Prototype Generation from Ontology Charts

Iman Poernomo
King’s College London
Computer Science Department
iman.poernomo@kcl.ac.uk

George Tsaramirsis
Accenture
UK Delivery Centre
georgios.tsaramirsis@accenture.com

Abstract

Semantic Analysis is a business analysis method designed to capture user requirements and represent them as text or in a graph that is called Ontology Chart. Ontology Charts can be mapped one to one to temporal Database schemas and Class diagrams that developers can use to produce software systems. In principle, these systems can be extended without the need to modify the database schema of the legacy system. However Semantic Analysis is not widely used hence there is a lack of experts in the field and no design patterns are available. These make it difficult for the analysts to pass the organizational knowledge to the engineers. This study will suggest an automated technique that can produce a prototype from an Ontology Chart. The prototype will also include both front and back end. The software support that is required to achieve the automation of these techniques is also presented in this research.

1. Introduction

Driven by the belief that organisations should be treated as information systems and not as computer systems, introduced Methods for Eliciting, Analysing and Specifying User’s Requirements (MEASUR) [5]. The MEASUR methods appear to have a number of potential benefits for organisations [2]. One of those methods, the Semantic Analysis (SAM) is dealing with information modelling [4]. The method works as follows: Initially the problem must be defined by domain experts and passed to the project analyst(s). The next step is the generation of candidate affordances. This step will generate a list of semantic units that may be included in the schema. The candidate grouping follows where some of the semantic units that will appear in the schema are placed in simple groups. Finally the groups will be integrated together into an ontology chart [3]. The ontology chart can be mapped directly to database schema. This schema appeared to be able to support future changes without removing anything in the structure of the schema, just by adding a new table [8]. This is cheaper because it just require new modules, where if something had to be removed or renamed from the schema then, changes had to be made to the front end as well as the links. Since usually the source code is not available, reverse engineering is required in the code. This is very expensive and some times more expensive than just doing everything from scratch.

The method however is not widely used since not many people are familiar with it and those who are found it difficult to use. This is assumed to be due to high complexity of the schema produced by SAM and the weakness of both analysts and developers to map it to an actual system.

The scope of this research is not to detail how semantic analysis can be conducted but to propose a method can mine enough information from the Ontology Chart so it can produce a prototype. The prototype will consist of a Graphical User Interface (GUI), linked to a database schema, capable of maintaining all the benefits that SAM promises.

2. Ontology Charts

Ontology Charts consists of nodes linked together. The graphical position of these nodes linked together with a line represent ontological dependency relations directed from left to right. The affordance at the left site is ontologically dependant to the affordance(s) of the right site [6]. If an affordance is finished, then all its dependants will finish as well. If a financial institution stop to exist all financial product that it is offering will also stop to exist. Hence no application for any of these financial products that do not exist will make sense. Apart from the ‘Root’ node, all the others must have either one or two antecedents. SAM is limited to binary associations.
The ontology chart in Fig. 1 is a generic model for loan applications. The problem statement that describes is that a person can apply for a loan product. The person has first name, last name and email. The loan product has a possible borrowed amount, lending fee, possible loan purpose and length of repayment. When a person applies for a loan, the person needs to specify the purpose of the loan, the amount and the number and frequency of repayments. All these must be with the limits of target loan product. For example if the maximum amount that a specific loan product offer is $10000, then the customer can not apply for a higher amount. Finally a person that works for the financial institution will consider the application and either approve it or not. Every node from the ontology chart has in build, start and finish times so the system is temporal, maintains history of changes and supports non destructive updates [7].

Figure 1. Ontology Chart for a loan application

Every node in the ontology chart is called an affordance. Affordances are divided to Agents (actors), communication acts, entities, determiners and other affordances [2]. The agents are affordances that can take legal responsibility and can afford an action. 'Person' is an agent since; a person can apply for a loan. Entities, on the other hand can not take legal responsibility and can not act, neither can represents a relationship. Loan Product is an example of an entity. Determiners are a special class of affordances that are used to maintain data of their antecedent. 'First Name' for example is a determiner of the 'Person'. Communication Acts are formal or informal, social communications of an agent talking about something. 'Applies' for example is a communication act initiated by the agent 'person' about a 'loan product'. The other affordances term is used to include all these affordances that do not fall in one of the above categories within a given domain and problem statement and usually they are relationships between affordances. In this paper the cycle shape is used to represent the agents, the square for other affordances, the rounded rectangle to represent entities, the hash character (#) to represent determiners and the exclamation character (!) to repent communication acts.

