An Approach to Conceptual Fit Analysis for COTS Selection

Antoni Olivé

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya – Barcelona Tech
antoni.olive@upc.edu

Abstract. One aspect that has received little attention in COTS selection is what we call conceptual fit, which we evaluate in terms of the existing misfits. We define the notion of conceptual misfit and we present an approach that determines the conceptual misfits between the user requirements and a set of candidate systems. The approach consists in defining a superschema, the mapping of the conceptual schemas of the candidate systems and of the user requirements to that superschema, and the automatic computation of the existing conceptual misfits. We believe that conceptual fit could be taken into account by almost all existing COTS selection methods.

Keywords. Conceptual Fit, COTS selection, Conceptual Modeling

1. Introduction

Nowadays, many organizations build many of their information systems by customizing and/or integrating Commercial-off-the-shelf (COTS) systems. Selecting the most convenient COTS system for a particular situation has become a critical activity in information systems engineering.

COTS system selection essentially consists in evaluating user requirements with respect to characteristics of candidate systems. The evaluation is performed by defining a set of criteria, assessing the importance of each criterion for the users and the degree to which the criterion is satisfied by a system. One kind of criterion that has received little attention is what we call conceptual fit. It is similar to what is called domain compatibility in OTSO [1] and suitability of data in GOThIC [2].

This paper analyzes the conceptual fit between user requirements and COTS systems. We define the notion of *conceptual misfit* and we present an approach to the determination of the existing conceptual misfits between a set of user requirements and a system. The absence of conceptual misfits indicates a perfect conceptual fit. Our notion of conceptual misfit has been inspired in the ontological expressiveness analysis [3].

The structure of the paper is as follows. Next section identifies the different kind of conceptual misfits that may exist between a set of user requirements and a COTS system. Section 3 formalizes the general problem of evaluating the conceptual fit. In

section 4 we describe the approach we propose for solving that problem. Finally, section 5 summarizes the conclusions.

2. Conceptual Fit

By conceptual fit we mean the fit between two structural conceptual schemas. In our context, one conceptual schema is that of the user requirements and the other one is that of a particular COTS system. For the purposes of this paper, we will assume simple structural conceptual schemas consisting only of entity types, ISA hierarchies, attributes and binary associations. This can be easily extended, if desired [4].

In the UML metamodel *M* (see Fig. 1) of the schemas that we consider in this paper, entity types have a name, may be abstract or concrete, may be a singleton or be unconstrained, and may have sub/supertype associations between them. Entity types may have attributes, which are properties. Properties have a minimum and a maximum cardinality, and a type. Cardinalities may be zero, one or unconstrained. Associations have two ordered participants, each of which is a property, as before.

Assume now that we have two instances of *M* that we call U_i (for user requirements) and S_j (for COTS system). We are interested in knowing how well U_i and S_j fit each other. To this end, we try to see whether there are misfits between them. Based on the simple metamodel *M* we identify three kinds of misfits in the schema elements, called deficits, incompatibilities and excesses, which we define in the following. The idea is that the degree of fit of U_i and S_j is inversely proportional to the number of misfits, the maximum being the absence of them.

2.1 Entity Type Misfits

We say that there is an *entity type deficit* between U_i and S_j with respect to (wrt) *E* if *E* is a concrete entity type of U_i but *E* is not an entity type of S_j. Note that we consider only the concrete entity types of U_i because these are the ones of interest to the users. Abstract entity types in U_i are unions of concrete ones.

There is an *entity type cardinality incompatibility* between U_i and S_j wrt *E* if *E* is a concrete entity type of U_i and an entity type of S_j, but *E* is unconstrained (not a singleton) in U_i and a singleton in S_j. Both U_i and S_j have the entity type *E* but, in U_i, *E* may have several instances while only one instance is allowed in S_j.

We say that there is an *entity type excess* between U_i and S_j wrt *E* if *E* is a concrete entity type of S_j but *E* is not an entity type of U_i. In this case, S_j includes an entity type that is not of interest to U_i.

2.2 Attribute Misfits

There is an *induced attribute deficit* between U_i and S_j wrt *A* if *A* is an attribute of the concrete entity type *E* in U_i and there is an entity type deficit between U_i and S_j wrt *E*. In this case, the deficit is induced by the entity type deficit.

There is an *attribute deficit* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in U_i, S_j includes E, but S_j does not include A.

There is an *attribute cardinality incompatibility* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in U_i, S_j includes A, but the cardinalities are incompatible. An incompatibility arises when the minimum cardinality in U_i is zero and one in S_j, or when the maximum cardinality is unconstrained in U_i and one in S_j.

There is an *induced attribute excess* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in S_j and there is an entity type excess between U_i and S_j wrt E. In this case, the excess is induced by the entity type excess.

