CHARM
Cultural Heritage Abstract Reference Model

CHARM Extension Guidelines

version 1.1.0 · 4 September 2019

This document and its contents were created by César González-Pérez with the help of Patricia Martín-Rodilla.

Partial funding was provided by Incipit · CSIC and projects MIRFOL (09SEC002606PR, INCITE Programme, Xunta de Galicia, Spain), ARIADNE (313193, FP7-INFRASTRUCTURES-2012-1) and MARIOL (HAR2013-41653-R, Retos de la Sociedad, Plan Estatal 2013-2016, Spain).

4 September 2019 12:28 revision 107

Table of Contents

Introduction

This document presents some guidelines on how to extend CHARM, the Cultural Heritage Abstract Reference Model. Some knowledge of CHARM is assumed throughout; if you are not familiar with CHARM, please refer to the *CHARM White Paper* [4] before reading this document. Some knowledge of object-oriented modelling, and in particular of ConML, is also assumed; if you are not familiar with this, please refer to the ConML web site [2] for introductory resources.

The CHARM extension guidelines rely extensively on ConML's extension mechanisms. This guide hides most of the unnecessary technicalities; however, if you are interested in the details, please see the *ConML Technical Specification* [1] for detailed information. If you need information on the academic underpinnings of CHARM, please see the *Research* and *Resources/Publications* areas, online [3].

For additional information on CHARM, please visit www.charminfo.org.

Background and Motivation

CHARM is an abstract reference model of cultural heritage. This means that it represents the elements of interest to cultural heritage, but from a very abstract point of view. This is intentional; by being abstract, CHARM is applicable to a wide range of problems, geographical areas, chronological periods, cultural scenarios and methodological approaches. Thus, CHARM contains concepts such as *MaterialEntity*, *Agent* or *Valorization*.

However, by being abstract, CHARM is not directly applicable, since it does not contain specific representations of those things that anyone in particular may be interested in. In other words, it is very unlikely that any user of CHARM would find the specific concepts they need already in CHARM. For example, an archaeological museum adopting CHARM would not find concepts such as *Beaker* or *PotteryShard* in CHARM; similarly, an anthropologist studying heritage formation processes would not find concepts such as *IdentityAppreciation* or *Cultural-Production* in CHARM.

You need to extend CHARM in order to use it. Although the word "extend" often suggests adding new contents to something, in the context of models "extending" means adding, removing or modifying classes, attributes, associations and other elements to the base model in order to obtain a resulting model that is perfectly adjusted to your task at hand. Following on our example above, the archaeological museum may add *Beaker* and *PotteryShard* classes to represent the concepts they are interested in, plus an association linking them together. By extending CHARM, you achieve two goals at the same time:

- You obtain a model that is tailored to your particular project or situation. In other words, you are not forced to use an off-the-shelf standard that may not be suitable for you.
- The model that you obtain is still based on CHARM, so that certain level of compatibility is maintained between your projects and other projects using CHARM-derived models. Besides giving you a head start, this facilitates the exchange, comparison and reuse of information.

The result of extending CHARM, i.e. the model that you obtain, is called a *particular model*. The following sections discuss how particular models are created, and describe an example.

If any of these concepts is not clear, please revise the *CHARM White Paper* [4].

Creating a Particular Model

This section describes the tasks related to the creation of a particular model, i.e. a CHARM-derived model that is perfectly adjusted to your needs. There are three major tasks that you need to carry out in order to create a particular model:

- Pick a base model. This may be CHARM itself or a CHARM-derived particular model.
- Study the selected model to rename, modify or remove elements.
- Add your own elements to the model.

These tasks are discussed in depth in the following sections.

Picking a Base Model

A particular model is never created from scratch. Rather, you start by selecting a base model. This may be CHARM itself, or another existing particular model if you have one. Since any particular model is ultimately based on CHARM, your model will also be based on CHARM, either directly or indirectly.

By default, every element in the base model that you select will become a part of your particular model (but see next section for exceptions), so you should select the base model that best suits your needs. You should try to find a model that is as close as possible to your project or situation, but perhaps a little bit more abstract, and use that model as a base. If your organization has developed an organization-level model that every project is expected to use, you may find that it provides valuable concepts that are missing in CHARM; similarly, if your department or team has a specific model that you repeatedly use for your projects, you may find that it contains almost every concept that you need, and that they are precisely arranged in the right manner. If this is not the case and there is no custom model that you can use, then you can always use CHARM as a base model, but you will need to put more work to adjust it to your project or situation.

