional arrows between applications and common files. The MbsML language has been designed to be clear and self-explanatory for a user who knows the basics of multibody system formulation. The structure of the language is simple, and all of the elements in the language are named such that documents are straightforward to follow. The problem with the MbsML language is that the user should be an expert in multibody system simulation in order to adopt the language. This may be particularly problematic when a software application from another domain needs to communicate with the software application of the multibody system domain. Another problem, which this approach omits, is the on-line integration of software applications. With a document-based exchange strategy, the user actively has to push the data to and pull the data from other applications. This may lead to situations where the data is outdated or the same information is stored in several places.

1.2.2 Data Management Software

Modelica[4] is an object-oriented, equation-based language designed specifically for the modelling and simulation of physical systems [1]. The attempt in the development of the languages has been to provide a unified representation for system models and the implementation of formulations. The language is efficient for system modelling but it is not general enough for the efficient capturing of data, information, and knowledge. Another general language for modelling, the System Modelling Language (SysML) [22] is an attempt to unify the description of system models used in systems engineering. The SysML language is based on the Unified Modelling Language (UML) version 2 [23, 24], and is implemented as a UML profile. SysML is intended to be a general modelling language for systems engineering, although the emphasis in its design is on representing model hierarchies rather than describing some domain specific models in detail [25]. There is work in progress to map the model representations in the SysML and Modelica modelling languages [26, 27].

Shephard et al. [28] introduced a concept of using the supervising software SEED for design software integration and communication. This approach may introduce fluent data exchange, but does not capture engineering knowledge

[4]Website of the Modelica Association: `http://www.modelica.org/`

and is related to selected software components, such as CAD systems and computer-aided engineering (CAE) tools. In addition, the SEED system was tailored for the design and development of automotive climate control systems, which is a narrow application area within machine design.

Eben-Chaime et al. [29] introduce a procedure that is based on the usage of an intermediate software layer between standard simulation tools and data management solutions. The intermediate layer in the architecture is divided into three different functional parts: input/output handling, autonomous operations, and control. The input/output handling module enables simulation definition for a set of separate simulations and manages the input parameter and analysis output mapping. The autonomous operations part in this architecture executes the batch runs for simulations that have been set by the user in the input/output handling. The control module manages the simulation runs and communicates with external tools, such as simulation solvers. The proposed architecture seems to lack the ability to store the model data in a centralised manner. In addition, the flexibility to add new and different kinds of simulation models related to the previously used is not described.

The use of a common database for model and result data management, depicted in Figure 1.5 c, resolves many of the problems of the document-based approach. Applications using a common database have access to data that is up-to-date, and thus the problem of incoherent design data is solved. In order to accomplish a common database, a number of designs of a common modelling database have been proposed. Tisell and Orsborn [30, 31] have proposed an approach in which the data is stored into a tailored object-relational database system called MECHAMOS. In this system, the multibody system modelling data is stored in an analytical form that enables case specific optimisation for the numerical model of the system. In the prototype implementation of the MECHAMOS system, external computational tools, such as MATLAB[5] and Maple[6], are used as solvers. The proposed system is tailored for multibody system simulation and is thus not suitable for other simulation disciplines. Daberkow and Kreuzer [32] have proposed a common modelling kernel called DAMOS-C for a CAD and dynamic simulation environment in order to integrate different tools. Their approach applies the object-oriented data model for modelling data exchange between different applications in the modelling and simulation process, such as CAD software and multibody system solvers. The DAMOS-C system focuses on the multibody simulation domain including an interface to external CAD systems.

Kübler and Schiehlen [33] discuss the simulation of complex engineering systems, concentrating on mechatronic systems and the integration of different models and simulation tools. They have divided the model description

[5]Website of Matlab software: `http://www.mathworks.com/products/matlab/`
[6]Website of Maple software: `http://www.maplesoft.com/products/Maple/index.aspx`

into three different levels: physical, mathematical, and behavioural model description. These levels refer to the description of the modelled system, the transformation of the system model into a mathematical formulation based on selected methods, and numerical results of the simulation based on the mathematical model, respectively. Several approaches are suggested for representing the physical model, such as ISO 10303 STEP, VHDL-AMS, and DSL, but none of these have gained any popularity in the community or among the software vendors for multibody system simulation. In the article, the focus is on the block representation of coupled multibody system models at the mathematical level.

There have been several proposals for model and design data management in the product development process using a centralised upper level model. Hoffman and Joan-Arinoy [34] propose a product master model approach to map the CAD data with, as they call it, downstream application processes, such as simulations and analyses. Their proposal concentrates on geometry management, but is not restricted to it. In the proposal, each software tool, such as the CAD system and the simulation tools, may have their own, possibly proprietary, model repositories, and only the data that is relevant to other software tools is communicated. In the approach, each software tool in the system communicates with the master model, retrieves data from it and sends modification requests e.g. for the CAD geometry. This approach, even though flexible as to the often closed proprietary software tools, lacks the ability to manage the model data in a centralised way. In this approach, all of the detailed modelling data is managed by the simulation tool in question. In addition, the system does not capture any additional engineering knowledge that may be present in the process. Siemers et al. [35] proposed a meta-model approach for managing the integration and co-simulation of different simulation tools. In this approach, the simulation software specific data is represented in the format of the software in question and only the data required for model integration and the communication between different software tools in simulation is described. Also this approach lacks the centralised data management and engineering knowledge capturing features.

1.3 Semantic Data Representation

The concepts of data, information, and knowledge can be defined in a hierarchical manner. The data is at the lowest level in the hierarchy, and thus data is defined as being values of parameters or variables, such as dimensions and quantities. Information is at a higher hierarchical level compared to data. Information is a collection of data forming a whole and describing a concept or entity. Thus, data needs to be interpreted to have a meaning. Knowledge

in this hierarchy is at the highest level. In addition to data and information, knowledge adds experience to information. The present data management approaches for modelling and simulation focus on capturing data and, in some cases, information. The challenge of data representation is to be able to capture knowledge in the data representation.

The growth of the World Wide Web (WWW) and the evolution of Web technologies have multiplied the amount of data and information available on the Internet. In this expansion of information, it has become cumbersome to find and distil the useful information from the data mass. While the Web search services develop further, the amount of search mass seems to keep growing even more rapidly. In order to utilise the information available on the Internet and to sort meaningful information out of the data mass, the World Wide Web Consortium (W3C), an organisation to maintain and develop technologies for WWW, began a development project for the next generation Web, the Semantic Web. As the name implies, the next generation Web focuses on the meaning of the data, its semantics, and methods to retrieve the hunted information out of the data mass. The main effort of the Semantic Web project has been to develop a set of fundamental technologies that can be used within the framework of the existing Internet infrastructure and enable this vision. The technologies enable software applications to combine and refine data from a number of sources. The Semantic Web integrates separate data sources, otherwise loosely connected, into a single network of information [36]. The latest technologies of knowledge representation and data semantics to create a framework of specifications, languages, and tools for this purpose are employed in the Semantic Web project [37]. To this end, concepts and terms can be defined explicitly and definitions can be domain and context specific. Definitions can contain rules and validity constraints increasing the value of the data. The Semantic Web technologies are general enough also to be adopted to other engineering domains. While the Semantic Web is still evolving, the technologies developed for it are adopted to different applications, including product data management [38].

The idea of a thinking machine, artificial intelligence (AI), has been under research since the early 1950s [39]. The research area has several branches, such as expert systems, natural language understanding, machine learning, and knowledge representation [40]. The work in the research area of knowledge representation concentrates on developing methods to store human knowledge in machine-usable form. This is the fundamental requirement for all systems that use knowledge and apply reasoning to it. The research in the knowledge representation area concentrates on methods to represent and use information and knowledge formally and in machine-readable form [41]. This is required in artificial intelligence in which a computer system solves problems based on available knowledge, in machine-readable form, of a specific domain. Artificial

intelligence can be applied for example to expert systems to speed up the solving process of e.g. system maintenance or to provide additional support for the innovation process. It is important to note that the crucial element of an expert system is the knowledge database and, particularly, the representation of human knowledge.

Semantic Web technologies are not ideal for system modelling data management, as will be discussed in more detail in chapter 2. This is mainly due to the fundamental objective of being able to capture imperfect data and knowledge and thus applying the open world assumption (OWA) e.g. in the Web Ontology Language (OWL). In addition, some data structures, such as vectors and arrays, are not included into the built-in data types of the OWL, although they can be represented as user defined data types. After all, the Semantic Web technologies have been selected as the implementation foundation of this work due to their maturity and wide application in research and industry.

1.4 Objectives of the Thesis

In Figure 1.6, a concept of using knowledge representation technologies for storing data and domain knowledge, and for sharing it in an organisation or community is illustrated. In this concept, all of the data is in one storage to keep the data up-to-date and to avoid overlapping and redundancy. Users and tools, such as solvers and analysis tools, can access the data storage. In addition, external data sources can be connected to the system. This concept raises many questions, such as the structure of the data model, access and version control, the integrity of the data and knowledge, and the scalability of the system. In this work, the focus is on the main principles of storing modelling data and knowledge in machine-readable form, and implementation questions of such a system are bypassed. Implementation and technical solutions for such a system are a matter of future research.

This work concentrates on the modelling data management of multibody system simulation within a simulation-based product process. The main objectives of this work can be summarised as follows:

1. Introduce a procedure for managing multibody system modelling data using semantic data model and ontology-based modelling approach,
2. Demonstrate that the semantic data model allows application-based reasoning on the model data, and
3. Show that ontology-based modelling is able to capture domain knowledge by using semantic data and constraint- and rule-based reasoning.

To this end, the domain of multibody system dynamics is narrowed to concern rigid bodies only. This is due to the fact that the description of structural

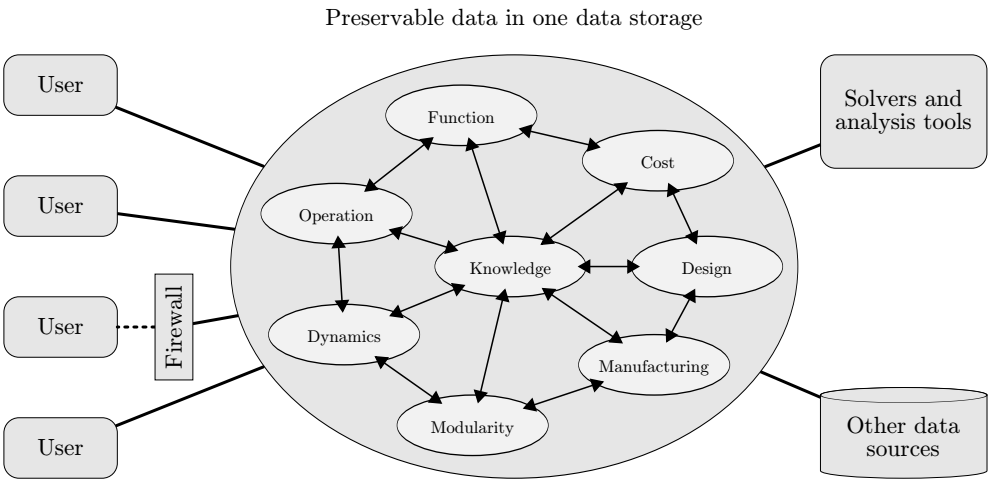


Figure 1.6: Vision of the model for common data management for modelling and simulation.

flexibility increases the complexity of the domain description, whereas it does not introduce new information about the general applicability of the method. In addition, the developed modelling ontologies are general and simplified, and they are not designed for practical industrial modelling. In this work, the emphasis is on the generality and extensibility of the concept.

Although the research deals with multibody system dynamics, the focus of the study is on modelling data management. For this reason, the mathematical background associated with multibody dynamics is explained at a general level only. The numerical examples are used for demonstrating that the objectives of the study are achieved in multibody dynamics.

1.5 Scientific Contribution

In this work, the semantic data representation approach is used for describing modelling data of a multibody system. This is accomplished by developing a multibody system modelling ontology, i.e. a semantic data model for multibody system model description, and then applying this ontology to model an example multibody system. The semantic data model allows the same data representation method to be used for the data representation of a wide variety of system models. This, on the other hand, enables the model data from different domains to be mapped together. This work represents a method which distinguishes the modelling data and the computational tools, such as solvers. This is an important aspect for data management in terms of preservability, i.e. the ability to archive and later reuse the valuable information about the

product and its functionality. Aspects of general system modelling and simulation combined with methods and technologies under research and development in general knowledge representation and data management are considered. The particular emphasis of this study is on the future bottlenecks of the application of modelling and simulation in product development.

The semantic data modelling approach used in this work is general and it allows different types of data and knowledge to be captured into the same data representation. The data model provides natural mechanisms to map the data of one domain with the data of other domains, which is the method to capture the knowledge into the data system. The principles of semantic data representation are general enough to represent data in different forms and structures, whether it is structured in a strict manner or has a complex and unstructured form. In practice, that could mean e.g. mapping the multibody system modelling data of a mechanical system with the engineering experience of that specific application domain.

This study presents a procedure to capture the domain knowledge of multibody system modelling into the system. The use of semantic data representation and semantic reasoning technologies enables inferring on modelling data to e.g. validate the data against modelling rules. In this work, semantic constraints, i.e. restrictions of data element connectivity or type, and rules are set for the semantic modelling data, and reasoning against these rules is applied to demonstrate the applicability of the method. This work presents a general semantic validation mechanism for multibody system modelling data, which enables both general model validation and modelling case specific modelling constraints.

The methods introduced in this study are general and applicable to other modelling and simulation domains. The use of semantic data representation for capturing knowledge is being extensively researched in several fields. This work applies existing technologies of the Semantic Web to new targets, and discusses the suitability of the method for modelling data representation and how the fundamental principles of the semantic data representation, such as the open world assumption, are related to this purpose.

The scientific contribution of this work can be summarised as follows:

1. The application of semantic data representation is introduced by developing an ontology for multibody system modelling and applying this ontology for modelling a multibody system.

2. The use of one simple but expressive data representation technique for both plain modelling data and engineering knowledge can be described using the presented method. The presented method enables the separation of modelling data and engineering knowledge from the computational tools, such as solver and analysis tools.

3. The implementation of both general and case specific modelling rules and the application of semantic reasoning to the modelling data are represented and discussed.

1.6 Structure of the Thesis

In the present chapter 1, the overall context of the thesis is described, namely the data management in a simulation-based product process, as well as its relevance. The mathematical formulation for solving multibody system dynamics is represented in chapter 2. In chapter 3, a multibody system modelling ontology is developed. In addition, the basic data model of the Modelica MultiBody library is presented in semantic form. The application of the multibody system modelling ontology is demonstrated with an example in chapter 4. Then, the suitability of the semantic approach is discussed and the future work in this research area is identified in 5. The conclusions of the work are presented in chapter 6.

Chapter 2

Multibody System Simulation and Semantic Data Management

2.1 Multibody System Simulation

2.1.1 Background

The simulation model of a system is an approximation of the real world system the model represents. An example of such a simplification is the representation of a mechanical system in the multibody system simulation domain. The simplification can been seen as a projection from a real world parameter space $\mathbb{R}^M$ to an ideal parameter space $\mathbb{R}^N$, i.e.

$$\mathbb{R}^M \to \mathbb{R}^N,\ M > N\,, \tag{2.1}$$

where M is the dimension of the real world parameter space and N the dimension of the parameter space of the general multibody system modelling domain. On the other hand, a different numerical formulation of the system in the same computational domain, in an ideal case, is a conversion from one representation to another, where no initial information about the source system representation is lost, or in another way

$$\mathbb{R}^N \to \mathbb{R}^O,\ N \simeq O\,, \tag{2.2}$$

where O is the dimension of the parameter space of a numerical formulation-specific modelling domain. In other words, the basic information required for describing the real world system in some numerical formulation in the simulation domain should be sufficient for describing the same system in another

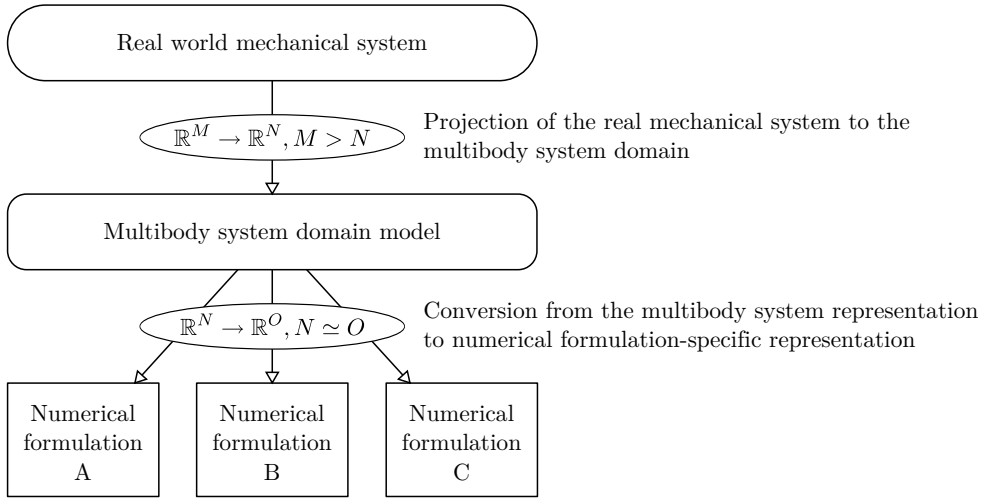


Figure 2.1: System representation phases from a real mechanical system via a general multibody system modelling representation to a numerical formulation-specific representation.

numerical formulation in the same simulation domain. This may require the information to be presented in a different form. Figure 2.1 illustrates the system representation phases from a real world system to a numerical formulation-specific representation.

A mechanical system can be classified on the basis of the type and dependencies of its constraints. A system is defined as holonomic if all constraints of the system are holonomic. A constraint is holonomic, if it can be expressed as

$$f(q_1, q_2, \ldots, t) = 0\,, \tag{2.3}$$

where q_i are the generalised coordinates of the connected bodies and t is time. A constraint that cannot be expressed in this manner is nonholonomic, and a system having nonholonomic constraints is a nonholonomic system. Furthermore, if a constraint is explicitly dependent on time, it can be classified as rheonomic, and if the function of a constraint is not explicitly dependent on time, the constraint can be classified as scleronomic. [42]

Multibody system dynamics relies upon classical mechanics as well as computation dynamics [12]. The equations of motion needed when analysing the dynamic responses of the multibody system, can be derived by using global or topological formulations. Global formulations can be implemented to computers in a straightforward manner due to the fact that both open and closed kinematic loops can be solved with the same algorithms. Global methods are based on the use of generalised coordinates that describe the position and orientation of each body. The equation of motion of a body can be derived by

expressing inertial and externally applied forces in terms of generalised coordinates. In practise, this can be accomplished using the concept of virtual work. In order to study a constrained system of bodies, the constraint equations that couple the generalised coordinates need to be defined. Constraint equations can be augmented to the equations of motion by using the penalty method or the augmented Lagrangian formulation [12, 8, 9]. Alternatively, constraint equations can be embedded in the equations of motion by using the coordinate partitioning approach [9, 43]. A drawback of global methods is that they use a large number of generalised coordinates, and for this reason, a global method may be computationally inefficient. In topological formulations, the topology of a mechanism is utilised in order to improve numerical efficiency [44]. A topology-based approach uses relative coordinates, allowing the kinematic analysis to be accomplished recursively. In this kinematic analysis approach, one body is studied at a time in a kinematic chain. The number of generalised coordinates required in the approach is equal to the number of degrees of freedom in the open kinematic chains of the system [45].

The equations of motion define a multibody system mathematically and they are the foundation for simulating multibody system behaviour. In the following sections, the equations of motion are derived for a multibody system of rigid bodies in three dimensions. These equations will show the fundamental elements of the formulation of multibody system dynamics and thus the necessary components to be described for the multibody system simulation model. The formulation follows the principles presented in [46] and [47]. For simplicity, only holonomic mechanical systems are considered in the following formulation of the equations of motion. In addition, only global methods are considered, and e.g. topological methods are not presented, because global methods are widely used and generally applicable for different kinds of simulation purposes. The mechanical system formulation presented below does not affect the fundamental information required to present a multibody system.

2.1.2 Equations of Motion for a Multibody System

Kinematics of Multibody Systems

In the following, the concept of the local frame of reference for a rigid body is introduced. Any particle P of a rigid body can be presented using the location and orientation of the local frame of reference of the body and the location of the particle in this local frame of reference. Figure 2.2 illustrates the location of a particle of a rigid body in the global frame of reference as

$$\boldsymbol{r} = \boldsymbol{R} + \mathbf{A}\bar{\boldsymbol{u}}\,, \tag{2.4}$$

where $\boldsymbol{r}$ is the location vector of particle P expressed in the global frame of reference, $\boldsymbol{R}$ is the location vector of the body frame of reference expressed in

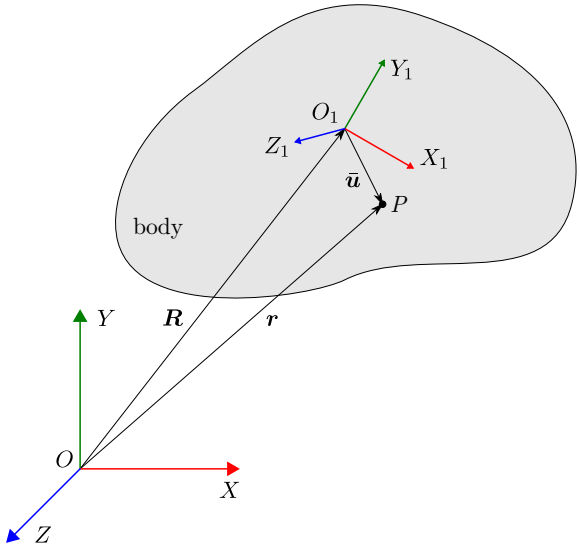


Figure 2.2: Location of a particle of a rigid body in the global frame of reference.

the global frame of reference, $\mathbf{A}$ is the rotation matrix, and $\bar{\boldsymbol{u}}$ is the location of particle P expressed in the body local frame of reference. Due to the assumption of an ideally rigid body, the location of every particle of the body is constant in the local frame of reference and can be expressed explicitly with Equation (2.4). To express the location of particle P in the global frame of reference using the location of the body frame of reference and the constant location of particle P in the body local frame of reference, a coordinate conversion in the form of rotation matrix $\mathbf{A}$ has to be used.

There are several methods to present orientation, of which the Euler angles and quaternions are often used in mechanical engineering. Of these two, Euler angles are more common in interactive use, due to the intuitive convention of use. The concept of Euler angles is based on a sequence of single rotation operations around a rotation axis, in three dimensions X, Y, and Z. The rotation operations are not commutative, and there are 12 different combinations for the sequences that can be selected, e.g. X-Y-Z or Z-X-Z [48, 46].

The selection of location coordinates in three-dimensional space is straightforward, but for expressing orientations there are more options, such as Euler angles or quaternions. The location and orientation coordinates together form the vector of generalised coordinates

$$\boldsymbol{q} = \begin{bmatrix} x & y & z & \boldsymbol{\theta}^T \end{bmatrix}^T , \tag{2.5}$$

where x, y, and z are the location coordinates in X, Y, and Z coordinate axis direction, respectively, and $\boldsymbol{\theta}$ is the vector of orientation coordinates. The

following representation is independent of the selected generalised coordinates for orientation.

Rotation matrix $\mathbf{A}$ is orthogonal [48], which means that its transpose is its inverse, and thus $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Differentiating this notation with respect to time and doing some equation manipulation gives

$$\mathbf{A}^T\dot{\mathbf{A}} = -\left(\mathbf{A}^T\dot{\mathbf{A}}\right)^T . \tag{2.6}$$

The above says that matrix $\mathbf{A}^T\dot{\mathbf{A}}$ is equal to its negative transpose, which means that this matrix is a skew-symmetric matrix. This skew-symmetric matrix can be expressed as

$$\mathbf{A}^T\dot{\mathbf{A}} = \tilde{\bar{\boldsymbol{\omega}}} = \begin{bmatrix} 0 & -\bar{\omega}_3 & \bar{\omega}_2 \\ \bar{\omega}_3 & 0 & -\bar{\omega}_1 \\ -\bar{\omega}_2 & \bar{\omega}_1 & 0 \end{bmatrix} , \tag{2.7}$$

where $\bar{\omega}_i$ are the angular velocities expressed in the body local frame of reference. The components of matrix $\tilde{\bar{\boldsymbol{\omega}}}$ can also be expressed as the angular velocity vector

$$\bar{\boldsymbol{\omega}} = \begin{bmatrix} \bar{\omega}_1 & \bar{\omega}_2 & \bar{\omega}_3 \end{bmatrix}^T . \tag{2.8}$$

From Equation (2.7), the time derivative of the rotation matrix can be written as

$$\dot{\mathbf{A}} = \mathbf{A}\tilde{\bar{\boldsymbol{\omega}}} . \tag{2.9}$$

For the formulation of the equations of motion, it is essential to note that the angular velocity vector $\bar{\boldsymbol{\omega}}$ is not the time derivative of the vector of orientation coordinates $\dot{\boldsymbol{\theta}}$.

To represent velocity using generalised coordinates $\boldsymbol{q}$, the angular velocity vector $\bar{\boldsymbol{\omega}}$ needs to be transformed to the time derivatives of the orientation coordinates. To this end, transformation matrix $\bar{\mathbf{G}}$ is introduced, so that

$$\bar{\boldsymbol{\omega}} = \bar{\mathbf{G}}\dot{\boldsymbol{\theta}} . \tag{2.10}$$

For transformation matrix $\bar{\mathbf{G}}$ in the body local frame of reference applies $\bar{\mathbf{G}} = \mathbf{A}^T\mathbf{G}$, where $\mathbf{G}$ is the transformation matrix in the global frame of reference.

The definition for the time differential of rotation matrix $\mathbf{A}$ in Equation (2.9) is used for defining the second time derivative of the rotation matrix

$$\begin{aligned} \ddot{\mathbf{A}} &= \dot{\mathbf{A}}\tilde{\bar{\boldsymbol{\omega}}} + \mathbf{A}\dot{\tilde{\bar{\boldsymbol{\omega}}}} \\ &= \mathbf{A}\tilde{\bar{\boldsymbol{\omega}}}\tilde{\bar{\boldsymbol{\omega}}} + \mathbf{A}\tilde{\bar{\boldsymbol{\alpha}}} , \end{aligned} \tag{2.11}$$

where $\tilde{\bar{\boldsymbol{\alpha}}}$ is a skew-symmetric rotational acceleration matrix

$$\tilde{\bar{\boldsymbol{\alpha}}} = \begin{bmatrix} 0 & -\bar{\alpha}_3 & \bar{\alpha}_2 \\ \bar{\alpha}_3 & 0 & -\bar{\alpha}_1 \\ -\bar{\alpha}_2 & \bar{\alpha}_1 & 0 \end{bmatrix} , \tag{2.12}$$

where $\bar{\alpha}_i$ are the angular accelerations expressed in the body local frame of reference. The components of the rotational acceleration matrix can be expressed as the angular acceleration vector $\bar{\boldsymbol{\alpha}} = \begin{bmatrix}\bar{\alpha}_1 & \bar{\alpha}_2 & \bar{\alpha}_3\end{bmatrix}^T$. Rotational acceleration can be derived from the angular velocity differentiating Equation (2.10) relative to time

$$\bar{\boldsymbol{\alpha}} = \dot{\bar{\boldsymbol{\omega}}} = \dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} + \bar{\mathbf{G}}\ddot{\boldsymbol{\theta}}\,, \tag{2.13}$$

and the angular acceleration vector can expressed as $\bar{\boldsymbol{\alpha}} = \begin{bmatrix}\dot{\bar{\omega}}_1 & \dot{\bar{\omega}}_2 & \dot{\bar{\omega}}_3\end{bmatrix}^T$.

Velocity in Generalised Coordinates The velocity of particle P of the body, i.e. the time derivative of location $\boldsymbol{r}$, presented in Equation (2.4), can be written as

$$\dot{\boldsymbol{r}} = \dot{\boldsymbol{R}} + \mathbf{A}\tilde{\bar{\boldsymbol{\omega}}}\bar{\boldsymbol{u}}\,. \tag{2.14}$$

Applying Equation (2.10) to the above Equation gives

$$\dot{\boldsymbol{r}} = \dot{\boldsymbol{R}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\dot{\boldsymbol{\theta}}\,, \tag{2.15}$$

where $\tilde{\bar{\mathbf{u}}}$ is the skew-symmetric matrix form of the location of particle P in the body local frame of reference

$$\tilde{\bar{\mathbf{u}}} = \begin{bmatrix} 0 & -\bar{u}_3 & \bar{u}_2 \\ \bar{u}_3 & 0 & -\bar{u}_1 \\ -\bar{u}_2 & \bar{u}_1 & 0 \end{bmatrix}\,, \tag{2.16}$$

where $\bar{u}_i$ are the components of vector $\bar{\boldsymbol{u}}$ expressed in the body local frame of reference. Partitioning Equation (2.15) on the basis of the generalised coordinates gives for velocity

$$\dot{\boldsymbol{r}} = \begin{bmatrix}\mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{R}} \\ \dot{\boldsymbol{\theta}}\end{bmatrix}\,. \tag{2.17}$$

Acceleration in Generalised Coordinates The acceleration of particle P of the body, i.e. the second time derivative of location $\boldsymbol{r}$, can be calculated with Equation (2.15)

$$\begin{aligned}\ddot{\boldsymbol{r}} &= \ddot{\boldsymbol{R}} - \dot{\mathbf{A}}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\dot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\ddot{\boldsymbol{\theta}} \\ &= \ddot{\boldsymbol{R}} - \mathbf{A}\tilde{\bar{\boldsymbol{\omega}}}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\dot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\ddot{\boldsymbol{\theta}}\,.\end{aligned} \tag{2.18}$$

By reorganising the above equation and using Equation (2.10), acceleration can be presented as

$$\begin{aligned}\ddot{\boldsymbol{r}} &= \ddot{\boldsymbol{R}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\ddot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\boldsymbol{\omega}}}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\dot{\boldsymbol{\theta}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} \\ &= \begin{bmatrix}\mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{R}} \\ \ddot{\boldsymbol{\theta}}\end{bmatrix} + \begin{bmatrix}0 & -\mathbf{A}\tilde{\bar{\boldsymbol{\omega}}}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{R}} \\ \dot{\boldsymbol{\theta}}\end{bmatrix}\,.\end{aligned} \tag{2.19}$$

Dynamics of Multibody Systems

The equations of motion for a multibody system are based on Newton's second law of motion and d'Alembert's principle for virtual work. Newton's second law of motion is

$$\boldsymbol{F}_a = \int_V \rho \ddot{\boldsymbol{r}} \, dV \,, \tag{2.20}$$

where $\boldsymbol{F}_a$ is the vector of force applied to the particle, ρ is density, and V is volume. D'Alembert's principle for virtual work is

$$\delta W = \sum \left(\boldsymbol{F}_a^T - \int_V \rho \ddot{\boldsymbol{r}}^T \, dV \right) \delta \boldsymbol{r} = 0 \,, \tag{2.21}$$

where δW is the virtual work and $\delta \boldsymbol{r}$ is the virtual displacement. Newton's formulation in Equation (2.20) of classical mechanics describes the dynamics of free particles (unconstrained, noninertial particles with finite mass) and d'Alembert's principle considers also the reaction forces in the constraints. In Equations (2.20) and (2.21), force $\boldsymbol{F}_a$ denotes applied forces expressed in Cartesian coordinates.

