Reference NEESgrid Data Model for Shake Table Experiment

Jun Peng [1], Kincho H. Law [1], and Gokhan Pekcan [2]

[1] Stanford University, Stanford, CA 94305
[2] University of Nevada, Reno, NV 89557

1 Introduction

The primary goal of NEESgrid data/metadata effort is to work collaboratively with NEESgrid team and the NEES community and to help define data requirements and needs for the George Brown Jr. Network for Earthquake Engineering Simulation instigated by the National Science Foundation. The NEESgrid promotes NEES as a distributed virtual laboratory for earthquake engineering research and simulation. The "collaboratory" will allow researchers gain remote, shared access to experimental equipment and data. This report serves to outline current tasks and approaches to define data models for supporting the activities involved in earthquake engineering simulations.

In order to facilitate collaboration within the NEES framework, one of the key services that NEESgrid needs to support is with respect to the ***data*** and ***metadata***, for earthquake engineering simulations. Following the NEESgrid Data and Metadata Advisory Group meeting that was held at the Argonne National Laboratory on November 5, 2003, a NEES Data/MetaData Task Group (DMD-TG) was formed to actively define and coordinate the data/metadata development tasks. Complete list of contributing members involved in this development is given in the Acknowledgements section of this report. The goal of DMD-TG is to develop and deploy end-to-end solutions that put tools and capabilities into the hands of the NEES community. The high level outline of this effort is to develop models for data representation based on existing data efforts combined with new information. For this purpose sample data sets are being developed that demonstrate the features of the data models along with scenarios for the use of the data and models. Ultimately, associated specifications for the tools necessary to support entering, importing, storing, searching, and extracting data from the repository are being established.

One major task of data/metadata effort is to develop a reference data model for supporting the major activities involved in earthquake engineering simulations. There are many existing data modeling techniques and tools that are available to help design and structure the data. A brief review of these relevant techniques and approaches has been reported earlier [1]. A **data model** is defined as the grammar, vocabulary and content that represent all types of "information" stored in one format or another in a "system". The **grammar** defines the relationships among **elements** in the system; the **vocabulary** defines the terminology used to describe these element; **content** defines what is to be included in the system. A data model is in essence a representation of the data and their relationship and provides a conceptual or implementation view of the data. Ideally, the data model should be independent of hardware/software platforms so that its implementation can be universal.

Within the scope of the NEESgrid data/metadata effort, ***data*** is defined as *all* of the project related information and encompasses ***observational*** (or *acquired*) *data* recorded during experiments by means of sensors, cameras and the like; ***computational*** (or *generated*) *data* for and as a result of simulations, post-processing, etc.; ***literature*** in the form of reports, journal papers, drawings, etc. It is noted that the utilities and tools to facilitate the processing, visualization, interpretation and dissemination of data are also included within the scope. Associated descriptive and related data, i.e. ***metadata***, are expected to be generated and published in the prescribed (by NEESgrid) formats and language (e.g. XML, RDF, NEESML, etc.). It is expected that a set of functional system-wide services for storage, retrieval, and management of data and metadata associated with a project will be available as part of the NEESgrid infrastructure. These services will be based on specialized *data models* with only limited *content* populated by *elements* that are most critical to (1) the execution of a project, i.e. conduct and control of experiments and simulations; (2) the equipment, collection of sensor and video/image data, visualization; (3) the storage, retrieval and management functions. However, it must be noted that the so-called limitations on the content and elements will not prevent future extensions of the data models and the integration of new project related elements in the NEESgrid infrastructure.

Based on the experience gained from the review and the suggestions/feedback from NEESgrid team and the NEES community, a reference data model has been developed. The preliminary design of the reference data model is presented in the following sections. A brief summary is also presented that discusses the approach currently undertaken to develop a project data model for NEES experimentations. Although the reference data model is intended to focus on the data requirements for shake table experiments, large portion of the model should be of sufficient generality to be useful for other types of experiments, such as centrifuge, Tsunami, or even field tests. In fact, the upcoming version of the data model will include details for the applications in centrifuge and geotechnical areas, currently being developed by the University of Southern California team.