See *Use Model Refinement Hierarchies in Organizations*, p. 9 for additional recommendations on how to develop and maintain a meaningful hierarchy of particular models.

Renaming, Modifying and Removing Elements

Once you have selected a base model, your particular model is initially identical to said base model. However, it is often necessary to change things in order to adapt them to your project or situations. This can be achieved by renaming, modifying or even completely removing elements in the model according to our needs.

These are the kinds of changes that are possible:

- An element in the model can be renamed. For example, the *Agent* class could be renamed as *Actor* if this name is more suitable for your purpose.
- An element in the model can be removed altogether. For example, the *TangibleEntity* class (and all its subclasses) could be removed from the model if your model is not concerned with tangible entities at all.

The following sections provide details about each of these kinds of changes.

Renaming Elements

You should rename an element in the model if the name provided by the base model is not suitable for your needs. You must bear in mind, however, that the semantics of the concept

represented by the model element are not expected to change by the renaming. If you need a different concept, do not use renaming; add a new concept (see *Any association may be* removed if it is not relevant.

Adding Your Own Content, p. 6); if, on the contrary, the semantics are suitable but the terminology is not, then renaming is an appropriate technique. For example, renaming a class *Structure* as *House* would not be advisable because, although it can be argued that houses are structures, the two terms clearly mean different things. Renaming *Person* as *Individual* would be acceptable, though.

You can rename almost anything in the model: enumerated types, enumerated items, classes, attributes and semi-associations can be renamed.

Removing Elements

Removing an element is useful when your particular model is not concerned with said element. Removal should be used when a simplification of the base model is convenient, and the semantics of the model allows for the resulting structural change. This latter part is important, since it imposes certain limitations to which elements can be removed and which cannot, as explained below.

Enumerated types, enumerated items, classes, attributes and associations can be deleted from a model, according to the following rules.

Enumerated Types

If you want to remove an enumerated type, you also need to remove all the enumerated types that specialize from it. Also, you will need to remove all the attributes having the enumerated type as type, either by themselves or by removing the owner classes.

For example, if you wanted to remove the *ConstructionTechnique* enumerated type from CHARM, you would need to remove the *ConstructedStructure* class, since this class owns an attribute, *ConstructionTechnique*, of type *ConstructionTechnique*. Any additional enumerated types that had been added as specializations of *ConstructionTechnique* would have to be removed as well.

Enumerated Items

If you want to remove an enumerated item, you also need to remove all the descendant enumerated items.

For example, if you wanted to remove the *Stonework* enumerated item from the *Construction-Technique* enumerated type in CHARM, you would also need to remove the enumerated items *AshlarStonework* and *RubbleStonework*, since they are subitems of the former. If any other enumerated type specializing from *ConstructionTechnique* existed, and had sub-items from any of these, they would have to be removed as well.

Classes

If you want to remove a class, you also need to remove all its descendant classes. In addition, you also need to remove all the associations connected to that class and its descendants.

For example, if you wanted to remove the *TangibleEntity* class in CHARM, you would also need to remove all the classes specializing from it, such as *Place* and *MaterialEntity*, plus all the classes specializing from these, such as *StructureEntity*, and so on. In addition, you would need to remove any association connected to any of them; it is the case of e.g. *NamedMeasure*, which *Describes* 1 *TangibleEntity*.

Please note that removing a class may lead to a considerable cascade of removals if the class being initially removed is high up in the generalization hierarchy. However, this may be the desired effect, since the resulting model will be much smaller and simpler.

Attributes

Any attribute may be removed if it is not relevant.

Associations

Any association may be removed if it is not relevant.

Adding Your Own Content

Renaming and removing elements allows you to customize the particular model according to your needs. However, you will most likely need to add extra elements as well. You can add whatever elements you need by following the rules that are described in the sections below.

Enumerated Types and Enumerated Items

You can add new enumerated types as necessary. This includes specializing existing enumerated types as well as adding totally new enumerated types.

