A Running Example

In this section, we provide a running example on an Automated Teller Machine (ATM) as a running example.

## Initial Informal Requirements

*The system interacts with the customer to gather transaction information. The bank computer gets the transaction information from the system to verify an account and to process a transaction. Each bank may be processing transactions from several ATMs. The customer interacts with the ATM network via the ATM. The complexity of the client system should be simple. The complexity of the system should be sophisticated. The color of customer service should be more brightened. The color of client service should be dark. Increasing complexity of the product recommender should not cause decreasing download speed. Increasing complexity of client service should not cause decreasing maintainability of client service. The system shall provide access to anonymous users for unrestricted functions. The system shall provide privacy to users. Response time of the system should be high. Capacity of the system should be high. Increasing capacity of users should not cause decreasing system response time*.

## Apply Preprocessing

*In the preprocessing module first we label all requirements as follows:*

R1: The ATM interacts with the customer to gather transaction information.

R2: The bank computer gets the transaction information from the ATM to verify an account and to process a transaction.

…….

*R18: Increasing capacity of users should not cause decreasing system response time.*

### POS tagging

The approach applies POS tagging to all requirements as follows:

*R15: Increasing/VBG complexity/NN of/IN client/NN service/NN should/MD not/RB cause/VB de-creasing/JJ maintainability/NN ./.*

### Stop word removing/stemming

We apply stop word removing and stemming to all requirements. Here we present the outcome of R8 to R12.

*R8: complexity client\_system simple*

*R9: complexity system sophisticate*

*R10: color customer\_service bright*

*R11: color client\_service dark*

*R12: Increase complexity product\_recommender not cause decrease download\_speed*

### Removing coordination ambiguity

We identify coordination ambiguity in requirement R2. It will split the requirement and make two separate requirements as:

*R2.1: The bank\_computer gets the transaction\_information from the ATM to verify an account.*

*R2.2: The bank\_computer gets the transaction\_information from the ATM to process a transaction.*

### Identify Anaphora Ambiguity

We identify anaphora ambiguity in requirement R5 as the requirement R5 contains the keyword *“it”* and “*them”*. The system resolve the ambiguity “*them”* automatically and rewrite the requirement R5 as

