Effective Sharing of Family History Information

Scott Woodfield
Computer Science Dept.
Brigham Young University
801-489-3046
woodfiel@cs.byu.edu

ABSTRACT

Family history research is significantly enhanced when we collaborate using automated communication. Unfortunately, machine-to-machine communication can lead to unintended information modification or information loss. We propose the development of an exchange model that includes two sections. One represents common family history information whose organization and content are specified by an accepted standards committee. The first section should be comprehensive, but need not attempt to be all things to all people. The second section would include less common information and a conceptual model (or schema) to define the organization and semantics of that information. We briefly describe a powerful meta-model for defining such conceptual models. With this structure we can reduce or eliminate information loss when exchanging information using computerized tools.

Keywords

Conceptual model, family history, information sharing

1. INTRODUCTION

Family history research is much more enjoyable and efficient when done in collaboration with others. Collaboration reduces redundant work, puts multiple eyes on a problem, and provides emotional support when things are difficult. Fifty years ago researchers used hand written letters or a telephone to communicate. Twenty-five years ago, we improved the speed of communication by using emails. Now we can automatically exchange information.

There is a basic problem with current family history communication software however. The information sent is not always what the receiver actually receives. The primary problem is a mismatch between the source's conceptual model and that of the receiver. For instance, if one program allows a user to assign probabilities to the different birthdates of a person, but the receiver's program does not, then the probability information will be lost.

We use the term conceptual model to mean a specification of what information can be stored and how it is organized. We use the term conceptual model rather than schema because it has less of an implementation connotation. In fact, a conceptual model does not specify a representation or implementation. For instance, in a family history program, it does not specify what kind of database to use. For transferring information, it does not specify the underlying format (e.g. XML[1]). A conceptual model only specifies content and its organization. It should also specify valid and invalid content. A conceptual model should be easy to read, understand, and if necessary, modify.

The first problem we have to solve is to make sure all the information in one model can be transferred to another model. If that cannot be done, no amount of agreement on representation will help.

The first problem is compounded by the lack of a widely accepted exchange model that does not distort or lose information. We need a single exchange model with an agreed upon representation, for without one, we are forced to create numerous exchange models. If there are *n* different programs (e.g. PAF[2] and Family Tree Maker[3]) then we would have to create up to $\frac{n(n-1)}{2}$ different custom exchange programs.

Even with a single exchange model, if it is not sufficiently powerful, there can be problems.
The information sent from program A through the intermediate model to program B may damage or lose the information during two transformations -- first, when converting from program A to the intermediate form, and second, when converting from the intermediate form to program B.

There is an even more subtle problem. Information send from A to B may be corrupted or missing segments. However, B does not recognize this fact and disseminates the recently received information to many other associates. Even if the transmission from B is correct, the information sent is incorrect and none of the recipients is the wiser.

To easily create and extend conceptual models requires a well-defined model of conceptual models, a meta-model. Tools for creating and editing conceptual models are a necessity. The meta-model should be formally defined. From a formally defined meta-model we can automatically generate implementations of a conceptual model. An

example of such a meta-model is UML[4]. A common problem with any conceptual modeling tool, such as UML, is that it is not powerful enough to represent some of the concepts needed for a genealogical conceptual model. For instance, while it can represent crisp logic, it cannot represent fuzzy logic. It can only represent sets, partially-ordered sets, totally-ordered sets, and lists informally. There is no distinction between hard constraints and soft constraints. It has no probabilistic constraints.

Even with current modeling techniques, we miss things that make conversion difficult. If a model does not provide adequate support for modeling sources, it makes it difficult to send or receive information from one that does. We have similar problems with research support such as the representation of personas and scenarios. Even if we did, customizing or extending existing models is significantly constrained or impossible.

2. PRIOR SOLUTIONS

There have been many attempts to solve the transmission problem. All try to create the one common model that rules them all. Basically, the common model has to be all things to all people, but it usually falls short.