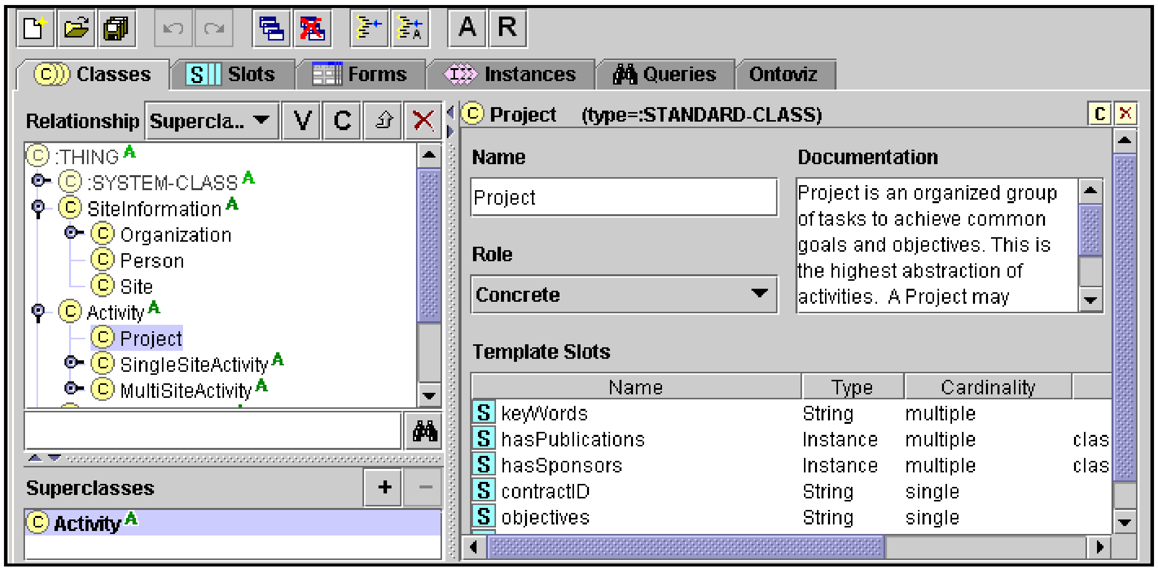
2 Data Modeling Tool and Approach

2.1 Protégé-2000 – Data Modeling Tool

There are many data modeling or software design tools that can be used to facilitate the design of a data model for specific application. In our work, we select Protégé-2000, which is an open-source software package designed to help developing knowledge-based systems [2]. Protégé-2000 (http://protege.stanford.edu) is a useful tool to build ontology for knowledge-based systems. Ontology represents explicit formal specifications of the terms in the domain and relations among them [3].

As an open source software, Protégé-2000 has attracted a wide variety of plug-ins from around the world to enhance its capabilities. Some of these software plug-ins allow a model developed in Protégé-2000 to be exported in many standard formats, including UML (Unified Modeling Language [4]), OWL (Web Ontology Language, http://www.w3.org/2001/sw/WebOnt/), XML Schema (http://www.w3.org/XML/Schema), and RDF (Resource Description Framework, http://www.w3.org/RDF/).

In Protégé-2000, a graphical user interface (GUI) is provided to facilitate ontology development. The interface enables the modeling of an ontology of classes to describe a particular subject with a set of concepts and their relationships. The interface also allows direct entering of specific instances of data and the creation of a knowledge base. Figure 1 shows an example of the GUI, with classes view shown in the left window, and detailed attributes view of a class (Project) shown in the lower right window.



:THING
:SYSTEM-CLASS
SiteInformation
Organization
Person
Site
Activity
Project
SingleSiteActivity
MultiSiteActivity

Superclasses: Activity

Project (type=:STANDARD-CLASS)

Name: Project

Documentation: Project is an organized group of tasks to achieve common goals and objectives. This is the highest abstraction of activities. A Project may

Role: Concrete

Template Slots

Name	Type	Cardinality	
keyWords	String	multiple	
hasPublications	Instance	multiple	clas
hasSponsors	Instance	multiple	clas
contractID	String	single	
objectives	String	single	

Figure 1 – Protégé-2000 Interface

2.2 Object-Oriented Data Modeling

Object-oriented data modeling approach is employed in the development of NEESgrid data model. In an object-oriented data model, information is modeled as **objects**, which can be any sorts of (real or abstract) entities [5]. The general representation of certain type of objects is called a **class**, which represents explicit description of concepts in a domain. The creation of an object of a certain class is called **instantiation**. The relationship between an object and a class can be viewed analogically in a procedural language in that a variable being a particular instance of a pre-defined type such as an integer. For example, ***Project*** is modeled as a class, and a MOST experiment [6] is an object instance of the ***Project*** class.

An object encapsulates certain related data as **slots**, which are also called **attributes** or **properties**. Slots can have different **facets** describing the value type, allowed values, the number of the values (**cardinality**), and other features of the values the slot can take. A value-type facet describes what types of values can fill in the slot. Some common value types are string, number, Boolean, enumerator, and object instance. Allowed objects and the variety of values (e.g. minimum number, maximum number) of a slot are often referred to as a **range** of a slot. Slot cardinality defines how many values a slot can have, such as single (at most one value) or multiple (more than one value). For example, "minute" is a slot of class ***Time***. The cardinality of minute is single, the data type of minute is integer, and the range is from 0 to 59.

An object connects with other related objects via some relationships. The relationship types commonly used include *classification, association, aggregation* and *generalization*. These relationships types may in turn impose certain "object-oriented" features and integrity constraints to help maintain consistency and correctness of the data in the model. One important feature of object-oriented modeling is the concept of class hierarchies, with slots of a **superclass** being inherited by its **subclasses**. This **inheritance** feature allows us to define the common slots used by several classes at the highest possible level in the hierarchy, which avoids the duplication of slots at the lower levels. A class can have subclasses that represent concepts that are more specific than the superclass. For example, we can divide the class ***Activity*** into Project, SingleSiteActivity, and MultiSiteActivity. The class ***SingleSiteActivity*** in turn can be divided into Task, EventGroup, and Event. The common slots for all these ***Activity*** classes are name, description, start Time and end Time.