For example, you could add a *MediterraneanConstructionTechnique* enumerated type that specializes from the *ConstructionTechnique* enumerated type in CHARM if you were developing a particular model for Mediterranean traditional architecture. You could also add a new *Colours* enumerated type to list a colours standard table, which does not specialize from anything in CHARM.

You can freely add enumerated items to the enumerated types that you create from scratch. You can add sub-items to existing items in enumerated types that you create as specializations from existing ones. However, you cannot add root enumerated items to enumerated types that you create as specializations from existing ones.

Following on with the example above, you could populate the newly added *Colours* enumerated type with items *White*, *Grey*, *Red*, etc., since this is an enumerated type that you have created from scratch. You could also add items to the *MediterraneanConstruction-Technique* enumerated type; in this case, and since this enumerated type inherits its items from *ConstructionTechnique* in CHARM, you could only add sub-items to the existing items. For example, you could add *DryStone* as a sub-item under *RubbleStonework*.

Classes and Features

You can add classes to your model as long as they specialize from existing classes. In other words, you cannot add standalone classes that do not specialize from anything. This limitation rarely has a practical impact, since CHARM provide the *Entity* class as a specialization root, so any class that you may want to add could specialize, directly or indirectly, from this class.

For example, you could add a *Celebration* class as a specialization of the *SocialAct* class in CHARM, since celebrations are a special case of social acts.

You can freely add attributes to classes that you add in this manner. Similarly, you can freely connect classes that you add in this manner by using associations. Please bear in mind that, since the classes that you add specialize from other classes, they will inherit their attributes and associations. For this reason, you need to take these inherited features into account when deciding which ones to add.

For example, you could add a *YearStarted* attribute to your *Celebration* class to document when a celebration was first observed. However, since *Celebration* specializes from *SocialAct*, it inherits the *HasUsualParticipant* association towards *Agent*, so you do not need to add anything to document what agents participate in each celebration.

You can also add attributes to classes that are taken straight from the base model, or connect associations to them. For example, you could add an attribute to *SocialAct* itself, or create a new association that connected one of your classes to *SocialAct*.

A Sample Particular Model

Let's assume the following scenario. A network of museums is about to start a campaign to collect feedback from visitors about the collections on display. The aim is to capture specific comments from visitors about each collection over a long period of time, so that each museum can assess how well received the changes to display and interaction techniques are. Each museum in the network will display multiple collections during the campaign, each collection being composed of multiple objects. Each object will be presented to visitors with a title and a text description. Visits to each museum will be recorded at exit points, and basic details for each visitor will be noted, such as name and nationality. It is important to keep track of how many visitors make up each visit, and what the reason is for each visit, in order to obtain meaningful statistics later. Then, visitors will be asked to volunteer to fill in a simple feedback form where they will be able to provide short comments about which collections they liked or disliked, and why.

A particular model is to be created to describe this situation. The aim of this model will be the development of a database that will allow each museum to store and manage the information related to the campaign, as well as to relate this information to other CHARM-based data sources.

We assume that the museum network has no organization-level model that we could use, so CHARM is selected as base model (see *Picking a Base Model*, p. 4). This is also the simplest case, so it works well for illustration purposes. The contents of CHARM can be found on the online *CHARM Reference* [3]. Once this decision is made, we proceed to modify the model as needed (see *Renaming, Modifying and Removing Elements*, p. 4) and add the necessary content to represent the concepts that are specific to the task at hand (see *Adding Your Own Content*, p. 6), namely museums, collections, visitors, etc. Figure 1 shows the result.

Figure 1. Sample particular model obtained for the scenario described above. Classes shown in white are taken directly from CHARM. Classes shown in green have been added new as part of the particular model. Red lines indicate removed elements.

In Figure 1, the classes representing concepts that are particular to our scenario are shown in green. They stand for the specific knowledge that makes our scenario different from most others, and hence the need to extend CHARM. It is easy to see that all these classes are mentioned, directly or indirectly, in the scenario description that is provided at the top of this section.

These classes need to become part of our particular model, and since they can only do so by specializing from classes in CHARM (see *Adding Your Own Content*, p. 6), it is crucial to determine what classes from CHARM we need to retain in the model, and which can be dropped. Figure 1 shows that, for example, *Museum* has been defined as a specialization of *Organization* in CHARM, and *Visit* as a specialization of *TimePoint* in CHARM.