2.1 GEDCOM

The most frequently used exchange model is GEDCOM[5]. It has many of the limitations described above, such as poor support for sources, inability to support formal conclusions using formal logic, and the difficulty in representing structures other than lists (e.g. partially-ordered lists).

2.2 The Union Model

Another solution is to try and create the total union model. That is, we will try to create a model that can represent anything stored in any family history program. This presents a few problems. First, if any model is extended, the union model might need changing. Second, we almost always leave out some obscure concept found in some program. Third, it is difficult to agree on what is in the union. An example of an attempt at a universal model can be found at opengen.org. It is interesting that such efforts create yet another conceptual model.

2.3 My Program Is Sufficient

A variation of the total union model is, ”my software can represent anything anyone ever would want to represent.” The Master Genealogist (TMG[6]) attempts to do this with its GenBridge[7] software. While better than GEDCOM, it is proprietary and thus controlled by Wholly Genes[8]. All information must conform to the internal format of GenBridge. If it can’t represent something (e.g. probabilistic constraints), it can’t be transferred. Support and requested improvements are done at the discretion of Wholly Genes (they do try to be very helpful). It is a windows only piece of software. Every transformation from a different model (e.g. Family Tree Maker) to GenBridge must be custom built. GenBridge is a good solution but we can do better.

2.4 The Text Bucket

Another common solution is the “text bucket” solution. That is, if information cannot be automatically converted from one model to another, it is converted to text and stored. There are three problems. First the conversion is not always accurate. Second, the information must be manually converted into any destination format. This is fraught with errors and bias. Third, it is difficult for machines to process text.

2.5 The General Model

Another solution is to create a general model that can be specialized to represent information. An example was the Gentech Genealogy Data Model[9]. As with other solutions we are constrained by the power of the underlying model. For instance, the description of the data model states that “*all* genealogical data can be broken down into a series of short, formal genealogical statements” [10]. Is that true, and, are their genealogical statements sufficient? For instance, hard and soft constraints are difficult to represent in their system. Another problem is the use of lists or sequences to represent information. In many cases it appears that the information is better represented as a set or ordered-set. The Gentech model also has a flaw in that it over constrains an implementation. It is a design model rather than conceptual model. In particular, it tends to “normalize” the organization of the information. It also defines foreign keys and primary keys, concepts not present in a conceptual model. Thus, the target storage mechanism is a relational database. An object-oriented database would find it cumbersome to implement the Gentech model.

2.6 Specific Attempts at an Exchange Model

There are many attempts at creating a replacement for GEDCOM. Some are mentioned above. A very good starting point for exploring the various attempts are Tamra Jone's article giving an overview of GEDCOM replacements [11]. The BetterGEDCOM project also provides similar information.

2.7 The Test

All prior solutions have merit but few can guarantee the goal of any conversion situation. For any piece of information m_1 that satisfies some conceptual model M_1, then for any transformation $T_{M_1M_2}()$ from M_1 to M_2, and transformation $T_{M_2M_1}()$ from M_2 to M_1, the following expression should hold:

Equation 1. $T_{M_2M_1}(T_{M_1M_2}(m_1)) = m_1$.

In other words, the conversion of information from one model to another and back should not result in the loss of any information.

3. A SOLUTION

First, there must be a powerful meta-model for defining conceptual models. Second, there should be a single common, but not necessarily complete, reference model for genealogical information that all agree on. Third, the information in any application should allow for the storage of any information defined by the common model, the recording of extension model definitions, and the storage of information defined by the extension models. Fourth, there should be a universally accepted information exchange model that supports both the common model and extensions.