In object-oriented data models, class can be **abstract** or **concrete** [3]. A concrete (or physical) class can have direct instances, as in the case that a MOST experiment is an instance of the class ***Project***. On the other hand, abstract class cannot have any direct instances. For example, the ***Activity*** class is defined as the general abstraction of action or process, and thus a direct instance cannot be created. In Protégé-2000, an "A" icon next to the class name indicates that the class is abstract, as shown in the left window of Figure 1.

3 Description of the Reference Data Model

As depicted in Figure 2, the NEESgrid data/metadata task group is working towards producing end-to-end solutions that integrate site specifications database, project level model, domain specific data models, and common elements. To capture all these data, the reference data model is designed to include six base classes, namely *SiteInformation, Activity, Apparatus, ApparatusSetup, DataElement,* and *ComplexDataType*. The high-level class diagram of the reference data model is presented in Figure 3, which shows the association relationship among classes. (The *ComplexDataType* class, which is employed to support other base classes, is not shown in the figure.) The association relationship exists between classes when an object of one class *knows/contains* an object of another class. For example, a ***Project*** object knows about its ***Task***s objects, a ***Project*** also contains ***Organization***s, ***Site***s, and ***RolePerson***s. RolePerson is in turn defined as the combination of a Person and his/her role in a Project. The arrow in Figure 3 denotes the direction of the relationship *contains*; i.e., A $\rightarrow$ B indicates that class A *contains* class B. In the following, the six base classes are briefly described.

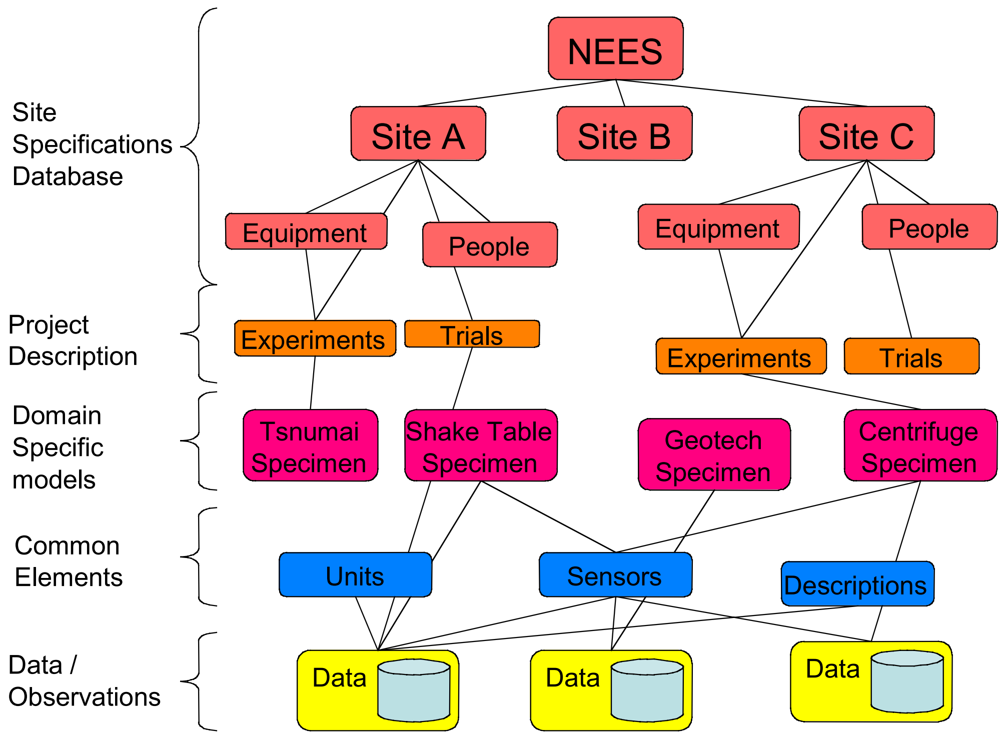


Figure 2 – Overall Data Model for NEESgrid (Courtesy of Chuck Severance)