Once all the particular classes have been incorporated in this manner, every class in CHARM that is not necessary can be removed. Since our scenario is small and involves only a handful of concepts, most of the classes in CHARM can be safely ignored. This is achieved by removing them (see *Removing Elements*, p. 5). We can safely assume that all the CHARM classes not shown or referenced in Figure 1 have been removed in this manner. In addition, the CHARM classes that are kept can be altered as needed (see *Renaming, Modifying and Removing Elements*, p. 4). In our example, no renaming is applied, but a number of attributes are removed. For example, Figure 1 shows that the *Name* attribute has been removed from two CHARM classes.

Finally, it is important to bear in mind that any classes added to the model will inherit the attributes and associations of the classes they specialize from according to the relevant rules. For example, in Figure 1, the class *Visit* is introduced as having attributes *Date*, of type Time, and *Reason*, of type *VisitReason*. The *Date* attribute is a redefinition of the inherited *Moment* attribute of *TimePoint*, as denoted by the square brackets in the figure. The *VisitReason* enumerated type, not present in CHARM, will need to be defined as well as part of the particular model. The *Visit* class, however, does not inherit the *Name* attribute as it has been removed.

Extension Process

Over the previous sections we have shown that the creation of a particular model involves three different tasks: picking a base model, modifying it, and adding your own content (see *Creating a Particular Model*, p. 4). These tasks are rarely performed as a sequence, because they are strongly interconnected. For example, it is often difficult to determine what modifications are necessary before the new content has been at least explored and sketched.

This section describes the suggested process that you should follow to create a particular model. The process outline is as follows:

1. The scope and goals for your particular model are determined.
2. A base model is selected, depending on the above-mentioned scope and goals.
3. Candidate new classes are identified, by determining the relevant concepts to capture in the particular model according to its goals and scope.
4. Ancestor classes in the model are determined for each of the candidate new classes, by finding the best semantic matches.
5. Existing classes in the model are marked as candidates for removal, as they are not relevant to the particular model.
6. New attributes and associations for candidate new classes are sketched. New enumerated types are sketched.
7. Attributes and associations of selected ancestor classes are marked as candidates for removal, as they are not relevant to the particular model.

8. Repeat from step 3, tightening relationships and converging towards a stable solution that captures all the relevant concepts at the appropriate level of detail as established by the particular model scope and goals.

Extension Best Practices

This section describes some extension guidelines that should be always taken into account.

Extension Means Refinement

Extending a model means that the resulting particular model will be a *refinement* of the base model. By "refinement" we mean a more specific, concrete and tailored variant. This means that model extension is applicable whenever a more specific, concrete and tailored model is to be obtained, but not in other cases.

For example, let's imagine that a model exists which has been used for a series of similar projects with great success. We are now faced with a new project that is different, and we wonder whether we should extend the existing model in order to reuse as much of it as possible, remove whatever parts are not relevant, and add the necessary new elements. This would not be advisable, because the new project does not entail a more specific, concrete and/or tailored situation as compared to the previous projects; rather, it is at the same level of abstraction. If anything, an organization-level model, if there is one, could be extended into a project-specific one, but a project-specific model should not be extended "sideways" into another project just to achieve reuse.

See *Use Model Refinement Hierarchies in Organizations*, p. 9 for further elaboration on this theme.

Use Model Refinement Hierarchies in Organizations

In *Background and Motivation*, p. 3 we say that CHARM is an abstract model, and presented this as the reason why it cannot be directly applied to any project or situation, and why the creation of particular models is a need. However, and as hinted at in *Picking a Base Model*, p. 4, not every particular model needs to be aimed at being directly applied to specific projects. Quite to the contrary, it is advisable to consider the creation of intermediate models that capture intermediate levels of abstraction, more specific than CHARM but more abstract than any particular project, whenever the organizational context so suggests. In other words, if the development of a particular model is seen as a dramatic reduction of abstraction, then we can say that this reduction does not need to be made in a single step (from CHARM to a highly specific model), but it can be better made in a sequence of steps (from CHARM to an intermediate model and from this to a highly specific model, for example).