3.1 The Meta-Model

A sufficiently powerful meta-model is required to define suitable conceptual models. We have developed an extended text-version of the Object Relationship Model(ORM) found in Object-oriented System Analysis[13]. We call it E-ORM. E-ORM supports many of the qualities required of a meta-model. It incorporates all of the concepts found in UML such as relations, formal 1st-order constraints, generalization-specialization, and aggregation. In addition, it has been extended to include formal sets, partially-ordered set, multi-sets, and lists. It also allows for soft and hard constraints and allows probabilistic constraints. Being text-based it is easy to edit. Its syntax is formally defined, making it possible to parse it and automatically generate data storage implementations. Because the representation of relations, generalizations, and aggregations appear as sentence-like structures, it is easier to read. While there are more extensions to be considered (e.g. fuzzy logic), E-ORM is suitable to modeling family history information. E-ORM will come in two forms, a definition form to be used by experts and a readable form to be used by normal users

As a comparison between UML, the definition version of E-ORM, and the readable version of E-ORM, see figures 1, 2, and 3.

The UML version of the model is readable with training. Editors for UML are expensive. Automatic

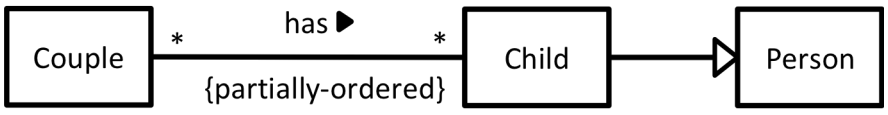

Figure 1. UML Version of Partial Family History Model

```
Couple[SOFT 0:15] has Child[SOFT 1][PARITALLY-ORDERED];
Child IS_A Person;
```

Figure 2. Extended ORM, Definition Version

A Couple should have 0 to 15 Child, a Child should belong to 1 and only 1 Couple.
The set of Child belonging to a Couple is partially ordered.
A Child is a Person;

Figure 3. Extended ORM, Readable Version

conversion of UML models to implementations is difficult and tools are expensive. Working with the definition version of E-ORM also requires training. However, it can be created and edited with simple text processing tools. There is also an alpha-version of a parser that can be used to automatically generate information structures such as dataset or XML schemas. At this time there is no tool for converting definition E-ORM models to or from readable E-ORM models. Such tools have been developed for meta-models such as Object Role Modeling[14] and should be able to be created for E-ORM.

3.2 A Single Common Reference Model

To communicate efficiently, there should be a single common reference model, M_c. A reference model is a conceptual model that all programs and exchange programs implement. This guarantees that any information stored in program A, whose organization satisfies the reference model, can be transformed and stored in another program B, whose organization also satisfies the reference model. This does not require that the reference model be implemented in the same manner in both programs. Program A may organize information according to individuals and families. Program B may organize information more along the lines of events and characteristics. It doesn't matter. If they both satisfy the common reference model then a transform must exist that can transform information in A to information in B.

The reference model need not, and probably cannot be, a "union" model, also known as "everything including the kitchen sink" model. Such a model isn't really possible. It should be more of an intersection model. It should represent all information that can be stored in any genealogy program. It may also store information that is commonly found in other programs, but perhaps not in all. For instance, not every family history program supports the recording of LDS information. However, most do. Thus, it might be wise to define such capabilities for all. As will be seen later, it is not critical that the reference model be complete.

To be widely, accepted the common reference model should be created and controlled by a sufficiently powerful standards committee. Preferably, the model and supporting software should be open source. All major players in family history software should be represented. Though beyond the scope of this paper, and not subject to a technical solution, the failure to create such an organization is a leading cause of family history communication difficulties.

3.3 Structure of Source and Destination Models

To support the exchange algorithm of section 3.6 we suggest that family history programs organize information as follows. 1) The data store must allow the storage of all information, m_c, that satisfies the common reference model, M_c. 2) It should explicitly store a conceptual model, M_{ue}, that defines all user-defined extensions to the reference model. 3) The data store must allow the storage of all information, m_{ue}, that satisfies M_{ue}. 4) A conceptual model, M_{xe}, must be explicitly stored. This model defines the structure of all information received from external parties whose structure is not defined by M_c or M_{ue}. 5) A means must be provided to store data, m_{xe}, that is received from other parties that does not satisfy M_c or M_{ue} and thus cannot be stored in m_c or m_{ue}.