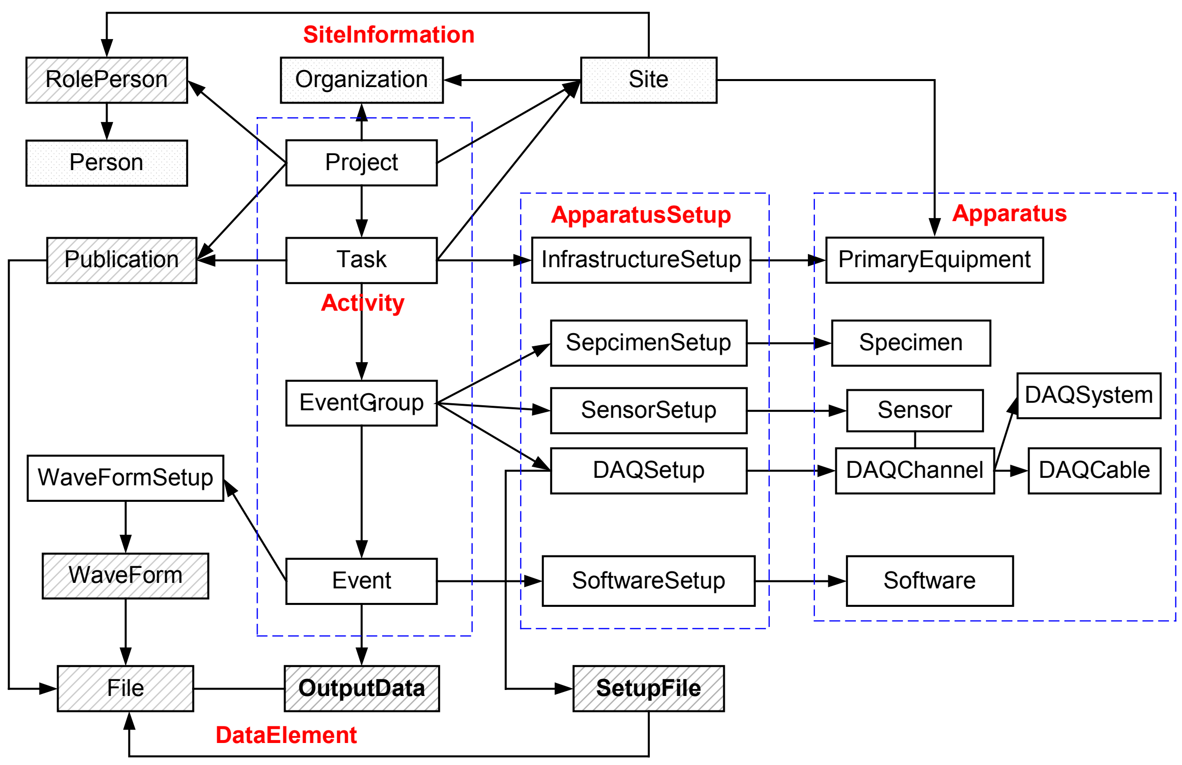


Figure 3 – High-level Class Diagram of the Reference Data Model

3.1 SiteInformation

A typical experiment site is hosted by certain organizations, and the site has personnel playing different roles, facilities, equipments and other information. In the reference data model, Site is modeled as the aggregation of other component classes, such as RolePerson, Organization, PrimaryEquipment, and SecondaryEquipment. The relationship of Site with other related classes is shown in Figure 4. This group of classes is intended to be associated with the site specifications database [7] that is currently under development by the NEES community.

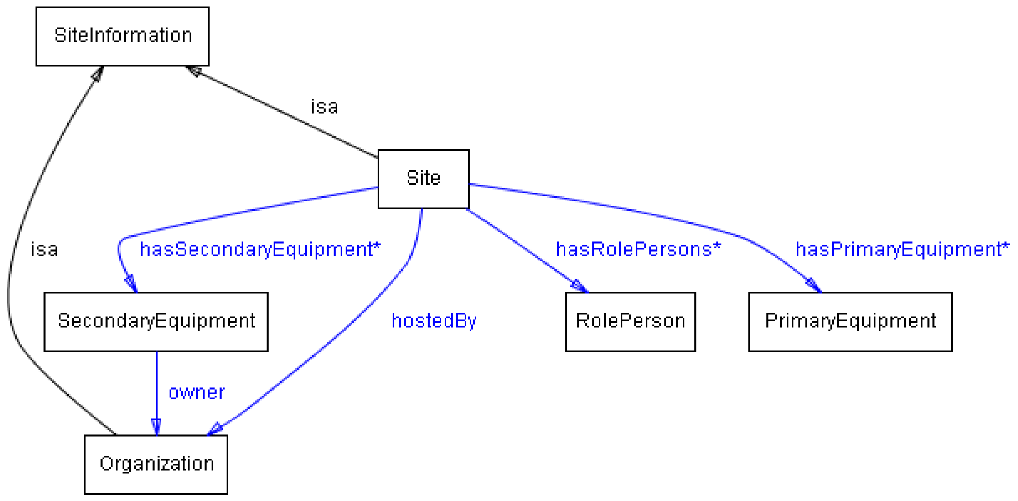


Figure 4 – Relationship of Site with Other Classes (generated by Protégé)

3.2 Activity

The Activity class is designed to support project level modeling. As shown in Figure 3, the Activity class has four hierarchical layers.

- A ***Project*** is a collection (organized group) of tasks designed to achieve specific goals and objectives of a project. A Project can be sponsored by one or more funding sources. A Project includes one or more related Tasks. For example, the CUREE-Caltech Woodframe project had many tasks/activities to study the performance of woodframe structures, with the objective to reduce earthquake losses to woodframe construction (http://www.curee.org/projects/woodframe/index.html).