Based on d'Alembert's principle and Newton's second law of motion, the virtual work done by the inertial forces can be written as

$$\delta W_i = \boldsymbol{F}_i^T \, \delta \boldsymbol{r} = \int_V \rho \ddot{\boldsymbol{r}}^T \, dV \delta \boldsymbol{r} \,, \tag{2.22}$$

where $\boldsymbol{F}_i$ is the vector of inertial forces, expressed in Cartesian coordinates. Applying the rule of chain derivation to the virtual displacement relative to the vector of generalised coordinates

$$\delta \boldsymbol{r} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \delta \boldsymbol{q} \tag{2.23}$$

and inserting this to Equation (2.22) gives

$$\delta W_i = \int_V \rho \ddot{\boldsymbol{r}}^T dV \, \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{q}} \delta \boldsymbol{q} \,. \tag{2.24}$$

For generalised inertial forces, acceleration $\ddot{\boldsymbol{r}}$ in Equation (2.24) is expressed in

generalised coordinates $\boldsymbol{q}$. Now, the generalised inertial forces become

$$\begin{aligned}\boldsymbol{Q}_i^T = &\int_V \rho \Bigg(\ddot{\boldsymbol{q}}^T \begin{bmatrix} \mathbf{I} \\ -\bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \mathbf{A}^T \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \end{bmatrix} \\ &+ \dot{\boldsymbol{q}}^T \begin{bmatrix} 0 \\ -\bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\boldsymbol{\omega}}^T \mathbf{A}^T - \dot{\bar{\mathbf{G}}}^T \tilde{\bar{\mathbf{u}}}^T \mathbf{A}^T \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \end{bmatrix} \Bigg) dV \\ = &\int_V \rho \Bigg(\ddot{\boldsymbol{q}}^T \begin{bmatrix} \mathbf{I} \\ -\bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \mathbf{A}^T \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \end{bmatrix} \\ &+ \begin{bmatrix} -\dot{\boldsymbol{\theta}}^T \bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\boldsymbol{\omega}}^T \mathbf{A}^T - \dot{\boldsymbol{\theta}}^T \dot{\bar{\mathbf{G}}}^T \tilde{\bar{\mathbf{u}}}^T \mathbf{A}^T & \dot{\boldsymbol{\theta}}^T \bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\boldsymbol{\omega}}^T \tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} + \dot{\boldsymbol{\theta}}^T \dot{\bar{\mathbf{G}}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \end{bmatrix} \Bigg) dV \,, \end{aligned} \tag{2.25}$$

where $\boldsymbol{Q}_i$ is the vector of generalised inertial forces. From this, the following components can be separated for simplifying the manipulation of the equations later:

$$\mathbf{M} = \int_V \rho \left(\begin{bmatrix} \mathbf{I} & -\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \\ -\left(\mathbf{A}\tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}}\right)^T & \bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\bar{\mathbf{u}}}\bar{\mathbf{G}} \end{bmatrix} \right) dV \tag{2.26}$$

$$\boldsymbol{Q}_v = -\int_V \rho \left(\begin{bmatrix} \mathbf{A}\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\tilde{\bar{\mathbf{u}}} - \mathbf{A}\tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} \\ \bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\bar{\mathbf{u}}}\dot{\bar{\mathbf{G}}}\dot{\boldsymbol{\theta}} - \bar{\mathbf{G}}^T \tilde{\bar{\mathbf{u}}}^T \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\bar{\boldsymbol{u}} \end{bmatrix} \right) dV \,, \tag{2.27}$$

where $\mathbf{M}$ is the mass matrix and $\boldsymbol{Q}_v$ is the quadratic velocity vector. In Equation (2.26), the term $\int_V \rho \tilde{\bar{\mathbf{u}}}^T \tilde{\bar{\mathbf{u}}} dV$ is the inertia tensor and it is usually shortened as $\mathbf{I}_{ij}$, where the terms of the inertia tensor are the moments of inertia of the body, and indexes i and j denote to coordinate directions X, Y, and Z.

From Equations (2.25), (2.26), and (2.27) it follows that the generalised inertial forces can be expressed as

$$\boldsymbol{Q}_i^T = \ddot{\boldsymbol{q}}^T \mathbf{M} - \boldsymbol{Q}_v^T \,. \tag{2.28}$$

The procedure for defining external forces $\boldsymbol{F}_e$ and further the vector of generalised external forces $\boldsymbol{Q}_e$ corresponds to the above procedure for representing inertial forces $\boldsymbol{F}_i$ in generalised form $\boldsymbol{Q}_i$.

The equations of motion for a multibody system of rigid bodies can be obtained by applying d'Alembert's principle and setting the virtual work done by inertial forces equal to the virtual work done by external forces

$$\delta W_i = \delta W_e \tag{2.29}$$

$$\boldsymbol{Q}_i^T \delta \boldsymbol{q} = \boldsymbol{Q}_e^T \delta \boldsymbol{q} \,, \tag{2.30}$$

where δW_e is the virtual work done by external forces. Combining the statement for the virtual work done by inertial and external forces, and the previous

equation for inertial forces gives thus

$$\left(\ddot{\boldsymbol{q}}^T\mathbf{M} - \boldsymbol{Q}_v^T - \boldsymbol{Q}_e^T\right)\delta\boldsymbol{q} \equiv 0\,. \tag{2.31}$$

The above statement is not unambiguously true due to the fact that the statement does not explicitly define the virtual work done by the forces generated into the joints connecting bodies. The often used methods to guarantee equality for the above statement in Equation (2.31) are:

- *Augmented Formulation*: a method of adding constraint equations using Lagrange multipliers, which leads to an extended set of equations for the multibody system. This method is explained in more detail below.
- *Embedding Technique*: a method of embedding constraint equations into the original set of equations, which leads to a minimum set of equations for the multibody system when using generalised coordinates. Even though the set of equations is smaller, the method requires the same initial information about the simulated system as augmented formulation.

Augmented Formulation In augmented formulation, the joint forces are added into the set of equations of motion using Lagrange multipliers. The equation for virtual work becomes

$$\delta W = \delta W_i - \delta W_e - \delta W_c = 0\,, \tag{2.32}$$

where δW_c is the virtual work done by constraint forces. The equations of motion can now be written as

$$\begin{aligned}\left(\ddot{\boldsymbol{q}}^T\mathbf{M} - \boldsymbol{Q}_v^T - \boldsymbol{Q}_e^T - \boldsymbol{Q}_c^T\right)\delta\boldsymbol{q} = 0\\ \Rightarrow \mathbf{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}_v - \boldsymbol{Q}_e - \boldsymbol{Q}_c = 0\,,\end{aligned} \tag{2.33}$$

where $\boldsymbol{Q}_c$ is the vector of generalised constraint forces. The vector of constraint forces is $\boldsymbol{Q}_c = \mathbf{C}_{\boldsymbol{q}}^T\boldsymbol{\lambda}$, where $\mathbf{C}_{\boldsymbol{q}}$ is a vector of constraint equations differentiated relative to generalised coordinates $\boldsymbol{q}$, and $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Now, the statement for virtual work is explicitly true, but the formulation adds a set of new variables in the form of Lagrange multipliers. To fulfil the set of equations, a set of algebraic constraint equations are added. Altogether, the equations of motion are

$$\left.\begin{aligned}\mathbf{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}_v - \boldsymbol{Q}_e - \mathbf{C}_{\boldsymbol{q}}^T\boldsymbol{\lambda} = 0\\ \boldsymbol{C} = 0\end{aligned}\right\}\,. \tag{2.34}$$

The joints and motion constraints in a multibody system restrict the motion of the connected bodies relative to each other. These restrictions are described

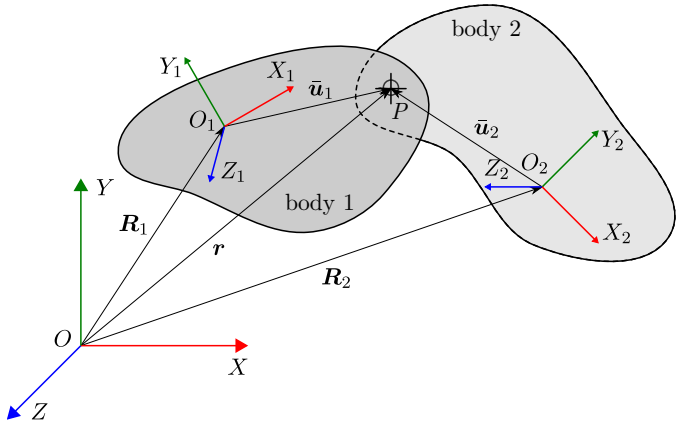


Figure 2.3: Definition of a spherical joint between two bodies using local frames of reference.

with constraint equations of the form $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{q}, t) = 0$. Constraint equations are additional equations that define e.g. that two bodies have a common location in the global frame of reference so that this point P is constant in the local frame of reference of both bodies. Figure 2.3 illustrates the definition of a spherical joint between two bodies, body 1 and body 2. The constraint equation for this case defines the location for point P in the global frame of reference so that the location for point P is constant in the local frame of reference of both bodies, i.e. $\bar{\boldsymbol{u}}_1$ and $\bar{\boldsymbol{u}}_2$ are constants.

The principle of virtual displacement can be applied to multibody system constraint equations. For that, virtual work on a multibody system is expressed by using independent generalised coordinates. The derivative of a constraint equation $\boldsymbol{C}$ respect to generalised coordinates $\boldsymbol{q}$ is

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{q}} \delta \boldsymbol{q} = 0\,. \tag{2.35}$$

The Jacobian matrix $\mathbf{C}_{\boldsymbol{q}}$ of a multibody system is a partial derivative of the constraint equations with respect to generalised coordinates $\boldsymbol{q}$

$$\mathbf{C}_{\boldsymbol{q}} = \begin{bmatrix} \partial C_1/\partial q_1 & \partial C_1/\partial q_2 & \dots & \partial C_1/\partial q_n \\ \partial C_2/\partial q_1 & \partial C_2/\partial q_2 & \dots & \partial C_2/\partial q_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial C_{n_c}/\partial q_1 & \partial C_{n_c}/\partial q_2 & \dots & \partial C_{n_c}/\partial q_n \end{bmatrix}, \tag{2.36}$$

where C_i $(i = 1 \dots n_c)$ are the components of the constraint vector, n_c is the number of constraint equations, and n is the number of generalised coordinates.

The equations of motion in Equation (2.34) are a set of differential algebraic equations, which is difficult to solve with common numerical solvers in a general

form. The algebraic constraint equations can be transformed to differential equations by differentiating them twice relative to time

$$\mathbf{C}_{\boldsymbol{q}}\dot{\boldsymbol{q}} = -\boldsymbol{C}_t \tag{2.37}$$

$$\mathbf{C}_{\boldsymbol{q}}\ddot{\boldsymbol{q}} = -\boldsymbol{C}_{tt} - \left(\mathbf{C}_{\boldsymbol{q}}\dot{\boldsymbol{q}}\right)_{\boldsymbol{q}}\dot{\boldsymbol{q}} + 2\mathbf{C}_{\boldsymbol{q}t}\dot{\boldsymbol{q}}\,, \tag{2.38}$$

where $\mathbf{C}_t$ is the vector of the first partial time derivative of the constraint equations, $\mathbf{C}_{tt}$ is the vector of the second partial time derivative of the constraint equations, and $\mathbf{C}_{\boldsymbol{q}t}$ is the matrix of the constraint equations differentiated relative to the generalised coordinates and time. Combining all the above into a matrix form, the equations for motion for a rigid body system can be written as

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\boldsymbol{q}}^T \\ \mathbf{C}_{\boldsymbol{q}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_e - \boldsymbol{Q}_v \\ \boldsymbol{Q}_c \end{bmatrix}\,. \tag{2.39}$$

2.1.3 Notes on Multibody System Simulation

The formulation of the equations of motion in the previous sections shows that a multibody system can be expressed using the following quantities:

- locations and orientations of the local frame of reference for each body in the system, expressed with generalised coordinates

$$\boldsymbol{q} = \left[x, y, z, \boldsymbol{\theta}^T\right]\,,$$

- mass matrix $\mathbf{M}$; expanding the matrix representation will produce for each body mass m and an inertia tensor with six components, called moments of inertia of the body

$$\mathbf{I}_{ij} = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ & I_{YY} & I_{YZ} \\ \text{Symm.} & & I_{ZZ} \end{bmatrix}$$

  and depending on the location of the body centre of mass relative to the body local frame of reference, quadratic velocity vector $\boldsymbol{Q}_v$,

- locations and orientations of the joints and motion constraints $\mathbf{C}$ in the system, expressed with generalised coordinates $\boldsymbol{q}$,

- locations and orientations of external forces $\boldsymbol{Q}_e$ in the system, expressed with generalised coordinates $\boldsymbol{q}$,

- definitions of the force function expressions in the system, and

- additional data that is used for describing the previous elements, such as additional state functions, variables, and data curves (e.g. in the form of spline functions) for function expressions.

Regardless of the applied formalism, the same quantities define the system. Thus, these quantities are the minimum but sufficient requirement for representing a multibody system model in a general form. The following sections present the background for the semantic data model. In addition, the special features and methods involved in the use of semantic data management, such as semantic rules and constraints, and semantic reasoning, are covered.

2.2 Semantic Data Management

2.2.1 Background

The fast growth in the amount of Web data, and on the other hand, the continual progress in Web information technology development has induced the World Wide Web Consortium (W3C) to embark on a project to design the next generation Web, the Semantic Web[1]. The objective of this development is to create a set of technologies that can be used within the framework of the existing Internet infrastructure, while enhancing the management and exploitation of the Web content. The Semantic Web presents a set of technologies that enables storing the meaning of the data within the data itself. This makes it possible for software applications to combine and refine data from a number of sources and thus form a single network of information [36]. The present Web is document-centric. This means that the content in the Web is passive, even though the content of individual pages may be composed from e.g. a database. Another feature in the present Web is that the data and the structures are designed for human manipulation, not for a machine processable form. This restricts the automatic composition of new Web content based on existing arbitrary Web data.

The same needs as in the Web can be seen in the area of product development and the management of modelling and simulation data. Due to the increase in computers' numerical computing power and the development of computational software, the amount of modelling and simulation data involved in a development process is increasing rapidly. As in the present Web, the modelling and simulation data management practice is document-centric. Model data is managed by using model files; product data management (PDM) systems focus on document management. In model files, the content of the model description is obscure for humans, and automatic information retrieval from these files for other applications is in most cases impossible. However, the value of the data is not in the format but in the content, the information. The dependencies of different design and simulation models in a design process are illustrated in Figure 2.4. Each tool in the process stores its data in a software

[1]Website of the Semantic Web: `http://www.w3.org/2001/sw/`

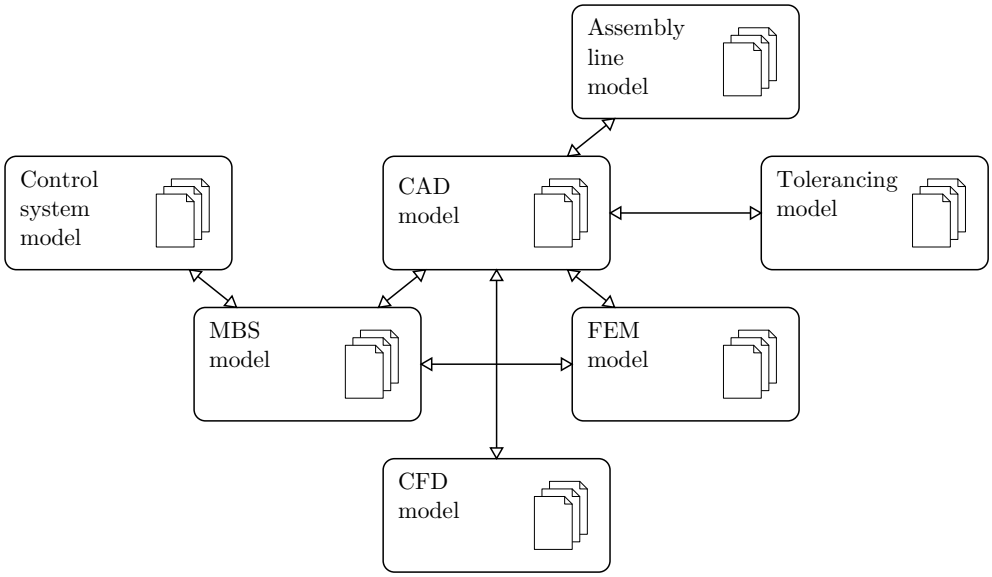


Figure 2.4: An example of the dependencies of different design and simulation models in a design process.

application-specific format. For each model, there may be one or more files to be saved. For each domain model there may be several model versions, either for different simulation model development states or for different simulation purposes.

While the performance of computers and the efficiency of simulation software keep increasing, also the process of using simulation in product development is changing. As described in chapter 1, the application of a simulation-based product development process, and especially a simulation-based product life-cycle process increases the amount of modelling and numerical result data drastically, compared to the present situation. In addition to the amount of data, also the complexity of the data and information increases due to the number of different versions of models needed in the product development process and the number of simulation cases that are run for understanding the product under development better. In this trend, the bottleneck of the improvement is the human being. Our capabilities to distill the valuable information out of the data mass is limited. The answer to this is to increase the abstraction level of the managed data. The best solution is to change the plain data into information. Thus, the need for a new method to manage the data in the product process is parallel to the one for the Semantic Web.

The latest technologies of knowledge representation and data semantics are employed to create a framework of specifications, languages, and tools for the Semantic Web project [37]. To this end, concepts and terms can be defined

explicitly and definitions can be domain- and context-specific. In addition, the definitions can contain rules and validity constraints to increase the value of the data. The main improvement is, however, that due to the semantics of the content and its representation in machine-readable form, the information can be condensed out of the data mass automatically by software.

The origin of knowledge representation as a separate discipline in research has its roots in the research and development of artificial intelligence [39, 40]. Due to the increase in research activity in the area of the Semantic Web, the interest in representing knowledge and reasoning on the existing data has increased. There are many designs of systems and methods for representing knowledge in a computer-processable form, but, in this thesis the emphasis is on the technologies developed for the Semantic Web.

The area of knowledge representation can be coarsely divided into two subtasks. On the first one, representing human knowledge in a machine-readable form including terms, their descriptions, relations, and other constraints are in focus. In the second subtask, applying reasoning to a mass of knowledge using well-formed queries is studied. The human knowledge representation can be seen as hierarchical and layered, starting from fundamental concept definitions, adding new concepts and terms based on them, and mapping related definitions. This is also the case in the models for knowledge representation, such as descriptive logic- and frame-based methods, which create a network of definitions that refines the definition of the subject while enabling tracing back to the origin of the definition chain.

The often used architecture of a system based on the semantic data model consists of a special database management system (DBMS) for triple data, the database itself, i.e. the content, and client applications that use the database. This can be related to the traditional relational database systems used for example in business data management. A database system with query capabilities can be used for retrieving desired data from the system. In relational database systems, the most common method is the Structured Query Language (SQL) [49], which enables simple methods to be used to retrieve relatively complex data from the database system. Similar capabilities are usually available in semantic database systems. When more complex rules for data retrieval and comparison are required, separate reasoning systems, referred to as reasoning engines, are often used. The major difference of a semantic representation compared to the relational data model used in the traditional database system, is its flexibility with respect to the content and structure of the data. On the other hand, one of the challenges associated with semantic data representation is the requirement for the computational resources of the semantic database management system.

A fundamental question of how the surrounding world is seen and how it is treated from the knowledge management point of view affects the procedure of

how the actual data is handled. For data storing and especially reasoning, the concepts of open world assumption and closed world assumption are essential. The following definitions are generally used in knowledge representation and reasoning:

Open world assumption: In the open world assumption (OWA), a statement is assumed true, if it is not explicitly false. This means that if a new statement is proposed and there is no information to prove it false, i.e. the statement is unspecified, then it is true. The open world assumption is well suited for knowledge representation and it is used e.g. in the Semantic Web approach. [50]

In the open world assumption, something can be said to be true only if it is always true, no matter if some new additional information is provided. One can imagine the WWW as the source of information; if something is to be inferred on the basis of the information available in the Web, no one can guarantee that the information in hand is the whole and complete information there is for this specific subject.

Closed world assumption: In the closed world assumption(CWA), a statement is assumed false, if it is not explicitly true. This means that the definition of the world under attention is explicit and closed. Statements about the objects in the definition can either be true or false, but statements about objects outside the definition are explicitly false. [50]

The closed world assumption is well suited for describing domains that are complete and well bounded, which is usually the case e.g. with system modelling domains. Modelling knowledge and reasoning based on the closed world assumption is closer to the concept of using object-oriented methods than when applying the open world assumption.

The Semantic Web relies upon the open world assumption [51]. The open world assumption is necessary for enabling the description of incomplete knowledge and information available in the Web. Modelling domains, on the other hand, are usually well defined and complete, which would make it appropriate to use the closed world assumption. To use Semantic Web technologies like Web Ontology Language (OWL), which make an open world assumption, to represent the modelling domain, additional rules and constraints have to be applied to complete the representation and close the definition.

The above assumptions of either the open or the closed world have fundamental influence on how the semantic data is interpreted by a semantic reasoner, a software that does logical reasoning on the semantic data based on the basic semantics, semantic constraints, and semantic rules. Thus, the fundamental and philosophical question of how the surrounding world is seen and

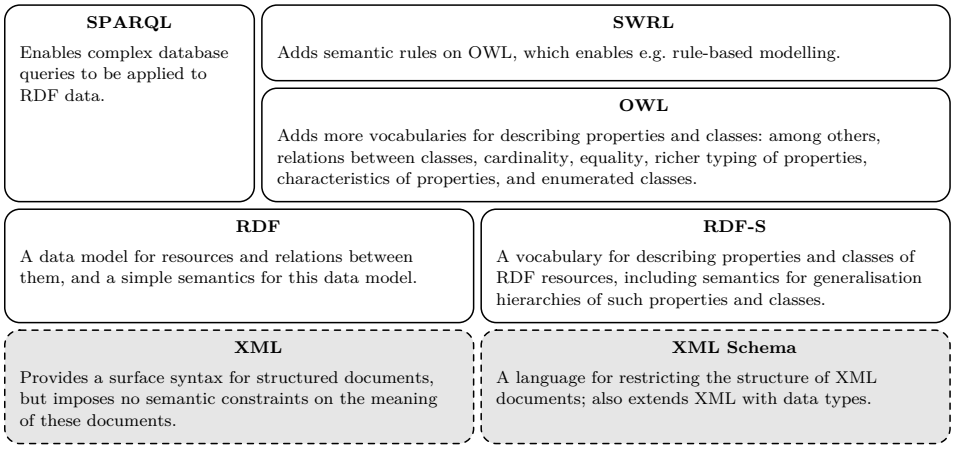


Figure 2.5: The layered structure of the enabling core technologies for the Semantic Web [2].

treated is also fundamental for semantic data management from the practical point of view.

The concept of the Semantic Web is strongly layered, as illustrated in Figure 2.5. This is also the case for the content of semantic data. The layered form of semantic data is bounded and finite, so it has to have an origin, a bottom layer that is used as the basis for other definitions. Fundamental definitions have been already collected into general ontologies, such as Dublin Core[2] [52], Cyc[3] [53], and General Formal Ontology[4] [54]. These are examples of storages of fundamental data, common knowledge, which are referred to but do not refer anywhere.

The W3C has developed and applied several technologies under the umbrella of the Semantic Web as depicted in Figure 2.5. These technologies include:

- the Resource Description Framework (RDF), a language for representing information about resources in the World Wide Web, and the RDF Schema (RDF-S), a vocabulary description language for RDF,
- the Web Ontology Language (OWL), an ontology description language developed for the Semantic Web,
- the SPARQL Protocol and RDF Query Language (SPARQL), a query language developed for the Semantic Web to be used together with RDF,

[2]Website of Dublin Core: `http://dublincore.org/`
[3]Websites of Cyc: `http://cyc.com/cyc` and OpenCyc.org: `http://www.opencyc.org/`
[4]Website of General Formal Ontology: `http://www.onto-med.de/ontologies/gfo/`

- the Semantic Web Rule Language (SWRL), an extension to the Web Ontology Language OWL to introduce Horn-like rules, and
- the Extensible Markup Language (XML) together with the Extensible Markup Language Schema (XML Schema), a natural choice for data serialisation for the Semantic Web; the grey background colour and dash line of the boxes in Figure 2.5 indicate that these technologies are general serialisation technologies and not specific to semantic data representation.

The most relevant technologies for this work are described briefly in the following sections. There are more technologies that are related to the Semantic Web, but they do not have relevance for this work.

2.2.2 Resource Description Framework

A number of notations have been developed for knowledge representation in the computational environment. The subject-predicate-object form, the data triple, for the description of data entities and their relationships has gained wide acceptance, due to its straightforward notation. This approach creates the basis for the data model selected for the Semantic Web [55].

One of the main components for the Semantic Web is the Resource Description Framework (RDF), which is a language for representing information with respect to resources in the WWW [2, 37, 55, 56]. The RDF language has a simple structure, making it attractive for storing heterogeneous machine-readable data. The data model in the RDF consists of data triples: a resource (subject), a property (predicate), and its value (object); values (objects) may be either other resources or literals (constant values). In the RDF, resources (subjects and objects in data triples) as well as properties (predicates in data triples) are identified using Unified Resource Identifiers (URIs) [57]. Figure 2.6 illustrates a representation of information in the triple form.

The RDF is extensible, making it possible to have a precise description of the semantics of the data. The language has been widely adopted for the technology of several semantic data applications, while a number of tools and application libraries are available for software developers. The RDF specification describes the triple form for the data model as well as the basic rules that can be applied to the data. The RDF can also be used for representing information with respect of issues that can be identified on the Web even when information cannot be directly retrieved from the Web [37]. This generality makes the RDF a suitable option also for system modelling data management.

The RDF defines the basic features for the data representation, but it is not sufficient to be used for generic semantic data representation. The description of the data model is overly general and does not have structures and mechanisms for e.g. defining ontologies and their properties. The Resource Description

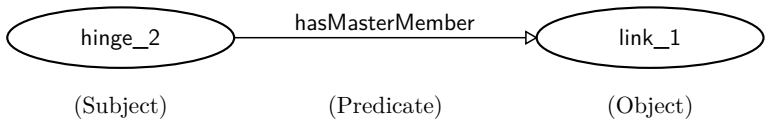


Figure 2.6: An example of the use of the subject-predicate-object, the data triple, representation of multibody system simulation modelling data.

Framework Schema (RDF-S), the RDF's vocabulary description language, is a semantic extension of the RDF. It can be seen as an extension in the sense of description of the semantics, but, on the other hand, it can be seen as a restriction to the RDF, which is otherwise open and permissive description. The RDF-S uses the RDF to describe new features, and thus it can be treated as an RDF vocabulary defining new concepts. The RDF-S adds the concept of class into the definition of RDF. The class concept in the RDF-S is similar to those in object-oriented languages. The RDF-S adds also other concepts to the RDF, such as a type system, as well as domain and range. The RDF and RDF-S can be considered as the foundation for upper layers in the Semantic Web data model.

In a semantic database having a triple-formed data model, all the data is described in triples, i.e. all component instances, their connections and relations, and all additional data like component data properties. The advantage of the triple-formed data model is that it simplifies the design and development of tools to handle the data. This approach resembles the XML data representation, which has already proved to be flexible and rich in the number of tools [20]. Like in the XML representation, the model is independent of the application area. In this approach, all the tools that are developed for database development and management are available for all application areas. For example, general semantic ontology development tools, such as Protégé[5] and NeOn Toolkit[6], are usable for system modelling ontology development.

2.2.3 Web Ontology Language

The Semantic Web and its knowledge representation are based on the triple-formed low level data model and the application of ontologies. In the Semantic Web context, an ontology is a special kind of vocabulary that has object-oriented features. An ontology consists of definitions of classes, object properties that connect the instances of the classes, and data properties, which define the properties of the classes. The classes can have subclasses, which inherit the properties from the superclass. An object in a data model can be an in-

[5]Website of Protégé: `http://protege.stanford.edu/`
[6]Website of NeOn Toolkit: `http://neon-toolkit.org/wiki/Main_Page`

stance of several classes at the same time, but classes can be defined as disjoint to prevent this. An example of an ontology could be a specification of the multibody system simulation domain. In that case, the ontology defines basic entities, such as a rigid body, joints and constraints, forces, and their properties. For a rigid body, the properties could be the location and orientation of the body, mass, centre of mass location, and the inertia tensor presented in a given frame of reference. To use the basic principles of the semantic data model, the measures and their units can be mapped from another ontology.

The Web Ontology Language (OWL) is an ontology language built on the RDF and RDF-S. While the RDF gives tools for describing arbitrary things using one simple formalism, the OWL bounds this permissive and thus difficult-to-handle domain to a clearly bounded and implementable definition. The OWL is one of the technologies of the Semantic Web, which has strongly influenced the development of this ontology development language. The first version of the OWL specification was completed in 2004 [58], and a new version of it, the OWL 2 Web Ontology Language [59], was completed in 2009. The new version of the OWL is backwards compatible with the old OWL definition.

The OWL has mechanisms to describe ontologies that are built by using classes and properties. The classes can be seen similar to the classes in object-oriented programming languages, such as Java or C++. The subclasses inherit properties from the superclass, and instances of a subclass are also instances of the particular superclass. The OWL properties are the predicates in the data triple, hence they are the connecting predicates between the subjects and objects. The properties can be either object properties, i.e. they connect class instances to other class instances, or data properties, when they are used for connecting RDF literals or typed values to class instances. In addition to the above basic features, the OWL defines class constructors, such as intersect, union, and complement, for composing complex classes.

Due to the primary design goal of the Web Ontology Language, it uses the open world assumption (OWA). This assumption enables incomplete data to be used for reasoning, which has to be assumed for the data from the Web. Another feature of the Web Ontology Language, coming from the RDF definition, is that it does not make unique name assumption (UNA). This means that e.g. instances of a class with different names may refer to one entity, i.e. one is an alias of the other.

The OWL 2 specification has two different sublanguages:

1. OWL DL (referring to Description Logic) is the practical subset of the specification, and
2. OWL Full, which is RDF and RDF-S -compliant; the complexity of OWL Full restricts its use in practical applications.

In addition to these sublanguages, OWL 2 introduces three language profiles.

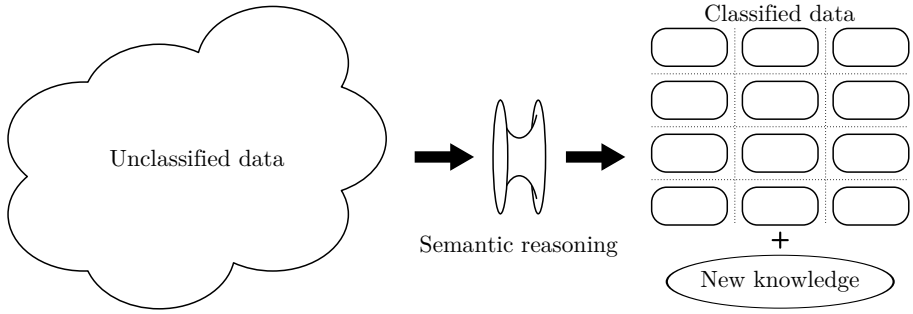


Figure 2.7: Semantic data and its classification.

The language profiles are tailored subsets of the OWL 2 language for specific purposes to simplify the otherwise unnecessarily difficult implementation. The profiles are [60]:

1. OWL 2 EL, which is relatively close to OWL 2 DL with just a few simplifications in the data model; OWL 2 EL is designed for large data sets with structurally complex data models (e.g. in biohealth and system description),

2. OWL 2 QL, which is designed in such a way that it can be implemented with relational database systems; the name refers to Query Language, and

3. OWL 2 RL, which is optimised for reasoning and rule-based modelling; RL refers to Rule Language.

OWL Features and Characteristics

Developing ontologies, describing knowledge semantically, and reasoning with the OWL are strongly influenced by the open world assumption made in the OWL. The class definitions and property restrictions in the language are not meant for validating if an instance of an ontology class is modelled correctly from the modelling domain point of view, but to classify the semantic data based on the applied ontologies. Except for datatype restrictions (e.g. `hasMass some double[≥0]`, meaning the property of the mass of a body is a datatype of double, and its value must be greater or equal to zero), the property restrictions are constructors for complex classes, i.e. they are instructions for the reasoner to filter the semantic data and classify it according to complex class definitions (Figure 2.7).

Some fundamental features of the Web Ontology Language which have a higher importance in this thesis, and some features which are consequences of the nature of the OWL, are described below.