Let's imagine an archaeology consultancy firm doing survey and excavation work under contract. They want to adopt CHARM for all their work, in order to homogenize their data management operations while being able to handle the peculiarities of each project. Instead of having each project create a project-specific particular model using CHARM as a base, this organization would better create an organization-wide model (using CHARM as a base), and then create project-specific particular models by using this organization-wide model as a base whenever necessary. By doing this, they would be able to capture organization conventions, procedures and standards into the organization-wide model, thus giving any project a head start by giving them added specificity by default. Figure 2 shows the difference between the two options.

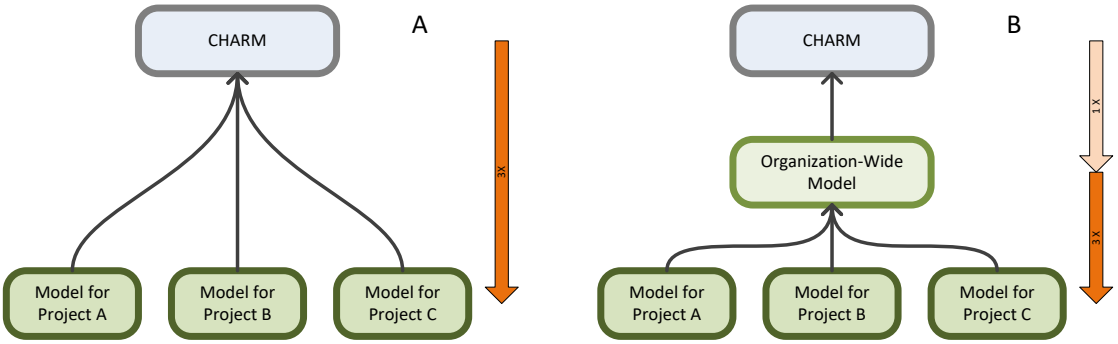

Figure 2. Model hierarchies in organizations. In scenario A, every project model is created from CHARM, which involves a large change in abstraction level. In scenario B, an organization-wide model has been created once, and project models are created from it rather than from CHARM, which involves a much smaller change in abstraction level.

Usually, the larger the change in abstraction level, the larger the modelling effort. For this reason, a scenario such as B in Figure 2, where an organization-wide model captures concepts that are common to every project, not only helps standardize information management in the organization and capture relevant methodological knowledge; it also reduces the effort that is necessary to develop project-specific models.

For example, the archaeology consultancy firm from our example above may create an organization-wide particular model comprising classes such as *Mound* and *Wall*, since these are kinds of material entities that they often deal with in most projects. They would invest a significant amount of time in obtaining good definitions for these terms, and linking them together, and to other classes, in the most appropriate way for their objectives. Then, projects in the firm would use this organization-wide model as a base in order to create their project-specific particular models, gaining direct access to the *Mound* and *Wall* classes and perhaps refining them.

Finally, it is worth noting that model hierarchies can occur in multiple rather than a single level. For example, a large organization with multiple departments may want to develop a CHARM-based organization-wide model, and then ask each department to refine that into a department-specific model. Projects in each department would be created by using the relevant departmental model as a base. Or, an organization that often carries out projects of different kinds (excavation and museum design, for example) could develop a separate particular model for each project type, and then particular models for each individual project within their type.

Acknowledgements

Very special thanks to Brian Henderson-Sellers, Chris Partridge, Isabel Cobas Fernández, Martin Doerr and Sergio España for their helpful insights.

Special thanks to the members of the MIRFOL team: Alejandro Güimil-Fariña, Camila Gianotti, César Parcero-Oubiña, Charlotte Hug, Patricia Martín-Rodilla, Pastor Fábrega-Álvarez and Rebeca Blanco-Rotea. They are the co-creators of CHARM. Also special thanks to the collaborators who helped the MIRFOL team on specific fields: Cristina Mato-Fresán, Lucía Meijueiro and Rocío Varela-Pousa.

Thanks to Isabel Cobas Fernández for her very useful reviews to a draft of this document.

References

[1] Incipit, 2018. *ConML Technical Specification*. ConML 1.4. http://www.conml.org/Resources_TechSpec.aspx

[2] Incipit, 2018. *ConML Web Site*. Accessed on 23/02/2018. http://www.conml.org

[3] Incipit, 2018. *CHARM Web Site*. Accessed on 23/02/2018. http://www.charminfo.org

[4] Incipit, 2018. *CHARM White Paper*, version 1.0.7. Incipit, CSIC. http://www.charminfo.org/Resources/Technical.aspx