- A ***Task*** belongs to a particular Project and contains one or more EventGroups. Each Task typically serves a specific role in a Project. In case of an experiment, each Task has a distinct InfrastructureSetup; any changes to the InfrastructureSetup would initiate a new Task. For example, Task 1.1.1 of the CUREE-Caltech Woodframe project refers to the shake table test of a simplified two-story single-family house [8].

- An ***EventGroup*** is defined as a collection of Events. Any change to the data acquisition setup, sensor setup or the specimen setup would initiate a new EventGroup. The sequence of EventGroups in a Task is determined by their startTime. For example, Test Phase 6 of the Task 1.1.1 of the CUREE-Caltech Woodframe project is identified as an

EventGroup because the test structure (specimen) has been changed after Test Phase 5 [8].

- An ***Event***, which is the atomic level of Activity, refers to each single run of an experiment or a simulation. Events within an EventGroup may have different input motions, loading protocols, etc. The sequence of Events in an EventGroup is determined by their startTime. Two types of Event are defined in the model, namely ExperimentEvent and SimulationEvent. An example Event is a particular test within Test Phase 6 of Task 1.1.1 of the CUREE-Caltech Woodframe Project. For each event, output, such as sensor readings or simulation results, are generated and recorded.

The reference data model explicitly models certain Activities that are carried out at multiple Sites. The class hierarchy of Activity in the reference data model is shown in Figure 5. In the reference data model, SingleSiteActivity is defined as Activity that is carried out only at a single Site, whereas MultiSiteActivity is defined as a collection of SingleSiteActivities. Figure 6 shows an example project that has a single site Task (Task1) and a MultiSiteTask (M_Task1). The M_Task1 has Tasks that are undertaken at both Site1 and Site2. The MultiSiteEvent M_E1 has an Event E2 at Site1 and an Event E4 at Site2, and the MultiSiteEvent M_E2 has an Event E3 at Site1 and an Event E5 at Site2. As shown in Figure 6, although Project does not directly *contain* Task2 that takes place at Site2, Task2 can still be accessed from the Project since M_Task1 *contains* Task2. This design enables the support of the types of experiments (such as the MOST experiment) that are carried out either simultaneously or independently at several Sites.

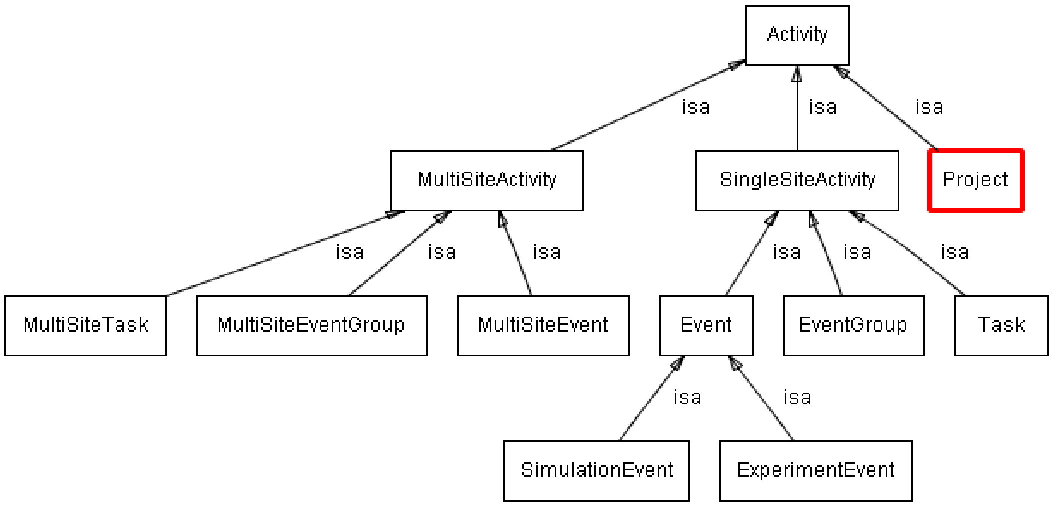


Figure 5 – Class Hierarchy of Activity (generated by Protégé)