There is an *attribute excess* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in S_j, U_i includes E, but U_i does not include A. In this case, S_j includes an attribute that is not of interest to U_i.

2.3 Association Misfits

There is an *induced association deficit* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, and there is an entity type deficit between U_i and S_j wrt E_1 or E_2. In this case, the deficit is induced by the entity type deficits.

There is an *association deficit* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, S_j includes E_1 and E_2, but S_j does not include R.

There is an *association cardinality incompatibility* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, S_j includes E_1 and E_2, but the cardinalities of one of its participants are incompatible. An incompatibility arises when the minimum cardinality in U_i is zero and one in S_j, or when the maximum cardinality is unconstrained in U_i and one in S_j.

There is an *induced association excess* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in S_j, and there is an entity type excess between U_i and S_j wrt E_1 or E_2. In this case, the excess is induced by the entity type excess.

There is an *association excess* between U_i and S_j wrt R if R is an association of the concrete entity types E_1 and E_2 in S_j, U_i includes E_1 and E_2, but U_i does not require R.

3. Evaluating the Conceptual Fit Criterion for COTS Selection

The general problem of evaluating the conceptual fit criterion can be defined as follows:

Given:

- The user requirements U_i of a system in some domain and
- A set $S_1,\ldots,S_n$ of n candidate COTS systems in that domain,

Determine:

- The conceptual misfits (deficits, misfits and excesses as defined in the previous section) between U_i and each of the $S_1,\ldots,S_n$.

Conceptual fit analysis can be performed considering the complete set of user requirements U_i and of the candidate systems $S_1,\ldots,S_n$, or considering only a fragment of them. The latter possibility is likely to be of much more practical interest in most cases.

The set of conceptual misfits found can be used as a basis for selection. If there are no misfits between U_i and S_j, then there is a perfect fit between them.

If there are one or more deficits or incompatibilities between U_i and S_j, then the selection of S_j would require either the change of the user requirements U_i (changing their intended way-of-working) or a customization of S_j for the user (customizing existing systems to accommodate users' requirements).

If there are one or more excesses between U_i and S_j, then the selection of S_j would imply dealing with the unneeded features related to those excesses, and the need of the corresponding resources.

4. A Method for Determining the Conceptual Fit

A straightforward approach to the solution of the general problem of determining the conceptual fit would be to consider each S_j ($j = 1,\ldots,n$) separately, and determine the conceptual misfits between U_i and S_j as indicated in Sect. 2. When the number n is large and/or the conceptual schemas are large, the evaluation effort may be large too.

However, in a context where the selection process must be performed several times with the same set of candidate systems $S_1,\ldots,S_n$, with different user requirements U_i, then a better solution would be to build an intermediate superschema S. That superschema S should integrate $S_1,\ldots,S_n$ in a way such that U_i and each of the $S_1,\ldots,S_n$ could be mapped to S, and such that the conceptual misfits of U_i and each of the $S_1,\ldots,S_n$ could then be computed automatically.

Based on the above idea, the method we propose consists of four parts:

1. A superschema S that is a union of all schemas $S_1,\ldots,S_n$ and all possible user requirements $U_1,\ldots,U_m$ in a given domain.
2. The definition of the schemas $S_1,\ldots,S_n$ in terms of S.
3. The definition of user requirements U_i in terms of S.
4. The (automatic) computation of the misfits between U_i and $S_1,\ldots,S_n$.

We describe these parts in the following.

4.1 The superschema

In our method, the superschema S is an instance of the metamodel M for a domain D such that:

- S includes the schemas of all possible COTS systems $S_1,\ldots,S_n$ in D.
- S includes all possible conceptual user requirements $U_1,\ldots,U_m$ in D.

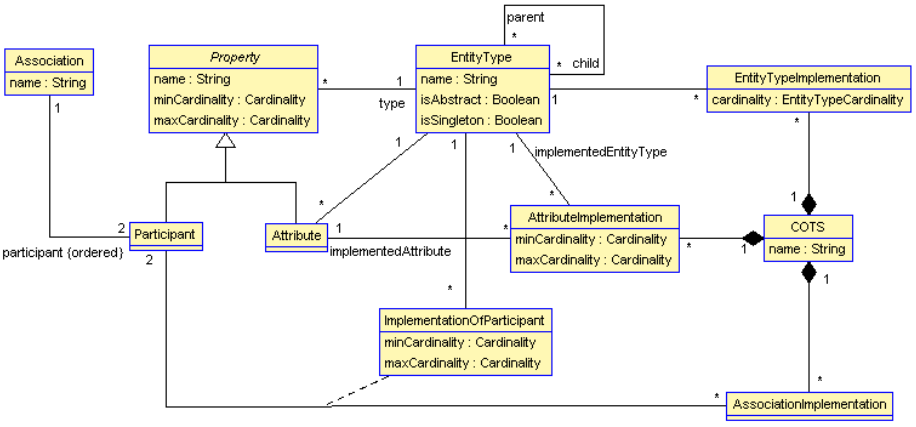

Fig.1. COTS implementation of a superschema

By inclusion of schemas here we mean that:

- *S* comprises all concrete entity types, attributes and associations that may be required by $U_1,\ldots,U_m$. On the other hand, the cardinalities of the attributes and associations in *S* must not be incompatible with those that may be required by $U_1,\ldots,U_m$.
- *S* comprises all concrete entity types, attributes and associations that are implemented in $S_1,\ldots,S_n$. On the other hand, the cardinalities of the attributes and associations in *S* must not be incompatible with those that are implemented in $S_1,\ldots,S_n$.

4.2 Mapping Conceptual Schemas of COTS Systems to the Superschema

For the purposes of conceptual fit analysis we need to know for each S_j ($j = 1,\ldots,n$) in *D*:

- The entity types of *S* implemented in S_j and their corresponding cardinalities. We are interested only in the entity types that are concrete in S_j. If S_j implements all subtypes of an abstract entity type *E* in *S*, then S_j also implements *E*.
- The attributes and associations of *S* implemented in S_j and their corresponding cardinalities.

Figure 1 shows the metamodel *M* and the extension needed to represent the part of *S* that is implemented by S_j. A COTS system is assumed to implement a set of concrete entity types (with a cardinality that may be Singleton or Unconstrained), a set of attributes and a set of associations.

Note that if *S* includes an abstract entity type *E* with subtypes $E_1,\ldots, E_m$ and *E* has an attribute *A*, then a system S_j that implements two or more of those subtypes could implement *A* differently in each case. Our metamodel of Fig. 1 takes this possibility into consideration by indicating in *AttributeImplementation* the implemented entity type. A similar reasoning applies to the association participants.

The mapping process can be superschema-driven or system-driven. In the former, the elements of *S* are taken in some convenient order, and for each of them it is

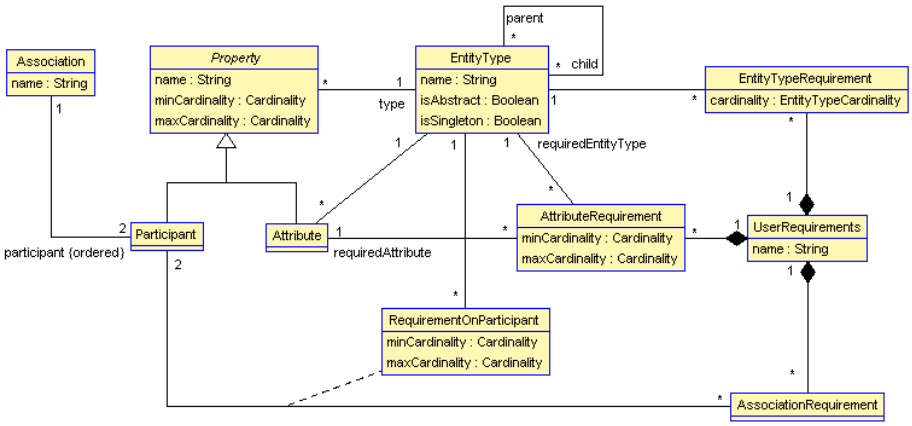

Fig.2. Extension of the metamodel *M* with user requirements

checked whether or not it is implemented by the system. If the element is a concrete entity type that is not implemented by S_j then there is no need to check the implementation of its attributes and associations. To use this process, the conceptual schema of S_j needs not to be explicit; what is needed to know is what entity types, attributes and associations of *S* are implemented in S_j.

In the system-driven process, the elements of the conceptual schema of S_j are taken in some convenient order, and each of them is mapped to *S*. To use this process the conceptual schema of S_j must be explicit.

4.3 Defining Conceptual User Requirements

For the purposes of conceptual fit analysis of U_i we need to know:

- The entity types of *S* required by U_i and their corresponding cardinalities. We need to know only the entity types that are concrete in U_i. If U_i requires all subtypes of an abstract entity type *E* in *S*, then U_i also requires *E*.
- The attributes and associations of *S* required by U_i and their corresponding cardinalities.

Figure 2 shows the extension of the metamodel *M* needed to represent the user requirements in terms of *S*. User requirements are assumed to consist of concrete entity types (with a cardinality that may be Singleton or Unconstrained), a set of attributes and a set of associations.

