A Superstructure for Models of Quality

David W. Embley[1], Stephen W. Liddle[2], and Scott N. Woodfield[1]

[1] Department of Computer Science
[2] Information Systems Department
Brigham Young University, Provo, Utah 84602, USA
embley@cs.byu.edu, liddle@byu.edu, woodfiel@cs.byu.edu

Abstract. With additional quality modeling features added to conceptual models, computers could play a greater role in ensuring a higher level of quality in the information we model. For information-discovery applications, these additional conceptual modeling features should automatically accommodate certainty and conflicting information, support evidence-based research, automate collaboration, and provide research guidance. To address these issues, we propose a superstructure that adds four additional abstraction layers to typical conceptual models: a knowledge layer, an evidence layer, a communication layer, and an action layer. We show by a running example the benefits these abstraction layers provide for increasing the quality of the information being modeled.

Keywords: conceptual modeling continuum, abstraction hierarchy for conceptual modeling, evidence-based conceptual modeling, information-discovery applications, automated collaboration, research guidance, and uncertainty.

1 Introduction

Improving quality is a means to an end—not an end in and of itself. To improve quality, we need to know how it relates to our end goal. For systems whose end goal is the discovery of correct information in the face of uncertainty, conceptual models can play two quality roles: They can (1) reliably model the information discovered and (2) reliably model the meta-information that supports reasoning and communication about the discovered information and guides users in resolving uncertainties.

Suppose, for example, that we wish to have reliable information about human intergenerational ancestry to check hypotheses about inherited diseases or to trace our roots and better understand ourselves in our historical context. Then our end goal is to discover and then correctly populate a conceptualization of ancestral information. Ideally, the conceptual model itself needs to be able to (1) reflect reality, (2) allow for contradictory assertions, (3) organize evidence to support and refute assertions, (4) gather, disseminate, and reason about assertions, and (5) guide users in resolving contradictions and adjusting assertions with the end goal of having as many discovered assertions as possible, all of which are correct.

Our end goal for information-discovery applications is an accurate model of reality—a quality model. To achieve this end, we need a model of quality, which for our information-discovery application is one that allows us to represent the relevant meta-information about the reality we seek to model. The combination is the superstructure we propose. To improve the quality of models, we investigate models of quality. We base our investigation on Meadow's continuum [1].[3] Meadow's continuum provides ever-higher levels of conceptualization, taking raw symbols from data to meaning and even wisdom. The first four layers are concerned with quality models (the information being modeled), while the last three are concerned with model quality (the meta information used to ensure the quality of the information being modeled).

Meadow's seven conceptual layers, renamed to better fit our needs, are:

Quality of Models:

1. Symbols: to represent and record information.
2. Classes: to classify and provide semantics for symbols.
3. Information: to relate and constrain class instances.
4. Knowledge: to allow for conflicting assertions and supporting documentation.

Models of Quality:

5. Evidence: to organize supporting evidence and automate evidentiary logic.
6. Communication: to send and receive information without distortion or loss.
7. Action: to provide automated guidance for user behavior.

2 Conceptualization Superstructure

Figures 1 and 2 are conceptual-model diagrams, which we use to illustrate our superstructure. In the subsections below, we explain the components of these diagrams that pertain to each of the seven layers and how these components provide an increase in power over each preceding layer.

As an illustration of the superstructure, we show how the increase in power helps a doctor determine the pain management regimen for his patient, Laura Williams. The proper regime depends on whether there is a genetic disposition to opioid addiction. An important indicator is whether two or more of Laura's parents, grandparents, or great-grandparents were alcoholics. Laura knows her mother was not an alcoholic nor was anyone on her father's side, but she knows little about her maternal ancestors since her mother died when she was three.

2.1 Symbols

Conceptualization: The symbol layer has no conceptual-modeling features. It is merely a collection of symbols: text files and digitized documents and images. The cloud in Figure 1 represents a collection of symbols.

[3] Others (e.g., [2] and [3]) have proposed similar hierarchies of conceptualization.