M_c need not be explicitly stored. All of the sections may be empty. If M_{ue} is empty then m_{ue} must be empty. If M_{xe} is empty then m_{xe} must be empty. M_{ue} and M_{xe} must be disjoint. It will make things easier if M_{ue} and M_{xe} are stored using the syntax and structure used to specify the exchange model extensions of the next section. We recommend that M_{ue} and M_{xe} be defined using meta-model as powerful as the one described in section 3.1.

3.4 A Single Exchange Model

There must be a single exchange model with a single implementation used by all parties. For every transmission of information from a source to a destination, there is an exchange packet *X*. It has three components: $X.m_c, X.M_e, X.m_e$. $X.m_c$ is all data that satisfies M_c. $X.M_e$ is a conceptual model that defines the organization of all information that cannot be represented by M_c. $X.M_e$ should be expressed by the meta-model used to define M_{ue} and M_{xe} $X.m_e$ is the data that cannot be described by M_c and thus cannot be stored in $X.m_c$. $X.M_e$ must be defined such that all of $X.m_e$ can be organized according to $X.M_e$.

A well-defined interface for storing information in and retrieving information from the exchange model should be provided.

3.5 Required Transformations

To effectively exchange information the following transforms are required. In the following equations, the prime symbol (“’”) means the selected information from a source, *S*, that is being transferred to a destination, *D*.

1) $T_{S.m_c,X.m_c}(S.m'_c)$
2) $T_{S.M_{ue},X.M_e}(S.M'_{ue})$
3) $T_{S.M_{xe},X.M_e}(S.M'_{xe})$
4) $T_{S.m_{ue},X.m_e}(S.m'_{ue})$
5) $T_{S.m_{xe},X.m_e}(S.m'_{xe})$
6) $T_{X.m_c,R.m_c}(X.m_c)$
7) $T_{X.M_e,D.M_e}(X.M_e, D.M_{ue}, D.M_{xe})$
8) $T_{X.m_e,D.m_{ue}}(X.m_e, X.M_e, D.M_{ue})$
9) $T_{X.m_e,D.m_{xe}}(X.m_e, X.M_e, D.M_{xe})$
10) $T_{X.m_e,D.m_e}(X.m_e, X.M_e, D.M_e)$

Transformations 1 through 5 convert the information in the source to the representation defined by the exchange model. Transform 1 converts all selected data in the source that is defined by the common conceptual model to the data representation of the common conceptual model in the exchange model. Transform 2 converts the representation of the conceptual model for user-defined extensions in the source to the representation of conceptual models for extensions in exchange model. Only that part of the user-defined model needed to define data selected from the user-defined model ($S.m'_{ue}$) is included. If the representation of conceptual models in the source is the same as that used in the exchange model, then the transformation is the identity function. Transform 3 does the same thing as transform 2, except it converts the external extensions stored in the source. Transforms 2 and 3 are greatly simplified if the representation of conceptual models for $S.M_{ue}$ and $S.M_{xe}$ is the same as that used to represent $X.M_e$. Transform 4 converts the selected user-defined data in the source and converts it to the format and organization defined in $X.M_e$. Transform 5 is analogous to transform 4 except it works on the externally-defined information.

Transformations 6 through 10 convert information stored in the exchange model to information stored in the destination. Transform 6 converts all of the data in the exchange model conforming to the common reference model to the format used to store data in $D.m_c$. Transform 7 creates a conceptual model, $D.M_e$, representing all concepts in $X.M_e$ not found in $D.M_c$, $D.M_{ue}$, or $D.M_{xe}$. $D.M_e$ is a temporary model and not stored. Transform 8 extracts the set of data from $X.m_e$ that can be stored in $D.m_{ue}$. Similarly, transform 9 extracts the set of data from $X.m_e$ that can be stored in $D.m_{xe}$. Transform 10 extracts the set of data from $X.m_e$ that cannot be stored in $D.m_c$, $D.m_{ue}$, or $D.m_{xe}$. It is organized according to the model created by transform 7. Transforms 6 through 10 are much easier to create if the conceptual modeling representation of the exchange model is the same as that used by the destination.