*R5: It must be very easy for customer to use the ATM.*

## SRS Ontology Formation

## Figure 10 shows the SRS ontology generated automatically by the approach based on fig. 1.

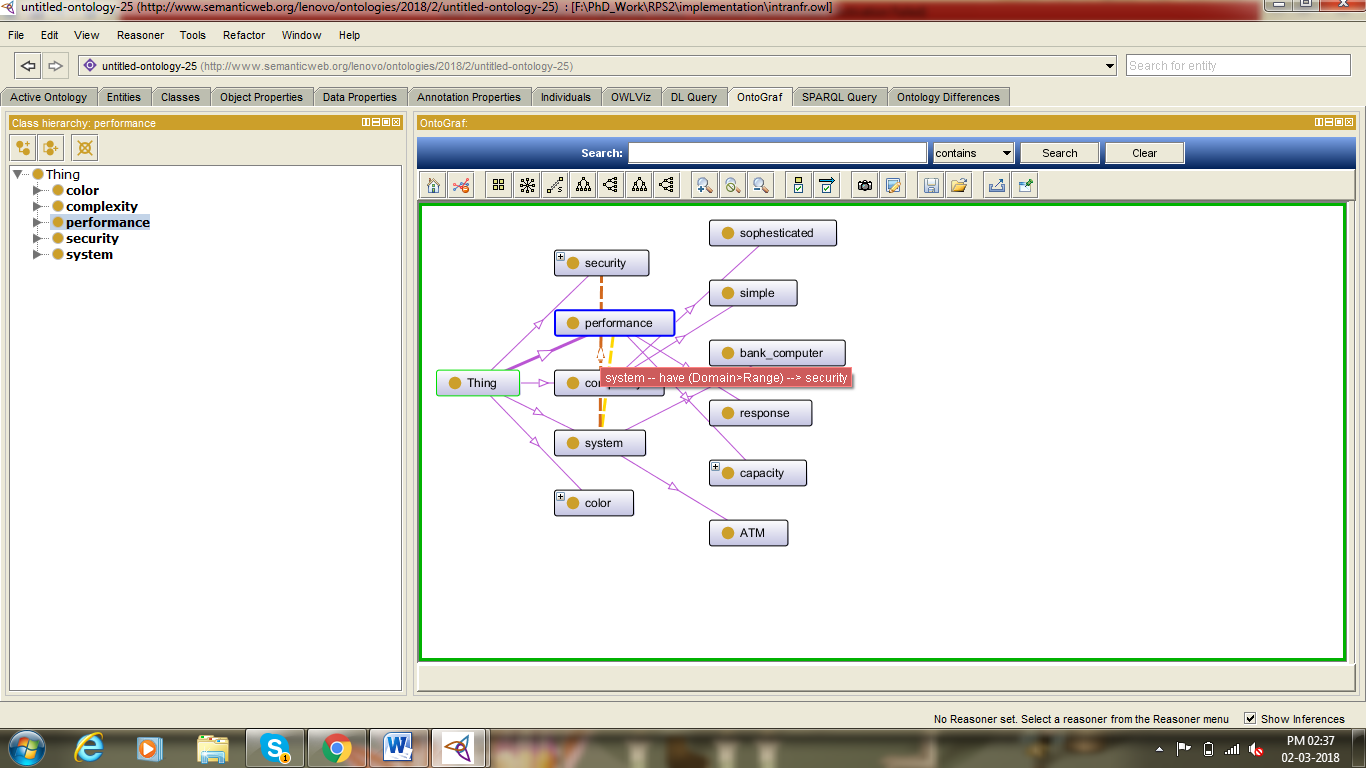


Fig. 1. SRS Ontology

## Classification of NFRs

Using pre-developed quality ontology (see fig. 11) based on the standard quality model, we extract the NFRs from the requirements as follows:

*NFR1: The complexity of the client system should be simple. NFR2: The complexity of the system should be sophisticated. NFR3: The color of customer service should be more brightened.*

*NFR4: The color of client service should be dark.*

*NFR5: Increasing complexity of the product recommender should not cause decreasing download speed.*

*NFR6: Increasing complexity of client service should not cause decreasing maintainability of client service.*

*NFR7: The system shall provide access to anonymous users for unrestricted functions.*