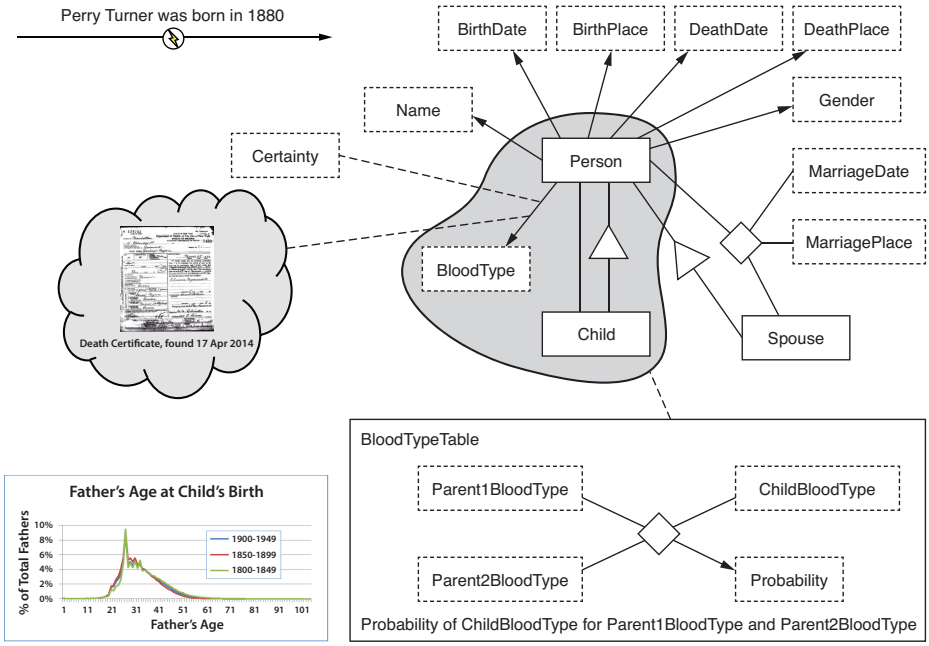

Fig. 1. Conceptualization of First Six Layers.

Example: Laura's maternal grandmother is Mary Turner. Laura has a transcribed copy of Mary's diary stored as a text file—symbols. She also has a scanned death certificate for Mary's husband, Bill Turner, stored as an image—symbols. Entries in the diary mention Bill's blood type as "A". The death certificate gives Bill's birth date, full name, and cause of death—cirrhosis of the liver, which implies that she has one ancestor who was alcoholic. She now needs help to determine whether either of Bill's parents were alcoholic.

Superstructure Motivation: At the symbol level, the computer is unable to to provide any assistance. While easily understood by humans, text documents and images are difficult for machines to organize and process. Further, with no semantic information, the symbol "A" appearing in the diary adjacent to the phrase "Blood Type" has no meaning.

2.2 Classes

At this level we are able to classify symbols and place them into classes—the named rectangles in Figure 1. The dashed-border rectangles are lexical classes whose members are symbols, often short strings of characters such as "AB-" for *BloodType*. The solid-border rectangles are nonlexical classes whose members are object identifiers. A so-called *data frame* [4] provides the semantics for each class. Figure 3 shows a data frame for the *BloodType* class. Note that it defines the expected external representation for blood types and context keywords that indicate that symbols such as "A-" are blood types, not grades or something