Cardinality The number of properties connected to an instance of a class is not restricted in the OWL. To restrict the number of properties attached to an instance, either by requiring a minimum, defining a maximum, or defining the exact number of properties, a cardinality constraint is used. These restrictions are *ObjectMinCardinality*, *ObjectMaxCardinality*, and *ObjectExactCardinality*. It is important to note that even though either a minimum or an exact cardinality restriction is set to an instance, this does not guarantee that the instance has any properties of a kind. In other words, if there are properties of a kind, the cardinality restriction sets constraints to the number of such properties. Related to the cardinality restrictions are existential (*ObjectSomeValuesFrom*) and universal (*ObjectAllValuesFrom*) restrictions. They define, respectively, that there has to be at least one property of a kind attached to the instance and that the only allowed properties are of the specified properties. [60, 61]

Class Disjointness The disjoint constraint restricts the instances of a class from being an instance of another, disjoint class. An example of class disjointness would be the subclasses *Man* and *Woman* of the superclass *Human*. It is quite natural that an instance of the class *Woman* cannot at the same time be an instance of the class *Man*.

Closure Axiom The open world assumption is made in the OWL specification. This general assumption may become a problem, if in some specific occasion a local closed world assumption is required. As an example, in a multibody system domain, a rigid body can be defined with and only with a local frame of reference, mass, and inertia tensor. Based on the open world assumption, if no other restrictions are done, it is a valid statement that a rigid body has a property temperature, even though it is not relevant in the ideal rigid multibody system domain. To close the definition of a rigid body, an additional statement, a closure axiom, is required to define that the mentioned rigid body properties are the required and the only allowed properties for a rigid body.

Domain and Range The domain and range constraints in the OWL limit the set of classes a property may refer from (domain) and may refer to (range). The domain and range constraints are not strict restriction, but meant for adding implicit information about the resources the property is connecting. An example of the added implicit information is the domain and range constraints for the *hasMasterMember* property (see chapter 3 for details). For the reasoner, the domain and range constraints in this case inform that the subject instance of the data triple is either a force or a constraint, and that the object instance

of the data triple is a body. This information decreases the need for further inferring and thus speeds up the process. [61]

Functional The OWL Web Ontology Language Guide [51] explains the functional property constraint in a compact manner: if a property D is tagged as functional, then for all a, b, and c

$$D(a,b) \wedge D(a,c) \rightarrow b = c\,. \tag{2.40}$$

That is, an instance of a class can have only one value attached by the specified property. It is important to note that functionality for a property does not guarantee that there is only one property of the specific kind relating to one individual. In case there are e.g. two instances of a functional property relating to the same individual, these two property instances must point to the same value, or the relation is inconsistent. This is the consequence of the fact that the RDF and thus the OWL do not make a unique name assumption. If it is required that there is only one instance of a property relating to one individual, the cardinality constraint has to be used.

An inverse functional property defines a unique subject for the specified property in the data triple, i.e. a value of a property can be connected only from one unique instance of a class. Based on Equation (2.40), this is

$$D(a,b) \wedge D(c,b) \rightarrow a = c\,. \tag{2.41}$$

Symmetric Symmetric property means that if instance a of some class has property D with a value of instance b of some class, then instance b has the same property D with value a

$$D(a,b) \rightarrow D(b,a)\,. \tag{2.42}$$

An asymmetric property states explicitly that this is not valid.

2.2.4 Semantic Database System

Implementing a software application that utilises the semantic data model approach is not a trivial task, especially if the application is meant for managing large data sets. The concept of storing system modelling data by using a semantic data model is illustrated in Figure 2.8. In addition to the basic structures of the data model, i.e. knowledge representation framework and ontology development languages, a special database management system (DBMS) is required. In case of triple data, this is called a triplestore. A triplestore is a specifically designed database system for triple formatted data. For this reason, it is computationally more efficient to use than for example a relational

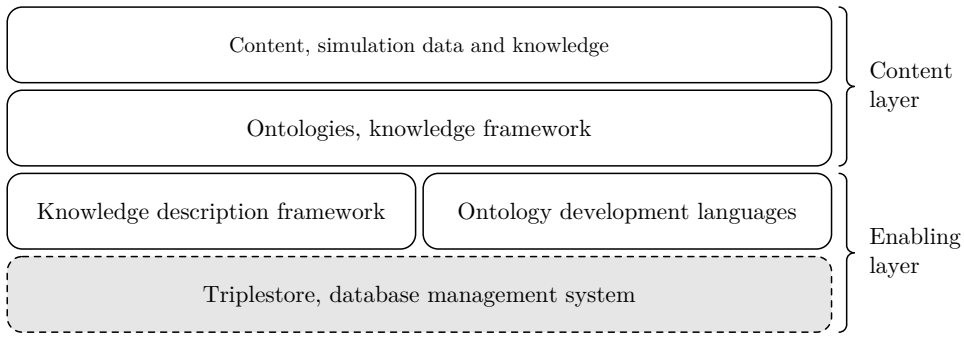


Figure 2.8: The components for semantic data management of system simulation.

database system applied for this purpose. In Figure 2.8, the dash line and the grey colour for the triplestore means that this is a real software component unlike the other, which are language definitions or actual data content. These three components form the enabling layer, the infrastructure and general technologies, of the semantic data management system. Using these components, knowledge can be described and stored into the system. The essential network of ontologies (the knowledge framework) constitutes the basis of common models of knowledge in a knowledge domain. More specific and case dependent data models can be created with these knowledge structures. They describe for example individual simulation models or parts of models. These two components form the content layer. The layered, hierarchical structure of the semantic data model itself encourages building the knowledge framework also hierarchically. This, on the other hand, simplifies the data management and enables knowledge accumulation into the semantic data management system.

For a general data management application, a database design with a clear data interface, data storing, manipulation and deleting functions, sophisticated data query mechanisms, and flexible client software connectivity is a popular solution. Due to the basic objectives discussed in chapter 1, one of the motivations to apply the semantic approach is to solve the problem of managing large and complex data structures. The functionality of a semantic database system can be compressed to the following:

1. representing and storing data in the semantic form,
2. being able to validate the data and its semantics against given ontologies,
3. applying explicit and implicit reasoning on the data on the basis of given ontologies,
4. being able to perform complex queries to large data sets, and

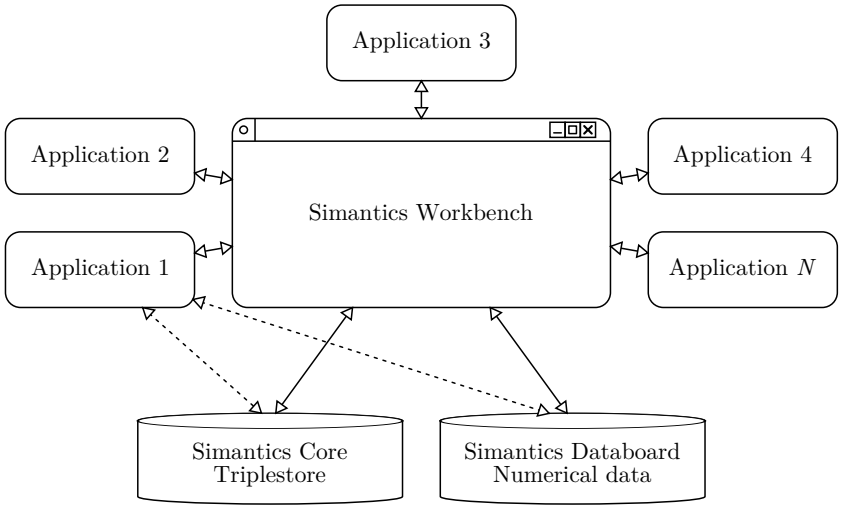


Figure 2.9: The architecture of the Simantics platform.

5. being able to map data on the ontology level and thus provide a projection from one data representation to another.

An example of a semantic database system for modelling is the Simantics[7] platform (Figure 2.9). The software architecture of Simantics is a traditional server-client model. The semantic database server, Simantics Core, holds all the semantic data of the system. The client software application, Simantics Workbench, provides the user interface to the system and is the framework for all interactive tools, such as model editors, database browser, and ontology development tools [62]. The third component in the system, Simantics Databoard, provides an unified interface for different data backends and operates as a connection framework between the data in Java classes and the semantic data graph. External simulation applications, such as numerical solvers, are connected to the client software application as application plug-ins. The application plug-ins can have direct connections to the database server and to each other.

Knowledge Mapping in a Semantic Environment

One of the major design objectives of the Semantic Web and its technologies has been to enable the mapping of data and information in the Web. The first generation of the Web introduced the concept of publishing information electronically and linking it manually with hyperlinks. The concept can be simplified as publishing information passively in electrical form in the Internet.

[7]Website of the Simantics platform: `https://www.simantics.org/simantics`

Even though there are technologies to compose a web page actively on client's (e.g. user's browser) request and retrieve up-to-date data from a database system, the structure and mechanisms are passive and predefined. [36, 63]

The next generation of the Web introduces interactive services. The information is not only published by pushing it to users, the Web content is interactive instead, and the users are able to produce and modify the content to some extent. Examples of the interactive Web are the social media applications in the Web, such as Wikipedia[8]. The technologies used in the Web 1.0 and Web 2.0 concepts do not solve the demand of being able to create new content using multiple data sources and automated procedures to retrieve relevant data. For this purpose, a project for the next generation Web, the Semantic Web, has been introduced. [36, 63]

In the Semantic Web, the third generation of the Web, the data concepts are defined with ontologies. The efficient and rational use of the semantic data model takes advantage of the existing ontologies and builds on them. This means that new ontologies are mapped to the existing ones, and if necessary, the existing ontologies are extended instead of creating completely new ontologies. An example of using mapping between ontologies is the use of an existing unit system ontology. The knowledge mapping can be seen as bridging the concepts of one domain with the concepts of another. [36, 63]

Semantic data mapping, or semantic integration, has also been seen as a method for integrating data sources in different environments than the Web [64, 65]. In general, semantic ontology-based data modelling has been seen as a tool for information integration. Application areas for semantic data mapping are e.g. database integration and integration of data that is already described with ontologies, such as biomedical databases [66].

In physical system modelling, such as multibody system and computational fluid dynamics (CFD), a unit system for quantities is required. The definition for a unit system, such as SI units, should be explicit and thus reusable. In addition, conversions between different unit systems, such as the SI unit system and the English customary unit system, are also explicit and can be systematised. To rationalise the development of physical modelling ontologies, the modelling domain ontologies should be mapped with the general unit system ontologies. In semantic modelling, there are also other approaches to implement unit systems than a fully-fledged units ontology, such as using literals for quantities and units (e.g. *length:quantity* and *mm:unit* denoting "length" is type of "quantity" and "mm" is type of "unit").

The mechanism of mapping data in a semantic environment is based on the fundamental feature of the low level data model. The data triple allows mappings between practically all kinds of data types. On the upper abstraction

[8]Website of Wikipedia: `http://www.wikipedia.org/`

level, the semantic modelling language, such as the OWL, defines mechanisms for mappings between ontologies. The mapping can be done by using built-in mechanisms of the OWL, such as *equivalentClass* and *equivalentProperty* axioms or, in complex mappings, specific mapping ontologies with intermediate constructing structures for complex classes.

Tool Interoperability in the Semantic Approach

In addition to the data and information mapping discussed in the previous section, the semantic approach is used for software application data exchange and integration. In principle, the approach is straightforward. The minimum requirement is to provide a data flow between software applications and the semantic environment. To use the semantic approach, the data models of the connected software applications need to be described semantically, i.e. also software application-specific ontologies have to be provided. After this, the software application data is described in the semantic environment, and all the semantic mapping and data modification features are available. Mapping between different software applications can be done straight between the application-specific ontologies, or they can be done by using domain-specific ontologies to simplify the mapping between different domain ontologies and software applications.

The concepts of integrating software applications by mapping application-specific ontologies directly and by using domain-generic ontology as an intermediate step are illustrated in Figure 2.10. As shown in the Figure, ontologies can build bridges between the data models. When the modelling is done inside the ontology database, the database forms the common data model for different applications.

The concept presented above is applied to modelling time data management and data transfer between applications. Runtime communication between software applications, such as numerical solvers, is a slightly different task. In addition to just sending and receiving data when the data is modified, runtime communication requires information about the order of precedence, i.e. which one of the software applications has the highest priority and which applications follow the orders.

The actual runtime data transfer can be implemented with several strategies. Direct communication between software applications can be done by using a selected communication protocol (e.g. TCP/IP) or document-based by using intermediate files, described as dash line in Figure 2.10. Depending on the semantic environment capabilities, the runtime data transfer can also be implemented through the semantic system. This option simplifies the implementation of several software application integrations even more, due to one communication interface, but can become complicated on the semantic envir-

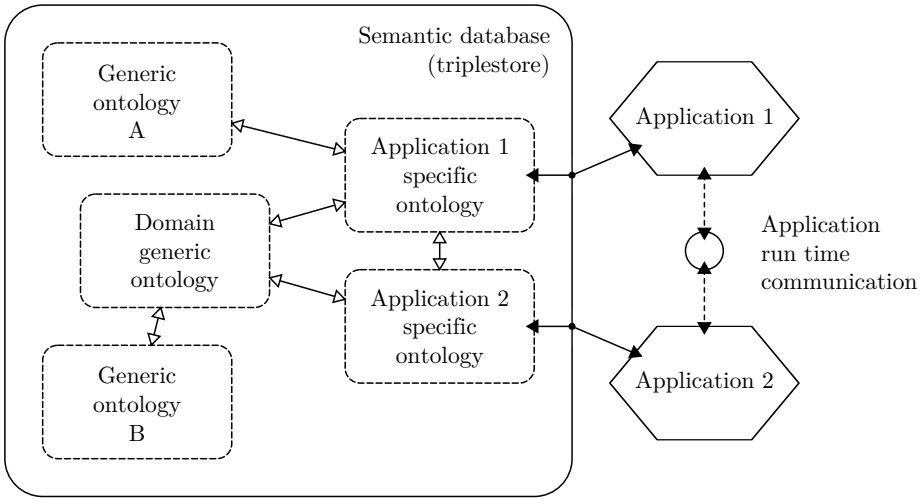


Figure 2.10: The architecture of semantic integration of applications.

onment side.

The advantage of semantic database data exchange compared to the one-to-one data exchange strategy between modelling and simulation tools, illustrated in Figure 1.5, is that once the communication capability is created for a new integrated software application, the whole set of existing integrations are available. Creating new data mappings inside the ontology database application does not require any software development, and for this reason it is available for a simulation user who may not be experienced in programming. As a conclusion, a common semantic framework introduces unified tools for data and knowledge mapping.

2.2.5 Semantic Reasoning and Rule-Based Modelling

In the context of this thesis, semantic reasoning means either checking the validity of the semantic data on the basis of the ontologies, or implicit inferring of new information on the basis of the existing data. An example of the former in the modelling and simulation domain could be the validity checking of the multibody system model based on the ontology definition and possible additional modelling case restrictions. An example of the latter could be an analysis of system features based on the modelling data and additional instructions about the features under analysis. The use of reasoning is one of the advantageous features of the use of the semantic data model. It is a tool for applying smart data analysis methods on the modelling data and thus increasing the value of the data. The motivation for using advanced data models and data management tools, and on the other hand, the application principle

of the rule-based modelling approach, are depicted in the following example.

Figure 2.11 presents a simplified example of the phases of the machine product development process. In the Figure, the subsystems represent e.g. the mechanical design, hydraulic subsystems, and control systems of the product. In this example, the design process begins with the concept design phase for the product, during which the general architecture and structure of the product are designed. During this phase, the product design is divided into subsystems and their common interfaces, and the design requirements are set. These can be seen as the declaration of the main global design variables. The process continues with the concept phase for the subsystem design, during which the architecture and implementation of each subsystem are decided so that the earlier set requirements are met. During this phase, the main global variables are brought to the subsystem design environment, and a set of subsystem-specific interfaces and design parameters are declared. The third phase in the process is the detailed design phase for the subsystem designs. This is a phase during which both the main global design parameters and the subsystem-specific design parameters are defined, and possibly more design parameters are declared. The result of this phase is the overall design of the product, which can be seen as the set of declared and defined design variables. If the design is good, the set of variables meets the requirements of the product and its subsystems. If the requirements cannot be met, the requirements have to be adjusted to enable it. The fundamental objective for efficient product development is to minimise the area of the rectangle in Figure 2.11 bounded by the time axis and the subsystem discipline axis. This can be done by decreasing the time-scale of the subtasks in the process rectangle and by filling the area with the subtasks as well as possible to avoid gaps between the consecutive subtasks. Usually the number of subsystems cannot be decreased.

In the example above, each of the subsystems has its own and established methods for design practices. For example, mechanical design is nowadays done mostly three-dimensionally using a CAD system, while automation and control system design tools are two-dimensional schematic graphs. In addition, the simulation methods in different engineering disciplines differ remarkably, which leads to many different modelling and simulation tools and many separate simulation models, which all describe the same product or part of it. When the application of simulation-based product development becomes more attractive and this approach is more widely used, the management of all the computational models of the product becomes more difficult. The models and simulation results are connected to each other via the product and its global design variables and interfaces. Thus, changing a parameter value in one discipline may have severe influences on other disciplines of the same product development process. This is a challenge for product data management, especially in the simulation-based product development process.

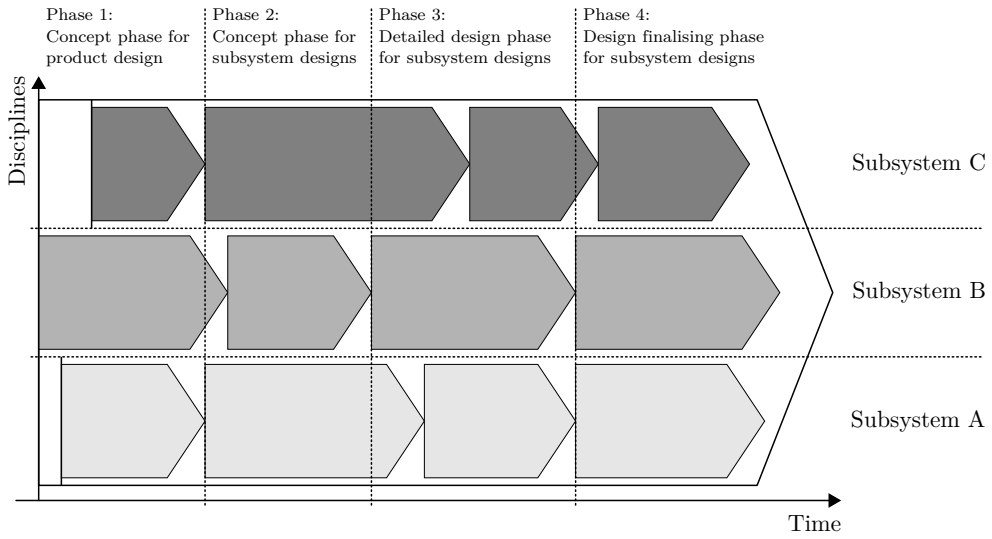


Figure 2.11: Simplified example of the phases of the machine product development process.

Figure 2.12 illustrates the application of semantic rules and constraints for a semantic modelling ontology. In the Figure, the diamonds present the parameter options for the design, the solid line bounding the diamonds show the available parameter space, the thick dash line is the applied semantic rule set, and the thin dot line and the dot line diamonds are the excluded parameter space. In the beginning of a modelling task, the whole ontology space and all the parameter values are open for use (Figure 2.12 a). Selecting a domain ontology, e.g. a multibody system modelling ontology, bounds the options for modelling based on the general modelling rules and constraints included in the ontology (Figure 2.12 b). Further, applying case-specific rules and constraints bounds the ontology space further and gives the modeller the design space for the current task (Figure 2.12 c). The modelling tools infer the semantic model during the process and limit the available options for the user. This decreases the possibility of error and makes it easier for the modeller to use the modelling tools with fewer options, and thus increases the efficiency of the work. The advantage of using semantic rules and constraints with reasoning simplifies the multi-domain modelling and data management. This is due to the fact that the same data management and rule definition mechanisms are used for all the modelling data. The difference in the semantic method compared to previously used methods in modelling data management is that instead of creating the model validation mechanisms and validation rules in the modelling tool, they are included in the modelling domain ontology (for generic domain rules) and individual simulation model (for case specific rules). This enables the use of

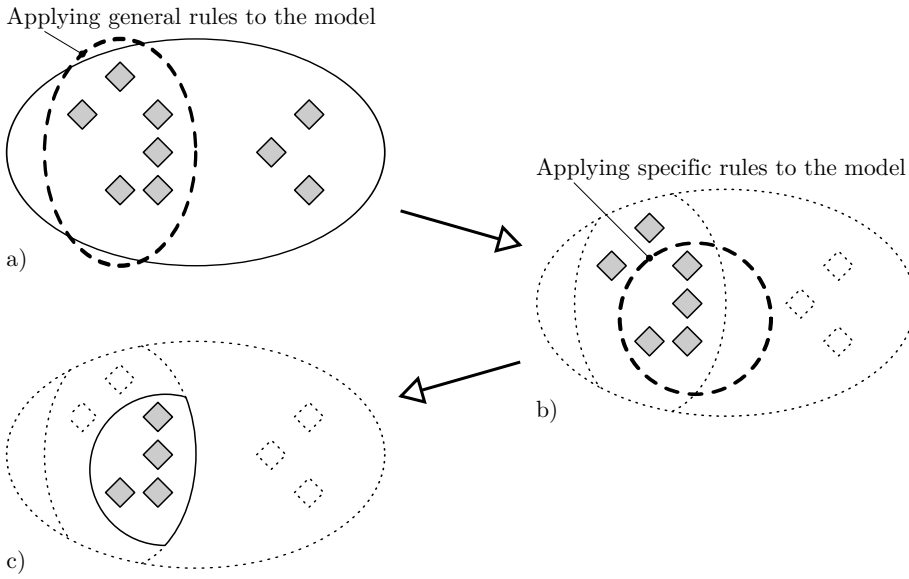


Figure 2.12: Applying rules to the domain ontology. a) Original set of options and application of general modelling rules. b) General rules-limited set of options and application of case-specific rules. c) Specific rules-limited set of options.

strict rules and also combining the model information with information about the validity of the model. This expansion in included information increases also the information value of the data. It can be said that this converges the product modelling data into information.

Semantic Reasoners

A semantic reasoner is a software application that can be used for checking the consistency of a semantic ontology, as well as for classifying semantic data based on ontologies. Here, classification means reasoning the data and classifying it on the basis of the class definitions. The conceptual difference to the object-oriented approach is that in the case of object-oriented programming, the user of a object class is restricted to use the methods and datatypes provided by the class, but in semantic reasoning, the classes are the basis for the classification and the data either fits a class or not. In semantic data reasoning using the OWL, there are no restrictions about the content of the data.

In this work, the open source semantic reasoner Pellet[9] [67] is used for

[9]Website of Pellet reasoner: `http://clarkparsia.com/pellet/`

checking the consistency of the developed ontologies, as well as for classification of the data, rule-based reasoning on SWRL rules, and data queries on the semantic data using SPARQL. The Pellet reasoner is compliant with the OWL DL version 2.

Constraints and Rules in the Web Ontology Language

The description of data and knowledge in a semantic form can be divided into two distinct areas:

1. plain description of content, i.e. data, facts, and details; and

2. description of data relations, restrictions, and rules.

The first area provides results especially for explicit knowledge retrieval, an example of which could be a query for all the bodies in a multibody system model that have mass equal or over 2 kg. This query does not require much reasoning or checking any complex validity of rules or constraints, but only the value of one property of each body in the model. This information is available as explicit data. The second area of the semantic data and knowledge representation may result in complex reasoning on the data. An example of the second area is finding all the bodies in the system that are constraint to the ground body via one other body, and the chain is done using spherical joints. This query requires analysis of the topology of the model instead of analysis of the model component parameters.

The Web Ontology Language includes only limited features for applying complex semantic rules and restriction on the data. These features are briefly described in section 2.2.3. The features are:

- existential and universal quantifications (*hasSomeValueFrom* and *hasAllValuesFrom*, respectively),

- cardinality restrictions (*hasMinCardinality*, *hasMaxCardinality*, *hasExactCardinality*),

- class disjointness,

- property domain and range restrictions,

- equality and inequality of individuals (class instances),

- data types, and

- property functionality.

Semantic Web Rule Language

The Web Ontology Language OWL contains only limited mechanisms to set semantic constraints to the data. These include data restrictions, such as setting cardinality or domain and range restrictions for properties. The actual semantic rules are not included in the OWL. For example, defining the following simple rule is not possible with the OWL: defining properties D and E, such as if instance a has property D that has the value of instance b and the same instance a has property E that has the value of instance c, the value instances b and c have to be different instances

$$D\,(a,b) \wedge E\,(a,c) \rightarrow b \neq c\,.$$

To fulfil this lack of functionality, a specific rule language has been introduced. The Semantic Web Rule Language (SWRL) is an extension to the OWL language axioms introducing elements for Horn-like rules [68]. The original Horn clause [69] is written as

$$L_1 \ldots L_n \Rightarrow L\,(\equiv \neg L_1 \vee \ldots \neg L_n \vee L)\;, \tag{2.43}$$

where L_1 to L_n are literals and $n \geq 0$, and L to is the only positive literal. A Horn clause is a definite clause when it has exactly one positive literal. This is often presented as

$$\Omega \bigwedge_{i<n} (\Phi_{i0} \wedge \cdots \wedge \Phi_{im_i} \rightarrow \Psi_i)\;, \tag{2.44}$$

where Ω is a string of quantifiers and the Φ_{i0} to Φ_{im_i} and Ψ_i are atomic formulas, and $m_i, n > 0$ [70]. Especially for first-order logic programming, this is represented in a simpler form as

$$(S_1 \wedge S_2 \wedge \cdots \wedge S_n) \rightarrow S\,. \tag{2.45}$$

This can be said as if statements S_1 to S_n are true ($n > 0$), statement S is true. In the SWRL, the rules are defined in the form

$$\text{antecedent} \Rightarrow \text{consequent}\,,$$

where *antecedent* is the body and *consequent* is the head of the rule, and both the body and the head can be conjunctions of atoms $a_1 \circ a_2 \circ \cdots a_n$, where $n \geq 0$ [68]. In practice, the SWRL rules are as in the following example

$$\text{father}\,(?a,?b) \wedge \text{father}\,(?b,?c) \rightarrow \text{grandFather}\,(?a,?c)\;,$$

where the question mark preceeding a symbol denotes a variable. This example sets a rule that if instance a has father b and if instance b has father c, then instance a has grandfather c.

From the point of view of the system modelling ontology development, the SWRL provides a mandatory extension to the expressiveness of the OWL. With semantic rules, modelling ontologies can include valuable statements about the validity of the semantic model. This is important from the usage point of view of both algorithmic and manual semantic model. For algorithmic use, semantic rules add more information for inferring; for manual use, such as semantic modelling, the rules enable necessary information for on-time model validation, which increases the efficiency and improves the modelling quality.

2.2.6 Queries in the Semantic Database

Even though semantic reasoning distills the ontology-specified instances out from the data mass, it is not exactly a database query operation. Semantic reasoning classifies the provided data based on the ontology information, and it is thus able to do implicit inferring on the data. For example, a data triple *Mary–hasBrother–Robert* implicitly tells that Robert is a man and he is a sibling of Mary. But for a large data mass, the reasoning-based approach for retrieving simple data, such as "print all rigid bodies" may be computationally too exhausting. For operations like this, a database query is a more convenient approach.

Similar operations are applied to a semantic database as to e.g. a relational database. The data is stored in and retrieved from the database, the data in the database is used, e.g. by mapping, in new applications, and database queries are applied e.g. to discover new patterns in the data or to use distilled data from the database. For data queries in the Semantic Web, a special query language, the SPARQL Protocol and RDF Query Language (SPARQL) [71, 72], has been developed. The language is similar to the Structured Query Language SQL [49], the standardised (ISO 9075) query language in relational database systems. The SPARQL unifies the data retrieval in the Semantic Web and simplifies the software application development.

A separate language for data queries in the semantic environment, together with other Semantic Web technologies and languages, forms a solid set of tools and methods for flexible and scalable data management. In addition, the modularised structure of tools and methods brings more flexibility to the software application and system development.

Chapter 3

Ontology Development for Multibody System Modelling

3.1 Design of Domain Ontology for Multibody System Modelling

A domain model for multibody system modelling describes the concepts, terms, and components for defining a multibody system unambiguously. The design of the domain model can be based on the requirements for describing some specific features in the model, or the application area of the model can be the dictating factor for the design. However, in most cases an important driving force for the domain model design is the software application implementation.

Even though some effort for the standardisation of multibody system model representation has been going on [12, 19, 20], there is no common model for representing multibody systems, and the community has not been able to agree on a unified, standardised model representation approach. All the widely used multibody system software in the market, such as LMS Virtual.Lab Motion[1], the Modelica MultiBody[2] library, MSC Adams[3], and SIMPACK[4], have their own specific formalism to describe a multibody model, and transferring a model from one system to another has been found cumbersome. The multibody system domain models used in these tools differ from each other e.g. in the granularity of the model components and in the approach of defining component information. The differences in the numerical formalisms used in the above mentioned software applications has also a significant influence on the domain model. Examples of different domain models are the Modelica MultiBody

[1]Website of LMS Virtual.Lab Motion: `http://www.lmsintl.com/simulation/virtuallab/motion`

[2]Website of Modelica Association: `http://www.modelica.org/`

[3]Website of MSC Adams: `http://www.mscsoftware.com/Products/CAE-Tools/Adams.aspx`

[4]Website of SIMPACK: `http://www.simpack.com/`

library [73] and MSC Adams [74]. In the Modelica MultiBody library, the granularity of the domain is small, whereas in MSC Adams the approach is to describe the model with generic components. For example, the Modelica MultiBody library provides thirteen components in its Parts class, while MSC Adams has two. Still, both these software applications are capable of describing the same multibody system.

To have a general multibody system modelling ontology, a new domain model for this work was designed and implemented. This multibody system domain model does not follow strictly any of the domain models used in commonly available software applications, but the applied design concepts are relatively close to those used in MSC Adams. The first version of the developed modelling ontology was introduced by the present author in [75]. The principles for the design of the multibody system modelling domain for this work are:

- clarity from the system modelling point of view,
- simplicity, and
- minimal number of components.

The management and use of modelling and simulation data and related product data, although in a concise context, resembles the objectives of the Semantic Web. Ontology design can be referred to as the design of a database or the design of an object-oriented software. In both of these cases, classes, their relations, and packaging are the key to the reusability and clarity of the design. The fundamental principles and practical aspects of database design have been described in [76]. For the ontology development of multibody system modelling, the following steps have been used [77]:

1. Classification: class data type definition and basic design of the ontology.
2. Aggregation: type attributes and composite types.
3. Generalisation: type derivation, data representation from general to special.

The used ontology development language is the OWL (version 2), although it has not been designed for engineering purposes. There are a number of tools available for OWL ontology development, and in this study Protégé and NeOn Toolkit are used for constructing the multibody system modelling ontology and for checking the validity of the ontology. The ontology model in the OWL is hierarchical, where every class is a subclass of some other class. In order to accomplish a hierarchical structure, a base class called *Thing* is provided. The property that connects subclasses to superclasses, such as *Thing*, is called *is-a*.

3.1.1 Class Hierarchy and Features of the *Mbs* Ontology

The classes that define the uppermost level of the *Mbs* ontology are presented in Figure 3.1. The developed ontology domain is called *Mbs*. The ontology is a collection of classes and subclasses, and properties, which define the predicates that connect the instances of classes. The classes can have data properties that define parameters for the class instances. The basic classes for the *Mbs* ontology are the following:

- *Analysis*, which collects all the necessary data associated to a simulation case;
- *AuxiliaryCoordinateSystem*, which can be used for defining for example constraints and forces;
- *Body*, including the general rigid body and the ground body;
- *Case*, a system simulation case, which holds all the elements of a simulation case organised under the elements model, analysis, and results;
- *Constraint*, including joints such as spherical joint or revolute joint;
- *DataElement*, including data elements variable, table, and spline curve data;
- *Force*, including vector forces, point-to-point forces, and field forces like gravity;
- *Geometry*, which contains parametrised geometries for basic shapes and a method to use external geometry e.g. from a CAD system;
- *Model* and *SubModel*, which collect all the data associated to a multibody system simulation model and enable modular modelling using nested submodels; and
- *Results*, which collects the results data associated with the analyses of the case.