3. Related work

Ontology Charts are currently used for business analysis and can produce database schemas and classes. The most recent and complete mapping of an ontology chart to class diagram is the transformation process proposed by Ades [1]. In that proposal the root is discarded and the rest of the affordances are mapped one to one to classes with start and finish time attributes. This structure is extendable and immune to malignant changes, since additional functionality can be added by adding new modules without the need to change the legacy system. However, even when a new class or a new table in the database are added (if it is mapped to a database instead), there is still some configuration and deployment overhead. That proposal does not mention anything about front end.

Earlier proposal to auto generate an information system [9] proposing mapping an ontology chart to a database schema suggested that a table needs to be created for every node of the ontology chart except the root node. The table names are the same as node’s label for each node and fields included start time, finish time, start record time, finish record time, antecedent1, antecedent2 and value. The start and finish time is used to capture the action time of the affordance. For example the start time of a person in the EU is the date and time when is born. The record start and finish times is when it is recorded in the information system. If a person that is born 20 years applies for a loan today the start time of that person will be 20 years ago and the record start time will be today. The antecedent1 is used as a foreign key that points to the antecedent of the affordance. If the antecedent is the root then both antecedent1 and antecedent2 keys will be null. Finally value is used to capture data. For example a possible value of the determiner First Name of person can be 'George'. Cascade updates are used to ensure that the start time of a dependant can not be earlier than its antecedent or its start authority and its finish time later than its antecedent. Also the finish of an antecedent must mean finish of all its dependants.
so the finish time of all its dependants that have null as finish time will be updated by the time that their antecedent had finished. That proposal also includes a process for auto generating an interface and linking it with the back end. However that process is auto generating individual screens with no link with each other and it was expecting the developer to manually assemble the different screens. This paper builds on the weaknesses of previous approaches and suggests a more complete and solid solution.

4. Mapping to the database

This section will introduce a database structure, capable of mapping any ontology chart, therefore it propose a mapping of ontology charts to an actual database instead of a database schema. Given that all the entities follow the structure of the template, an entity with the structure of the surrogate table, described in Liu’s book [2] could be used as well to store all the universals and particulars of the ontology chart.

**Figure 2. Surrogate Template**

In Fig 2 the ‘Sort’ can be either ‘Universal’ or ‘Particular’, the Label is the name of the node, e.g. ‘Person’ and the type can be agent, entity, other affordance, communication act or determiner. However from design point of view is good to differentiate the Structure of the system from the data of the system. That is why this paper proposes a four table structure, one table for Universals and one for Particulars, a table for OCL statements that will be used to capture the business norms and a separate table that will store the Label of Universals. The last allow Universals to have many Labels in different languages and scripts enabling the system to be language independent. This proposed schema can be seen in Fig 3.

Ontology charts contain only universals and they will be mapped in the universals table. Table 1 shows the mapping of the Ontology chart from Fig 1. Start and Finish times were included as they deal will the concept of universals. For example the Start Time of the universal ‘Person’ means when the concept of person started to exist. Usually this is lost in history and out the scope of the system. However it is useful for affordances of type ‘entity’ as the object or concept that it is representing in the information system may stop to exist. For example consider the case of email. If after 100 years emails are replaced by an alternative and the concept of email is lost then the universal email should finish. In a similar way start time can be used for scheduling. For example assume that emails need to be captured after a certain date. Then the start time of the universal email can be in the future hence it will be active only after a certain date. Finally another use of start and finish times at the universal level, is for switching on/off branches of the information system. If the universal employment is switched off (finish) then the universal ‘Consider’ will be switched off (finish) as well. This however will not stop the rest of the system from operating. People will still be able to apply for loans. There will be no consideration of these applications until the universal ‘consider’ is back, but still no loan applications will be lost. For simplicity reasons only three determiners were included in the mapping. The column Label in table 1 was placed there for visibility reason, but the labels will be stored in table ‘Label’.