3.6 The Exchange Algorithm

The exchange algorithm is performed by the following sequence of actions.

1) Create an empty exchange packet *X*.
2) $X.m_{cT} = T_{m_{cS},m_{cT}}(m'_{cS})$

   Convert the common information selected in the source to the format used to represent data in the exchange model and store it in the section of *X* used to store common information.
3) $X.M_e = T_{S.m_{ue},X.M_e}(S.M'_{ue}) + T_{S.m_{xe},X.M_e}(S.M'_{xe})$

   Convert subsets of models defined by $S.M_{ue}$ and $S.M_{xe}$ to the format used to store the extension models in the exchange model. The smallest subset is chosen that totally defines the selected data to be sent. If the conceptual model representation is textual and the same in the source as in the exchange model this step becomes simple. Because $S.M_{ue}$ and $S.M_{xe}$ are assumed to be disjoint we need only concatenate $S.M'_{ue}$ and $S.M'_{xe}$ and store them in $X.M_e$.
4) $X.m_e = T_{S.m_{ue},X.m_e}(S.m'_{ue}) + T_{S.m_{xe},X.m_e}(S.m'_{xe})$

   Convert the selected data from the user-defined and externally-defined sections of the source data to the exchange data format and store it in the exchange packet. The

exchange data format should satisfy the $X.M_e$ specification.

5) Send the exchange packet X to the destination D.
6) $D.m_c = D.m_c + T_{X.m_c,D.m_c}(X.m_c)$
   Convert the common data in the exchange packet to the format of the destination's common data and add it to the destination's common data.
7) $D.m_{ue} = D.m_{ue} + T_{X.m_e,D.m_{ue}}(X.m_e, X.M_e, D.M_{ue})$
   Extract the data in $X.m_e$ that, according to $D.M_{ue}$, can be stored in $D.m_{ue}$ and add it to the destination's user-defined data.
8) $D.M_{xe} = D.M_{xe} + T_{X.M_e,D.M_e}(X.M_e, D.M_{ue}, D.M_{xe})$
   Extract conceptual model constructs from $X.M_e$ not found in the user-defined or externally-defined models of the destination. Add them to the externally-defined conceptual model of the destination.
9) $D.m_{xe} = D.m_{xe} + T_{X.m_e,D.m_{xe}}(X.m_e, D.M_{xe})$
   Extract the data in $X.m_e$ that, according to $D.M_{xe}$, can be stored in $D.m_{xe}$ and add it to the destination's externally-defined data.

4. CONSEQUENCES

The core of the solution to the information exchange problem is to first send any selected information that fits the common model. This, in some sense, is like the old GEDCOM model. For any information in the source that does not fit the common model, send it and its corresponding conceptual model to the destination. At the destination store all data from the source that fits the common model in the common data section. All other data that can be stored in user-defined section is converted and stored there. The remaining data, along with any needed conceptual modeling components, is stored in the externally-defined section.

If done in this manner, we should be able to satisfy equation 1 above. That is, if a source sends any information to a destination and that in turn is sent back to the source, nothing is lost. Another advantage of this solution is that the organization and representation of information in the source, destination, and exchange packet may all be different and it will still work. It is now possible for users or others to extend the conceptual model of any program and still be able to share information without loss.

5. FUTURE WORK

Much of what has been described does not exist. It will probably not succeed unless involved parties expend efforts to make it happen. Some committee needs to be formed and tasked to create a solution. There are some obvious first steps.

The most difficult step will be to algorithmically determine how the $X.M_e$ model relates to $X.M_{ue}$ and $X.M_{xe}$ models. It will also be difficult to convert $X.m_e$ data to $D.m_{ue}$ and $D.m_{xe}$ data. Beyond using the common reference model, we can reduce the problem if sub-models are created and certified by some recognized standards committee. For instance, a "Civil War" sub-model could be created and used by all who wish to do family history research for people that existed during the Civil War. Such a model would be stored in a user's M_{ue} model. Anyone using the Civil War extensions can send their information to anyone, even those who have not extended their common model with the Civil War sub-model. In the latter case the Civil War information will be stored in the recipient's $X.m_e$ portion of their data store. When the information is in turn sent on to someone who does use the Civil War sub-model, they will be able to receive the information, without loss. It will appear as if it came from the first party.