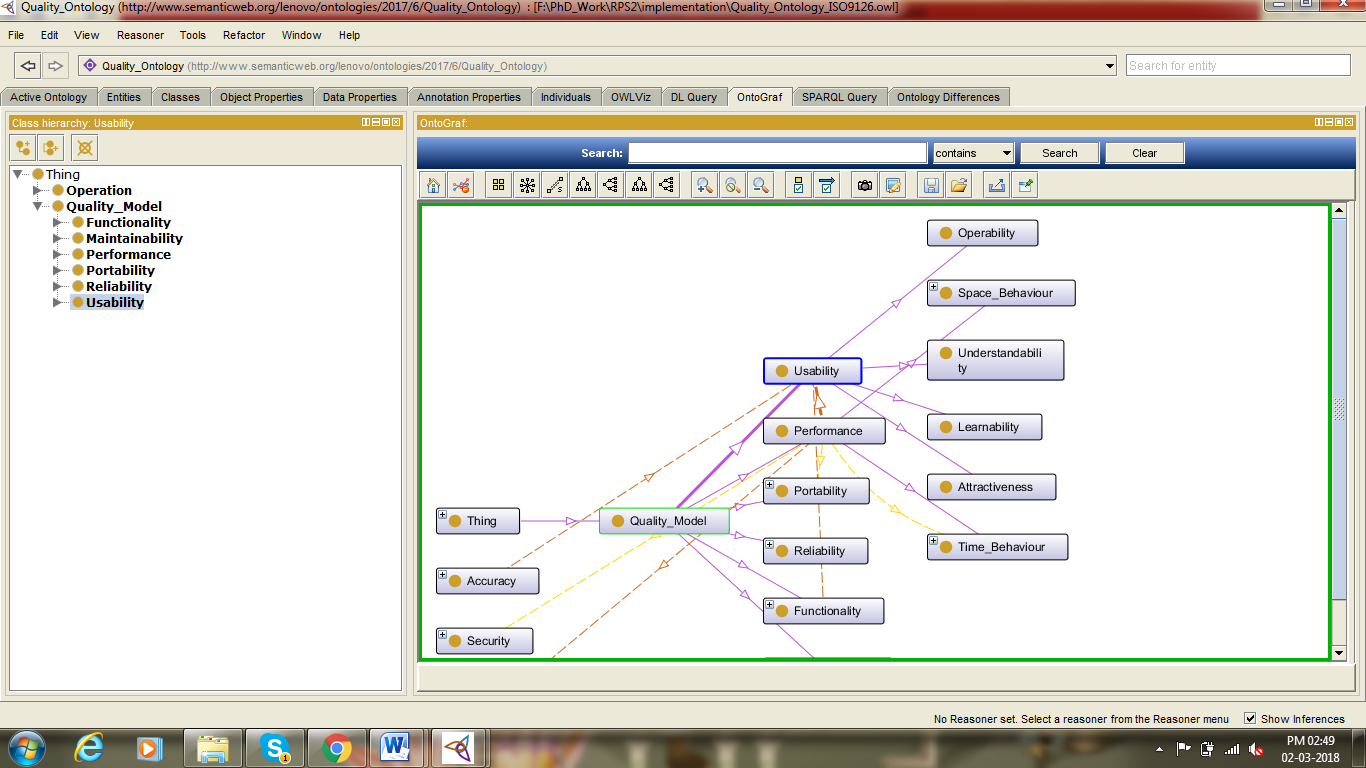
*NFR8: The system shall provide privacy to users.*

*NFR9: Response time of the system should be high.*

*NFR10: Capacity of the system should be high.*

*NFR11: Increasing capacity of users should not cause decreasing system response time.*

## As, we aim to identify intra-conflicts among NFRs further processing steps consider NFRs from the requirements document.



*Fig.2. Quality Model Ontology*

## Correlative NFR Clustering

### Find TF-IDF(Term Frequency – Inverse Document Frequency) of NFRs:

A set of NFR documents:

NFRs= {NFR1, NFR2…NFR11}

NFR1: (complexity, client\_system, simple)

NFR2: (complexity, system, sophisticate)

NFR3: (color, customer\_service, bright)

NFR4: (color, client\_service, dark)

NFR5 :( Increase, complexity, product\_recommender,

not, cause, decrease, download\_speed)

NFR6: (Increase, complexity, client\_service, not, cause, decrease, maintainability, client\_service)

NFR7: (system, provide, access, anonymous, user, unrestricted, function)

NFR8: (system, provide, privacy, user)

NFR9: (Response\_time, system, high)

NFR10: (Capacity, system, high)

NFR11: (Increase, capacity, user, not cause, decrease, system, response\_time)

Total terms: [complexity, client\_system, simple, system, sophisticate, color, customer\_service, bright, dark, increase, product\_recommender, not, cause, decrease, download\_speed, maintainability, provide, access, anonymous, user, unrestrict, function, privacy, response\_time, high, capacity]. For all the NFRs, we calculate the *TF* scores as defined in fig.7:

NFR1: (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

NFR2: (1, 0, 0, 1, 1,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

NFR3: (0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

NFR4: (0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

NFR5: (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)

NFR6: (1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,0, 1, 1,1,0, 1, 0, 0, 0, 0, 0, 0, 0, 0)

NFR7: (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0)

Some terms appear in two NFR documents, some appear only in one NFR document. The total number of NFR documents is *N* = 11 Therefore, the *IDF* values of the terms are:

complexity = *log2*(11/4) = 1.459, client\_system= *log2*(11/1) = 3.459, simple= *log2*(11/1)= 3.459, system= *log2*(11/6) =0.872

Now we multiply the *TF* scores by the *IDF* values of each term, obtaining the following matrix of NFR documents-by-terms (*TF-IDF)*: here we present the results for term

T1 to T7 for NFR1 to NFR 4 in table 3.