The use of subclasses is a method to structure the ontology. In addition, the subclasses inherit assigned properties from the superclass. For instance, for the class *Force*, all the subclasses inherit the data properties *hasMasterMember* and *hasSlaveMember* from the superclass. The classes and their subclasses are described in more detail in the following sections.

Class *Analysis* The class *Analysis* includes all the basic information needed to run a simulation, such as analysis type, duration of the simulated time, and the size of the time step. One simulation case can have at the most one defined analysis.

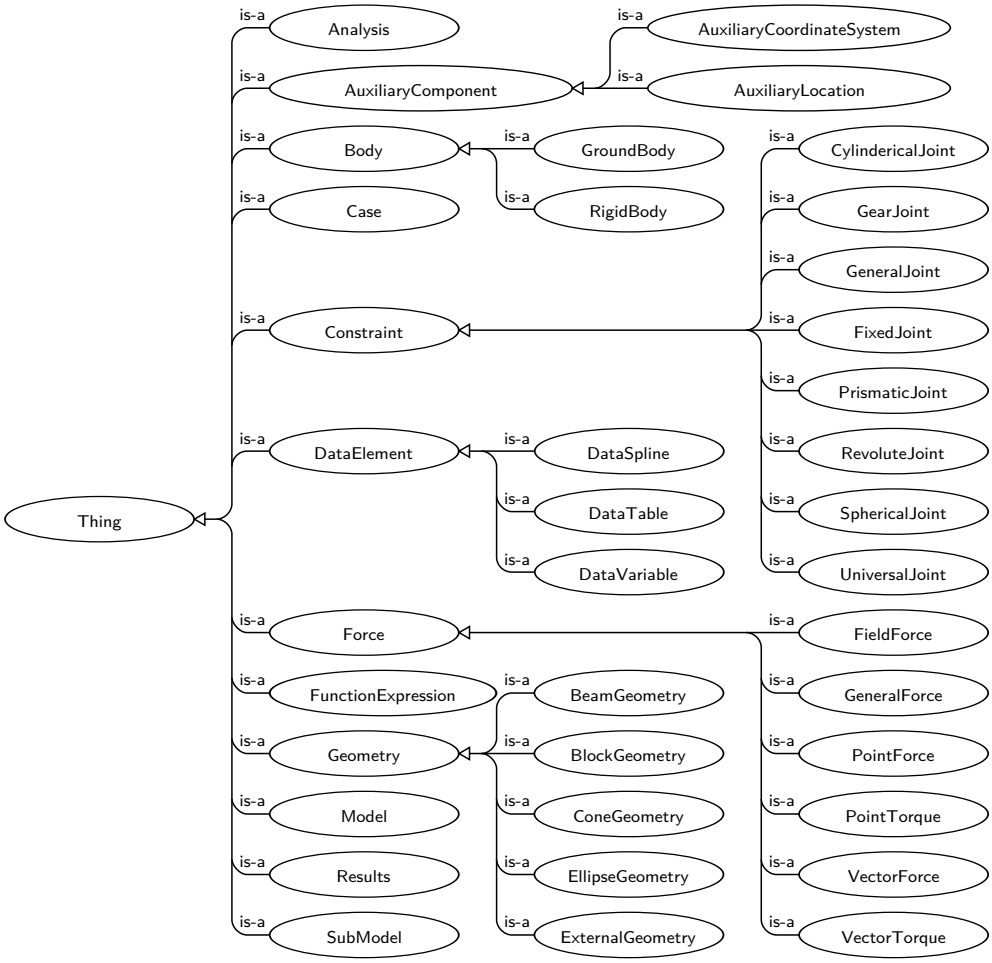


Figure 3.1: Semantic graph of the classes and subclasses of the *Mbs* ontology.

Class *AuxiliaryComponent* The class *AuxiliaryComponent* contains the subclasses *AuxiliaryCoordinateSytem* and *AuxiliaryLocation*. The former is used for positioning and orienting and the latter for positioning entities in the local reference frame of a body. An example is e.g. positioning a constraint.

Class *Body* The class *Body* defines all body and reference frame components. In this version of the *Mbs* ontology, only rigid bodies are defined, i.e. flexible bodies are not included in the ontology definition. There is no technical reason for excluding the definition of flexible bodies; flexible bodies are not included to keep the ontology research simple. Two different body subclasses are defined, *RigidBody* and *GroundBody*. *RigidBody* is one of the fundamental modelling components. It defines the local frame of reference for the body,

the mass, and the inertial tensor. *GroundBody* defines the global frame of reference, thus every valid model having instances of the class *RigidBody* has to have exactly one instance of the class *GroundBody*.

Class *Case* In the modelling hierarchy, the uppermost component in the domain is the class *Case*, which consists of at the most one component of the class *Model*, at the most one component of the class *Analysis*, and at the most one component of the class *Results*. Thus, the instances of the class *Case* are containers for all the modelling, analysis, and results data for one simulation case. There are no associated data properties for the class *Case*. The top level of the simulation case hierarchy is illustrated in Figure 3.2 with an example. In the example, the vehicleDynamics case contains the vehicle model, the cornering analysis, and the cornering results components.

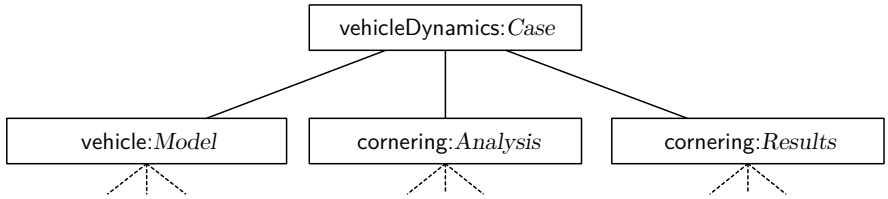


Figure 3.2: An example of simulation case hierarchy. Model components under *Model*, *Analysis*, and *Results* are omitted.

Class *Constraint* The class *Constraint* contains definitions for all the subclasses of multibody system joints and primitive joints: *CylindricalJoint*, *FixedJoint*, *GeneralJoint*, *GearJoint*, *SphericalJoint*, *RevoluteJoint*, *PrismaticJoint*, and *UniversalJoint*. All the constraints require definition of two bodies that the constraint connects, a master member and a slave member, and the location of the constraint. The master and slave member set the action and reaction force definition, respectively, for the constraint. In addition, most of the constraints require additional information for definition, such as the orientation of the constraint.

Class *DataElement* Data elements are designed for storing simulation runtime variable data. The class *DataElement* includes three subclasses: *DataSpline*, *DataTable*, and *DataVariable*. *DataSpline* is used for defining continuous and derivative two- and three-dimensional data as a function of an independent variable, such as time. The spline control points are defined in a *DataTable* element. *DataTable* is used for storing N dimensional tabular data, e.g. for defining *DataSpline* or discrete values that are a function of an independent variable. *DataVariable* is used for computing algebraic functions during the

simulation runtime. The value of an instance of *DataVariable* can be used e.g. in force function expressions.

Class *Force* The class *Force* contains definitions for all the subclasses of multibody system forces: *FieldForce*, *GeneralForce*, *PointForce*, *PointTorque*, *VectorForce*, and *VectorTorque*. In addition to the force classes, there is a class *FunctionExpression* that is used for defining the force function expression. The forces are divided into two fundamentally different types: forces acting between two points along the connecting line, and forces defined as vector elements. Forces belonging to the former type are defined between two bodies, using either the body local frames of reference or the *AuxiliaryComponent* instances. Forces belonging to the latter type are defined between two bodies by using a reference coordinate system, which can be arbitrary (either the local frame of reference or an instance of an *AuxiliaryComponent* of some body). The *FunctionExpression* of the force instance defines the magnitude of the force.

Class *Geometry* The class *Geometry* defines a small set of primitive geometries for modelling. The class contains the following subclasses: *BeamGeometry*, *BlockGeometry*, *ConeGeometry*, *EllipseGeometry*, and *ExternalGeometry*. The first four primitive geometries can be used for defining mass properties (mass and inertia) based on the basic geometrical parameters and density. The *ExternalGeometry* class is used for defining complex geometries using external sources, such as CAD models.

Class *Model* The class *Model* is the container element for all the modelling components. It includes the definition of the modelled system, its topology and relation in the modelling state. The class *Model* has a subclass *SubModel* for modular modelling. A model can have an unlimited number of submodels. An instance of the class *Model* can also be a reference to another instance of *Model* in another simulation case. This enables using one definition of a simulation model in several analyses and results in other cases, without duplicating the information. The use of submodels under a model is illustrated in Figure 3.3 with an example. In the example, the vehicle model contains submodels, such as chassis, powerTrain, and wheelSupport4. The wheelSupport4 submodel contains other modelling components, such as the upperArm4 body.

Class *SubModel* The class *SubModel* is a container element for the subassemblies of a model. A submodel requires always a model as a parent component. A simulation model can have an unlimited number of submodels. This class enables a modular modelling approach by encapsulating parts of a model into one assembly. Because a *submodel* is like any body component in the

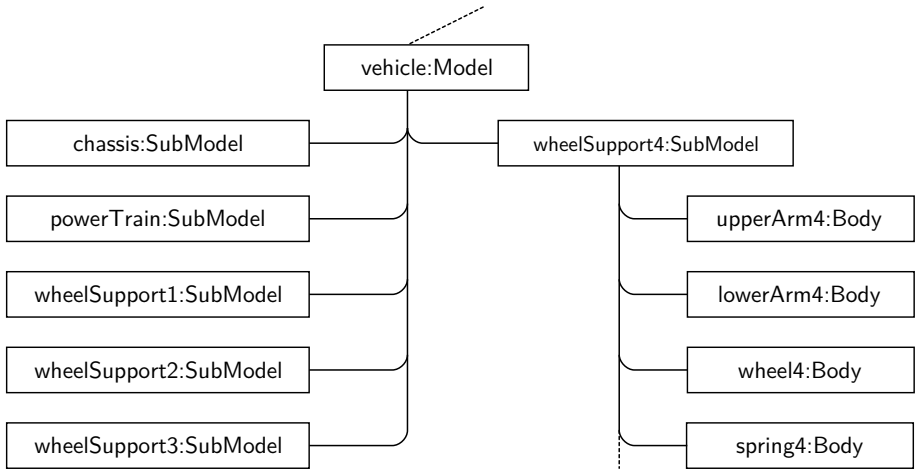


Figure 3.3: An example of simulation model hierarchy and use of submodels.

model, it must have location and orientation properties to locate and orient it under a model.

Class *Results* The class *Results* includes the stored simulation results for the case. The *Results* class is a container for the numerical data produced by the numerical solver for the analysis of the case, defined by the instance of the class *Analysis*. Thus, a case can have at the most one instance of the class *Results*.

3.1.2 Object Properties for the *Mbs* Ontology

The instances in a semantic model are connected with properties, which form the predicate in a triple. For force vectors and constraints, the natural choice for properties are the definitions for acting bodies, and in the case of instances of *VectorForce*, *VectorTorque*, and *GeneralForce*, also the definition for the reference coordinate system.

The object properties for the *Mbs* ontology are listed below. The domain and range restrictions and property characteristics are summarised in Table 3.1. The characteristics are explained briefly in chapter 2.

hasAuxiliaryComponent: In a model, attaches a local point (a definition of a location) or local frame of reference to a body.

hasCaseObject: In a modelling case, attaches case objects to the simulation case.

hasAnalysis: Connects an analysis hierarchically into a simulation case. A simulation case can have at the most one analysis.

hasModel: Connects a model hierarchically into a simulation case. A simulation case can have at the most one model; the model can be referenced from another simulation case.

hasResults: Connects a results component hierarchically into a simulation case. A simulation case can have at the most one results set.

hasFunction: In a model, attaches a function expression into a force component.

hasGeometry: In a model, attaches a geometry into a body. A body may have several geometries.

hasMasterMember: In a model, defines the reaction force element for a constraint or a force. A constraint or a force can have at the most one master member.

hasSubModel: In a model, attaches a submodel structure into the model. A model can have several submodels.

hasModelObject: This superclass contains object properties for including modelling components into a model or a submodel. It has the following subproperties:

hasBody: Attaches bodies of the model to the model.

hasConstraint: Attaches constraints of the model to the model.

hasDataElement: Attaches data elements of the model to the model.

hasForce: Attaches forces of the model to the model.

hasGroundBody: Attaches the ground body of the model to the model. The ground body is one of the rigid bodies in the model, except that it defines the global frame of reference and its mass properties are neglected.

hasSubModel: Attaches the submodels of the model to the model.

hasReferenceMember: In a model, defines the frame of reference (a body or an auxiliary coordinate system) for a force.

hasSlaveMember: In a model, defines the action force element for a constraint or a force. A constraint or a force can have at the most one master member.

Table 3.1: Object properties for the *Mbs* ontology.

Object Property	Domain	Range	Functional
hasAuxiliaryComponent	*Body*	*AuxiliaryComponent*	No
hasCaseObject	–	–	No
hasAnalysis	*Case*	*Analysis*	Yes
hasModel	*Case*	*Model*	Yes
hasResults	*Case*	*Results*	Yes
hasFunction	*Force*	*FunctionExpression*	Yes
hasGeometry	*Body*	*Geometry*	No
hasMasterMember	*Constraint* or *Force*	*AuxiliaryComponent* or *Body*	Yes
hasModelObject	–	–	No
hasBody	*Model* or *SubModel*	*Body*	No
hasConstraint	*Model* or *SubModel*	*Constraint*	No
hasDataElement	*Model* or *SubModel*	*DataElement*	No
hasForce	*Model* or *SubModel*	*Force*	No
hasGroundBody	*Model* or *SubModel*	*GroundBody*	Yes
hasSubModel	*Model*	*SubModel*	No
hasReferenceMember	*Force*	*AuxiliaryComponent* or *Body*	Yes
hasSlaveMember	*Constraint* or *Force*	*AuxiliaryComponent* or *Body*	Yes

3.1.3 Data Properties for the *Mbs* Ontology

The data properties define the features and characteristics of a class. The data properties have a type, which can be an OWL primitive datatype, like a double (double precision number) or Boolean value (true or false), or the datatype can be explicitly defined. The OWL allows restrictions to be defined for data properties. In addition, the datatypes may be restricted to e.g. set the range for the value. These datatype restrictions are treated as rules, and thus they can be explicitly reasoned. Restricted data properties can be used for checking the validity of the ontology, and especially models that are created using the ontology. As an example of data properties, the data properties for the class *RigidBody* are presented in Table 3.2. In the *Mbs* ontology, some physical properties are under a reasoning procedure, according to which the mass of a rigid body has to be zero or positive. In this case, there are data property restrictions for the mass and inertia properties. These general restrictions can be used for reasoning the validity of the multibody system simulation model. The data properties of the class Body are set as functional. Consequently, an instance of the class *RigidBody* must have only one of each of these properties. The data properties of other classes are defined similarly. The full listing of the *Mbs* ontology is presented in Appendix A.

3.1.4 Modelling Domain Constraints and Rules in the *Mbs* Ontology

The semantic constraints and restrictions in the developed *Mbs* ontology are limited mostly to data type constraints, definitions of domains and ranges, and

setting property characteristics. The constraints and restrictions have two major functions. First, they restrict the use of the ontology in such a way that only limited combinations are allowed, thus guaranteeing the validity of the model to some extent, and second, they simplify and guide the reasoning process by providing additional implicit information about the modelled system.

As an example, the restrictions of the class *RigidBody* are presented in Table 3.2. Due to the open world assumption used in the OWL, e.g. the cardinality constraint `hasLocationX exactly 1 double` does not constrain a rigid body to have one and only one property *hasLocationX*, but it defines that if an instance has a *hasLocationX* data property, there can be only one data property of that kind attached to the specific instance. This is one of the features that make the OWL a questionable tool for system modelling ontology development.

For the class *Analysis* of the *Mbs* Ontology, the restrictions are (in the OWL Manchester syntax [78]):

```
Class: Analysis
  Annotations:
    rdfs:label "Analysis"@
  SubClassOf:
    hasAnalysisType exactly 1 {"dynamic", "kinematic", "linear", "static"},
    hasStartTime max 1 xsd:double[>= 0.0],
    hasStopTime max 1 xsd:double[>= 0.0],
    hasTimeStep max 1 xsd:double[>= 0.0]
```

Here, an enumerated list is used for defining the allowed values for *hasAnalysisType*. The object properties in the developed ontology have a domain and a range restrictions set. For *hasGeometry* property these restrictions are:

domain: *Body*
range: *Geometry*

This means that *hasGeometry* can relate instances of the class *Body* (or its subclasses) to instances of the class *Geometry* (or its subclasses). If the ontology and also the model based on the ontology have been classified as valid, a reasoner can infer already from the object property *hasGeometry* that there is an instance of the class *Geometry* attached to an instance of the class *Body*.

General Modelling Domain Rules in the *Mbs* Ontology

The ontology restrictions and constraints that can be set with the OWL are limited, and some essential constraints cannot be defined with it. For example, defining a general rule for *hasMasterMember* and *hasSlaveMember* of the *Mbs* ontology so that for a given instance, another instance cannot be related by both of these properties. This inconsistent relation is illustrated in Figure 3.4, where the hinge revolute joint is connected to the door rigid body with the *hasMasterMember* and *hasSlaveMember* property.

Table 3.2: Data properties for the *RigidBody* class in the *Mbs* ontology. These data properties are all functional.

Object property	**Restriction**			**Description**
hasLocationX	exactly	1	double	Initial position component in x-direction of the global frame of reference.
hasLocationY	exactly	1	double	Initial position component in y-direction of the global frame of reference.
hasLocationZ	exactly	1	double	Initial position component in z-direction of the global frame of reference.
hasOrientationR1	exactly	1	double	Initial orientation around R1-axis of the global frame of reference according to the selected rotation sequence.
hasOrientationR2	exactly	1	double	Initial orientation around R2-axis of the global frame of reference according to the selected rotation sequence.
hasOrientationR3	exactly	1	double	Initial orientation around R3-axis of the global frame of reference according to the selected rotation sequence.
hasRotationSequence	exactly	1	”xyx”, ”xyz”, ”xzx”, ”xzy”, ”yxy”, ”yxz”, ”yzx”, ”yzy”, ”zxy”, ”zxz”, ”zyx”, ”zyz”	The rotation sequence for Euler angles.
hasVelocityX	max	1	double	Initial velocity in x-direction of the global frame of reference.
hasVelocityY	max	1	double	Initial velocity in y-direction of the global frame of reference.
hasVelocityZ	max	1	double	Initial velocity in z-direction of the global frame of reference.
hasAngularVelocityAlpha	max	1	double	Initial angular velocity around x-axis of the global frame of reference.
hasAngularVelocityBeta	max	1	double	Initial angular velocity around y-axis of the global frame of reference.
hasAngularVelocityGamma	max	1	double	Initial angular velocity around z-axis of the global frame of reference.
hasAccelerationX	max	1	double	Initial acceleration in x-direction of the global frame of reference.
hasAccelerationY	max	1	double	Initial acceleration in y-direction of the global frame of reference.
hasAccelerationZ	max	1	double	Initial acceleration in z-direction of the global frame of reference.
hasAngularAccelerationAlpha	max	1	double	Initial angular acceleration around x-axis of the global frame of reference.
hasAngularAccelerationBeta	max	1	double	Initial angular acceleration around y-axis of the global frame of reference.
hasAngularAccelerationGamma	max	1	double	Initial angular acceleration around z-axis of the global frame of reference.
hasMass	exactly	1	double[>= 0.0]	The mass of the body.
hasMassInertiaIxx	exactly	1	double[>= 0.0]	Mass inertia component I_{xx} of the body in body frame of reference.
hasMassInertiaIxy	exactly	1	double[>= 0.0]	Mass inertia component I_{xy} of the body in body frame of reference.
hasMassInertiaIxz	exactly	1	double[>= 0.0]	Mass inertia component I_{xz} of the body in body frame of reference.
hasMassInertiaIyy	exactly	1	double[>= 0.0]	Mass inertia component I_{yy} of the body in body frame of reference.
hasMassInertiaIyz	exactly	1	double[>= 0.0]	Mass inertia component I_{yz} of the body in body frame of reference.
hasMassInertiaIzz	exactly	1	double[>= 0.0]	Mass inertia component I_{zz} of the body in body frame of reference.

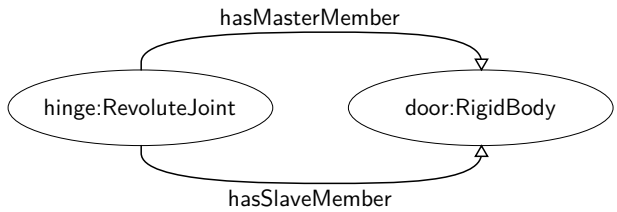


Figure 3.4: Example of an inconsistent relation in the *Mbs* Ontology.

For the constraints that cannot be modelled with the plain OWL, the SWRL is used. For example, the inconsistent situation in Figure 3.4 is restricted with the following rule:

$$\text{hasMasterMember}(?a, ?b), \text{hasSlaveMember}(?a, ?c) \rightarrow \text{differentFrom}(?b, ?c)$$

This can be read as for any given instance a related to any instance b by the property *hasMasterMember*, and for the given instance a related to any instance c by the property *hasSlaveMember*, instances b and c must be different individuals.

3.2 Semantic Representation of the Modelica MultiBody Library

Modelica[5] is an object-oriented, equation-based modelling language for system modelling [79, 80, 81]. As any other programming language, Modelica has a formal specification of its semantics and syntax. The language specification is maintained and further developed by the Modelica Association. A part of the Modelica Standard Library is the MultiBody library [73]. This library provides all the fundamental components for the description of a multibody system in three-dimensional space. The Modelica language and tools supporting the language definition are under active development. In addition, there is a development effort for creating an UML profile, ModelicaML, for the Modelica language representation.

The approach to defining components in the Modelica MultiBody library differs from the one used for the *Mbs* ontology. Instead of defining the minimum but general set of components, the Modelica MultiBody library contains many specific components that are special cases of the general corresponding component. A example of these are the components in the *Parts* class, *BodyBox* and *BodyCylinder*, which are rigid body components with a predefined geometry shape box and cylinder, respectively. The geometry is used for defining the mass properties, i.e. body mass and inertia, as well as the visualisation

[5]Website of Modelica Association: `http://www.modelica.org/`

for the body. This diversity in the library class components may simplify modelling in some cases, but on the other hand makes model translation from one representation to another more complex than the approach of a generic minimal component set.

Modelica is an object-oriented language, which leads to a naturally layered structure of high-level modelling libraries. One of the strengths of the Modelica language is its effortless support for encapsulating submodels into one modelling component. Another fundamental feature of the Modelica language is the concept of the connector. Connectors, which are implemented using connect-equation in Modelica, are the public interface of a component that are typed and must follow the following restrictions [1]:

- the primitive components of the two connectors must have the same primitive types,
- flow-variables may only connect to other flow-variables, and
- causal variables (input/output) only to causal variables (input/output).

The *Modelica MultiBody* OWL ontology is modelled only from the relevant parts. The principle of the OWL representation is to capture the structure and relations of the Modelica representation of the multibody library. In the OWL representation, the overall class hierarchy is represented, in addition to those object and data properties that are needed to show the ontology mapping between the *Modelica MultiBody* and *Mbs* ontologies. The Modelica Multibody library introduces highly packed and high-abstraction level interface to the modelling of a mechanical system. This, together with the concept of typed connection interfaces introduces some challenges to the semantic representation. Let us consider a simple model of a door with just one hinge (Figure 3.5 a). If the OWL representation of the Modelica MultiBody library is implemented in a straightforward manner, the information about the connection type has to be included in the predicates of a data triple, in the example that is for instance *has_frame_b_ConnectedTo_frame_a_Of* (Figure 3.5 b). As there are many different types of connectors in the Modelica MultiBody library, and because also other Modelica libraries can be used together with the MultiBody library, this approach leads to a large set of object properties. If the Modelica MultiBody library is opened in some parts and the internals of the modelling components are exposed to the OWL model, the *Modelica MultiBody* OWL ontology becomes easier to understand, and the number of object properties can be decreased (Figure 3.5 c). The drawback of this approach is that the component structure of the original Modelica MultiBody library is broken, and the user of the ontology is required to model instances, such as frames associated with bodies and joints, that are not explicitly present in the corresponding model in the Modelica language.

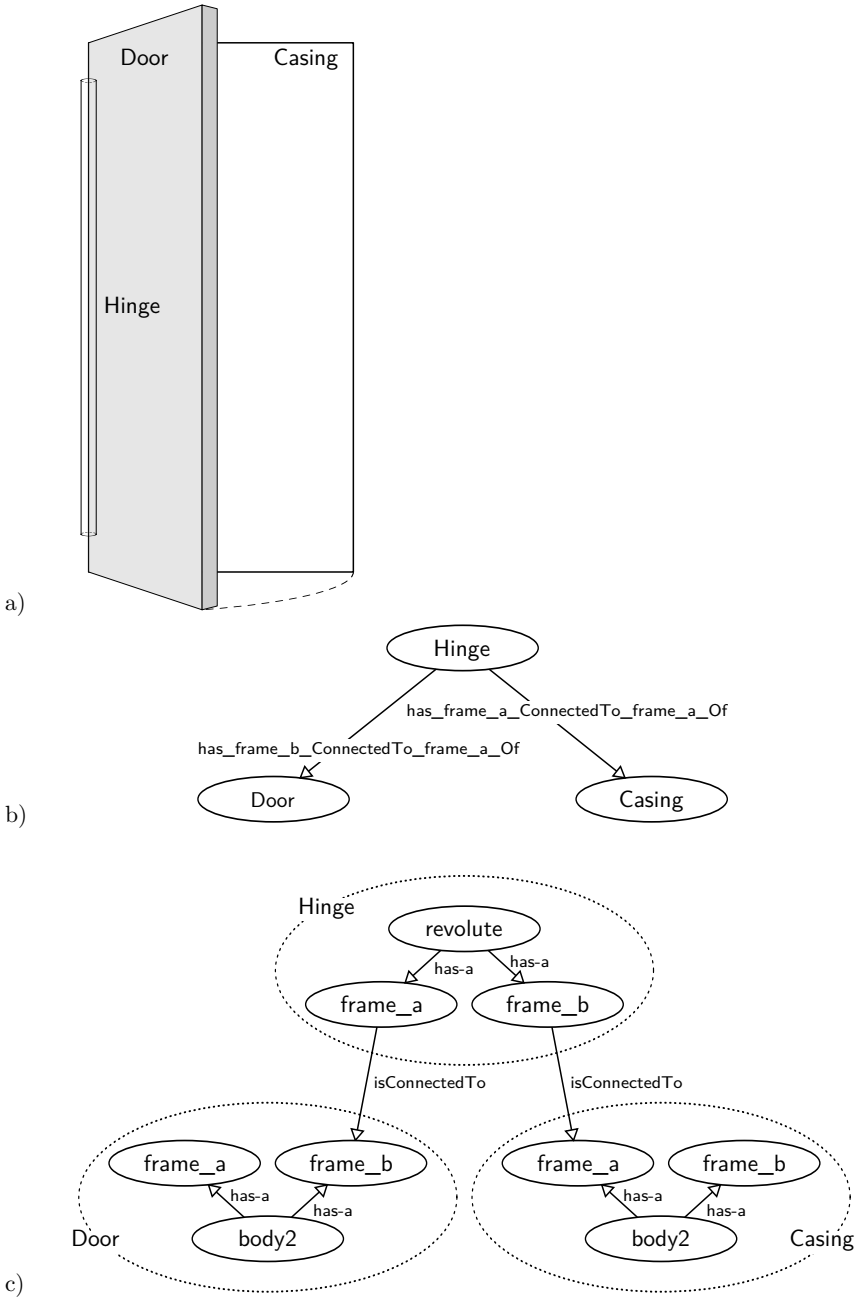


Figure 3.5: An example of the semantics of a Modelica MultiBody model of a one-hinge door. a) The model of a door with one hinge. b) The model of a door in a straightforward OWL presentation of the Modelica MultiBody library. c) The model of a door in an opened OWL presentation of the Modelica MultiBody library.

Table 3.3: Object properties of the *Modelica MultiBody* OWL ontology. All the combinations of the rows and columns are replaced to the placeholders <*row*> and <*column*> of the template respectively, i.e. $7 \times 7 = 49$ different object properties.

	axis	**bearing**	**frame_a**	**frame_b**	**frame_ia**	**frame_ib**	**support**
axis	*has_<row>_ConnectedTo_<column>_Of*						
bearing							
frame_a							
frame_b							
frame_ia							
frame_ib							
support							

In this work, due to the use of the *Modelica MultiBody* OWL ontology for demonstrating ontology mappings, the strict Modelica MultiBody library representation has been selected. The *Modelica MultiBody* OWL ontology class hierarchy is visualised in Figure 3.6 to illustrate the complexity of this ontology related to the *Mbs* ontology developed in this work. The class and subclass structure of the Modelica MultiBody library is listed in Appendix B. The object properties for the ontology are listed in Table 3.3. As this is just a partial ontology for the Modelica MultiBody library, and for selected classes and their typed connectors there are 49 object properties, this indicates that the selected ontology modelling strategy is probably not efficient and does not lead to a clear and easy use of the modelling ontology. As an example of attaching data properties to classes, the data properties for the classes *Revolute*, *World*, and *Body* are listed in Tables 3.4, 3.5, and 3.6, respectively.

Figure 3.6: OWL ontology of the Modelica MultiBody library modelling components. The object property in the graph is *hasSubClass*.

Table 3.4: Data properties associated with the class *Revolute* in the *Modelica MultiBody* OWL ontology. The descriptions for data properties are from the Modelica Standard Library version 3.1 documentation [1].

Data property	**Restriction**			**Description**	**Units**
animation	exactly	1	boolean	True, if *animation* shall be enabled.	–
cylinderColor	exactly	1	string	Color of cylinder representing the joint axis.	–
cylinderDiameter	exactly	1	double	Diameter of cylinder representing the joint axis.	[m]
cylinderLength	exactly	1	double	Length of cylinder representing the joint axis.	[m]
n	exactly	1	double	Axis of rotation resolved in frame_a.	[1]
specularCoefficient	exactly	1	double	Reflection of ambient light.	–
stateSelect	exactly	1	double	Priority to use joint angle phi and $w = der(phi)$ as states.	–
useAxisFlange	exactly	1	boolean	True, if axis flange is enabled.	–

Table 3.5: Data properties associated with the class *World* in the *Modelica MultiBody* OWL ontology. The descriptions of the data properties are from the Modelica Standard Library version 3.1 documentation [1].

Data property	**Restriction**		**Description**	**Units**
enableAnimation	exactly	1 boolean	Animation of all components is enabled.	–
animateWorld	exactly	1 boolean	World coordinate system shall be visualized.	–
animateGravity	exactly	1 boolean	Gravity field shall be visualized.	–
label1	exactly	1 string	Label of horizontal axis in icon.	–
label2	exactly	1 string	Label of vertical axis in icon.	–
gravityType	exactly	1 string	Type of gravity field.	–
g	exactly	1 double	Constant gravity acceleration.	[m/s^2]
n	exactly	1 double	Direction of gravity resolved in world frame.	–
mue	exactly	1 double	Gravity field constant.	[m^3/s^2]
driveTrainMechanics3D	exactly	1 boolean	True, if 3-dim. mechanical effects of *Parts.Mounting1D/Rotor1D/BevelGear1D* shall be taken into account.	–
axisLength	exactly	1 double	Length of world axes arrows.	[m]
axisDiameter	exactly	1 double	Diameter of world axes arrows.	[m]
axisShowLabels	exactly	1 boolean	True, if labels shall be shown.	–
axisColor_x	exactly	1 double	Color of x-arrow.	–
axisColor_y	exactly	1 double	Color of y-arrow.	–
axisColor_z	exactly	1 double	Color of z-arrow.	–
gravityArrowTail	exactly	1 double	Position vector from origin of world frame to arrow tail, resolved in world frame.	[m]
gravityArrowLength	exactly	1 double	Length of gravity arrow.	[m]
gravityArrowDiameter	exactly	1 double	Diameter of gravity arrow.	[m]
gravityArrowColor	exactly	1 double	Color of gravity arrow.	–
gravitySphereDiameter	exactly	1 double	Diameter of sphere representing gravity center.	[m]
gravitySphereColor	exactly	1 double	Color of gravity sphere.	–
nominalLength	exactly	1 double	Nominal length of multibody system.	[m]
defaultAxisLength	exactly	1 double	Default for length of a frame axis.	[m]
defaultJointLength	exactly	1 double	Default for the fixed length of a shape representing a joint.	[m]
defaultJointWidth	exactly	1 double	Default for the fixed width of a shape representing a joint.	[m]
defaultForceLength	exactly	1 double	Default for the fixed length of a shape represening a force.	[m]
defaultForceWidth	exactly	1 double	Default for the fixed width of a shape represening a force.	[m]
defaultBodyDiameter	exactly	1 double	Default for diameter of sphere representing the center of mass of a body.	[m]
defaultWidthFraction	exactly	1 double	Default for shape width as a fraction of shape length.	–
defaultArrowDiameter	exactly	1 double	Default for arrow diameter.	[m]
defaultFrameDiameterFraction	exactly	1 double	Default for arrow diameter of a coordinate system as a fraction of axis length.	–
defaultSpecularCoefficient	exactly	1 double	Default reflection of ambient light.	–
defaultN_to_m	exactly	1 double	Default scaling of force arrows.	[N/m]
defaultNm_to_m	exactly	1 double	Default scaling of torque arrows.	[Nm/m]

Table 3.6: Data properties associated with the class *Body* in the *Modelica MultiBody* OWL ontology. The descriptions of the data properties are from the Modelica Standard Library version 3.1 documentation [1].