Note that similarly to the previous case, if *S* includes an abstract entity type *E* with subtypes $E_1,\ldots, E_m$ and *E* has an attribute *A*, then if U_i requires two or more of those subtypes, it could require *A* differently in each case. The same applies to association participants.

As in the mapping of systems, the definition of user requirements can be superschema-driven or requirements-driven.

4.4 Computing Misfits

In our method, once we have defined the instance of M corresponding to the superschema S for a domain D, the instances of the candidate COTS systems $S_1,\ldots,S_n$ in D and their mapping to S (Fig. 1), and the instance of the user requirements U_i and its mapping to S (Fig. 2) we can then automatically compute the misfits between U_i and $S_1,\ldots,S_n$. In what follows we explain the details of the computation in terms of the UML schemas shown in Figs. 1 and 2.

Entity type deficit. Let E be an entity type required by U_i. There is a deficit of E in S_j if E is not implemented in S_j. E can be implemented in S_j directly or by exclusion. There is a direct implementation when E is also an entity type of S_j. There is an implementation by exclusion when there is an entity type E' implemented by S_j such that E' is a supertype of E, $E_1,\ldots, E_p$ ($p > 0$) and $E_1,\ldots, E_p$ are not required by U_i. The exclusion of $E_1,\ldots, E_p$ by U_i implies that the population of E and E' will always be the same, and therefore E' can implement E in S_j.

Entity type incompatibility. Let E be an unconstrained entity type required by U_i. There is an incompatibility when E is implemented by a singleton entity type in S_j.

Entity type excess. Let E be an entity type in S_j. There is a misfit of this kind when E does not implement any entity type in U_i.

Induced attribute deficit. This happens when U_i requires an attribute of entity type E and there is an entity type deficit between U_i and S_j wrt E.

Attribute deficit. This happens when U_i requires an attribute A of an entity type E that is implemented in S_j, but that implementation does not include A.

Attribute cardinality incompatibility. This happens when the cardinalities of an attribute required by U_i are incompatible with those of its implementation in S_j.

Induced attribute excess. Let A be an attribute of a concrete entity type E in S_j. There is a misfit of this kind when E is an entity type excess for U_i. In OCL:

Attribute excess. Let A be an attribute of a concrete entity type E in S_j. There is a misfit of this kind when E is an implementation of an entity type required by U_i but A is not implemented.

Induced association deficit. There is misfit of this kind when U_i requires an association R between the concrete entity types E_1 and E_2 and there is an entity type deficit between U_i and S_j wrt E_1 or E_2.

Association deficit. There is misfit of this kind when U_i requires an association R between the concrete entity types E_1 and E_2 that are implemented in S_j, but S_j does not include R.

Association cardinality incompatibility. This happens when the cardinalities of an association required by U_i are incompatible with those of the implemented association in S_j.

Induced association excess. Let R be an association between the concrete entity types E_1 and E_2 in S_j. There is a misfit of this kind when E_1 and E_2 are an entity type excess for U_i.

Association excess. Let R be an association between the concrete entity types E_1 and E_2 in S_j. There is a misfit of this kind when E_1 and E_2 are implementations of entity types in U_i but R is not.

5. Conclusions

We have proposed a new aspect for COTS system selection, which we call conceptual fit, which assesses the fit between the conceptual structure of a given system and of the user requirements. We have identified three kinds of misfits in the schema elements, called deficits, incompatibilities and excesses. The idea is that the degree of conceptual fit is inversely proportional to the number of misfits, the maximum being the absence of them.

We have formally defined the general problem of evaluating the conceptual fit between the user requirements and a set of COTS systems, and we have proposed a new method for its solution. The method consists in defining a superschema, the mapping of the conceptual schemas of the candidate systems and of the user requirements to that superschema, and the computation of the conceptual misfits.

The main effort required by our method is the development of the superschema and the mapping of the candidate systems to it. However, this must be done only once per domain and the result could be reused in all COTS selections of a domain.

Acknowledgments. This work has been partly supported by the Ministerio de Economía y Competitividad and FEDER under project TIN2008-00444, Grupo Consolidado.

References

1. Kontio, J.: OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland Technical Reports. College Park, University of Maryland. CS-TR-3478, UMIACS-TR-95-63, (1995)
2. Ayala, C.P., Franch, X.: Domain Analysis for Supporting Commercial Off-the-Shelf Components Selection. In: D.W. Embley, A. Olivé, and S. Ram (Eds.): ER 2006, LNCS 4215, pp. 354 – 370, (2006)
3. Wand, Y.: Ontology as a foundation for meta-modelling and method engineering. Information & Software Technology 38(4): 281-287 (1996)
4. Olive, A.: Conceptual Modeling of Information Systems. Springer, Berlin (2007)