**Figure 3. Proposed Structure**

Table 3: Mapping to Universals table
The last two rows record the word Person in Greek. Row 10 is the Greek word for Person written using Latin script. Row 11 is the word Person in Greek language written in Greek script. This allows the system to use any representation the word person in any recorded by the system language and script when execute queries or commands. Table OCL is used to capture the business norms written in OCL. Table 3 shows an example of a business norm.

The Particulars table is used to store data corresponding to universals. Every particular corresponds to one universal and a universal can have many particulars. The following scenario will be used to populate the particulars table.

The Financial Institution ‘good bank’ offers a loan product called ‘good loan’. The ‘good loan’ can lend up to $10000. George Smith applies for $500 ‘good loan’ loan product. John Smith, an employee of ‘good bank’ considers the application and approves it.

5. Generating the Graphical User Interface

The method provides techniques for aiding the designer to produce a front end. The result front end is html and JavaScript code associated with each particular. Any technology could have been used instead of JavaScript which was selected just to prove the concept.

Fig. 4 shows a Person Appling for a loan.

Figure 4. Applicant Screen

The system identifies all the directly linked affordances with that person. All the non determiners affordances directly linked with the current affordance will be displayed as tabs on the top of the screen. In this case this client can access the ‘applies’ affordance. The ‘Applies’ hyperlink will move the application screen. If this person was linked with the ‘employment’ affordance would have another tab next to ‘applies’ called ‘employment’. Similar an employee has just an ‘employment’ tab. Fig. 5 shows this.

Figure 5. Employee Screen

The last column of table 4 is present just for visibility purposed and is not present in the system. This section presented a flexible structure that can map any ontology chart. The structure builds on the surrogate table by differentiating between universals and particulars, allow each universal to have more than one labels in different languages and scripts and include a table for commands.
sal 'Employment' was not null. That would have been the reflection of switching off the employment Branch of the system. The determiners were presented in the same form displaying the last entered value. Other values for each determiners or modification of the existing values can be done by clicking the 'more' link. Fig. 6 show the determiner screen.

![Figure 6. Determiner Screen](image)

The above figure, selects all the 'FirstName' records for the employee with person id 2. If that employee had more names, they would appear in different rows.

If a person clicks on 'Applies' tab the screen in Fig. 7 will appear.

![Figure 7. Applies Screen](image)

This section showed how GUI can be auto generated from Ontology Charts. This GUI is linked with the structure described in the previous paragraph. This is a prototype and not the final system. This research is not trying to replace programmers but to help programmers and designers to produce a prototype that can clarify requirements. The programmers can either continue to build on this prototype and produce the final system or just use it to understand requirements.

6. Semantic Analysis Toolkit

The above motioned approaches for generating prototypes were implemented in an open source software tool developed at 2006 by the Authors, called Semantic Analysis Toolkit (SAT). SAT provide a graphical interface implemented in Java Swing that allow that designer to drag and drop affordances and form an Ontology Charts. The tool then can evaluate the chart against the basic rules of semantic analysis spotting basic mistakes and produce a prototype. Extending an existing prototype is also possible. The tool can read the 'Universal' table of an already existing system and extend its schema by adding new affordances. This will not affect the legacy system. The tool can also produce an XMI file that can be imported by MDA tools, if the users decide to use the MDA approach instead. The tool can be downloaded from [http://sourceforge.net/projects/semantool].

7. Conclusions

This paper demonstrated how a prototype system can be generated from an Ontology Chart. The paper proposed a flexible database structure originating from extending the surrogate template structure. The new
structure is capable of storing any ontology chart and allows each universal to have more than one label in different languages and scripts, producing language independent systems. Also the systems are flexible enough to allow partly switch off of their functionality without effecting the rest of the system. The GUI automatically reflects changes to the database. These methods were implemented in open source software called SAT.

There are a lot of areas of improvement for the proposed methods. Some of more interesting areas include improving the effect of the business logic and its effect on the system, improving the wording used in the ontology charts so it servers both the purpose of readability of the ontology chart and more meaningful interface.

8. Acknowledgments

This work would have not been possible without the support of Accenture UK that allowed time off for the completion of this paper. Also the Accenture colleagues and friends must be thanked for their positive attitude that was a driving force for this paper.

References