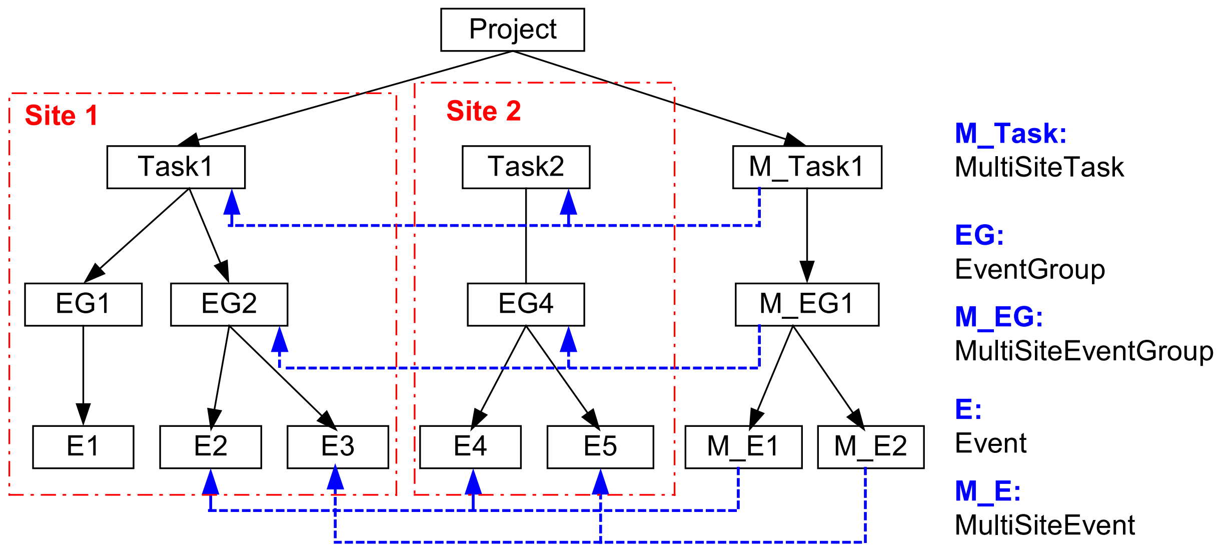


Figure 6 – Layout of an Example Project

3.3 Apparatus

Apparatus is defined as any equipment, specimen, or software that may be used in an Activity. In the current version of the reference data model, the direct subclasses of Apparatus include Specimen, SoftwareProgram, PrimaryEquipment, and SecondaryEquipment. Explicit modeling of Specimen or SoftwareProgram is not considered in the reference model. Only the most basic modeling is provided (as a collection of descriptive files, drawings, and/or photos). This design reflects current approach used to describe specimen in earthquake engineering experiments. However, if so desired, the Specimen class can be extended to support other, more detailed, models.

PrimaryEquipment is the major equipment that is used for the execution of an experiment with respect to a specific research area. Direct subclasses of PrimaryEquipment are ShakeTableEquipment, CentrifugeEquipment, WaveBasinEquipment, FieldTestEquipment, and LargeScaleTestEquipment. Further description of individual PrimaryEquipment is assumed to be contained in the site specifications database [7]. New types of primary equipment can be added to the model as needed.

SecondaryEquipment may be a component of the PrimaryEquipment or may be a piece of equipment that facilitates the execution of an Event, data collection, and/or observation. One important type of SecondaryEquipment are the equipment and sensors used for data acquisition. Data schemas for describing sensors are available; one example is the SensorML [9] developed by OpenGIS Consortium. The reference data model includes a sensor model that is designed specifically to support earthquake engineering experiments. The data acquisition equipment is modeled as a collection of classes, including Sensor, DAQCable, DAQChannel, and DAQSystem. Figure 7 shows the relationships and the slots of these classes. Typically a data acquisition system involves at least three main components: (1) the sensors which respond to a physical stimulus and generate analog voltage signals; (2) a DAQchannel (a.k.a. signal conditioner as part of a DAQSystem) which receives the signal and uses predefined filter, gain, offset, excitation, sensitivity (calibration) information for Analog-to-Digital (A/D) and Engineering Unit (EU) conversions; and (3) a PC unit which uses some communications link (serial port, phone modem, radio modem, etc.) to retrieve the data. It is noted that A/D hardware can be either external to or as part of the signal conditioner. DAQDevice model will be further detailed in future versions of the data model.