Data property	Restriction			Description	Units
animation	exactly	1	boolean	True, if *animation* shall be enabled.	–
r_CM	exactly	1	double	Vector from *frame_a* to center of mass, resolved in *frame_a*.	[m]
m	exactly	1	double	Mass of rigid body.	[kg]
I_11	exactly	1	double	Element (1,1) of inertia tensor.	[kgm^2]
I_21	exactly	1	double	Element (2,1) of inertia tensor.	[kgm^2]
I_22	exactly	1	double	Element (2,2) of inertia tensor.	[kgm^2]
I_31	exactly	1	double	Element (3,1) of inertia tensor.	[kgm^2]
I_32	exactly	1	double	Element (3,2) of inertia tensor.	[kgm^2]
I_33	exactly	1	double	Element (3,3) of inertia tensor.	[kgm^2]
r_0.start	exactly	1	double	Position vector from origin of world frame to origin of *frame_a*.	[m]
v_0.start	exactly	1	double	Absolute velocity of *frame_a*, resolved in world frame (= $der(r_0)$).	[m/s]
a_0.start	exactly	1	double	Absolute acceleration of *frame_a* resolved in world frame (= $der(v_0)$).	[m/s^2]
angles_fixed	exactly	1	boolean	True, if *angles_start* are used as initial values, else as guess values.	–
angles_start	exactly	1	double	Initial values of angles to rotate *frame_a* around *sequence_start* axes into *frame_b*.	[rad]
sequence_start	exactly	1	double	Sequence of rotations to rotate *frame_a* into *frame_b* at initial time.	–
w_0_fixed	exactly	1	boolean	True, if *w_0_start* are used as initial values, else as guess values.	–
w_0_start	exactly	1	double	Initial or guess values of angular velocity of *frame_a* resolved in world frame.	[rad/s]
z_0_fixed	exactly	1	boolean	True, if *z_0_start* are used as initial values, else as guess values.	–
z_0_start	exactly	1	double	Initial values of angular acceleration $z_0 = der(w_0)$.	[rad/s^2]
sphereDiameter	exactly	1	double	Diameter of sphere.	[m]
sphereColor	exactly	1	double	Color of sphere.	–
cylinderDiameter	exactly	1	double	Diameter of cylinder.	[m]
cylinderColor	exactly	1	double	Color of cylinder.	–
specularCoefficient	exactly	1	double	Reflection of ambient light.	–
enforceStates	exactly	1	boolean	True, if absolute variables of body object shall be used as states.	–
useQuaternions	exactly	1	boolean	True, if quaternions shall be used as potential states otherwise use 3 angles as potential states.	–
sequence_angleStates	exactly	1	integer	Sequence of rotations to rotate world frame into frame_a around the 3 angles used as potential states.	–

Chapter 4

Semantic Approach in Multibody System Modelling

In the previous chapter, an OWL ontology was developed for representing multibody system models. In this chapter, the ontology is used for representing an example multibody system model of a double pendulum. In addition, the validation of the model data is demonstrated by first including obvious modelling faults and then using semantic reasoning to figure out the validity of the model. The use of semantic queries and semantic ontology mapping are briefly demonstrated. Along the demonstrations, details concerning the approach in general and the selected technology specifically are notified. The suitability of the semantic data model approach in general and the application of the Semantic Web technologies for multibody system modelling data representation are discussed in more detail in section 5.1.

4.1 Application of the Mbs Ontology in Multibody System Modelling

The objective of the example is to present a multibody system so that all the necessary information is available for formulating a simulation model explicitly. The necessary information for a multibody simulation model includes:

- the components of the system;
- the parameter values for the component features, such as mass properties for bodies and location and orientation of bodies, constraints, and force; and
- the topology of the model, i.e. the connectivity of the components.

A tailored model data representation method would allow straightforward description of the domain model, i.e. the design of the modelling ontology and flexible means for presenting domain restrictions and modelling constraints. The Web Ontology Language fulfills the requirements for presenting the domain modelling ontology as well as the additional information that is related to it, but it lacks of explicitness and simplicity for describing well-defined closed domains, such as multibody system modelling. With the OWL, the above-mentioned necessary information for explicit multibody system representation is presentable, but the additional requirements for model validity checking and rule-based modelling become more challenging.

4.1.1 The Multibody System Model

The developed *Mbs* ontology is used for an example of modelling a double pendulum, to demonstrate the use of the semantic data model. The model doublePendulum contains two rigid bodies (link_1 and link_2), two revolute joints (hinge_1 and hinge_2), and the gravitational force gravity. The double pendulum system is illustrated in Figures 4.1 and 4.2. In the model, the rigid body link_1 is attached to the rigid body ground with the revolute joint hinge_1, where the axis of rotation is parallel to the global Z coordinate axis and it is defined with location and direction data properties. The rigid body link_2 is attached to the rigid body link_1 with the revolute joint hinge_2, where the axis of rotation is again parallel to the global Z coordinate axis. The direction of gravity is opposite to the global Y coordinate axis.

The semantic model of the double pendulum in the form of a graph is presented in Figure 4.3. All the rigid body instances are mapped to model the instance doublePendulum with *hasObject* property. The revolute constraint instances are mapped to rigid body instances with *hasMasterMember* and *hasSlaveMember* properties. The instances, their types and data properties are presented in Table 4.1. The model properties (predicates) are presented in Table 4.2.

In a semantic database, all the model data, such as the multibody system model, and for example the control system model, are represented with the triple data model. Mappings, i.e. connecting predicates, can be defined between entities, whether or not they belong to the same domain ontology. This enables mapping of data from different modelling domains using the same procedure as within a single domain ontology. For example, in this case the geometry and mass properties of the rigid bodies link_1 and link_2 could have been retrieved by mapping the multibody system model to a semantic CAD design model. Another advantageous feature of the semantic data representation, the use of rules and constraints and the ontology reasoning based on these, can be used for increasing the value of the data and, indeed, enabling knowledge of the

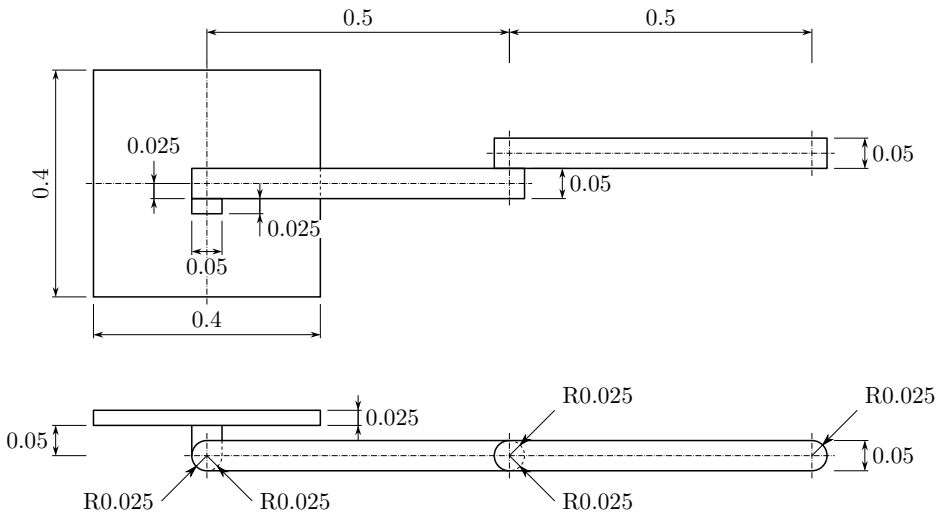


Figure 4.1: Dimensions of the doublePendulum model. The dimensions are expressed in SI units.

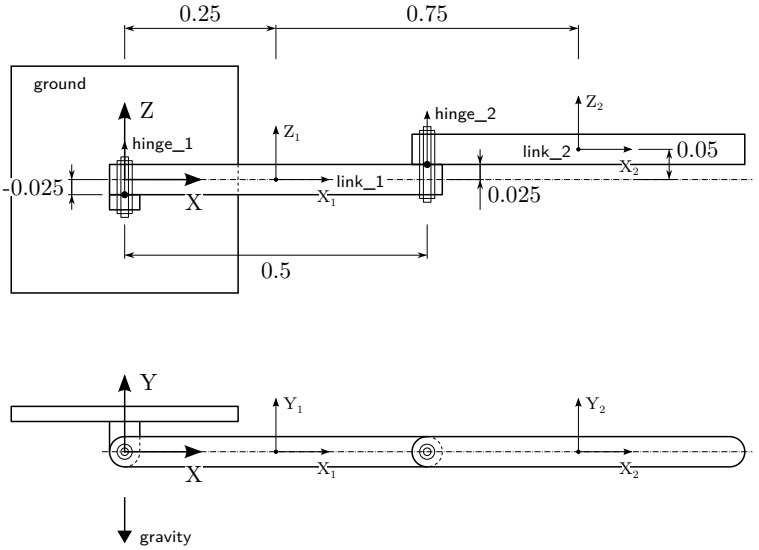


Figure 4.2: Multibody system components of the doublePendulum model and their locations and orientations. The revolute joints are assumed massless. The dimensions are expressed in SI units.

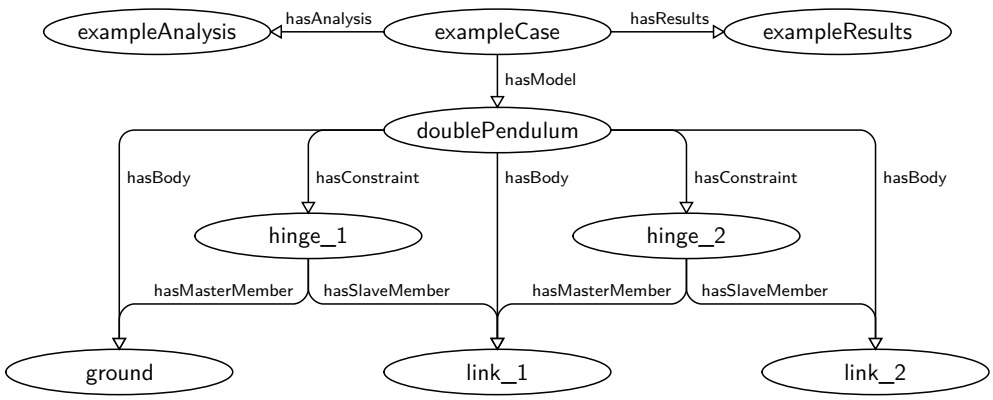


Figure 4.3: Semantic graph of the doublePendulum model as a part of the example case.

modelled domain to be stored into the semantic model. In the example above, only simple rules for mass and the mass moment of inertia components have been set. The same mechanisms of applying rules and constraints could have been used for setting for example the boundaries for geometrical dimensions. The use of reasoning with the semantic data model increases the apparent intelligence of the modelling data system.

4.1.2 Using Modelling Constraints and Rules

The modelling rules for the *Mbs* ontology were explained in chapter 3. The modelling rules can be divided into general rules applying in all the semantic models based on the ontology, and case-specific rules that are applied to the model during the modelling phase, based on special conditions. The advantage of using ontology-separate rule sets is to provide more flexibility in applying reasoning in the data.

The general rules can be used for checking the model validity against the principles of the specific domain. There are many restrictions in multibody system dynamics that should be considered. To mention some, e.g. the mass of the bodies should be zero or positive, and the constraints and forces should be implemented between the different bodies. Some of the fundamental restrictions were modelled in the *Mbs* ontology in chapter 3.

Modelling case-specific rules can be used for bounding the modelling design space tighter than what the general modelling rules do. As the general rules were defined to be used for checking the model validity against general domain modelling principles, the modelling case-specific rules may restrict also some physically meaningful features, such as the maximum value for the body mass or the number of individual bodies in the model. Modelling case-specific rules

Table 4.1: Instances (in sans serif typeface), their types (in parenthesis) and data properties with their values of the doublePendulum model. The data property values are in SI units.

exampleCase (*Case*):					
–		–		–	
doublePendulum (*Model*):					
–		–		–	
ground (*Ground*):					
hasFunctionX	”0.0”	*hasFunctionY*	”-9.80665”	*hasFunctionZ*	”0.0”
link_1 (*RigidBody*):					
hasLocationX	”0.25”	*hasLocationY*	”0.0”	*hasLocationZ*	”0.0”
hasOrientationR1	”0.0”	*hasOrientationR2*	”0.0”	*hasOrientationR3*	”0.0”
hasRotationSequence	”ZXZ”	–		–	
hasMassInertiaIxx	”5.557941E-04”	*hasMassInertiaIyy*	”3.288061E-02”	*hasMassInertiaIzz*	”3.287534E-02”
hasMassInertiaIxy	”0.0”	*hasMassInertiaIxz*	”0.0”	*hasMassInertiaIyz*	”0.0”
hasMass	”1.346567”	–		–	
link_2 (*RigidBody*):					
hasLocationX	”0.75”	*hasLocationY*	”0.0”	*hasLocationZ*	”0.05”
hasOrientationR1	”0.0”	*hasOrientationR2*	”0.0”	*hasOrientationR3*	”0.0”
hasRotationSequence	”ZXZ”	–		–	
hasMassInertiaIxx	”5.557941E-04”	*hasMassInertiaIyy*	”3.288061E-02”	*hasMassInertiaIzz*	”3.287534E-02”
hasMassInertiaIxy	”0.0”	*hasMassInertiaIxz*	”0.0”	*hasMassInertiaIyz*	”0.0”
hasMass	”1.346567”	–		–	
hinge_1 (*RevoluteJoint*):					
hasLocationX	”0.0”	*hasLocationY*	”0.0”	*hasLocationZ*	”-0.025”
hasDirectionX	”0.0”	*hasDirectionY*	”0.0”	*hasDirectionZ*	”1.0”
hinge_2 (*RevoluteJoint*):					
hasLocationX	”0.5”	*hasLocationY*	”0.0”	*hasLocationZ*	”0.05”
hasDirectionX	”0.0”	*hasDirectionY*	”0.0”	*hasDirectionZ*	”1.0”

Table 4.2: The predicates that map the instances in the doublePendulum model.

Subject	Predicate	Object
exampleCase	*hasAnalysis*	exampleAnalysis
exampleCase	*hasModel*	doublePendulum
exampleCase	*hasResults*	exampleResults
doublePendulum	*hasBody*	ground
doublePendulum	*hasBody*	gravity
doublePendulum	*hasBody*	link_1
doublePendulum	*hasBody*	link_2
doublePendulum	*hasConstraint*	hinge_1
doublePendulum	*hasConstraint*	hinge_2
hinge_1	*hasMasterMember*	ground
hinge_1	*hasSlaveMember*	link_1
hinge_2	*hasMasterMember*	link_1
hinge_2	*hasSlaveMember*	link_2

can be defined separately from the main ontology, or they can be applied during the case modelling. The flexibility to add rule sets on the data allows the design of special rule sets to be used as stencils to check the model against the requirements.

To check the consistency of the model, the Pellet[1] OWL reasoner (version 2.2.2) is used [67]. First, the modelling ontology consistency is checked:

```
1  %> pellet consistency mbs_ontology_09_06.owl
2  There are 1 input files:
3  /home/.../Ontologies/mbs_ontology_09_06.owl
4  Start loading
5  Finished loading in 00:00:00.684
6  Input size: Classes = 40, Properties = 109, Individuals = 0
7  Expressivity: ALCHQ(D)
8  Start consistency check
9  Finished consistency check in 00:00:00.031
10 Consistent: Yes
11
12 Timer summary:
13 Name               | Total (ms)
14 ============================
15 main               |       1073
16 loading            |        684
17 consistency check  |         31
```

The consistency check tells that the modelling ontology, the semantic class definition for multibody system modelling, is consistent and usable for the modelling. Next, the general modelling rules are checked for the original model. In the listing below, the output of the reasoner is presented:

[1]Website of Pellet reasoner: `http://clarkparsia.com/pellet/`

```
%> pellet realize mbs_ontology_09_06.owl 02_testModel.owl
Classifying 41 elements
Classifying:  100% complete in 00:00
Classifying finished in 00:00
Realizing 41 elements
Realizing:  100% complete in 00:00
Realizing finished in 00:00

 owl:Thing
    mbs:Analysis - (mbstestmodel:exampleAnalysis)
    mbs:AuxiliaryComponent
       mbs:AuxiliaryCoordinateSystem
       mbs:AuxiliaryLocation
    mbs:Body
       mbs:GroundBody - (mbstestmodel:ground)
       mbs:RigidBody - (mbstestmodel:link_2, mbstestmodel:link_1)
    mbs:Case - (mbstestmodel:exampleCase)
    mbs:Constraint
       mbs:CylindricalJoint
       mbs:FixedJoint
       mbs:GearJoint
       mbs:GeneralJoint
       mbs:PrismaticJoint
       mbs:RevoluteJoint - (mbstestmodel:hinge_2, mbstestmodel:hinge_1)
       mbs:SphericalJoint
       mbs:UniversalJoint
    mbs:DataElement
       mbs:DataSpline
       mbs:DataTable
       mbs:DataVariable
    mbs:Force
       mbs:FieldForce
       mbs:GeneralForce
       mbs:PointForce
       mbs:PointTorque
       mbs:VectorForce
       mbs:VectorTorque
    mbs:FunctionExpression
    mbs:Geometry
       mbs:BeamGeometry
       mbs:BlockGeometry
       mbs:ConeGeometry
       mbs:EllipseGeometry
       mbs:ExternalGeometry
    mbs:Ground - (mbstestmodel:ground)
    mbs:Model - (mbstestmodel:doublePendulum)
    mbs:Results - (mbstestmodel:exampleResults)
    mbs:SubModel
```

To demonstrate the usability of the ontology, an inconsistency of a rigid body mass is created in the model; the mass of link_1 is set to the value -1 kg, and the reasoning on the model is run again:

```
%> pellet realize mbs_ontology_09_06.owl 03_testModel.owl
ERROR: Ontology is inconsistent, run "pellet explain" to get the reason

%> pellet explain mbs_ontology_09_06.owl 03_testModel.owl
Axiom: Thing subClassOf Nothing
```

```
6
7  Explanation(s):
8  1)   Functional hasMass
9       link_1 type RigidBody
10      RigidBody subClassOf hasMass exactly 1 double[>= "0.0"^^double]
11      link_1 hasMass "-1.0"^^double
```

Similarly, the topology of the original double pendulum model is changed so that the constraint hinge_2 is related twice to link_2:

```
1  %> pellet realize mbs_ontology_09_06.owl 04_testModel.owl
2  ERROR: Ontology is inconsistent, run "pellet explain" to get the reason
3
4  %> pellet explain mbs_ontology_09_06.owl 04_testModel.owl
5  Axiom: Thing subClassOf Nothing
6
7  Explanation(s):
8  1)   hinge_2 hasMasterMember link_2
9       Rule(hasMasterMember(?a, ?b), hasSlaveMember(?a, ?c) -> differentFrom(?b, ?c))
10      hinge_2 hasSlaveMember link_2
```

The modelling case-specific rules are set to the rigid body mass properties; the mass value is limited to 2 kg. This rule is not a physical restriction, but limits the specific design to a given body mass. The modelling case-specific rule is set by defining to the superclass *Body* restriction:

```
hasMass exactly 1 double[<=2.0]
```

When setting the mass of the rigid body link_2 to be 2.5 kg and checking the model with the reasoner, the result is:

```
1  %> pellet realize mbs_ontology_09_06.owl 05_testModel.owl
2  ERROR: Ontology is inconsistent, run "pellet explain" to get the reason
3
4  %> pellet explain mbs_ontology_09_06.owl 05_testModel.owl
5  Axiom: Thing subClassOf Nothing
6
7  Explanation(s):
8  1)   Functional hasMass
9       link_2 hasMass "2.5"^^double
10      RigidBody subClassOf hasMass exactly 1 double[<= "2.0"^^double]
11      link_2 type RigidBody
```

The defined additional rule set can be stored and applied separately from the general *Mbs* ontology.

The use of the reasoner above shows some fundamental features of a semantic model applied for a multibody system simulation model. The complexity of the examples is modest, but the principle for applying this approach to complex models will be the same. It is important to note in the example above that the procedure for checking the consistency of the modelling ontology – which is done usually only at the time of developing the ontology – and the checking of the model validity follow the general approach for ontology-based

models. In addition, there are general tools, such as ontology development tools, modelling tools for semantic models and checking and validation tools (reasoner). For ontology and data representation, standardised or openly specified data formats are used. In addition to separating the data from the tools, also the functionality requirement for specific tools, such as a reasoner, is specified so that the tools in the process chain can be replaced. The description of data semantics with ontologies realises the requirement for data preservability, while the flexible and modifiable tool chain realises the requirement for a more reliable working process.

4.1.3 Queries into the Semantic Modelling Data Database

As the semantic data model is often implemented with a semantic database system, the natural requirement for such a system is to be able to perform database queries to it. To provide a general interface for data queries, a query language and applications implementing the query functionality support the quest for separating the modelling data and the tools. In the context of the Semantic Web, the SPARQL query language is the natural choice for describing the queries. This requires the database system to support the use of the language. The software architecture introduced in section 2.2.6 enables clear and open interfaces between the modelling database system and the client software. This interface can be used e.g. to perform modelling database queries to retrieve information about the models and their components.

As an example, a query to retrieve all the database components that are of the class *Body* would be:

```
1  PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
3  PREFIX mbs:      <http://www.simantics.org/ontologies/mbs.owl#>
4  PREFIX owl:      <http://www.w3.org/2002/07/owl#>
5  PREFIX sparqldl: <http://pellet.owldl.com/ns/sdle#>
6
7  SELECT ?rigid_bodies
8  WHERE {
9          ?rigid_bodies rdf:type mbs:RigidBody .
10 }
```

where `PREFIX` defines the namespace bindings and "?" denotes a variable. In the example above, there are five namespace bindings, from which the mbs is used explicitly in the query statement. The statement says: select (print) all instances that are type (class) *RigidBody* in the *Mbs* ontology. Running this query with the Pellet reasoner gives:

```
1  %> pellet query -q 06_query.sparql mbs_ontology_09_06.owl 06_testModel.owl
2  Query Results (2 answers):
3  rigid_bodies
4  ============
```

```
5  link_2
6  link_1
```

More specific and complex data queries can be applied, which makes this approach flexible for client software application purposes, especially when large databases are involved in modelling and simulation process.

4.1.4 Mapping Ontologies and Models

Ontology mapping, or ontology alignment, is the approach to enable model transformation from one representation format to another. Mappings can be used to transfer modelling data from one modelling tool to another or to use simulation results as the input in another application. Software applications in one simulation domain often use the same methods and principles in computing, and thus similar data models. An example is the area of multibody system simulation. As mentioned in chapter 2, to describe a multibody system, certain basic information about the system has to be provided. With this information, the case can be formulated for many different software applications used for multibody system simulation. This is because the data models of these tools in the multibody simulation domain are close to each other, and in that sense transforming model data from one application to another is possible. Another example is the finite element method applied for structural analysis. In this method, most of the data is transferable from one system to another, including the computational mesh, definitions of boundary regions, loads and constraints, initial conditions, and usually element and material types.

Mapping data using ontology mappings become more difficult when the data transfer is done between tools from different simulation domains. The disconformity of the data models and, more generally, the lack of overlapping on the conceptual level of describing systems may cause some required data to be missing on one domain side, or the presence of contradictory data for transforming the data from one domain to another. The ontology mapping procedure using intermediate mapping ontology is illustrated in Figure 4.4. In this example, six of the nine classes in both modelling domain ontologies are defined to be equivalent, but the remaining three classes in both ontologies require more complex mappings, which can be done by creating mapping classes in the mapping ontology.

Mapping the *Mbs* Ontology and *Modelica MultiBody* OWL Ontology

In chapter 3, a multibody system model representation and an ontology based on it were developed. This representation was based on the principle of having a minimum number of components for modelling and, instead of having

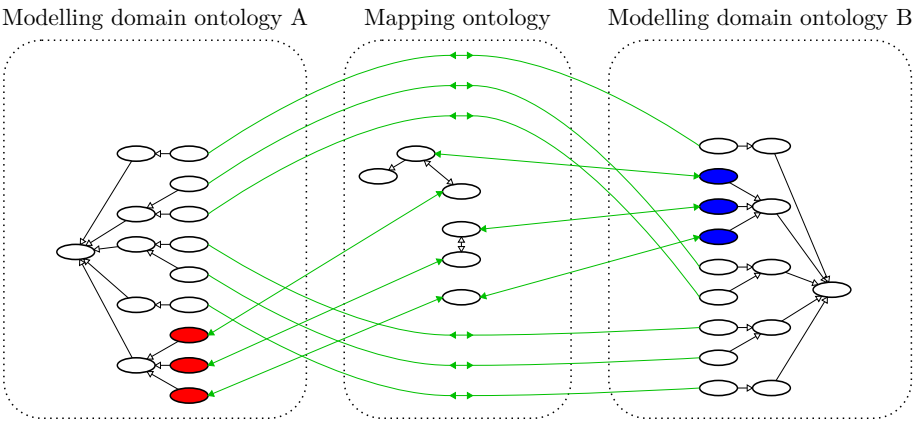


Figure 4.4: Schematic example of ontology mapping using an intermediate mapping ontology.

different variants of one basic component, the generality of the components was emphasised. In the same chapter, the partial *Modelica MultiBody* OWL ontology was introduced. The ontology implementation follows strictly the structure and naming of the Modelica MultiBody library. To demonstrate the fifth fundamental use-case of the semantic modelling data representation, i.e. data mapping, pointed out in chapter 2 on page 60, a part of the data model mapping between the *Mbs* ontology and the *Modelica MultiBody* OWL ontology is presented. Table 4.3 presents the direct mappings of data properties for counter classes *RigidBody* and *Body* in the *Mbs* ontology and *Modelica MultiBody* OWL ontology, respectively. The data property equivalences are defined with the property *equivalentProperty*. For this particular class mapping, the table shows good equivalence between the data properties. The data properties in the lower part of the table are for Modelica model visualisation, and, for this reason, not important for solving the simulation case. The parameters in the upper part of the table that have no equivalence, such as *angles_fixed* and *r_0.start_x*, can either be omitted or have default values for the computation. Due to the lack of a semantic database system, the function of the mapping cannot be demonstrated. In the demonstrated ontology class mapping, no data transformations are required. In case that for instance angular values need to be mapped from degrees to radians, the mapping class has to define the conversion which the semantic database system executes at the time when data is exported from one ontology domain to another.

Table 4.3: Ontology mapping between the *RigidBody* and *Body* classes of the *Mbs* ontology and the *Modelica MultiBody* OWL ontology, respectively. The data properties in the lower part of the table are for Modelica model visualisation.

***Mbs* ontology**	**Equivalence**	***Modelica MultiBody* OWL ontology**
hasLocationX	*equivalentProperty*	*r_CM_x*
hasLocationY	*equivalentProperty*	*r_CM_y*
hasLocationZ	*equivalentProperty*	*r_CM_z*
–	(no equivalence)	*angles_fixed* (false)
hasOrientationR1	*equivalentProperty*	*angles_start_x*
hasOrientationR2	*equivalentProperty*	*angles_start_y*
hasOrientationR3	*equivalentProperty*	*angles_start_z*
hasRotationSequence	*equivalentProperty*	*sequence_angleStates*
hasVelocityX	*equivalentProperty*	*v_0.start_x*
hasVelocityY	*equivalentProperty*	*v_0.start_y*
hasVelocityZ	*equivalentProperty*	*v_0.start_z*
–	(no equivalence)	*w_0_fixed* (false)
hasAngularVelocityAlpha	*equivalentProperty*	*w_0_start_x*
hasAngularVelocityBeta	*equivalentProperty*	*w_0_start_y*
hasAngularVelocityGamma	*equivalentProperty*	*w_0_start_z*
hasAccelerationX	*equivalentProperty*	*a_0.start_x*
hasAccelerationY	*equivalentProperty*	*a_0.start_y*
hasAccelerationZ	*equivalentProperty*	*a_0.start_z*
–	(no equivalence)	*z_0_fixed* (false)
hasAngularAccelerationAlpha	*equivalentProperty*	*z_0_start_x*
hasAngularAccelerationBeta	*equivalentProperty*	*z_0_start_y*
hasAngularAccelerationGamma	*equivalentProperty*	*z_0_start_z*
hasMass	*equivalentProperty*	*m*
hasMassInertiaIxx	*equivalentProperty*	*I_11*
hasMassInertiaIxy	*equivalentProperty*	*I_21*
hasMassInertiaIxz	*equivalentProperty*	*I_22*
hasMassInertiaIyy	*equivalentProperty*	*I_31*
hasMassInertiaIyz	*equivalentProperty*	*I_32*
hasMassInertiaIzz	*equivalentProperty*	*I_33*
–	(no equivalence)	*r_0.start_x*
–	(no equivalence)	*r_0.start_y*
–	(no equivalence)	*r_0.start_z*
–	(no equivalence)	*sequence_start*
–	(no equivalence)	*animation* (true)
–	(no equivalence)	*sphereDiameter* (default)
–	(no equivalence)	*sphereColor* (default)
–	(no equivalence)	*cylinderDiameter* (default)
–	(no equivalence)	*cylinderColor* (default)
–	(no equivalence)	*specularCoefficient* (default)
–	(no equivalence)	*enforceStates* (false)
–	(no equivalence)	*useQuaternions* (true)

Chapter 5

Discussion

5.1 Suitability of the Semantic Approach for Multibody System Modelling

The challenges in using computational methods in large-scale and especially applying a simulation-based product development approach where discussed briefly in chapters 1 and 2. The main requirements for the future can be summarized as follows:

- *Data and tool separation*: the modelling data must be independent of the tools that are used in the design process, to enable true modelling data preservability, connectivity, and integrity.
- *Knowledge capture*: all the relevant domain knowledge should be captured and linked to the modelling data, including informal silent engineering knowledge, to allow complex reasoning and e.g. model validation.
- *Data integration*: the original data source should be linked to all possible data uses; the appearance of redundant data should be avoided.

Even though the scope of this thesis is on multibody system modelling, the principles can be extended to cover, at least, the system simulation domain in general. Application of the semantic data model to multibody system modelling is an attempt to solve the above challenges. In the following sections the suitability of the approach, based on the work described in the previous chapters, is discussed. Modelling data management, especially in the product development process, should not be separated from the process context. Data management and selected methods and tools can be considered as just one part of the overall field. The process itself, the used practices and other constraints, such as available resources, influence the results and efficiency of the process. As mentioned in chapter 1, one of the main reasons for applying simulation on

product development is the need for concurrent engineering. This, on the other hand, is much dependent on the prevailing design practices, e.g. design documentation during the design process. The application of the simulation-based product development paradigm instead of the traditional design approach requires changes both in tools and data management, and processes and practices. New tools alone do not create completeness, but they provides good premisses for it.