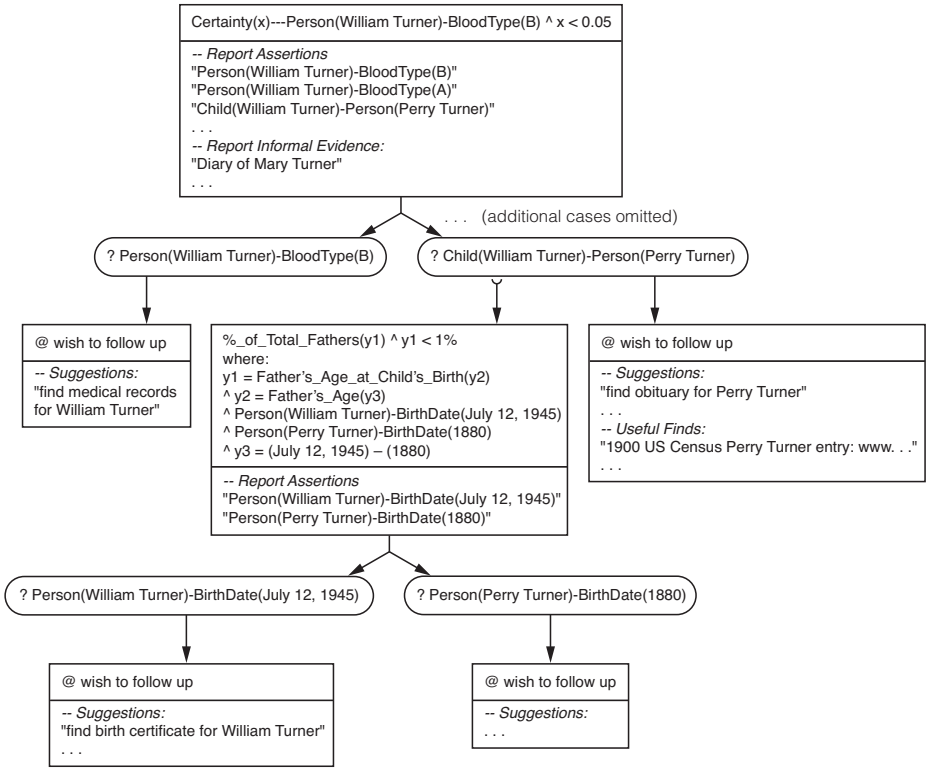

Fig. 2. Generated Action Conceptualization (partial).

else. Like abstract data types, data frames have I/O methods and other operations. Operators may also include expected phrase templates used to indicate the applicability of an operation. Data frames for nonlexical classes contain object-existence rules, which state that when an instance of some specified lexical class is recognized, an instance of a corresponding nonlexical class exists (e.g., when a *Name* instance is seen, a corresponding *Person* object exists).

Example: Some of Laura's information can now be classified. Using representation and context information in the *BloodType* data frame, a computer can identify the symbol "A" as a *BloodType*. Similarly, the computer can also extract the symbol "July 12, 1945" as a date and "William Turner" as a person's name.

Superstructure Motivation: Even with this information the computer can do little to assist Laura in determining who Bill's parents are and whether they are alcoholic. At the *Class Layer* the computer can classify information but cannot record how the instances in one class relate to instances in another class. We cannot associate the object identifier for "William Turner" in the class *Person* with his name, date of birth, and blood type.

```
BloodType
external representation: \b(A|A+|A−|B|B+|B−|AB|AB+|AB−|O|O+|O−)\b
context keywords: \b[Bb]lood\s[Tt]ype\b
input method: BloodTypeToString
output method: StringToBloodType
operator methods:
    CanDonateTo(x: BloodType, y:BloodType) returns (Boolean)
    external representation: \b[Cc]an\s.{0,30}{x}\sdonate\sto\s.{0,30}{y}
    end;
end;
```

Fig. 3. Data Frame for Blood Type.

2.3 Information

Conceptualization: All typical conceptual modeling features are included at this level: classes, relationships, generalization/specialization, and cardinality constraints. In Figure 1, lines connecting object-set classes represent relationship sets (e.g., *Person-BloodType*). Lines with triangles represent *isa* abstractions (e.g., *Child-isa-Person*). Decorations on lines express cardinality constraints (e.g., the arrowhead on *Person-BloodType* designates a functional constraint, limiting a person to at most one blood type). Furthermore, at this level, we retain and expand data-frame semantics to full extraction ontologies [4], which allows a computer to "read" text—extract information from text and populate conceptualizations. When the *Person* object-existence rules fire, for example, not only is an object identifier added to *Person* and the string added to *Name*, but a relationship between the instances is also added to the *Person-Name* relationship set. Similarly, with extraction-ontology recognizers for phrases that represent relationships, it is possible to automatically "read" information from semi-structured documents like an OCRed version of the death certificate.