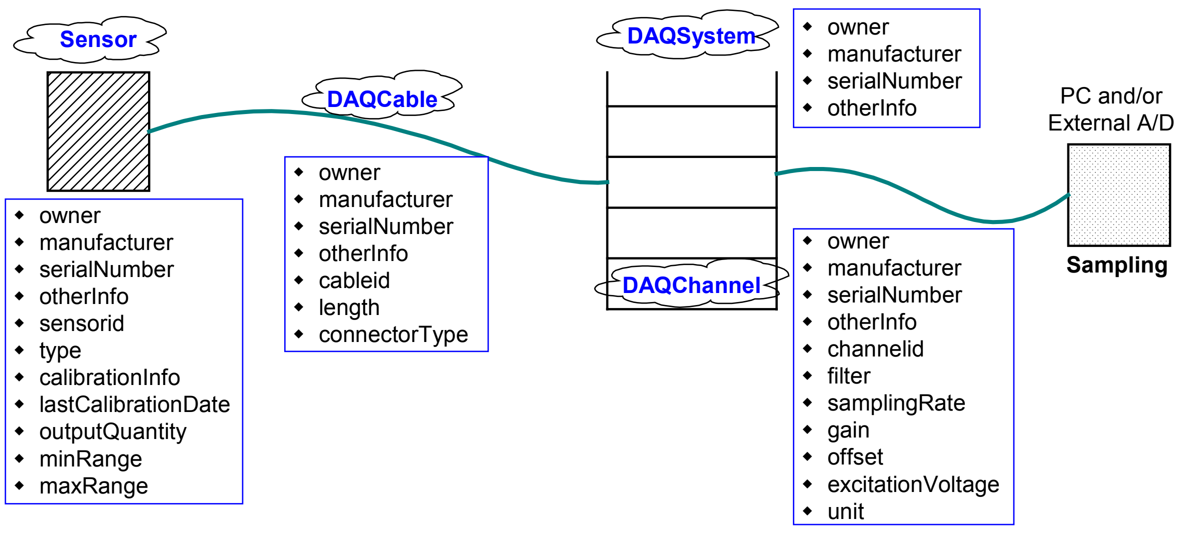


Figure 7 – Setup and Modeling of DAQ Devices

3.4 ApparatusSetup

Universal modeling of the arrangement and setup of apparatus for all experiments is very difficult if not impossible. Not only are there different types of experiments (such as shake table, pseudo-dynamic tests, centrifuge, and tsunami) and different materials (such as concrete, steel, wood, etc.), but also the geometry of specimen, the arrangement of sensors, and the configuration of PrimaryEquipment may be too complicated and cumbersome to model. For example, the "as-built" locations of sensors may be different from the "design" locations, and the exact physical locations (i.e., the coordinates x, y, z values) of sensors are very hard to be recorded. Therefore, it is recommended that the development of ApparatusSetup model be focused on tools and methodologies that can capture and organize CAD drawings, sketched drawings and notes, photos, narrative descriptions, electronic notes, and etc.

The class hierarchy of AppartusSetup in the current reference data model is shown in Figure 8. The InfrastructureSetup models the assembly and arrangement of the PrimaryEquipment used for a specific Task; any changes in InfrastructureSetup would trigger the launch of a new Task. The SpecimenSetup deals with the information on how the specimen is set up with respect to PrimaryEquipment. The SensorSetup includes the arrangement (location, orientation, etc.) of Sensors used in an experiment. The DAQSetup models the physical and electrical setup of one or more devices whose primary purpose is to acquire data. Any major change to SpecimenSetup, SensorSetup, or DAQSetup initiates a new EventGroup. The InputDataSetup deals with the choice and organization of input data to an Event. Any change to a new InputDataSetup indicates the beginning of a new Event.

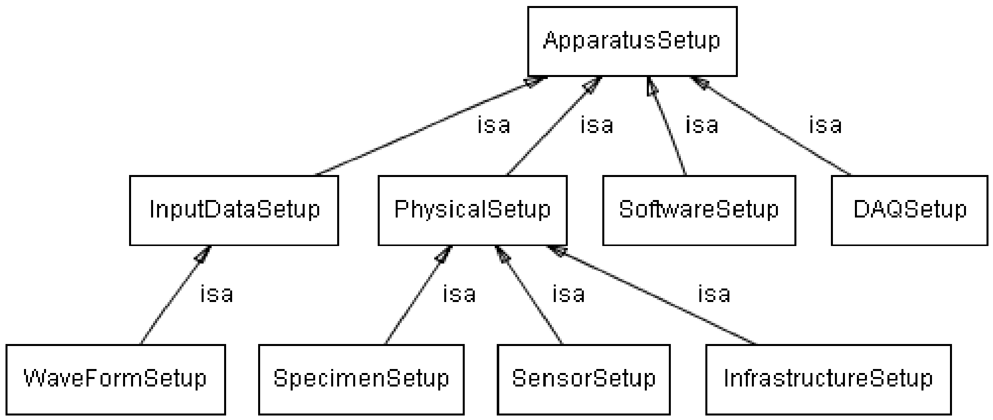


Figure 8 – Class Hierarchy of ApparatusSetup (generated by Protégé)

3.5 DataElement

DataElement represents all types of data that may be generated or processed during an Activity. The DataElement normally serves as Input/Output to an Activity. Types of DataElement include text document, publication, earthquake record, photo, CAD drawing, movie, etc. In NEESgrid data/metadata effort, it is assumed that the data is saved in or translated into computer-readable format. Therefore, a DataElement object is represented in the format (such as a file) that can be saved in computer memory, on disks, or in some kind of data storage repository.

3.6 ComplexDataType

ComplexDataType is defined in the reference data model to represent any data type that is not a simple data type such as integer, Boolean, or character string. In the current version of the reference data model, the following ComplexDataType are provided:

- ***Folder***, which is a collection of DataElements as files.
- ***RolePerson***, which is defined as the combination of a Person and his/her role in an Activity or in an Organization.
- ***Unit***, which is modeled as a name and its description. Currently there is a prototype unit library included in the reference data model. The unit library supports certain basic unit conversion. Other types of unit representation, such as the compact representation [10], can also be incorporated, if needed.
- ***Measurement***, which is defined as a value with associated unit.
- ***Date/Time***, which is externally represented as year, month, day, hour, minute, second, millisecond, etc., and internally saved as a long integer.
- ***Geometry/Location***, the geometry/location is needed for finding sensor location, representing specimen model, and etc. The spatial location is currently modeled as the values in a coordinate system (i.e. x, y, z values). It should be noted that, very often, geometry/location information are specified within CAD drawings or text documents,

etc. Referencing scheme may be added to relate an entity to the source that defines the location.