There are still problems converting subsets of $X.m_e$ corresponding to user-defined or uncertified sub-models appearing in $X.M_e$. While semantics may be the same, models may not. For instance, one model may record a marriage date as being associated with a couple. Another may store it as the date of a marriage event. Determining which information is the same, and which is not, is difficult. The problem is similar to exchanging data between heterogeneous databases. We hope to partially solve the problem using an ontological approach and natural language processing. With the limited family history domain, we believe progress can be made.

The quality of the alpha version of E-ORM must be improved until it becomes a 1.0 version of the software. It would be nice to add fuzzy logic to the model but that is a lower priority. It is also desirable to create a program that can take the definition version of

E-ORM and convert it to the more readable form. Such programs have been done before and should be feasible. Some conversions will be difficult, such as how to make probabilistic constraints understandable. If we consider the readable version of E-ORM to be a stylized version of English with a suitable grammar, we should also be able to take a readable version and convert it back into the definition version of E-ORM.

It is assumed that the transforms described in section 3.5 are manually created. It would be a lot easier to create automatic and easily extensible transforms if we could automatically generate XML and relational database models from E-ORM. More work should be done in this area.

Finally, it is assumed that the information in the common model can be easily viewed and manipulated. This is usually the selling point of any version of family history software. However, the views are manually crafted. It is more difficult to create interfaces for user-defined information and especially for externally-defined data, even if we have the corresponding conceptual models. It would be preferable to automatically generate interfaces from model descriptions. It would also be helpful if, when sending information from a source to a destination, we could send GUI information for viewing the extended data.

6. REFERENCES

[1] (2008, November) Extensible Markup Language(XML) 1.0 (Fifth Edition) [Online]. Available: http://www.w3.org/TR/REC-xml/.

[2] (2009, October) Personal Ancestral File [Online]. Available: http://www.familysearch.org/eng/paf/.

[3] (2012, October) Family Tree Maker [Online]. Available: http://www.familytreemaker.com/.

[4] (2011, July) Unified Modeling Language [Online]. Available: http://www.uml.org/.

[5] (1999, October) The GEDCOM Standard, Draft Release 5.5.1 [Online]. Available: http://www.phpgedview.net/ged551-5.pdf.

[6] (2011, December) The Master Genealogist, Version 8 [Online]. Available: http://www.whollygenes.com/Merchant2/merchant.mvc?screen=TMG.

[7] Genbridge [Online]. Available: http://www.whollygenes.com/Merchant2/merchant.mvc?Screen=PROD&Product_Code=GENBRDG-DL.

[8] Wholly Genes Software, 5144 Flowertuft Court, Columbia, Maryland 21044 [Online]. Available: http://www.whollygenes.com/.

[9] (2000, May) Genealogical Data Model Phase 1 [Online]. Available: http://xml.coverpages.org/GENTECH-DataModelV11.pdf

[10] (2000, May) Genealogical Data Model Phase 1, p. 12 [Online]. Available: http://xml.coverpages.org/GENTECH-DataModelV11.pdf

[11] (November, 2005) Jones, Tamera, GEDCOM Alternatives [Online]. Available: http://www.tamurajones.net/GEDCOMAlternatives.xhtml.

[12] Data Models, Build A Better GEDCOM Project [Online] Available: http://bettergedcom.wikispaces.com/Data+Models

[13] D. W. Embley, B. Kurtz, S. N. Woodfield, *Object-oriented Systems Analysis: A Model-Driven Approach*. New York: Yourdon Press, 1990.

[14] T. Halpin, Object Role Modeling [Online]. Available: http://www.orm.net/.