The fundamental concept of the open world assumption (OWA) adopted in the Semantic Web technologies makes it for its part a difficult choice for system modelling data representation. As mentioned in chapter 2, system modelling domains are typically well and explicitly defined, and thus suitable for the closed world approach. Being able to describe data and knowledge with the closed world assumption enables e.g. closing out irrelevant data, which might be a problem for model validity checking. The reason for selecting the Semantic Web technologies for this work was the availability and maturity of the technologies for semantic data representation. In addition, the fact that the technologies are already relatively well-known and there is plenty of material, including technology specifications, software documentation, and research publications available, makes it a justified choice for research purposes.

5.1.1 Separation of Data and Tools

The separation of the modelling data from the computational tools means that the data format and representation are independent of the tools that can make use of it. Practically this means that the data is presented in a format that is either standardised or specially designed for the purpose.

As computational methods are becoming more important tools in the product development process, the ability to select the right computational tool for different purposes becomes necessary. Different numerical solvers for the same computational domain, such as structural analysis, have different proficiencies in some specific detail areas. An example of this is contact modelling and non-linear phenomena in general in the finite element method for structural analysis. To be able to exploit the best of the available tools, one should be able to select the tool at the state of beginning of the solving, not at the beginning of the overall modelling phase.

Data preservability becomes increasingly important when the amount of modelling data in the product life-cycle span is increasing and more modelling data is accumulated. As in the traditional design process based on design documents, the new design is usually strongly based on the previous designs and the design history of the organisation, also when the simulation-based approach is applied. The existing simulation models are used as the basis for the new design and only necessary parts are updated. This shows the value of

the modelling data in the long run. In addition, the reusability of the modelling data is one of the key factors for increasing the efficiency of the modelling process. In a long period of time, accessibility to tool-specific data decreases due to changes in software applications. If the modelling data is stored in a tool-specific format, which is binary, and data model specification is not available, the retrieval of the information from the data format may become a laborious effort. Thus, presenting the modelling data in a software-application independent form and using standardised or openly specified formats would increase the preservability of the valuable data capital.

There are several solutions for separating the data and the tools using it. One of the oldest solutions is standardisation. As discussed in chapter 1, there have also been efforts for standardising the representation multibody model data, such as the MbsML. Even though semantic data representation may simplify the integrating of pieces of data, it does not automatically solve the challenges in differences on the conceptual level. In addition, to be able to reuse the ontologies and rely on existing knowledge modelling, some ontology standardisation is required. The standardisation should include definitions for formal issues, but also guidelines and definitions for conceptual matters.

Another issue that has an influence on the separation of data and the tools using it is the willingness to support open and reusable data. For the end user, openness of data formats would be ideal. It would make it possible to add import and export capabilities for new formats even when using present techniques. However, due to either economical or other business strategic reasons, some companies prohibit others of having the specifications for the data formats they use in their products, or even protect their data formats with licensing constraints or software patents. For requests for data and tool separation concerning situations like this, the application of semantic data representation is probably not a viable solution.

5.1.2 Knowledge Capture and Reasoning

One of the most important driving forces for the application of computational methods in product development is fulfilling the multiobjective optimisation task to produce better products (in quality), in a shorter time (time-to-market), at a lower price (design and development, and manufacturing) [82, 83]. All these separate requirements aim at better economical productivity in business. This is also the main spur to apply the computational approach in product development. The use of modelling and simulation together improves the design process concerning the requirements mentioned above. Modelling itself helps to understand the structure of the product and the dependencies of substructures and subsystems. In addition, modelling is a method to document the product, especially concerning functionality and dynamics. Simulation, on the

other hand, is a tool to understand the function and dynamics of the overall system. Thus both phases of the process increase the understanding about the product, its function, and dynamics. The understanding is especially important in the development of complex products with several subsystems from different engineering domains. With modelling and simulation, designers and engineers can gain understanding of the consequences and influences of their decisions on the function of the overall product.

5.1.3 Data Integration

When the amount of modelling data involved in a product process increases, the effort to keep the data solid and valid becomes more difficult. Means to increase the integrity of the modelling data include e.g. avoiding storage of unnecessary redundant data, and linking pieces of the data to form an overall product data model. Application of the simulation-based design paradigm requires designers from different engineering disciplines to work together and to use the same source information for building simulation models. To automate the model update, the pieces of data in the simulation models should be linked to a common source, so that when the source is updated, either the simulation model is automatically updated or at least the model user is informed that the simulation model is not up-to-date. The latter may be required in cases where the model update is not explicit, but there are choices that cannot be inferred on the basis of the existing data. As a conclusion, data connectivity is mandatory for enabling data integrity.

The objectives listed above are easy to underwrite, but already the simple data mapping between the *Mbs* ontology and *Modelica MultiBody* OWL ontology demonstrated that the correspondence, even inside a single strict modelling domain, may be partial. The methods developed for general semantic knowledge representation and ontology alignment are not directly applicable for a strictly defined system modelling domain. The strictness of system modelling does not give any space for such a concept as class mapping matching factor, which is common when mapping ontologies concerning common knowledge. In the precise modelling and simulation field, the information that e.g. a concept in one ontology matches 82.3 % of another concept in another ontology does not have other value than the information that the concepts are not exactly matching and thus some additional mapping operations are needed. This means that the methods developed for automatic ontology alignment for common knowledge do not necessarily work in the modelling and simulation domain, but the ontology alignment is most likely necessary to be done manually, adding conversion concepts and methods into the intermediate mapping ontology.

The concept of modelling a master model and its implementation with ad-

vanced data modelling methods and intelligent tools does not fully solve the data integration issue in product development by itself. The application of the model and the related tools, and the fitting of the process to the objectives also play an important role. The fact that the time constraint in a development process is one directional and irreversible makes the application of concurrent engineering using the simulation-based approach a challenging task. Shortening the product development time by applying concurrent engineering causes compression to the process and introduces a new risk of wasted resources, due to wrong decisions based on false conclusions on the simulation results or even faulty simulation results. This sets new demands for the simulation and data management system to help the users to estimate the risk and its consequences on the process. This can be expanded to include also, in addition to technical and technological aspects, the economical, ecological and environmental, social, and other aspects that are related to the process and the decisions made. While simulation-based product life-cycle management is evidently out of the scope of this thesis, these questions are relevant for designing data management solutions that will scale up when this kind of implementations are topical.

5.2 Future Work

The purpose of this study was to introduce a new method for representing multibody system modelling data using semantic data representation and to demonstrate the use of this representation with an example case. The objective was to introduce the principle of using semantic data modelling for modelling data management, not to favour any specific technology to implement it. The study has left many open questions for further research. The concept of using semantic data representation for multibody system modelling data management was shown useful, but the applied Semantic Web technologies were not found to be optimal for system simulation modelling data management. An ontology language, specifically designed for system simulation model representation, would solve many of the problems pointed out in this thesis, such as the complexity of defining modelling ontology restrictions due to the open world assumption, lack of built-in structured data, such as vectors and tables, and limitations in data reasoning, again due to the open world assumption. Along with the development of the fundamental concept, also the tools and software applying the concept and the methods should be developed. This includes integration of existing modelling and simulation tools to semantic data management systems and the development of semantic modelling tools, such as ontology development tools.

Unsystematic application of semantic data management can lead to a situation of scattered concept ontologies and unnecessary complexity in ontology

alignment. Thus, standardisation and consensus on common public ontologies are required. This should be done from the very beginning of applying semantic methods in the industry . This is also necessary to guarantee a strict separation of modelling data and computational tools. If the modelling data field is scattered to numerous different ontologies, part of the advantage for data and software integration will be lost.

This work has shown that ontology mapping, especially related to system simulation data management, is not very straightforward. The exactness of the modelling data does not leave much room for approximate matching, where a concept in one ontology is almost matching to another concept in another ontology. Ontology mappings can be done, in principle, manually, but in the case of large and complex ontologies this is difficult. On the other hand, data transfer between different simulation applications using the present technologies, i.e. either using file-based direct data transfer between two software applications or using some standardised intermediate file formats, struggles with the same problem of mapping two different software-internal data models. This leaves room for further research, tailored to simulation data management, in the area of ontology mapping and alignment, including automated methods and algorithms for defining mappings between ontologies programmatically.

Chapter 6

Conclusions

In this thesis, a new concept for representing multibody system modelling data was proposed. The concept is based on the use of a semantic data model, modelling ontologies, and a triplestore, a specific database for triple-formed semantic data. The concept was demonstrated by developing a simple semantic ontology for multibody system modelling. As an application example of the developed semantic data modelling approach, the description of a double pendulum mechanism was studied. In addition, the concept of including semantic modelling rules and constraints into multibody system modelling data was demonstrated by first adding the rules and constraints into the developed modelling ontology and then using a semantic reasoner to check the validity of the created double pendulum model against the rules and constraints.

Semantic Web technologies, such as the Resource Description Framework (RDF), the Web Ontology Language (OWL), and the Semantic Web Rule Language (SWRL), were selected for the ontology development, as well as for demonstrating the concept. This was done even though these technologies, due to the applied open world assumption and the general design objectives of the technologies, were known not to be well-suited for system modelling purposes. These technologies are mature and they are general enough for research purposes. In addition, there exist many tools for e.g. ontology development and reasoning. As the concept of this approach is new, there does not exist any commonly known ontology language designed especially for system simulation modelling data management.

The presented concept can be applied to any kind of multibody system model having the components and their relations that are available in the developed modelling ontology. By modifying the ontology, new component types, such as contacts or flexible bodies, can be presented. Semantic modelling rules and constraints enable adding validation data for the modelling, both on the general modelling domain level but also on the modelling case level, e.g. in the form of product design envelope boundaries. In addition to extending the

ontology by adding new component types, the ontology can be extended by adding new modelling constraints and rules. In fact, the presented concept is applicable to any system modelling domain that is well defined, when a modelling ontology for that system domain exists and is available. Moreover, because all the data is described using the same low-level mechanism, semantic triples, mapping modelling data between different modelling domains is straightforward.

The introduced method is versatile in describing both bare modelling data and also higher-level engineering experience and knowledge of the modelled system together with the system model. Compared to direct connections between modelling and simulation tools, this approach differs fundamentally by having a common data model for all modelling and simulation tools. The difference from document-based integration and data sharing is the flexibility of creating, modifying and using ontologies. The hierarchical nature of the semantic data model itself encourages ontologies to be applied on top of each other, so that for example dimensioning rules are applied on the model under development. The use of common standardised ontologies unifies the management of the modelling data and simplifies model exchange between different modelling tools, while allowing local extensions to ontologies to be applied. This, in turn, allows the local design and modelling practices to be stored into the modelling data management system in a formalised manner. The proposed semantic database architecture enables common databases to be used for distributed modelling and simulation. Mapping other product data with the modelling data increases the integration of all product life-cycle data and enables functional product models to be used as primary data models for all product life-cycle data management. The proposed architecture combines flexibility, impressiveness, and simplicity into the same solution.

The ability to apply reasoning on the available data is one of the design objectives of the Semantic Web. Reasoning can be used for checking the consistency of the data, e.g. to validate the multibody system modelling data, but also to infer new implicit information about the data, such as absence of some specific constructs in the design. This information is not explicitly stored into the modelling data, but can be inferred from the data on the basis of the existing class instances and their relations, i.e. object properties. This can introduce new possibilities to increase the information content, and thus the value of the modelling data.

As this work has introduced the concept of applying semantic data management methods for the management of multibody system modelling data, not all the details and possibilities of the approach have been studied. There are still many interesting subjects in this area that are open for further studies. To mention one, the seemingly modest detail of the open world assumption of the OWL and the consequences of applying this assumption to strictly defined

system modelling data are still not clear. Related to this, combining engineering knowledge and modelling data, both presented in a semantic form, can introduce a new challenge of combining semantic data that includes both the open world assumption and the closed world assumption. How this is treated by the reasoner would require both basic research and practical implementation work. Another question that this work has not provided an answer to is the scalability of the information systems when using the semantic data model. The triple-formed data representation is known to be computer resource extensive, thus scaling the data model to cover all the product development data can require more computational resources than is acceptable for an industrial-level system. On the other hand, this opens new research topics for semantic database technologies, which could overcome this challenge.

All the above gives the impression that the presented method of applying semantic data representation for system modelling data management would solve most of the presented problems in modelling data management in the product process. The phases of applying simulation in the product process presented in chapter 1 Figure 1.4, and especially the fourth phase, i.e. the simulation-based product process, include tight connection to the process, i.e. the way how people implement the design work using the available tools and systems. To get the optimal advantage of the new systems and methods, also the process has to be adjusted. This is often more difficult to realise, because the process is not always as well-defined and documented than the information systems and tools. The presented approach provides methods to utilise the information systems better and to connect the knowledge to the modelling data of developed products.

Bibliography

[1] Modelica Association. Modelica – a unified object-oriented language for physical systems modeling, language specification, version 3.1. Language specification, Modelica Association, `http://www.modelica.org/documents/ModelicaSpec31.pdf`, May 2009. Accessed April 18, 2011.

[2] Hayes, P. RDF semantics. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, 2004. Accessed April 18, 2011.

[3] Oden, J., Belytschko, T., Fish, J., Hughes, T., Johnson, C., Keyes, D., Laub, A., Petzold, L., Srolovitz, D., Yip, S., and Bass, J. Simulation-based engineering science. Report, National Science Foundation, `http://www.nsf.gov/pubs/reports/sbes_final_report.pdf`, 2005. Accessed April 18, 2011.

[4] Glotzer, S., Kim, S., Cummings, P., Deshmukh, A., Head-Gordon, M., Karniadakis, G., Petzold, L., Sagui, C., and Shinozuka, M. International assessment of research and development in simulation-based engineering and science. WTEC panel report, World Technology Evaluation Center, Inc. WTEC, 2009.

[5] Malola, S. *Computational Studies of Defects in Graphene and Carbon Nanotubes*. Doctoral thesis, University of Jyväskylä, July 2009.

[6] Laiho, A. *Electromechanical Modelling and Active Control of Flexural Rotor Vibration in Cage Rotor Electrical Machines*. Doctoral thesis, Helsinki University of Technology, August 2009.

[7] Benioff, M. and Lazowska, E. Computational science: Ensuring America's competitiveness. Report, President's Information Technology Advisory Committee (PITAC), `http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf`, 2005. Accessed April 18, 2011.

[8] Haug, E. J. *Computer Aided Kinematics and Dynamics of Mechanical Systems*, volume 1: Basic Methods. Allyn and Bacon, Inc., Massachusetts, USA, 1989. ISBN 0-205-11669-8.

[9] García de Jalón, J. and Bayo, E. *Kinematic and Dynamic Simulation of Multibody Systems — The Real-Time Challenge*. Springer-Verlag New York, Inc., Secaucus, New Jersey, USA, 1994. ISBN 0-387-94096-0.

[10] Bossak, M. Simulation based design. *Journal of Materials Processing Technology*, 76(1–3):8–11, April 1998.

[11] Aziz, H., Gao, J., Maropoulos, P., and Cheung, W. Open standard, open source and peer-to-peer tools and methods for collaborative product development. *Computers in Industry*, 56(3):260–271, April 2005.

[12] Schiehlen, W. Multibody system dynamics: Roots and perspectives. *Multibody System Dynamics*, 1(2):149–188, June 1997.

[13] Schiehlen, W. Computational dynamics: Theory and applications of multibody systems. *European Journal of Mechanics – A/Solids*, 25(4):566–594, July–August 2006.

[14] Schiehlen, W. Research trends in multibody system dynamics. *Multibody System Dynamics*, 18(1):3–13, August 2007.

[15] Schiehlen, W., Guse, N., and Seifried, R. Multibody dynamics in computational mechanics and engineering applications. *Computer Methods in Applied Mechanics and Engineering*, 195(41–43):5509–5522, 2006.

[16] ISO 10303-203. Product Data Representation and Exchange – Part 203: Application Protocol: Configuration Controlled 3D Design of Mechanical Parts and Assemblies. Standard, International Organization for Standardization, 2005.

[17] ISO 10303-214. Product Data Representation and Exchange – Part 214: Application Protocol: Core Data for Automotive Mechanical Design Processes. Standard, International Organization for Standardization, 2003.

[18] Product Data Association, U. Initial graphics exchange specification IGES 5.3. Technical specification, IGES/PDES Organization, U.S. Product Data Association, Trident Research Center, Suite 204, 5300 International Blvd., N. Charleston, SC 29418, 1996. Formerly ANS US PRO/IPO-100-1996.

[19] González, M., González, F., Luaces, A., and Cuadrado, J. Interoperability and neutral data formats in multibody system simulation. *Multibody System Dynamics*, 18(1):59–72, August 2007.

[20] González, M., Dopico, D., Lugrís, U., and Cuadrado, J. A benchmarking system for MBS simulation software: Problem standardization and performance measurement. *Multibody System Dynamics*, 16(2):179–190, September 2006.

[21] González, M. *A Collaborative Environment for Flexible Development of MBS Software*. Doctoral thesis, Universidade da Coruña Escola Politécnica Superior, February 2005.

[22] Object Management Group, Inc. OMG systems modeling language (OMG SysML). Specification 1.2, Object Management Group, Inc., 140 Kendrick Street, Building A, Suite 300 Needham, MA 02494 USA, June 2010.

[23] Object Management Group, Inc. OMG unified modeling language (OMG UML), infrastructure. Specification 2.3, Object Management Group, Inc., 140 Kendrick Street, Building A, Suite 300 Needham, MA 02494 USA, May 2010.

[24] Object Management Group, Inc. OMG unified modeling language (OMG UML), superstructure. Specification 2.3, Object Management Group, Inc., 140 Kendrick Street, Building A, Suite 300 Needham, MA 02494 USA, May 2010.

[25] Sjöstedt, C. *Modeling and Simulation of Physical Systems in a Mechatronic Context*. Doctoral thesis, KTH School of Industrial Engineering, 2009.

[26] Pop, A., Akhvlediani, D., and Fritzson, P. Towards unified system modeling with the ModelicaML UML profile. In Fritzson, P., Cellier, F., and Nytsch-Geusen, C., editors, *Proceedings of the 1st International Workshop on Equation-Based Object-Oriented Languages and Tools*, pp. 13–24, July 2007.

[27] Süß, J., Fritzson, P., and Pop, A. The impreciseness of UML and implications for ModelicaML. In Fritzson, P., Cellier, F., and Broman, D., editors, *2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, pp. 17–26, July 2008.

[28] Shephard, M. S., Beall, M. W., O'Bara, R. M., and Webster, B. E. Toward simulation-based design. *Finite Elements in Analysis and Design*, 40(12):1575–1598, 2004. The Fifteenth Annual Robert J. Melosh Competition.

[29] Eben-Chaime, M., Pliskin, N., and Sosna, D. An integrated architecture for simulation. *Computers & Industrial Engineering*, 46(1):159–170, March 2004.

[30] Tisell, C. and Orsborn, K. A system for multibody analysis based on object-relational database technology. *Advances in Engineering Software*, 31(12):971–984, November–December 2000.

[31] Tisell, C. and Orsborn, K. Using an extensible object-oriented query language in multibody system analysis. *Advances in Engineering Software*, 32(10-11):769–777, October–November 2001.

[32] Daberkow, A. and Kreuzer, E. An integrated approach for computer aided design in multibody system dynamics. *Engineering with Computers*, 15(2):155–170, 1999.

[33] Kübler, R. and Schiehlen, W. Modular simulation in multibody system dynamics. *Multibody System Dynamics*, 4(2-3):107–127, August 2000.

[34] Hoffman, C. and Joan-Arinyo, R. CAD and the product master model. *Computer-Aided Design*, 30(11):905–918, September 1998.

[35] Siemers, A., Fritzson, D., and Nakhimovski, I. General meta-model based co-simulations applied to mechanical systems. *Simulation Modelling Practice and Theory*, 17(4):612–624, April 2009.

[36] Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web – a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, May 2001.

[37] Manola, F. and Miller, E. RDF primer. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`, 2004. Accessed April 18, 2011.

[38] Böhms, M., Leal, D., Graves, H., and Clark, K. Product modelling using Semantic Web technologies. W3C Incubator Group report, The World Wide Web Consortium, `http://www.w3.org/2005/Incubator/w3pm/XGR-w3pm-20091008/`, October 2009. Accessed April 18, 2011.

[39] Buchanan, B. A (very) brief history of artificial intelligence. *AI Magazine*, 26(4):53–60, 2005.

[40] Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey, USA, 3rd edition, 2010. ISBN 0-13-604259-7.

[41] Davis, R., Shrobe, H., and Szolovits, P. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[42] Goldstein, H. *Classical Mechanics*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, USA, 2nd edition, 1980. ISBN 0-201-02918-9.

[43] Bayo, E., García de Jalón, J., and Serna, M. A. A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computational Methods in Applied Mechanics and Engineering*, 71(2):183–195, 1988.

[44] Serna, M., Avilés, R., and García de Jalón, J. Dynamic analysis of plane mechanisms with lower pairs in basic coordinates. *Mechanism and Machine Theory*, 17(6):397–403, 1982.

[45] Kim, S. and Vanderploeg, M. A general and efficient method for dynamic analysis of mechanical systems using velocity transformations. *Journal of Mechanisms Transmissions and Automation in Design-Transactions of the ASME*, 108(2):176–182, June 1986.

[46] Shabana, A. A. *Computational Dynamics*. John Wiley & Sons, Inc., 605 Third Avenue, New York, New York 10158-0012, USA, first edition edition, September 1994. ISBN 0-471-30551-0.

[47] Shabana, A. A. *Dynamics of Multibody Systems*. Cambridge University Press, 40 West 20th Street, New York, New York 10011-4211, USA, 3rd edition, 2005. ISBN 0-521-85011-8.

[48] Kuipers, J. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 3 Market Place, Woodstock, Oxfordshire OX20 1SY, 2002. ISBN 0-691-05872-5.

[49] ISO/IEC 9075. Information Technology – Database Languages – SQL. Standard, International Organization for Standardization, 2008.

[50] Reiter, R. On closed world data bases. In Gallaire, H. and Minker, J., editors, *Logic and Data Bases*, pp. 55–76, New York, USA, 1978. Plenum Press.

[51] Smith, M., Welty, C., and McGuinness, D. OWL Web Ontology Language guide. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`, 2004. Accessed April 18, 2011.

[52] ISO 15836. Information and Documentation – The Dublin Core Metadata Element Set. Standard, International Organization for Standardization, 2009.

[53] Lenat, D. CYC - a large-scale investment in knowledge infrastructure. *Communications Of The ACM*, 38(11):33–38, November 1995.

[54] Herre, H. and Heller, B. Ontology of time and situoids in medical conceptual modeling. In *Proceedings of the 10th Conference on Artificial Intelligence in Medicine (AIME 05), July 23 – 27, Aberdeen, Scotland*, volume 3581, pp. 266–275. Springer-Verlag, 2005.

[55] Klyne, G. and Carroll, J. Resource Description Framework (RDF): Concepts and abstract syntax. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, 2004. Accessed April 18, 2011.

[56] Brickley, D. and Guha, R. RDF vocabulary description language 1.0: RDF Schema. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, 2004. Accessed April 18, 2011.

[57] Berners-Lee, T., Fielding, R., and Masinter, L. Uniform Resource Identifier (URI): Generic syntax. Request for Comments 3986, Network Working Group, `http://tools.ietf.org/html/rfc3986`, 2005. Accessed April 18, 2011.

[58] Patel-Schneider, P. F., Hayes, P., and Horrocks, I. OWL Web Ontology Language – semantics and abstract syntax. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-owl-semantics-20040210/semantics-all.html`, 2004. Accessed April 18, 2011.

[59] Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., and Smith, M. OWL 2 Web Ontology Language structural specification and functional-style syntax. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`, 2009. Accessed April 18, 2011.

[60] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., and Rudolph, S. OWL 2 Web Ontology Language primer. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2009/REC-owl2-primer-20091027/`, 2009. Accessed April 18, 2011.

[61] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., and Stein, L. A. OWL Web Ontology Language reference. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, 2004. Accessed April 18, 2011.

[62] Gayer, M., Kortelainen, J., and Karhela, T. CFD modelling as an integrated part of multi-level simulation of process plants – semantic modelling approach. In Wainer, G. A., Tolk, A., and Kropf, P., editors, *Proceedings of the Summer Computer Simulation Conference (SCSC'10)*, Annual. Society for Modeling and Simulation International (SCS), ACM Press, (SCS): The Society for Modeling and Simulation International, July 2010.

[63] Shadbolt, N., Hall, W., and Berners-Lee, T. The Semantic Web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May–June 2006.

[64] Ullman, J. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, May 2000.

[65] Ehrig, M. and Sure, Y. Ontology mapping – an integrated approach. In Bussler, C., Davies, J., Fensel, D., and Studer, R., editors, *Semantic Web: Research and Applications*, volume 3053 of *Lecture Notes in Computer Science*, pp. 76–91. Springer-Verlag Berlin, Heidelberger Platz 3, D-14197 Berlin, Germany, 2004.

[66] Patel, C. O. and Cimino, J. J. A network-theoretic approach for decompositional translation across Open Biological Ontologies. *Journal of Biomedical Informatics*, 43(4):608–612, August 2010.

[67] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, June 2007.

[68] Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., and Dean, M. SWRL: A Semantic Web Rule Language combining OWL and RuleML. W3C member submission, The World Wide Web Consortium, `http://www.w3.org/Submission/SWRL/`, 2004. Accessed April 18, 2011.

[69] Horn, A. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21, March 1951.

[70] Keisler, H. Reduced products and Horn classes. *Transactions of the American Mathematical Society*, 117:307–328, 1965.

[71] Prud'hommeaux, E. and Seaborne, A. SPARQL query language for RDF. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`, 2008. Accessed April 18, 2011.

[72] Harris, S. and Seaborne, A. SPARQL 1.1 query language. W3C working draft, The World Wide Web Consortium, `http://www.w3.org/TR/2010/WD-sparql11-query-20100601/`, 2010. Accessed April 18, 2011.

[73] Otter, M., Elmqvist, H., and Mattsson, S. E. The new Modelica Multibody Library. In Fritzson, P., editor, *Proceedings of the 3rd International Modelica Conference*, pp. 311–330, 2003.

[74] MSC.Software Corporation. *Adams/Solver help – MSC Adams 2010*. Documentation ID: DOC9391.

[75] Kortelainen, J. and Mikkola, A. Semantic data model in multibody system simulation. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 224(4):341–352, 2010.

[76] Gruber, T. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, November–December 1995.

[77] Smith, J. and Smith, D. Database abstractions: Aggregation and generalization. *Transactions on Database Systems*, 2(2):105–133, June 1977.

[78] Horridge, M. and Patel-Schneider, P. F. OWL 2 Web Ontology Language – manchester syntax. W3C working group note, The World Wide Web Consortium, `http://www.w3.org/2009/pdf/NOTE-owl2-manchester-syntax-20091027.pdf`, 2009. Accessed April 18, 2011.

[79] Mattsson, S., Elmqvist, H., and Otter, M. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, April 1998.

[80] Fritzson, P. Modelica – a language for equation-based physical modeling and high performance simulation. In Kagstrom, B., Dongarra, J., Elmroth, E., and Wasniewski, J., editors, *Applied Parallel Computing – Large Scale Scientific and Industrial Problems*, volume 1541 of *Lecture Notes in Computer Science*, pp. 149–160. Springer, 1998.

[81] Schwarz, P. Physically oriented modeling of heterogeneous systems. *Mathematics and Computers in Simulation*, 53(4–6):333–344, October 2000.

[82] Mujber, T., Szecsi, T., and Hashmi, M. A new hybrid dynamic modelling approach for process planning. *Journal of Materials Processing Technology*, 167(1):22–32, 2005.

[83] Zorriassatine, F., Wykes, C., Parkin, R., and Gindy, N. A survey of virtual prototyping techniques for mechanical product development. *Proceedings of the Institution of Mechanical Engineers Part B – Journal of Engineering Manufacture*, 217(4):513–530, 2003.

[84] Motik, B., Parsia, B., and Patel-Schneider, P. F. OWL 2 Web Ontology Language – XML serialization. W3C recommendation, The World Wide Web Consortium, `http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/`, 2009. Accessed April 18, 2011.

[85] Modelica Association. *Modelica Standard Library (Version 3.1) Build 6*. Modelica Association, January 2010. Cited on July 19, 2010.