Example: For our story, we now have Bill's blood type "A", birth date "July 12, 1945", and father "Perry Turner" with his birth date "1880". Laura is excited because further research shows that Perry was a teetotaler. Because she only has one ancestor with an apparent alcohol problem she concludes that she is probably not susceptible to opioid addiction. Unfortunately, what the computer "says" with its conclusion-based model at this level of the conceptualization superstructure could be invalid—a potential life threatening mistake.

Superstructure Motivation: At the information layer, models must be valid—all constraints must hold—but in historical research we often have conflicting information. Further, the computer has no principled way to associate explanations and justifications with assertions.

2.4 Knowledge

Conceptualization: This layer allows for invalid models, unstructured meta-information for justification, and soft constraints for expressing likelihood. In Figure 1, the dashed line connecting the cloud with the *Person-BloodType* relationship set designates a set of links between relationship instances and unstructured

Person(x_1)-BloodType(x_2), Person(x_3)-BloodType(x_4), Person(x_5)-BloodType(x_6),
Child(x_1)-Person(x_3), Child(x_1)-Person(x_4),
Probability(x_7)_of_ChildBloodType(x_2)_for_Parent1BloodType(x_4)_and_Parent2BloodType(x_6),
⇒ Certainty(x_7)---Person(x_1)-BloodType(x_2)

Fig. 4. Inference Rule for Obtaining Certainty for Blood Type.

meta-information. Distributions like the "Father's Age at Child's Birth" in Figure 1 represent soft constraints. "Violation" of a soft constraint does not make a model instance invalid but suggests some assertions are unlikely. Since the *Knowlege Layer* permits conflicts, we can define valid models without weakening constraints. We can, for instance, record two birth mothers for a child without having to reduce the quality of the model, which can still declare that there is at most one mother for a child.

Example: With birth dates in 1880 and 1945, Perry Turner was apparently 65 years old when Bill was born, and the computer can use the distribution "Father's Age At Child's Birth" to suggest to Laura the unlikelihood that Perry is Bill's father. Armed with this computer-provided insight, Laura digs deeper and discovers a medical form stating that Bill has blood type "B". Since we can violate constraints, we can add this assertion while retaining the assertion that Bill's blood type is "A". Furthermore, we can attach a scanned image of the medical form as justification that Bill has blood type "B".

Superstructure Motivation: It would be nice if the computer could assist Laura in resolving these seeming contradictions. Unfortunately, since the evidence is informal, the computer cannot reason about it.

2.5 Evidence

Conceptualization: This layer of our superstructure allows justifying meta-information to be formally organized as a conceptual-model instance. In Figure 1, the *BloodTypeTable* is a populated conceptual-model instance yielding the probability of a child's blood type being x when its parents have blood types y and z. In Figure 1, a dashed line connects the blood-type table to a conceptual-model subcomponent in which the information to compute the probability resides. Since the justifying meta-information is formal, the computer can reason with it as the datalog-like rule in Figure 4 shows. The inference rule in Figure 4 yields for each relationship instance in *Person-BloodType* its *Certainty* as recorded in the *BloodType* table. These results can be added as a formal *Certainty* justification as Figure 1 shows.

Example: Laura looks for and finds a medical record documenting that Perry Turner and his wife have blood type "A", but the computer tells her that there is a 0% chance that their son Bill has blood type "B". Thus something is wrong.

Superstructure Motivation: Information becomes more valuable and better validated when we share it with others. We can easily do this by encoding the information in human readable form and sending it. The human receiver must then read, understand, and manually store it in their own machine. It would

model structure:
Probability[1:*] of ChildBloodType[1:*] for Parent1BloodType[1:*] and Parent2BloodType[1:*];
ChildBloodType, Parent1BloodType, Parent2BoodType --> Probability;
end;
model instance:
Probability(.9375) of ChildBloodType(A) for Parent1BloodType(A) and Parent2BloodType(A);
Probability(.0625) of ChildBloodType(O) for Parent1BloodType(A) and Parent2BloodType(A);
...
end;

Fig. 5. Model Structure and Instance Data.

be better if the source machine could directly communicate with the receiving machine.