**Table 3**. TF-IDF Matrix

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
| NFR1 | 1.459 | 3.459 | 3.459 | 0 | 0 | 0 | 0 |
| NFR2 | 1.459 | 0 | 0 | 0.872 | 3.459 | 0 | 0 |
| NFR3 | 0 | 0 | 0 | 0 | 0 | 2.459 | 1.872 |
| NFR4 | 0 | 0 | 0 | 0 | 0 | 2.459 | 1.872 |

### Find TMI (Term Mutual Information) using cosine similarity and expressed as a matrix:

Using equestion 1 we update the TF values as:

NFR1: (7.6, 5.8, 6.3, 3.6, 4.3, 4.6, 3.6, 4.3, 4.6, 3.9, 4.1, 3.6, 3.6, 3.3, 4.2, 3.9, 0.9, 2.4, 1.8, 0.5, 0.9, 0.9, 0.8, 0.5, 2.9, 3.5)

Here, we use cosine similarity to measure *TMI* on ontology based vector space model and represent it as a matrix M.

### Find the Euclidean distance with the TMI

Using equation 2, we calculate (table 4) euclidean distance of NFR documents (partial results):

**Table 4.** Euclidean Distance using Ontology based Distance Measure

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | NFR1 | NFR2 | NFR3 | NFR4 | NFR5 | NFR6 |
| NFR1 | 1.0 | 1.79 | 0.72 | 0.69 | 1.76 | 1.76 |
| NFR2 | 1.79 | 1.0 | 0.61 | 0.72 | 1.79 | 1.79 |
| NFR3 | 0.72 | 0.31 | 1.0 | 0.61 | 0.64 | 1.64 |
| NFR4 | 0.69 | 0.72 | 0.61 | 1.0 | 0.55 | 1.55 |
| NFR5 | 1.76 | 1.79 | 0.65 | 0.55 | 1.0 | 1.79 |
| NFR6 | 1.76 | 1.79 | 1.79 | 1.79 | 1.79 | 1.0 |
| NFR7 | 1.72 | 1.72 | 0.57 | 0.84 | 0.76 | 0.84 |
| NFR8 | 1.72 | 1.72 | 0.26 | 0.84 | 0.75 | 0.84 |
| NFR9 | 1.62 | 1.72 | 0.90 | 0.71 | 0.75 | 0.55 |

We assign each NFR to its own cluster. If there are N NFRs, then there will be N clusters. We find the two closest pairs of clusters based on TMI calculated in table VI. For example, the closest two NFRs are NFR2, NFR5 and NFR6, with a distance of zero among them. They are merged into a single cluster (Cluster1: NFR2, NFR5, NFR6). The next closest NFRs are NFR1 and NFR2, so NFR1 is merged with the cluster that already contains NFR2, NFR5 and R6 (Cluster2: NFR1, NFR2, NFR5, NFR6). NFR3 and NFR4 are the next closest NFRs. They are merged into a single cluster (Cluster3: NFR3, NFR4). NFR7 and NFR8 are the next closest NFRs. They are merged into a single cluster (Cluster4: NFR7, NFR8). After grouping correlative NFRs, we identify the diversity of correlative NFRs by means of analyzing syntactic nature of NFRs. For example, cluster 2 contains following NFRs:

NFR1: The complexity of the client system should be simple.

NFR2: The complexity of the system should be sophisticated.

NFR5: Increasing complexity of the product recommender should not cause decreasing download speed.

NFR6: Increasing complexity of client service should not cause decreasing maintainability of client service.

Syntactic analysis separates cluster 2 as cluster 2.1:NFR1, NFR2; and cluster 2.2: NFR5, NFR6; based on homogeneity of NFRs. Then we identify term diversity as mentioned in fig. 8.