Appendix A

Definition of the *Mbs* Ontology

A.1 Data Properties of the *Mbs* Ontology

Below are presented the data properties associated with the classes of the *Mbs* ontology in the Manchester syntax [78]. The rendering presents the *Mbs* ontology only partially, but is more readable than the XML Presentation syntax (OWL/XML) syntax [84]. The ontology is described in more detail in chapter 3, section 3.1 and listed in whole in section A.2.

```
...
Class: owl:Thing
Class: EllipseGeometry
  Annotations:
    rdfs:label "EllipseGeometry"@
  SubClassOf:
    Geometry,
    hasMajorRadius exactly 1 xsd:double[>= 0.0],
    hasMinorRadius exactly 1 xsd:double[>= 0.0]
Class: Analysis
  Annotations:
    rdfs:label "Analysis"@
  SubClassOf:
    hasAnalysisType exactly 1 {"dynamic" , "kinematic" , "linear" , "static"},
    hasStartTime max 1 xsd:double[>= 0.0],
    hasStopTime max 1 xsd:double[>= 0.0],
    hasTimeStep max 1 xsd:double[>= 0.0]
Class: Model
  Annotations:
    rdfs:label "Model"@
Class: CylindricalJoint
  Annotations:
    rdfs:label "CylindricalJoint"@
  SubClassOf:
    Constraint,
    hasRotationalDegreesOfFreedom value 1,
    hasTranslationalDegreesOfFreedom value 1,
    hasDirectionX exactly 1 xsd:double,
    hasDirectionY exactly 1 xsd:double,
    hasDirectionZ exactly 1 xsd:double
Class: DataElement
  Annotations:
    rdfs:label "DataElement"@
Class: Constraint
  Annotations:
    rdfs:label "Constraint"@
  SubClassOf:
```

```
      hasMasterMember exactly 1 Body,
      hasSlaveMember exactly 1 Body,
      hasLocationX exactly 1 xsd:double,
      hasLocationY exactly 1 xsd:double,
      hasLocationZ exactly 1 xsd:double
Class: ConeGeometry
  Annotations:
    rdfs:label "ConeGeometry"@
  SubClassOf:
    Geometry,
    hasBottomRadius exactly 1 xsd:double[>= 0.0],
    hasLength exactly 1 xsd:double[>= 0.0],
    hasTopRadius exactly 1 xsd:double[>= 0.0]
Class: BeamGeometry
  Annotations:
    rdfs:label "BeamGeometry"@
  SubClassOf:
    Geometry,
    hasDepth exactly 1 xsd:double[>= 0.0],
    hasHeight exactly 1 xsd:double[>= 0.0],
    hasLength exactly 1 xsd:double[>= 0.0]
Class: PrismaticJoint
  Annotations:
    rdfs:label "PrismaticJoint"@
  SubClassOf:
    Constraint,
    hasRotationalDegreesOfFreedom value 0,
    hasTranslationalDegreesOfFreedom value 1,
    hasDirectionX exactly 1 xsd:double,
    hasDirectionY exactly 1 xsd:double,
    hasDirectionZ exactly 1 xsd:double
Class: Case
  Annotations:
    rdfs:label "Case"@
  SubClassOf:
    hasAnalysis max 1 Analysis,
    hasModel max 1 Model,
    hasResults max 1 Results
Class: VectorTorque
  Annotations:
    rdfs:label "VectorTorque"@
  SubClassOf:
    Force,
    hasReferenceMember exactly 1 (AuxiliaryComponent
     or Body)
Class: FieldForce
  Annotations:
    rdfs:label "FieldForce"@
  SubClassOf:
    Force,
    hasReferenceMember exactly 1 (AuxiliaryComponent
     or Body)
Class: SubModel
  Annotations:
    rdfs:label "SubModel"@
  SubClassOf:
    owl:Thing,
    hasLocationX exactly 1 xsd:double,
    hasLocationY exactly 1 xsd:double,
    hasLocationZ exactly 1 xsd:double,
    hasOrientationR1 exactly 1 xsd:double,
    hasOrientationR2 exactly 1 xsd:double,
    hasOrientationR3 exactly 1 xsd:double,
    hasRotationSequence exactly 1 {
      "xyx", "xyz", "xzx", "xzy", "yxy", "yxz", "yzx", "yzy", "zxy", "zxz", "zyx", "zyz"}
Class: RigidBody
  Annotations:
    rdfs:label "RigidBody"@
  SubClassOf:
    Body,
    hasLocationX exactly 1 xsd:double,
    hasLocationY exactly 1 xsd:double,
    hasLocationZ exactly 1 xsd:double,
    hasMass exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIxx exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIxy exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIxz exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIyy exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIyz exactly 1 xsd:double[>= 0.0],
    hasMassInertiaIzz exactly 1 xsd:double[>= 0.0],
```

```
118       hasOrientationR1 exactly 1 xsd:double,
119       hasOrientationR2 exactly 1 xsd:double,
120       hasOrientationR3 exactly 1 xsd:double,
121       hasRotationSequence exactly 1 {
122         "xyx", "xyz", "xzx", "xzy", "yxy", "yxz", "yzx", "yzy", "zxy", "zxz", "zyx", "zyz"},
123       hasAccelerationX max 1 xsd:double,
124       hasAccelerationY max 1 xsd:double,
125       hasAccelerationZ max 1 xsd:double,
126       hasAngularAccelerationAlpha max 1 xsd:double,
127       hasAngularAccelerationBeta max 1 xsd:double,
128       hasAngularAccelerationGamma max 1 xsd:double,
129       hasAngularVelocityAlpha max 1 xsd:double,
130       hasAngularVelocityBeta max 1 xsd:double,
131       hasAngularVelocityGamma max 1 xsd:double,
132       hasVelocityX max 1 xsd:double,
133       hasVelocityY max 1 xsd:double,
134       hasVelocityZ max 1 xsd:double
135   Class: Geometry
136     Annotations:
137       rdfs:label "Geometry"@
138   Class: ExternalGeometry
139     Annotations:
140       rdfs:label "ExternalGeometry"@
141     SubClassOf:
142       Geometry,
143       hasDescription exactly 1 xsd:string
144   Class: PointForce
145     Annotations:
146       rdfs:label "PointForce"@
147     SubClassOf:
148       Force
149   Class: UniversalJoint
150     Annotations:
151       rdfs:label "UniversalJoint"@
152     SubClassOf:
153       Constraint,
154       hasRotationalDegreesOfFreedom value 2,
155       hasTranslationalDegreesOfFreedom value 0,
156       hasMasterAngle exactly 1 xsd:double,
157       hasMasterDirectionX exactly 1 xsd:double,
158       hasMasterDirectionY exactly 1 xsd:double,
159       hasMasterDirectionZ exactly 1 xsd:double,
160       hasSlaveDirectionX exactly 1 xsd:double,
161       hasSlaveDirectionY exactly 1 xsd:double,
162       hasSlaveDirectionZ exactly 1 xsd:double
163   Class: AuxiliaryCoordinateSystem
164     Annotations:
165       rdfs:label "AuxiliaryCoordinateSystem"@
166     SubClassOf:
167       AuxiliaryComponent,
168       hasOrientationR1 exactly 1 xsd:double,
169       hasOrientationR2 exactly 1 xsd:double,
170       hasOrientationR3 exactly 1 xsd:double,
171       hasRotationSequence exactly 1 {
172         "xyx", "xyz", "xzx", "xzy", "yxy", "yxz", "yzx", "yzy", "zxy", "zxz", "zyx", "zyz"}
173     DisjointWith:
174       AuxiliaryLocation
175   Class: FixedJoint
176     Annotations:
177       rdfs:label "FixedJoint"@
178     SubClassOf:
179       Constraint,
180       hasRotationalDegreesOfFreedom value 0,
181       hasTranslationalDegreesOfFreedom value 0
182   Class: DataSpline
183     Annotations:
184       rdfs:label "DataSpline"@
185     SubClassOf:
186       DataElement,
187       hasDataTable exactly 1 xsd:anyType
188   Class: Force
189     Annotations:
190       rdfs:label "Force"@
191     SubClassOf:
192       hasFunction exactly 1 FunctionExpression,
193       hasMasterMember exactly 1 (AuxiliaryComponent
194        or Body),
195       hasSlaveMember exactly 1 (AuxiliaryComponent
196        or Body)
197   Class: FunctionExpression
```

```
    Annotations:
      rdfs:label "FunctionExpression"@
  Class: GearJoint
    Annotations:
      rdfs:label "GearJoint"@
    SubClassOf:
      Constraint,
      hasRotationalDegreesOfFreedom value 1,
      hasTranslationalDegreesOfFreedom value 0,
      hasGearRatio exactly 1 xsd:double,
      hasMasterDirectionX exactly 1 xsd:double,
      hasMasterDirectionY exactly 1 xsd:double,
      hasMasterDirectionZ exactly 1 xsd:double,
      hasSlaveDirectionX exactly 1 xsd:double,
      hasSlaveDirectionY exactly 1 xsd:double,
      hasSlaveDirectionZ exactly 1 xsd:double
  Class: RevoluteJoint
    Annotations:
      rdfs:label "RevoluteJoint"@
    SubClassOf:
      Constraint,
      hasRotationalDegreesOfFreedom value 1,
      hasTranslationalDegreesOfFreedom value 0,
      hasDirectionX exactly 1 xsd:double,
      hasDirectionY exactly 1 xsd:double,
      hasDirectionZ exactly 1 xsd:double
  Class: PointTorque
    Annotations:
      rdfs:label "PointTorque"@
    SubClassOf:
      Force
  Class: SphericalJoint
    Annotations:
      rdfs:label "SphericalJoint"@
    SubClassOf:
      Constraint,
      hasRotationalDegreesOfFreedom value 3,
      hasTranslationalDegreesOfFreedom value 0
  Class: VectorForce
    Annotations:
      rdfs:label "VectorForce"@
    SubClassOf:
      Force,
      hasReferenceMember exactly 1 (AuxiliaryComponent
       or Body)
  Class: GeneralForce
    Annotations:
      rdfs:label "GeneralForce"@
    SubClassOf:
      Force,
      hasReferenceMember exactly 1 (AuxiliaryComponent
       or Body)
  Class: DataTable
    Annotations:
      rdfs:label "DataTable"@
    SubClassOf:
      DataElement,
      hasDataTable exactly 1 xsd:anyType
  Class: AuxiliaryComponent
    Annotations:
      rdfs:label "AuxiliaryComponent"@
    SubClassOf:
      hasLocationX exactly 1 xsd:double,
      hasLocationY exactly 1 xsd:double,
      hasLocationZ exactly 1 xsd:double
  Class: GroundBody
    Annotations:
      rdfs:label "Ground"@
    SubClassOf:
      Body,
      hasGravityX exactly 1 xsd:double,
      hasGravityY exactly 1 xsd:double,
      hasGravityZ exactly 1 xsd:double
  Class: Body
    Annotations:
      rdfs:label "Body"@
  Class: DataVariable
    Annotations:
      rdfs:label "DataVariable"@
    SubClassOf:
```

```
278       DataElement,
279       hasDataValue exactly 1 xsd:anyType
280   Class: AuxiliaryLocation
281     Annotations:
282       rdfs:label "AuxiliaryLocation"@
283     SubClassOf:
284       AuxiliaryComponent
285     DisjointWith:
286       AuxiliaryCoordinateSystem
287   Class: Results
288     Annotations:
289       rdfs:label "Results"@
290   Class: BlockGeometry
291     Annotations:
292       rdfs:label "BlockGeometry"@
293     SubClassOf:
294       Geometry,
295       hasDepth exactly 1 xsd:double[>= 0.0],
296       hasHeight exactly 1 xsd:double[>= 0.0],
297       hasLength exactly 1 xsd:double[>= 0.0]
298   Class: GeneralJoint
299     Annotations:
300       rdfs:label "GeneralJoint"@
301     SubClassOf:
302       Constraint,
303       hasLockedRotationX exactly 1 xsd:boolean,
304       hasLockedRotationY exactly 1 xsd:boolean,
305       hasLockedRotationZ exactly 1 xsd:boolean,
306       hasLockedTranslationX exactly 1 xsd:boolean,
307       hasLockedTranslationY exactly 1 xsd:boolean,
308       hasLockedTranslationZ exactly 1 xsd:boolean,
309       hasOrientationR1 exactly 1 xsd:double,
310       hasOrientationR2 exactly 1 xsd:double,
311       hasOrientationR3 exactly 1 xsd:double,
312       hasRotationSequence exactly 1 {
313         "xyx", "xyz", "xzx", "xzy", "yxy", "yxz", "yzx", "yzy", "zxy", "zxz", "zyx", "zyz"}
314   DisjointClasses:
315     Analysis, AuxiliaryComponent, Body, Case, Constraint, DataElement,
316     Force, FunctionExpression, Geometry, Model, Results, SubModel
317   DisjointClasses:
318     BeamGeometry, BlockGeometry, ConeGeometry, EllipseGeometry, ExternalGeometry
319   DisjointClasses:
320     CylindricalJoint, FixedJoint, GearJoint, GeneralJoint, PrismaticJoint,
321     RevoluteJoint, SphericalJoint, UniversalJoint
322   DisjointClasses:
323     FieldForce,GeneralForce,PointForce,PointTorque,VectorForce,VectorTorque
324   DisjointClasses:
325     DataSpline,DataTable,DataVariable
326   ...
```

A.2 *Mbs* Ontology

The whole *Mbs* ontology in the XML Presentation syntax (OWL/XML) [84] is listed below. This listing contains all the definitions for the ontology. The ontology metrics are given in Table A.1.

Table A.1: *Mbs* ontology metrics.

Metric	Value
Class count	39
Object property count	17
Data property count	82
DL expressivity	ALCRQ(D)
Subclass axioms count	151
Disjoint axioms classes count	6
Subobject property axioms count	9
Functional object property axioms count	8
Asymmetric object property axioms count	17
Object property domain axioms count	15
Object property range axioms count	14
Subdata property axioms count	67
Functional data property axioms count	49
Data property domain axioms count	10
AnnotationAssertion axioms count	136

A.2.1 *Mbs* Ontology Listing

```
<?xml version="1.0"?>
<!-- Copyright (C) Juha Kortelainen 2011 -->
<!DOCTYPE Ontology
[
  <!ENTITY xsd
    "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY xml
    "http://www.w3.org/XML/1998/namespace" >
  <!ENTITY rdfs
    "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
xml:base="http://www.simantics.org/ontologies/mbs.owl"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
ontologyIRI="http://www.simantics.org/ontologies/mbs.owl">
<Prefix name="rdf"
IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
<Prefix name="rdfs"
  IRI="http://www.w3.org/2000/01/rdf-schema#" />
<Prefix name="xsd"
  IRI="http://www.w3.org/2001/XMLSchema#" />
<Prefix name="owl"
  IRI="http://www.w3.org/2002/07/owl#" />
<Declaration>
<Class IRI="#Analysis" />
</Declaration>
<Declaration>
<Class IRI="#AuxiliaryComponent" />
</Declaration>
<Declaration>
<Class IRI="#AuxiliaryCoordinateSystem" />
</Declaration>
<Declaration>
<Class IRI="#AuxiliaryLocation" />
</Declaration>
<Declaration>
<Class IRI="#BeamGeometry" />
</Declaration>
<Declaration>
<Class IRI="#BlockGeometry" />
</Declaration>
<Declaration>
<Class IRI="#Body" />
</Declaration>
<Declaration>
<Class IRI="#Case" />
</Declaration>
<Declaration>
<Class IRI="#ConeGeometry" />
```

```
</Declaration>
<Declaration>
<Class IRI="#Constraint" />
</Declaration>
<Declaration>
<Class IRI="#CylindricalJoint" />
</Declaration>
<Declaration>
<Class IRI="#DataElement" />
</Declaration>
<Declaration>
<Class IRI="#DataSpline" />
</Declaration>
<Declaration>
<Class IRI="#DataTable" />
</Declaration>
<Declaration>
<Class IRI="#DataVariable" />
</Declaration>
<Declaration>
<Class IRI="#EllipseGeometry" />
</Declaration>
<Declaration>
<Class IRI="#ExternalGeometry" />
</Declaration>
<Declaration>
<Class IRI="#FieldForce" />
</Declaration>
<Declaration>
<Class IRI="#FixedJoint" />
</Declaration>
<Declaration>
<Class IRI="#Force" />
</Declaration>
<Declaration>
<Class IRI="#FunctionExpression" />
</Declaration>
<Declaration>
<Class IRI="#GearJoint" />
</Declaration>
<Declaration>
<Class IRI="#GeneralForce" />
</Declaration>
<Declaration>
<Class IRI="#GeneralJoint" />
</Declaration>
<Declaration>
<Class IRI="#Geometry" />
</Declaration>
<Declaration>
<Class IRI="#GroundBody" />
</Declaration>
<Declaration>
<Class IRI="#Model" />
</Declaration>
<Declaration>
<Class IRI="#PointForce" />
</Declaration>
<Declaration>
<Class IRI="#PointTorque" />
</Declaration>
<Declaration>
<Class IRI="#PrismaticJoint" />
</Declaration>
<Declaration>
<Class IRI="#Results" />
</Declaration>
<Declaration>
<Class IRI="#RevoluteJoint" />
</Declaration>
<Declaration>
<Class IRI="#RigidBody" />
</Declaration>
<Declaration>
<Class IRI="#SphericalJoint" />
</Declaration>
<Declaration>
<Class IRI="#SubModel" />
</Declaration>
<Declaration>
<Class IRI="#UniversalJoint" />
</Declaration>
<Declaration>
<Class IRI="#VectorForce" />
</Declaration>
<Declaration>
<Class IRI="#VectorTorque" />
</Declaration>
<Declaration>
<Class abbreviatedIRI="owl:Thing" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasAnalysis" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasAuxiliaryComponent" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasBody" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasCaseObject" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasContraint" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasDataElement" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasForce" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasFunction" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasGeometry" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasGroundBody" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasMasterMember" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasModel" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasModelObject" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasReferenceMember" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasResults" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasSlaveMember" />
</Declaration>
<Declaration>
<ObjectProperty IRI="#hasSubModel" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAcceleration" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAccelerationX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAccelerationY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAccelerationZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAnalysisParameter" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAnalysisType" />
</Declaration>
```

```
<Declaration>
<DataProperty IRI="#hasAngularAcceleration" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularAccelerationAlpha" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularAccelerationBeta" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularAccelerationGamma" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularVelocity" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularVelocityAlpha" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularVelocityBeta" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasAngularVelocityGamma" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasBottomRadius" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDataElementValue" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDataTable" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDataValue" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDegreesOfFreedom" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDepth" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDescription" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDirection" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDirectionX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDirectionY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasDirectionZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGearRatio" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGeometryParameter" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGravity" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGravityX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGravityY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasGravityZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasHeight" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLength" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLocation" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLocationX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLocationY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLocationZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedRotation" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedRotationX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedRotationY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedRotationZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedTranslation" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedTranslationX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedTranslationY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasLockedTranslationZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMajorRadius" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMass" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertia" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIxx" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIxy" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIxz" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIyy" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIyz" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMassInertiaIzz" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMasterAngle" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMasterDirectionX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMasterDirectionY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMasterDirectionZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasMinorRadius" />
</Declaration>
<Declaration>
```

```
<DataProperty IRI="#hasOrientation" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationQ0" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationQ1" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationQ2" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationQ3" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationR1" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationR2" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasOrientationR3" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasRotationSequence" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasSlaveDirectionX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasSlaveDirectionY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasSlaveDirectionZ" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasStartTime" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasStopTime" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasTimeStep" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasTopRadius" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasVelocity" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasVelocityX" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasVelocityY" />
</Declaration>
<Declaration>
<DataProperty IRI="#hasVelocityZ" />
</Declaration>
<Declaration>
<DataProperty abbreviatedIRI="owl:topDataProperty" />
</Declaration>
<Declaration>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
</Declaration>
<Declaration>
<Datatype abbreviatedIRI="xsd:anyType" />
</Declaration>
<SubClassOf>
<Class IRI="#Analysis" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasAnalysisType" />
<DataOneOf>
<Literal datatypeIRI="&xsd;string">dynamic</Literal>
<Literal datatypeIRI="&xsd;string">kinematic</Literal>
<Literal datatypeIRI="&xsd;string">linear</Literal>
<Literal datatypeIRI="&xsd;string">static</Literal>
</DataOneOf>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Analysis" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasStartTime" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Analysis" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasStopTime" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Analysis" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasTimeStep" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryComponent" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryComponent" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryComponent" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryCoordinateSystem" />
<Class IRI="#AuxiliaryComponent" />
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryCoordinateSystem" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR1" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryCoordinateSystem" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR2" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
```

```
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryCoordinateSystem" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR3" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryCoordinateSystem" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasRotationSequence" />
<DataOneOf>
<Literal datatypeIRI="&xsd;string">xyx</Literal>
<Literal datatypeIRI="&xsd;string">xyz</Literal>
<Literal datatypeIRI="&xsd;string">xzx</Literal>
<Literal datatypeIRI="&xsd;string">xzy</Literal>
<Literal datatypeIRI="&xsd;string">yxy</Literal>
<Literal datatypeIRI="&xsd;string">yxz</Literal>
<Literal datatypeIRI="&xsd;string">yzx</Literal>
<Literal datatypeIRI="&xsd;string">yzy</Literal>
<Literal datatypeIRI="&xsd;string">zxy</Literal>
<Literal datatypeIRI="&xsd;string">zxz</Literal>
<Literal datatypeIRI="&xsd;string">zyx</Literal>
<Literal datatypeIRI="&xsd;string">zyz</Literal>
</DataOneOf>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#AuxiliaryLocation" />
<Class IRI="#AuxiliaryComponent" />
</SubClassOf>
<SubClassOf>
<Class IRI="#BeamGeometry" />
<Class IRI="#Geometry" />
</SubClassOf>
<SubClassOf>
<Class IRI="#BeamGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDepth" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#BeamGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasHeight" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#BeamGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLength" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#BlockGeometry" />
<Class IRI="#Geometry" />
</SubClassOf>
<SubClassOf>
<Class IRI="#BlockGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDepth" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#BlockGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasHeight" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#BlockGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLength" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Case" />
<ObjectMaxCardinality cardinality="1">
<ObjectProperty IRI="#hasAnalysis" />
<Class IRI="#Analysis" />
</ObjectMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Case" />
<ObjectMaxCardinality cardinality="1">
<ObjectProperty IRI="#hasModel" />
<Class IRI="#Model" />
</ObjectMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Case" />
<ObjectMaxCardinality cardinality="1">
<ObjectProperty IRI="#hasResults" />
<Class IRI="#Results" />
</ObjectMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#ConeGeometry" />
<Class IRI="#Geometry" />
</SubClassOf>
<SubClassOf>
<Class IRI="#ConeGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasBottomRadius" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#ConeGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLength" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
```

```
</SubClassOf>
<SubClassOf>
<Class IRI="#ConeGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasTopRadius" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Constraint" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasMasterMember" />
<Class IRI="#Body" />
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Constraint" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasSlaveMember" />
<Class IRI="#Body" />
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Constraint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Constraint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Constraint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">1</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">1</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#CylindricalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#DataSpline" />
<Class IRI="#DataElement" />
</SubClassOf>
<SubClassOf>
<Class IRI="#DataSpline" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDataTable" />
<Datatype abbreviatedIRI="xsd:anyType" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#DataTable" />
<Class IRI="#DataElement" />
</SubClassOf>
<SubClassOf>
<Class IRI="#DataTable" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDataTable" />
<Datatype abbreviatedIRI="xsd:anyType" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#DataVariable" />
<Class IRI="#DataElement" />
</SubClassOf>
<SubClassOf>
<Class IRI="#DataVariable" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDataValue" />
<Datatype abbreviatedIRI="xsd:anyType" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#EllipseGeometry" />
<Class IRI="#Geometry" />
</SubClassOf>
<SubClassOf>
<Class IRI="#EllipseGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMajorRadius" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#EllipseGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMinorRadius" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#ExternalGeometry" />
<Class IRI="#Geometry" />
</SubClassOf>
<SubClassOf>
<Class IRI="#ExternalGeometry" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDescription" />
<Datatype abbreviatedIRI="xsd:string" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
```

```
<Class IRI="#FieldForce" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#FieldForce" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasReferenceMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#FixedJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#FixedJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#FixedJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#Force" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasFunction" />
<Class IRI="#FunctionExpression" />
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Force" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasMasterMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#Force" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasSlaveMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">1</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasGearRatio" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GearJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralForce" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralForce" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasReferenceMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLockedRotationX" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLockedRotationY" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
```

```
<DataProperty IRI="#hasLockedRotationZ" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLockedTranslationX" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLockedTranslationY" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLockedTranslationZ" />
<Datatype abbreviatedIRI="xsd:boolean" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR1" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR2" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR3" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GeneralJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasRotationSequence" />
<DataOneOf>
<Literal datatypeIRI="&xsd;string">xyx</Literal>
<Literal datatypeIRI="&xsd;string">xyz</Literal>
<Literal datatypeIRI="&xsd;string">xzx</Literal>
<Literal datatypeIRI="&xsd;string">xzy</Literal>
<Literal datatypeIRI="&xsd;string">yxy</Literal>
<Literal datatypeIRI="&xsd;string">yxz</Literal>
<Literal datatypeIRI="&xsd;string">yzx</Literal>
<Literal datatypeIRI="&xsd;string">yzy</Literal>
<Literal datatypeIRI="&xsd;string">zxy</Literal>
<Literal datatypeIRI="&xsd;string">zxz</Literal>
<Literal datatypeIRI="&xsd;string">zyx</Literal>
<Literal datatypeIRI="&xsd;string">zyz</Literal>
</DataOneOf>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GroundBody" />
<Class IRI="#Body" />
</SubClassOf>
<SubClassOf>
<Class IRI="#GroundBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasGravityX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GroundBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasGravityY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#GroundBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasGravityZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#PointForce" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#PointTorque" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">1</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#PrismaticJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">1</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionX" />
```

```
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RevoluteJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<Class IRI="#Body" />
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMass" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIxx" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIxy" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIxz" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIyy" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIyz" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMassInertiaIzz" />
<DatatypeRestriction>
<Datatype abbreviatedIRI="xsd:double" />
<FacetRestriction facet="&xsd;minInclusive">
<Literal datatypeIRI="&xsd;double">0.0</Literal>
</FacetRestriction>
</DatatypeRestriction>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR1" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR2" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR3" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasRotationSequence" />
<DataOneOf>
<Literal datatypeIRI="&xsd;string">xyx</Literal>
<Literal datatypeIRI="&xsd;string">xyz</Literal>
<Literal datatypeIRI="&xsd;string">xzx</Literal>
<Literal datatypeIRI="&xsd;string">xzy</Literal>
<Literal datatypeIRI="&xsd;string">yxy</Literal>
<Literal datatypeIRI="&xsd;string">yxz</Literal>
<Literal datatypeIRI="&xsd;string">yzx</Literal>
<Literal datatypeIRI="&xsd;string">yzy</Literal>
```

```
<Literal datatypeIRI="&xsd;string">zxy</Literal>
<Literal datatypeIRI="&xsd;string">zxz</Literal>
<Literal datatypeIRI="&xsd;string">zyx</Literal>
<Literal datatypeIRI="&xsd;string">zyz</Literal>
</DataOneOf>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAccelerationX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAccelerationY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAccelerationZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularAccelerationAlpha" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularAccelerationBeta" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularAccelerationGamma" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularVelocityAlpha" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularVelocityBeta" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasAngularVelocityGamma" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasVelocityX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasVelocityY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#RigidBody" />
<DataMaxCardinality cardinality="1">
<DataProperty IRI="#hasVelocityZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataMaxCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SphericalJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#SphericalJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">3</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#SphericalJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<Class abbreviatedIRI="owl:Thing" />
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasLocationZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR1" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR2" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasOrientationR3" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#SubModel" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasRotationSequence" />
<DataOneOf>
```

```
<Literal datatypeIRI="&xsd;string">xyx</Literal>
<Literal datatypeIRI="&xsd;string">xyz</Literal>
<Literal datatypeIRI="&xsd;string">xzx</Literal>
<Literal datatypeIRI="&xsd;string">xzy</Literal>
<Literal datatypeIRI="&xsd;string">yxy</Literal>
<Literal datatypeIRI="&xsd;string">yxz</Literal>
<Literal datatypeIRI="&xsd;string">yzx</Literal>
<Literal datatypeIRI="&xsd;string">yzy</Literal>
<Literal datatypeIRI="&xsd;string">zxy</Literal>
<Literal datatypeIRI="&xsd;string">zxz</Literal>
<Literal datatypeIRI="&xsd;string">zyx</Literal>
<Literal datatypeIRI="&xsd;string">zyz</Literal>
</DataOneOf>
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<Class IRI="#Constraint" />
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataHasValue>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">2</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataHasValue>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<Literal datatypeIRI="&xsd;integer">0</Literal>
</DataHasValue>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterAngle" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasMasterDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionX" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionY" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#UniversalJoint" />
<DataExactCardinality cardinality="1">
<DataProperty IRI="#hasSlaveDirectionZ" />
<Datatype abbreviatedIRI="xsd:double" />
</DataExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#VectorForce" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#VectorForce" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasReferenceMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<SubClassOf>
<Class IRI="#VectorTorque" />
<Class IRI="#Force" />
</SubClassOf>
<SubClassOf>
<Class IRI="#VectorTorque" />
<ObjectExactCardinality cardinality="1">
<ObjectProperty IRI="#hasReferenceMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectExactCardinality>
</SubClassOf>
<DisjointClasses>
<Class IRI="#Analysis" />
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
<Class IRI="#Case" />
<Class IRI="#Constraint" />
<Class IRI="#DataElement" />
<Class IRI="#Force" />
<Class IRI="#FunctionExpression" />
<Class IRI="#Geometry" />
<Class IRI="#Model" />
<Class IRI="#Results" />
<Class IRI="#SubModel" />
</DisjointClasses>
<DisjointClasses>
<Class IRI="#AuxiliaryCoordinateSystem" />
<Class IRI="#AuxiliaryLocation" />
</DisjointClasses>
<DisjointClasses>
<Class IRI="#BeamGeometry" />
<Class IRI="#BlockGeometry" />
<Class IRI="#ConeGeometry" />
<Class IRI="#EllipseGeometry" />
<Class IRI="#ExternalGeometry" />
</DisjointClasses>
<DisjointClasses>
<Class IRI="#CylindricalJoint" />
<Class IRI="#FixedJoint" />
<Class IRI="#GearJoint" />
<Class IRI="#GeneralJoint" />
<Class IRI="#PrismaticJoint" />
<Class IRI="#RevoluteJoint" />
<Class IRI="#SphericalJoint" />
<Class IRI="#UniversalJoint" />
</DisjointClasses>
<DisjointClasses>
<Class IRI="#DataSpline" />
<Class IRI="#DataTable" />
<Class IRI="#DataVariable" />
</DisjointClasses>
<DisjointClasses>
<Class IRI="#FieldForce" />
<Class IRI="#GeneralForce" />
<Class IRI="#PointForce" />
<Class IRI="#PointTorque" />
<Class IRI="#VectorForce" />
<Class IRI="#VectorTorque" />
</DisjointClasses>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasAnalysis" />
```

```
<ObjectProperty IRI="#hasCaseObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasBody" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasContraint" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasDataElement" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasForce" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasGroundBody" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasModel" />
<ObjectProperty IRI="#hasCaseObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasResults" />
<ObjectProperty IRI="#hasCaseObject" />
</SubObjectPropertyOf>
<SubObjectPropertyOf>
<ObjectProperty IRI="#hasSubModel" />
<ObjectProperty IRI="#hasModelObject" />
</SubObjectPropertyOf>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasAnalysis" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasFunction" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasGroundBody" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasMasterMember" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasModel" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasReferenceMember" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasResults" />
</FunctionalObjectProperty>
<FunctionalObjectProperty>
<ObjectProperty IRI="#hasSlaveMember" />
</FunctionalObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasAnalysis" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasAuxiliaryComponent" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasBody" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasCaseObject" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasContraint" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasDataElement" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasForce" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasFunction" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasGeometry" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasGroundBody" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasMasterMember" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasModel" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasModelObject" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasReferenceMember" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasResults" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasSlaveMember" />
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
<ObjectProperty IRI="#hasSubModel" />
</AsymmetricObjectProperty>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasAnalysis" />
<Class IRI="#Case" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasAuxiliaryComponent" />
<Class IRI="#RigidBody" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasBody" />
<ObjectUnionOf>
<Class IRI="#Model" />
<Class IRI="#SubModel" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasContraint" />
<ObjectUnionOf>
<Class IRI="#Model" />
<Class IRI="#SubModel" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasDataElement" />
<ObjectUnionOf>
<Class IRI="#Model" />
<Class IRI="#SubModel" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasForce" />
<ObjectUnionOf>
<Class IRI="#Model" />
<Class IRI="#SubModel" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasFunction" />
<Class IRI="#Force" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasGeometry" />
<Class IRI="#RigidBody" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasGroundBody" />
<Class IRI="#Model" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasMasterMember" />
<ObjectUnionOf>
```

```
<Class IRI="#Constraint" />
<Class IRI="#Force" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasModel" />
<Class IRI="#Case" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasReferenceMember" />
<Class IRI="#Force" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasResults" />
<Class IRI="#Case" />
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasSlaveMember" />
<ObjectUnionOf>
<Class IRI="#Constraint" />
<Class IRI="#Force" />
</ObjectUnionOf>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
<ObjectProperty IRI="#hasSubModel" />
<Class IRI="#Model" />
</ObjectPropertyDomain>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasAnalysis" />
<Class IRI="#Analysis" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasAuxiliaryComponent" />
<Class IRI="#AuxiliaryComponent" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasContraint" />
<Class IRI="#Constraint" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasDataElement" />
<Class IRI="#DataElement" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasForce" />
<Class IRI="#Force" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasFunction" />
<Class IRI="#FunctionExpression" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasGeometry" />
<Class IRI="#Geometry" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasGroundBody" />
<Class IRI="#GroundBody" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasMasterMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasModel" />
<Class IRI="#Model" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasReferenceMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasResults" />
<Class IRI="#Results" />
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasSlaveMember" />
<ObjectUnionOf>
<Class IRI="#AuxiliaryComponent" />
<Class IRI="#Body" />
</ObjectUnionOf>
</ObjectPropertyRange>
<ObjectPropertyRange>
<ObjectProperty IRI="#hasSubModel" />
<Class IRI="#SubModel" />
</ObjectPropertyRange>
<SubDataPropertyOf>
<DataProperty IRI="#hasAcceleration" />
<DataProperty abbreviatedIRI="owl:topDataProperty" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAccelerationX" />
<DataProperty IRI="#hasAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAccelerationY" />
<DataProperty IRI="#hasAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAccelerationZ" />
<DataProperty IRI="#hasAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAnalysisType" />
<DataProperty IRI="#hasAnalysisParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularAcceleration" />
<DataProperty abbreviatedIRI="owl:topDataProperty" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularAccelerationAlpha" />
<DataProperty IRI="#hasAngularAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularAccelerationBeta" />
<DataProperty IRI="#hasAngularAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularAccelerationGamma" />
<DataProperty IRI="#hasAngularAcceleration" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularVelocity" />
<DataProperty abbreviatedIRI="owl:topDataProperty" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularVelocityAlpha" />
<DataProperty IRI="#hasAngularVelocity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularVelocityBeta" />
<DataProperty IRI="#hasAngularVelocity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasAngularVelocityGamma" />
<DataProperty IRI="#hasAngularVelocity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasBottomRadius" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDataTable" />
<DataProperty IRI="#hasDataElementValue" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDataValue" />
<DataProperty IRI="#hasDataElementValue" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDepth" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
```

```
<SubDataPropertyOf>
<DataProperty IRI="#hasDescription" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDirectionX" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDirectionY" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasDirectionZ" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasGravityX" />
<DataProperty IRI="#hasGravity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasGravityY" />
<DataProperty IRI="#hasGravity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasGravityZ" />
<DataProperty IRI="#hasGravity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasHeight" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLength" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLocationX" />
<DataProperty IRI="#hasLocation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLocationY" />
<DataProperty IRI="#hasLocation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLocationZ" />
<DataProperty IRI="#hasLocation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedRotationX" />
<DataProperty IRI="#hasLockedRotation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedRotationY" />
<DataProperty IRI="#hasLockedRotation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedRotationZ" />
<DataProperty IRI="#hasLockedRotation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedTranslationX" />
<DataProperty IRI="#hasLockedTranslation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedTranslationY" />
<DataProperty IRI="#hasLockedTranslation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasLockedTranslationZ" />
<DataProperty IRI="#hasLockedTranslation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMajorRadius" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIxx" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIxy" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIxz" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIyy" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIyz" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMassInertiaIzz" />
<DataProperty IRI="#hasMassInertia" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMasterDirectionX" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMasterDirectionY" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMasterDirectionZ" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasMinorRadius" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationQ0" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationQ1" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationQ2" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationQ3" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationR1" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationR2" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasOrientationR3" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasRotationSequence" />
<DataProperty IRI="#hasOrientation" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
<DataProperty IRI="#hasDegreesOfFreedom" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasSlaveDirectionX" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasSlaveDirectionY" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
```

```
<SubDataPropertyOf>
<DataProperty IRI="#hasSlaveDirectionZ" />
<DataProperty IRI="#hasDirection" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasStartTime" />
<DataProperty IRI="#hasAnalysisParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasStopTime" />
<DataProperty IRI="#hasAnalysisParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasTimeStep" />
<DataProperty IRI="#hasAnalysisParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasTopRadius" />
<DataProperty IRI="#hasGeometryParameter" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
<DataProperty IRI="#hasDegreesOfFreedom" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasVelocity" />
<DataProperty abbreviatedIRI="owl:topDataProperty" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasVelocityX" />
<DataProperty IRI="#hasVelocity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasVelocityY" />
<DataProperty IRI="#hasVelocity" />
</SubDataPropertyOf>
<SubDataPropertyOf>
<DataProperty IRI="#hasVelocityZ" />
<DataProperty IRI="#hasVelocity" />
</SubDataPropertyOf>
<FunctionalDataProperty>
<DataProperty IRI="#hasAnalysisType" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasBottomRadius" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDataTable" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDataValue" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDepth" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDescription" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDirectionX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDirectionY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasDirectionZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasGearRatio" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasGravityX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasGravityY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasGravityZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasHeight" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLength" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLocationX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLocationY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLocationZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedRotationX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedRotationY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedRotationZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedTranslationX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedTranslationY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasLockedTranslationZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMajorRadius" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMass" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIxx" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIxy" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIxz" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIyy" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIyz" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMassInertiaIzz" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMasterAngle" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMasterDirectionX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMasterDirectionY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMasterDirectionZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasMinorRadius" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasOrientationQ1" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasOrientationQ2" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasOrientationQ3" />
</FunctionalDataProperty>
```

```
<FunctionalDataProperty>
<DataProperty IRI="#hasRotationalDegreesOfFreedom" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasSlaveDirectionX" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasSlaveDirectionY" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasSlaveDirectionZ" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasStartTime" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasStopTime" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasTimeStep" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasTopRadius" />
</FunctionalDataProperty>
<FunctionalDataProperty>
<DataProperty IRI="#hasTranslationalDegreesOfFreedom" />
</FunctionalDataProperty>
<DataPropertyDomain>
<DataProperty IRI="#hasAnalysisParameter" />
<Class IRI="#Analysis" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasDegreesOfFreedom" />
<Class IRI="#Constraint" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasGearRatio" />
<Class IRI="#GearJoint" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasGeometryParameter" />
<Class IRI="#Geometry" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasGravity" />
<Class IRI="#GroundBody" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasLockedRotation" />
<Class IRI="#GeneralJoint" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasLockedTranslation" />
<Class IRI="#GeneralJoint" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasMass" />
<Class IRI="#Body" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasMassInertia" />
<Class IRI="#RigidBody" />
</DataPropertyDomain>
<DataPropertyDomain>
<DataProperty IRI="#hasMasterAngle" />
<Class IRI="#UniversalJoint" />
</DataPropertyDomain>
<DLSafeRule>
<Body>
<ObjectPropertyAtom>
<ObjectProperty IRI="#hasMasterMember" />
<Variable IRI="urn:swrl#a" />
<Variable IRI="urn:swrl#b" />
</ObjectPropertyAtom>
<ObjectPropertyAtom>
<ObjectProperty IRI="#hasSlaveMember" />
<Variable IRI="urn:swrl#a" />
<Variable IRI="urn:swrl#c" />
</ObjectPropertyAtom>
</Body>
<Head>
<DifferentIndividualsAtom>
<Variable IRI="urn:swrl#b" />
<Variable IRI="urn:swrl#c" />
</DifferentIndividualsAtom>
</Head>
</DLSafeRule>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Analysis</IRI>
<Literal>Analysis</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#AuxiliaryComponent</IRI>
<Literal>AuxiliaryComponent</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#AuxiliaryCoordinateSystem</IRI>
<Literal>AuxiliaryCoordinateSystem</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#AuxiliaryLocation</IRI>
<Literal>AuxiliaryLocation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#BeamGeometry</IRI>
<Literal>BeamGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#BlockGeometry</IRI>
<Literal>BlockGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Body</IRI>
<Literal>Body</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Case</IRI>
<Literal>Case</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#ConeGeometry</IRI>
<Literal>ConeGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Constraint</IRI>
<Literal>Constraint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#CylindricalJoint</IRI>
<Literal>CylindricalJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#DataElement</IRI>
<Literal>DataElement</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#DataSpline</IRI>
<Literal>DataSpline</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#DataTable</IRI>
<Literal>DataTable</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#DataVariable</IRI>
```

```
<Literal>DataVariable</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#EllipseGeometry</IRI>
<Literal>EllipseGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#ExternalGeometry</IRI>
<Literal>ExternalGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#FieldForce</IRI>
<Literal>FieldForce</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#FixedJoint</IRI>
<Literal>FixedJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Force</IRI>
<Literal>Force</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#FunctionExpression</IRI>
<Literal>FunctionExpression</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#GearJoint</IRI>
<Literal>GearJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#GeneralForce</IRI>
<Literal>GeneralForce</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#GeneralJoint</IRI>
<Literal>GeneralJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Geometry</IRI>
<Literal>Geometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#GroundBody</IRI>
<Literal>Ground</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Model</IRI>
<Literal>Model</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#PointForce</IRI>
<Literal>PointForce</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#PointTorque</IRI>
<Literal>PointTorque</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#PrismaticJoint</IRI>
<Literal>PrismaticJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#Results</IRI>
<Literal>Results</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#RevoluteJoint</IRI>
<Literal>RevoluteJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#RigidBody</IRI>
<Literal>RigidBody</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#SphericalJoint</IRI>
<Literal>SphericalJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#SubModel</IRI>
<Literal>SubModel</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#UniversalJoint</IRI>
<Literal>UniversalJoint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#VectorForce</IRI>
<Literal>VectorForce</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#VectorTorque</IRI>
<Literal>VectorTorque</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAcceleration</IRI>
<Literal>hasAcceleration</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAccelerationX</IRI>
<Literal>hasAccelerationX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAccelerationY</IRI>
<Literal>hasAccelerationY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAccelerationZ</IRI>
<Literal>hasAccelerationZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAnalysis</IRI>
<Literal>hasAnalysis</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAnalysisParameter</IRI>
<Literal>hasAnalysisParameter</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAnalysisType</IRI>
<Literal>hasAnalysisType</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularAcceleration</IRI>
<Literal>hasAngularAcceleration</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularAccelerationAlpha</IRI>
```

```
<Literal>hasAngularAccelerationAlpha</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularAccelerationBeta</IRI>
<Literal>hasAngularAccelerationBeta</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularAccelerationGamma</IRI>
<Literal>hasAngularAccelerationGamma</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularVelocity</IRI>
<Literal>hasAngularVelocity</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularVelocityAlpha</IRI>
<Literal>hasAngularVelocityAlpha</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularVelocityBeta</IRI>
<Literal>hasAngularVelocityBeta</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAngularVelocityGamma</IRI>
<Literal>hasAngularVelocityGamma</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasAuxiliaryComponent</IRI>
<Literal>hasAuxiliaryComponent</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasBody</IRI>
<Literal>hasBody</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasBottomRadius</IRI>
<Literal>hasBottomRadius</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasCaseObject</IRI>
<Literal>hasCaseObject</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasContraint</IRI>
<Literal>hasContraint</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDataElement</IRI>
<Literal>hasDataElement</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDataElementValue</IRI>
<Literal>hasDataElementValue</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDataTable</IRI>
<Literal>hasDataTable</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDataValue</IRI>
<Literal>hasDataValue</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDegreesOfFreedom</IRI>
<Literal>hasDegreesOfFreedom</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDepth</IRI>
<Literal>hasDepth</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDescription</IRI>
<Literal>hasDescription</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDirection</IRI>
<Literal>hasDirection</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDirectionX</IRI>
<Literal>hasDirectionX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDirectionY</IRI>
<Literal>hasDirectionY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasDirectionZ</IRI>
<Literal>hasDirectionZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasForce</IRI>
<Literal>hasForce</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasFunction</IRI>
<Literal>hasFunction</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGearRatio</IRI>
<Literal>hasGearRatio</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGeometry</IRI>
<Literal>hasGeometry</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGeometryParameter</IRI>
<Literal>hasGeometryParameter</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGravity</IRI>
<Literal>hasGravity</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGravityX</IRI>
<Literal>hasGravityX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGravityY</IRI>
<Literal>hasGravityY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGravityZ</IRI>
<Literal>hasGravityZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasGroundBody</IRI>
```

```
<Literal>hasGround</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasHeight</IRI>
<Literal>hasHeight</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLength</IRI>
<Literal>hasLength</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLocation</IRI>
<Literal>hasLocation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLocationX</IRI>
<Literal>hasLocationX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLocationY</IRI>
<Literal>hasLocationY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLocationZ</IRI>
<Literal>hasLocationZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedRotation</IRI>
<Literal>hasLockedRotation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedRotationX</IRI>
<Literal>hasLockedRotationX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedRotationY</IRI>
<Literal>hasLockedRotationY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedRotationZ</IRI>
<Literal>hasLockedRotationZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedTranslation</IRI>
<Literal>hasLockedTranslation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedTranslationX</IRI>
<Literal>hasLockedTranslationX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedTranslationY</IRI>
<Literal>hasLockedTranslationY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasLockedTranslationZ</IRI>
<Literal>hasLockedTranslationZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMajorRadius</IRI>
<Literal>hasMajorRadius</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMass</IRI>
```

```
<Literal>hasMass</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertia</IRI>
<Literal>hasMassInertia</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIxx</IRI>
<Literal>hasMassInertiaIxx</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIxy</IRI>
<Literal>hasMassInertiaIxy</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIxz</IRI>
<Literal>hasMassInertiaIxz</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIyy</IRI>
<Literal>hasMassInertiaIyy</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIyz</IRI>
<Literal>hasMassInertiaIyz</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMassInertiaIzz</IRI>
<Literal>hasMassInertiaIzz</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMasterAngle</IRI>
<Literal>hasMasterAngle</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMasterDirectionX</IRI>
<Literal>hasMasterDirectionX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMasterDirectionY</IRI>
<Literal>hasMasterDirectionY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMasterDirectionZ</IRI>
<Literal>hasMasterDirectionZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMasterMember</IRI>
<Literal>hasMasterMember</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasMinorRadius</IRI>
<Literal>hasMinorRadius</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasModel</IRI>
<Literal>hasModel</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasModelObject</IRI>
<Literal>hasModelObject</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientation</IRI>
```

```
<Literal>hasOrientation</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationQ0</IRI>
<Literal>hasOrientationQ0</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationQ1</IRI>
<Literal>hasOrientationQ1</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationQ2</IRI>
<Literal>hasOrientationQ2</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationQ3</IRI>
<Literal>hasOrientationQ3</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationR1</IRI>
<Literal>hasOrientationR1</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationR2</IRI>
<Literal>hasOrientationR2</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasOrientationR3</IRI>
<Literal>hasOrientationR3</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasReferenceMember</IRI>
<Literal>hasReferenceMember</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasResults</IRI>
<Literal>hasResults</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasRotationSequence</IRI>
<Literal>hasRotationSequence</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasRotationalDegreesOfFreedom</IRI>
<Literal>hasRotationalDegreesOfFreedom</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasSlaveDirectionX</IRI>
<Literal>hasSlaveDirectionX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasSlaveDirectionY</IRI>
<Literal>hasSlaveDirectionY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasSlaveDirectionZ</IRI>
<Literal>hasSlaveDirectionZ</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasSlaveMember</IRI>
<Literal>hasSlaveMember</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasStartTime</IRI>
<Literal>hasStartTime</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasStopTime</IRI>
<Literal>hasStopTime</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasSubModel</IRI>
<Literal>hasSubModel</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasTimeStep</IRI>
<Literal>hasTimeStep</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasTopRadius</IRI>
<Literal>hasTopRadius</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasTranslationalDegreesOfFreedom</IRI>
<Literal>hasTranslationalDegreesOfFreedom</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasVelocity</IRI>
<Literal>hasVelocity</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasVelocityX</IRI>
<Literal>hasVelocityX</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasVelocityY</IRI>
<Literal>hasVelocityY</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label" />
<IRI>#hasVelocityZ</IRI>
<Literal>hasVelocityZ</Literal>
</AnnotationAssertion>
</Ontology>
<!-- Generated by the OWL API (version 3.0.0.1469)
     http://owlapi.sourceforge.net -->
```