2.6 Communication

Conceptualization: To communicate on its own, the computer must be able both to write/send and receive/read information. In Figure 1 the arrow with a lightning bolt in the center shows the computer receiving a message from some unknown source. Our proposed superstructure has three forms of communication.

1. When a statement of fact is sent, a previously agreed-upon format is used to decode the message—a common form of communication that only works if the sender and receiver agree on the format and semantics.
2. With extraction ontologies [4] the receiver can read, decode, and store facts; how well this works depends on how well the extraction ontology's recognizers can read.
3. If the sender and receiver are using the same conceptual model, both the statements of fact and the corresponding sub-model can be shared. In Figure 5 we demonstrate this using a model-equivalent programming language [5]. The textual model structure represents the *BloodTypeTable* in Figure 1, including its functional constraint, and the model-instance statements populate the model structure.

Example: In our example Laura remembers that Perry Turner may have been too old to be William Turner's father. She asks William Turner's sister, who is still alive, when Perry Turner was born. The reply comes as the message in Figure 1, "Perry Turner was born in 1880", which can be read using extraction-ontology technology and which confirms the fact that Perry was born in 1880. The question Laura asked was based on the assumption that Perry Turner was Bill's father. A better question would have been, "Is Perry Bill's father?"

Superstructure Motivation: Although helped by the computer to some extent, Laura has been on her own to decide what to do. Could a computer have provided her with the relevant information and, more importantly, could it have guided her reasoning and search for additional relevant information?

2.7 Action

Conceptualization: At this level of abstraction, we add object-behavior modeling along with object-interaction modeling and object-relationship modeling [6]. We represent object-behavior models with multi-threaded, enhanced state/transition diagrams as Figure 2 shows. Interestingly, we observe here that for fact-discovery applications as in our running example, useful object-behavior models can be generated automatically, rather than being specified by an expert in the field.

Example: Having reasoned (based on the inference rule in Figure 4) that the *Certainty* of *Person(William Turner)-BloodType(B)* is highly unlikely, the computer generates the object-behavior model in Figure 2 as follows:

- The trigger in the first transition in Figure 2 is the conclusion of the inference rule in Figure 4. In the action part of the first transition the *Reported Assertions* are the antecedent predicates of the inference rule with their variables bound to the instances for which the 0% certainty was derived, and the *Reported Informal Evidence* is the informal information in the cloud linked to these instances.
- Since if the conclusion is wrong, one or more of the antecedent assertions must be wrong, the computer can generate the subsequent states of the first transition as hypotheses to be considered. (Figure 2 shows only two of the six subsequent hypothesis states.)
- The *@ wish to follow-up* transitions depend on Laura's deciding to activate them. The provided *Suggestions* in these transitions are automatically generated if the conceptual modeler has included them in the formal evidence model and linked them as meta-information to the hypothesis relationship sets in question. The provided *Useful Finds* also come automatically if, knowing what to look for from the *Suggestions* list, the computer can send a query to a service such as `FamilySearch.org` or `Ancestry.com` and retrieve an image of a document that the computer suggests should be sought.
- Finally, when a constraint applies, as does the soft constraint in Figure 1 for the *? Child(William Turner)-Person(Perry Turner)* hypothesis in Figure 2, the computer generates a trigger that fires when the condition holds. The transition's action is to report assertions and informal evidence and then generate subsequent hypothesis states as explained previously.

Being guided by the computer-generated research plan, Laura follows up on the hypothesis that William Turner is not Perry Turner's child by searching for obituaries. She finds one for Perry and one for William Turner. Neither suggested a *Person-Child* relationship between the two. However, Laura also finds an obituary of a Steven Turner in which a William Turner was the son of Steven, and Steven was the son of Perry. The article stated that Steven Turner, a well known alcoholic, had died in a car accident. William Turner had been raised by his grandparents! Now Laura has found that two ancestors had suffered from apparent alcohol addictions, and she and her doctor now believe she may have a genetic disposition for opioid addiction.