NFR1-> simple conflicts with NFR2->sophisticate;

NFR3-> bright conflicts with NFR4->dark;

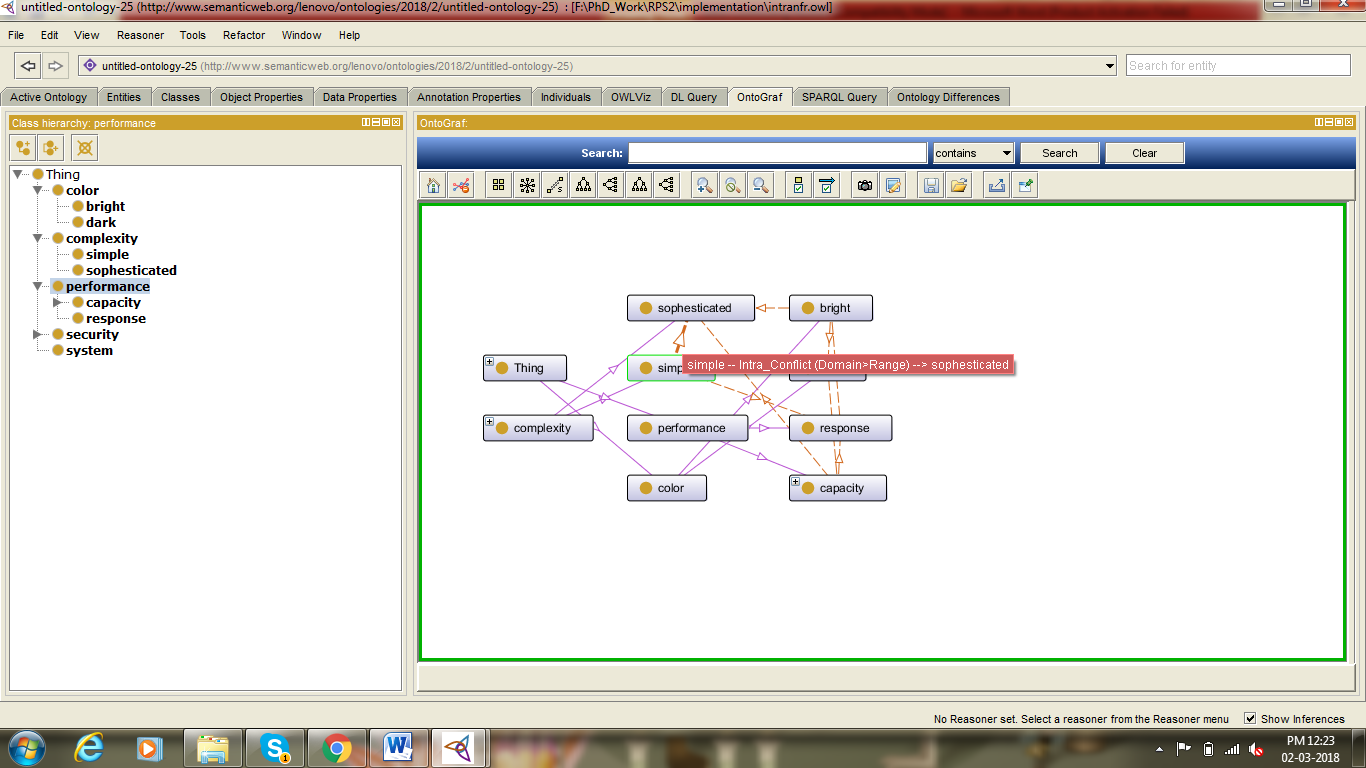
NFR7-> unrestrict conflicts with NFR8->privacy;

NFR5-> complexity conflicts wit2h NFR5->download\_speed;

NFR9-> capacity conflicts with NFR10->response\_time;

NFR11->capacity conflicts with NFR11->repsonse\_time.

Finally, we produce a list and ontological representation (fig. 3) of intra-conflicting NFRs.



***Fig. 3. Ontological Representation of Intra conflicting NFRs***