A.3 The doublePendulum Model

The doublePendulum model in the XML Presentation syntax (OWL/XML) [84] is listed below. This listing contains all the definitions for the semantic model. The model uses the Mbs ontology listed in section A.2

```
<?xml version="1.0"?>

<!DOCTYPE Ontology [
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<Ontology xmlns="http://www.w3.org/2002/07/owl#"
   xml:base="http://www.semanticweb.org/ontologies/2010/0/mbstestmodel.owl"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   ontologyIRI="http://www.semanticweb.org/ontologies/2010/0/mbstestmodel.owl">
  <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
  <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
  <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
  <Import>http://www.simantics.org/ontologies/mbs.owl</Import>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Analysis"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Case"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Ground"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Model"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Results"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RevoluteJoint"/>
  </Declaration>
  <Declaration>
    <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RigidBody"/>
  </Declaration>
  <Declaration>
    <Class abbreviatedIRI="owl:Thing"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasAnalysis"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasBody"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasContraint"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGround"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMasterMember"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasModel"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasResults"/>
  </Declaration>
  <Declaration>
    <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasSlaveMember"/>
  </Declaration>
  <Declaration>
    <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionX"/>
```

```
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionY"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionZ"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityX"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityY"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityZ"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationX"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationY"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationZ"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMass"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxx"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxy"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxz"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyy"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyz"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIzz"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR1"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR2"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR3"/>
</Declaration>
<Declaration>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasRotationSequence"/>
</Declaration>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Model"/>
  <NamedIndividual IRI="#doublePendulum"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#doublePendulum"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Analysis"/>
  <NamedIndividual IRI="#exampleAnalysis"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#exampleAnalysis"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Case"/>
  <NamedIndividual IRI="#exampleCase"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
```

```
  <NamedIndividual IRI="#exampleCase"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Results"/>
  <NamedIndividual IRI="#exampleResults"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#exampleResults"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#Ground"/>
  <NamedIndividual IRI="#ground"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#ground"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RevoluteJoint"/>
  <NamedIndividual IRI="#hinge_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#hinge_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RevoluteJoint"/>
  <NamedIndividual IRI="#hinge_2"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#hinge_2"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RigidBody"/>
  <NamedIndividual IRI="#link_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#link_1"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="http://www.simantics.org/ontologies/mbs.owl#RigidBody"/>
  <NamedIndividual IRI="#link_2"/>
</ClassAssertion>
<ClassAssertion>
  <Class abbreviatedIRI="owl:Thing"/>
  <NamedIndividual IRI="#link_2"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasBody"/>
  <NamedIndividual IRI="#doublePendulum"/>
  <NamedIndividual IRI="#link_2"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasBody"/>
  <NamedIndividual IRI="#doublePendulum"/>
  <NamedIndividual IRI="#link_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasContraint"/>
  <NamedIndividual IRI="#doublePendulum"/>
  <NamedIndividual IRI="#hinge_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasContraint"/>
  <NamedIndividual IRI="#doublePendulum"/>
  <NamedIndividual IRI="#hinge_2"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGround"/>
  <NamedIndividual IRI="#doublePendulum"/>
  <NamedIndividual IRI="#ground"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasAnalysis"/>
  <NamedIndividual IRI="#exampleCase"/>
  <NamedIndividual IRI="#exampleAnalysis"/>
</ObjectPropertyAssertion>
```

```
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasModel"/>
  <NamedIndividual IRI="#exampleCase"/>
  <NamedIndividual IRI="#doublePendulum"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasResults"/>
  <NamedIndividual IRI="#exampleCase"/>
  <NamedIndividual IRI="#exampleResults"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMasterMember"/>
  <NamedIndividual IRI="#hinge_1"/>
  <NamedIndividual IRI="#ground"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasSlaveMember"/>
  <NamedIndividual IRI="#hinge_1"/>
  <NamedIndividual IRI="#link_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMasterMember"/>
  <NamedIndividual IRI="#hinge_2"/>
  <NamedIndividual IRI="#link_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasSlaveMember"/>
  <NamedIndividual IRI="#hinge_2"/>
  <NamedIndividual IRI="#link_2"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityX"/>
  <NamedIndividual IRI="#ground"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityY"/>
  <NamedIndividual IRI="#ground"/>
  <Literal datatypeIRI="&xsd;double">-9.80665</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasGravityZ"/>
  <NamedIndividual IRI="#ground"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionX"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionY"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionZ"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">1.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationX"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationY"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationZ"/>
  <NamedIndividual IRI="#hinge_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionX"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
```

```
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionY"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasDirectionZ"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">1.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationX"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">0.5</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationY"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationZ"/>
  <NamedIndividual IRI="#hinge_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationX"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.25</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationY"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationZ"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMass"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">1.346567</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxx"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">5.557941e-04</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxy"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxz"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyy"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">3.288061e-02</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyz"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIzz"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">3.287534e-02</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR1"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
```

```
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR2"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR3"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasRotationSequence"/>
  <NamedIndividual IRI="#link_1"/>
  <Literal datatypeIRI="&xsd;string">zxz</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationX"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.75</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationY"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasLocationZ"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMass"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">1.346567</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxx"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">5.557941e-04</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxy"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIxz"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyy"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">3.288061e-02</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIyz"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasMassInertiaIzz"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">3.287534e-02</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR1"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR2"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasOrientationR3"/>
  <NamedIndividual IRI="#link_2"/>
  <Literal datatypeIRI="&xsd;double">0.0</Literal>
</DataPropertyAssertion>
```

```
  <DataPropertyAssertion>
    <DataProperty IRI="http://www.simantics.org/ontologies/mbs.owl#hasRotationSequence"/>
    <NamedIndividual IRI="#link_2"/>
    <Literal datatypeIRI="&xsd;string">zxz</Literal>
  </DataPropertyAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#doublePendulum</IRI>
    <Literal>doublePendulum</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#exampleAnalysis</IRI>
    <Literal>exampleAnalysis</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#exampleCase</IRI>
    <Literal>exampleCase</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#exampleResults</IRI>
    <Literal>exampleResults</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#ground</IRI>
    <Literal>ground</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#hinge_1</IRI>
    <Literal>hinge_1</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#hinge_2</IRI>
    <Literal>hinge_2</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#link_1</IRI>
    <Literal>link_1</Literal>
  </AnnotationAssertion>
  <AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <IRI>#link_2</IRI>
    <Literal>link_2</Literal>
  </AnnotationAssertion>
</Ontology>

<!-- Generated by the OWL API (version 3.0.0.1469) http://owlapi.sourceforge.net -->
```

Appendix B

Modelica MultiBody OWL Ontology

The Modelica MultiBody library is represented below as an OWL ontology. The hierarchy and naming conventions are the same as in the Modelica Standard Library [85].

The Modelica MultiBody library contains the following main modelling classes:

- *World*: world coordinate system, gravity field, and default animation definition;
- *Forces*: components that exert forces and/or torques between frames;
- *Interfaces*: connectors and partial models for 3-dimensional mechanical components;
- *Joints*: components that constrain the motion between two frames;
- *Parts*: rigid body components, such as bodies with mass and inertia, and massless rods;
- *Sensors*: sensors to measure variables;
- *Types*: constants and types with choices; and
- *Visualizers*: 3-dimensional visual objects used for animation.

In addition to the above, the library contains a set for functions in the *Frames* class to transform rotational frame quantities, and constant and type definitions in the *Types* class. The structure of the Modelica MultiBody library is presented in Table B.1.

Table B.1: The class and subclass structure of the Modelica MultiBody library.

- *Forces*
  - *WorldForce*
  - *WorldTorque*
  - *WorldForceAndTorque*
  - *Force*
  - *Torque*
  - *ForceAndTorque*
  - *LineForceWithMass*
  - *LineForceWithTwoMasses*
  - *Spring*
  - *Damper*
  - *SpringDamperParallel*
  - *SpringDamperSeries*
- *Interfaces*
  - *Frame*
  - *Frame_a*
  - *Frame_b*
  - *Frame_resolve*
  - *FlangeWithBearing*
  - *FlangeWithBearingAdaptor*
  - *PartialTwoFrames*
  - *PartialTwoFramesDoubleSize*
  - *PartialOneFrame_a*
  - *PartialOneFrame_b*
  - *PartialElementaryJoint*
  - *PartialForce*
  - *PartialLineForce*
  - *PartialAbsoluteSensor*
  - *PartialRelativeSensor*
  - *PartialVisualizer*
  - *ZeroPosition*
- *Joints*
  - *Prismatic*
  - *Revolute*
  - *RevolutePlanarLoopConstraint*
  - *Cylindrical*
  - *Universal*
  - *Planar*
  - *Spherical*
  - *FreeMotion*
  - *SphericalSpherical*
  - *UniversalSpherical*
  - *GearConstraint*
  - *RollingWheel*
  - *RollingWheelSet*
  - *Assemblies*
- *Parts*
  - *Fixed*
  - *FixedTranslation*
  - *FixedRotation*
  - *Body*
  - *BodyShape*
  - *BodyBox*
  - *BodyCylinder*
  - *PointMass*
  - *Mounting1D*
  - *Rotor1D*
  - *BevelGear1D*
  - *RollingWheel*
  - *RollingWheelSet*
- *Sensors*
  - *AbsoluteSensor*
  - *RelativeSensor*
  - *AbsolutePosition*
  - *AbsoluteVelocity*
  - *AbsoluteAngles*
  - *AbsoluteAngularVelocity*
  - *RelativePosition*
  - *RelativeVelocity*
  - *RelativeAngles*
  - *RelativeAngularVelocity*
  - *Distance*
  - *CutForce*
  - *CutTorque*
  - *CutForceAndTorque*
  - *Power*
  - *TansformAbsoluteVector*
  - *TansformRelativeVector*
- *Visualizers*
  - *FixedShape*
  - *FixedShape2*
  - *FixedFrame*
  - *FixedArrow*
  - *SignalArrow*
  - *Ground*
  - *Advanced*

Series title, number and
report code of publication

VTT Publications 766
VTT-PUBS-766

Author(s)
Juha Kortelainen

Title

Semantic Data Model for Multibody System Modelling

Abstract

Multibody system simulation, as one area of system simulation, represents a considerable improvement in predicting machine dynamic performance compared to previous methods that are often based on analytic equations or empirical testing. With multibody system simulation, it is fast and efficient to study the effects of design variables on the dynamic behaviour of the mechanism compared to the experimental approach. Accordingly, the use of multibody system simulation in the design process can decrease the need for physical prototypes, thus accelerating the overall design process and saving resources.

In the product development sense, the interaction between computational tools can be cumbersome. For this reason, multibody system simulation is often omitted in terms of the main stream of product development. Due to the increase of computational resources, the trend is towards extensive usage of simulation, including multibody system simulation, in the product development.

The research emphasis in the field of multibody system dynamics has been on the areas of improved multibody system formulations, such as recursive methods, representation of flexible structures, application of multibody simulation in new areas such as biomechanics, and including multibody simulation into multitechnical and multiphysics simulation. The research on modelling data management and integration approaches concerning tools applied in product design and development has not been in the main stream.

The growth of the World Wide Web and the evolution of Web technologies have multiplied the amount of data and information available on the Internet. In this expansion of information, it has become cumbersome to find and distil the useful information from the data mass. In order to utilise the information available on the Internet and to sort meaningful information out of the data mass, the World Wide Web Consortium began a development project for the next generation Web, the Semantic Web. In this development effort, methods and technologies based on semantic data representation are developed and utilised. These methods and technologies are general enough to be applied to other application areas as well.

This work concentrates on the modelling data management of multibody system simulation within a simulation-based product process. The main objectives of this work can be summarised as follows: introduce a procedure for managing multibody system modelling data using semantic data model and ontology-based modelling approach, demonstrate that the semantic data model allows application-based reasoning on the model data, and show that ontology-based modelling is able to capture domain knowledge by using semantic data and constraint- and rule-based reasoning.

In this work, the semantic data representation approach is used for describing modelling data of a multibody system. This is accomplished by developing a multibody system modelling ontology, i.e. a semantic data model for multibody system model description, and then applying this ontology to model an example multibody system.

ISBN
978-951-38-7742-2 (soft back ed.)
978-951-38-7743-9 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN
VTT Publications
1235-0621 (soft back ed.)
1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)

Project number

Date
Espoo, June 2011

Language
English, Finnish abstr.

Pages
119 p. + app. 34 p.

Name of project

Commissioned by

Keywords
multibody application, data management,
semantic data model, reasoning

Publisher
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
Phone internat. +358 20 722 4520
Fax +358 20 722 4374

Julkaisun sarja, numero ja raporttikoodi

VTT Publications 766
VTT-PUBS-766

Tekijä(t)
Juha Kortelainen

Nimike
Semanttinen tietomalli monikappaledynamiikan mallitiedon hallinnassa

Tiivistelmä
Monikappaledynamiikka, yhtenä systeemisimuloinnin erikoisalueena, edustaa merkittävää parannusta koneiden ja mekaanisten järjestelmien suorituskyvyn arvioinnissa verrattuna muihin menetelmiin, jotka usein perustuvat joko analyyttisiin laskelmiin tai kokeellisiin menetelmiin. Verrattuna kokeellisiin menetelmiin monikappaledynaamiikkaa soveltamalla on nopeaa ja tehokasta selvittää eri muuttujien vaikutus mekanismin dynaamisiin ominaisuuksiin. Täten monikappaledynamiikan soveltaminen suunnitteluprosessissa voi vähentää tarvetta todellisten prototyyppien käytölle ja siten nopeuttaa suunnitteluprosessia kokonaisuutena sekä säästää resursseja.

Laskennallisten ohjelmistojen yhteiskäyttö voi olla hankalaa tuotekehityksen näkökulmasta. Tästä syystä tuotesuunnittelun valtavirrassa usein vältetään monikappaledynamiikan käyttöä. Laskennallisten resurssien kasvusta johtuen suuntaus on kuitenkin kohti enenevää simuloinnin käyttöä tuotekehityksessä, mukaan lukien monikappaledynamiikka.

Monikappaledynamiikan tutkimus on painottunut monikappaledynamiikan formuloinnin, kuten esimerkiksi rekursiivisten menetelmien, kehittämiseen, rakenteellisen joustavuuden huomioimiseen, monikappaledynamiikan soveltamiseen uusilla tutkimusalueilla, kuten biomekaniikassa, sekä monikappaledynamiikan soveltamiseen moniteknisessä ja monifysikaalisessa simuloinnissa. Monikappaledynamiikan tiedonhallinta sekä tuotekehitykseen käytettyjen laskennallisten ohjelmistojen ja menetelmien integrointi eivät ole kuuluneet keskeisiin tutkimusalueisiin monikappaledynamiikassa.

Internetin kasvu sekä Web-teknologioiden kehitys ovat moninkertaistaneet Internetissä tarjolla olevan tiedon määrän. Tämän tiedon määrän kasvun myötä oikean tiedon löytämisestä Internetin tietomassasta on tullut hankalaa. Internetissä tarjolla olevan tiedon paremman hyödynnettävyyden mahdollistamiseksi World Wide Web Consortium on käynnistänyt kehityshankkeen nimeltä Semanttinen Web, jossa kehitetään seuraavan sukupolven verkkoa. Kehityshankkeessa kehitetään ja sovelletaan tiedon semanttiseen kuvaukseen perustuvia menetelmiä ja tekniikoita, jotka ovat riittävän yleisiä sovellettaviksi myös muille alueille, kuten laskennallisen tiedon hallintaan.

Tässä työssä keskitytään tuotekehitykseen liittyvän monikappaledynamiikan mallitiedonhallintaan. Työn tavoitteet voidaan tiivistää seuraavalla tavalla: esitellä periaate monikappaledynamiikan mallitiedon hallintaan käyttäen semanttista tietomallia sekä ontologiapohjaista mallinnusmenetelmää; osoittaa, että semanttisen tietomallin soveltaminen mahdollistaa sovelluspohjaisen päättelyn monikappaledynamiikan mallitiedosta; sekä osoittaa, että ontologiapohjainen mallintaminen mahdollistaa myös tietämyksen tallentamisen yhdessä mallitiedon kanssa soveltaen semanttista tietomallia sekä rajoite- ja sääntöpohjaista päättelyä. Tämä osoitetaan kehittämällä mallinnusontologia monikappaledynamiikan mallitiedon hallintaan ja soveltamalla tätä ontologiaa monikappaledynaamisen esimerkkitapauksen mallin kuvaukseen.

ISBN
978-951-38-7742-2 (soft back ed.)
978-951-38-7743-9 (URL: http://www.vtt.fi/publications/index.jsp)

Avainnimeke ja ISSN
VTT Publications
1235-0621 (soft back ed.)
1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)

Projektinumero

Julkaisuaika
Espoo, kesäkuu 2011

Kieli
Englanti, suom. tiiv.

Sivuja
119 s. + liitt. 34 s.

Projektin nimi

Toimeksiantaja(t)

Avainsanat
multibody application, data management, semantic data model, reasoning

Julkaisija
VTT
PL 1000, 02044 VTT
Puh. 020 722 4520
Faksi 020 722 4374

Technological evolution has rapidly transformed the world. We are now heading into a new era of information and knowledge. Humankind is producing data and information at a continuously increasing speed and, instead of finding information in general, the challenge is how to distil the desired information and knowledge out of the data masses. The development of information technology, including progress in algorithms, computational methods and computer hardware, has provided us with new tools to search and analyse the data masses. Data semantics, i.e. managing the meaning of the data, is one of the most promising methods for answering this challenge.

From the standpoint of product development, the interaction between computational tools can be cumbersome. For this reason, multibody system simulation is often excluded from the mainstream of product development. But due to the increase in computational resources, the trend is towards extensive usage of simulation, including multibody system simulation, in product development. This increases the pressure to improve data management in multibody system simulation as well.

This doctoral thesis focuses on combining new methods for data and knowledge management and system simulation. The former provides solutions to the challenges rising from the latter in order to ensure further improvements in the overall efficiency of applying computational methods for purposes such as product development. The main contributions of this work can be summarised as follows: it introduces a procedure for managing multibody system modelling data using a semantic data model and ontology-based modelling approach, demonstrates that the semantic data model allows application-based reasoning on the model data, and shows that ontology-based modelling is able to capture domain knowledge by using semantic data and constraint- and rule-based reasoning.

ISBN 978-951-38-7742-2 (soft back ed.)
ISSN 1235-0621 (soft back ed.)

ISBN 978-951-38-7743-9 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)