4 Summary and Discussions

In this report, a reference NEESgrid data model currently under development has been presented. The intention of this document is not to give a detailed description of the data model, but rather to present the preliminary design and to solicit feedbacks and comments from the NEES community. Although the reference data model is focused on shake table experiments, many of the features can be applied or extended to centrifuge, tsunami, pseudo-dynamic and other types of experiments. Six base classes and the relationships among these classes are defined and they represent the essential elements to support the end-to-end solution of NEESgrid data efforts. We believe the proposed reference data model is sufficiently flexible that new classes can easily be introduced; the slots of a particular class can be added, deleted, or modified; and the relationships among the classes can be altered. Other models, such as specimen model, unit model, geometry/location model, and Site model, can be appended to (or even replace certain parts of) the reference model. Data model development is an iterative and evolving process, and the reference data model will continue to be tested, validated, modified and revised, even beyond the current development effort.

We would like to emphasize that the data model development is a community effort. Suggestions and feedback from the NEES community and stakeholders are in the development process. The reference data model described in this document is based on version 0.4, which will be released for the review of NEES community during the second half of April 2004. We look forward to receiving and to incorporating any valuable suggestions from the NEES community.

Acknowledgements

This report is drafted as an interim report by the authors as part of the NEES's System Integration effort, WBS No. 2.4 Data and Metadata Management. The authors would like to acknowledge the active collaboration and contributions of the NEESgrid's Data/Metadata task committee members (sorted alphabetically by their first name):

Andrei Reinhorn	State University of New York, Buffalo
Bill Spencer	University of Urbana-Champaign
Chuck Severance	University of Michigan
Gokhan Pekcan	University of Nevada, Reno
Hank Ratzesberger	University of California, Santa Barbara
Jean-Pierre Bardet	University of Southern California
Jennifer Swift	University of Southern California
Jim Eng	University of Michigan
Jun Peng	Stanford University
Ken Ferschweiler	Northwest Alliance for Computational Science and Engineering
Kincho Law	Stanford University
Lelli Van Den Einde	University of California, San Diego

The authors would like to thank Joe Futrelle of UIUC, Chuck Severance and Jim Eng of U. Michigan for their time and discussions related to NEESgrid developments. The authors would also like to thank Dr. Patrick Laplace of University of Nevada, Reno and Professors Steve Mahin, Bozidar Stojadinovic and Greg Fenves of University of California, Berkeley for their time to discuss the data and metadata issues related to earthquake engineering experiments and simulations. Any opinions, findings, and conclusions or recommendations expressed in this material are, however, those of the authors and do not necessarily reflect the views of others and the National Science Foundation.

References

[1] Jun Peng and Kincho H. Law. *A Brief Review of Data Models for NEESgrid*, Technical Report NEESgrid-2004-01, 2004. (http://www.neesgrid.org/documents/TR_2004_01.pdf)

[2] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*, Stanford Medical Informatics, Stanford University, 2002. (http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html)

[3] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford University, Stanford, CA, 2002. (http://protege.stanford.edu/publications/ontology_development/ontology101.html)

[4] J. Arlow and I. Neustadt. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley Pub Co., Boston, MA, 2001.

[5] J. R. Rumbaugh, M. R. Blaha, W. Lorensen, F. Eddy, and W. Premerlani. *Object-Oriented Modeling and Design*, Prentice Hall, 1990.

[6] NEESgrid Team. *Multi-site Online Simulation Test (MOST)*, 2003. (http://www.neesgrid.org/most/index.html)

[7] Bruce L. Kutter, Daniel W. Wilson, Cherri Pancake, and Sally Haerer. *Introduction to the Site Specifications Database (SSDB)*, Network for Earthquake Engineering Simulation, 2004. (http://nees.orst.edu/IT/site.specs.db/cohorts/Introduction.pdf)

[8] D. Fischer, A. Filiatrault, B. Folz, C.-M. Uang, and F. Seible. *CUREE-Caltech Woodframe Project: Shake Table Tests of a Two-Story Woodframe House*, Consortium of Universities for Research in Earthquake Engineering, 2001.

[9] M. Botts (ed.). *Sensor Model Language (SensorML) for In-situ and Remote Sensors*, OpenGIS Interoperability Program Report, OGC 02-026, Open GIS Consortium Inc, 2002. (http://vast.uah.edu/SensorML/OGC-02-026_SensorML_0.07.doc)

[10] B. Hamilton. *A Compact Representation of Units*, Hewlett-Packard Laboratories, 1996. (http://www.hpl.hp.com/techreports/96/HPL-96-61.pdf)