Feature	Status	Feature	Status
Comprehensive conceptual models	yes	Data frames	yes
Extraction ontologies	yes	Contradictory information	yes
Probability/likelihood data	no	"Soft" constraints	no
Informal evidence	partial	Formal evidence	partial
Communication support	yes	Action specification generation	no
Action specification execution	yes	Component integration	partial

Table 1. Matrix of Implementation Status

3 Implementation Status

Our running example is reminiscent of the medical appointment scheduling example of Berners-Lee et al. when they first proposed the Semantic Web [7]. Indeed, our application fits well with the Semantic Web vision. However, as with the Semantic Web, even though a great deal has already been accomplished, there is still much to be done. The lower layers in our proposed superstructure—Symbols, Classes, Information, and Knowledge—tend to be thoroughly implemented, while the upper layers—Evidence, Communication, and Action—show a good amount of progress but need more work. Over the years we have implemented numerous software projects that support the proposed superstructure. Most of these projects fit within a Java-based graphical workbench that runs as a desktop application. Table 1 summarizes our implementation status.

We have implemented a comprehensive object-oriented conceptual model (OSM), data frames, and an ontology-based extraction system, OntoES, that works with high precision and recall in ontologically narrow domains [4]. Our workbench supports editing both schema and data-layer information within OSM, and we are able to communicate this information both graphically and textually (see Figures 1 through 5) and in an interchange-friendly XML format we call OSM-X. We can both manually and automatically (e.g. via OntoES) annotate assertions based on unstructured source documents, and we capture full linkage information between assertions and their supporting sources. This forms the basis for the evidence layer in our superstructure. Though our front-end tools do not yet support this, our OSM-X storage format allows for arbitrary user comments about annotations and assertions. We have implemented tools to execute state/transition diagrams (e.g. Figure 2): (1) automatically when fully formal triggers and actions are present [5] and (2) synergistically with the user's help for cases where triggers and actions may be informal or semi-formal [8]. We have not yet implemented the automatic generation of state/transition diagrams as explained in Section 2.7. While we have experimented with various types of uncertain data, we have not yet implemented "soft" constraints in our current toolset. Because we have a tool that converts our proprietary OSM-X interchange format to RDF/OWL, we are able to use Semantic Web reasoning tools like Jena to perform various kinds of automated inferencing [9] (e.g. for the rule in Figure 4). In addition to implementing additional features, we also need to better integrate some of the components in our toolset.

4 Concluding Remarks

Our superstructure includes both "models of quality" and "quality of models." The first four layers of the superstructure address models of quality: we can model our observations of the real world accurately with hard and soft constraints and ADT data frames, and we can allow our models to accept uncertain and conflicting assertions as we discover them. The last three layers address the quality of models: we can analyze the quality of the assertions in our models and improve them. The evidence layer enables computer-assisted reasoning and finding hard and soft constraint violations; the communication layer supports automated information collection both about the assertions and the evidence supporting the assertions; and the action layer supports improving the quality of the data in our conceptualizations and our search for truth under the guidance of an automated expert. Although developing our superstructure is a massive undertaking, we are well along in its implementation.

Meadow adds a last layer that he calls "Wisdom," and which we interpret to mean the proper application of knowledge based on truth (via evidence), communication, and action. It is our hope that the superstructure we propose here can more effectively enable us to proceed wisely—from our story, for example, to avoid placing Laura in a potentially life-threatening medical regimen.

References

1. C.T. Meadow. *Text Information Retrieval Systems.* Academic Press, Orlando, Florida, 1992.
2. C. Zins. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4), February 2007.
3. J. Rowley. The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science*, 33, 2007.
4. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, 1999.
5. S.W. Liddle. *Object-Oriented Systems Implementation: A Model-Equivalent Approach.* PhD thesis, Department of Computer Science, Brigham Young University, Provo, Utah, June 1995.
6. D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.
7. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 36(25):34–43, May 2001.
8. R.B. Jackson. *Object-Oriented Requirements Specification: A Model, A Tool and a Technique.* PhD thesis, Brigham Young University, 1994.
9. J.S. Park and D.W. Embley. Extracting and organizing facts of interest from ocred historical documents. In *Proceedings of the 13th Annual Family History Technology Workshop*, Salt Lake City, Utah, USA, March 2013.