# AOV-graph Modeling to the HealthWatcher

Author: Lyrene Fernandes

Version: 2.0 Data: 23/September/2006

**Health Watcher's description**

The Health Watcher's goals are collect and control the complaints and notifications, and provide important information to the people about the Health System. The citizen can access the system through the internet or dialing 1520, making his/her complaint or asking information about the health services. In the event of a complaint, it will be registered on the system and addressed by a specific department which will be able to carry out the procedure and return an answer when the analysis is accomplished. This solution will be registered on the system, being available for queries. The product will be put to public usage in kiosks in several strategic points, on which the citizen itself will make its complaints and information requests.

**Index**



# 1. Activities

This modeling was made as follows: Firstly I read the HealthWatcher's documentation. After that, I modeled a first version of the v-graph using the structure of the documentation. Next, I specified this first version in XML and added some concerns reused from other case studies, such as Persistence, Security, Realiability and Usability.

After that I had to change them in order to attend the Health Watcher System, including and excluding some goals, softgoals and tasks. Therefore, I analyzed the relationships in V-graph and identified crosscutting relationships. Next, I specified these crosscutting relationships in XML, applied the composition mechanism and generated some views (automatically using XLST). These views were used in order to help me to identify mistakes or trade-offs, so the cycle Specify in XML→ Composition → Generate views → Analyze views happened many times. When I found a stable version I begun to format the documentation, but some changes in modeling again happened. In summary, these activities are listed below:

- Reading the HW's documentation (30 minutes)
- **Modeling the HW using V-graph – Identifying goals, softgoals and tasks, and the relationships (55 minutes writing from scratch plus 55 minutes rewriting in XML)**
- **Analyzing the relationships and identifying crosscutting relationships**
- **Specifying the crosscutting relationships (in XML)**
- **Investigating trade-offs and early aspects interaction**
- **Formatting the specification document and fixing some mistakes**

| **Date** | **Start** | **Stop** | **Total** | **Number of people** | **Activity** |
|---|---|---|---|---|---|
| 05/Sept/2006<br>06/Sept/2006 | 21:20<br>18:00 | 22:15<br>18:55 | 110 m | 1 | Modeling the HW using V-graph – Identifying goals, softgoals and tasks, and the relationships |
| 11/Sept/2006 | 10:00 | 10:45 | 45 m | 1 | Analyzing the relationships and identifying crosscutting relationships (the first version) |
| 11/Sept/2006 | 11:55 | 12:40 | 45 m | 1 | Specifying the crosscutting relationships (the first version, in XML) |
| 11/Sept/2006<br>12/Sept/2006 | 14:30<br>20:00 | 15:15<br>22:00 | 165 m | 1 | Investigating trade-offs and early aspects interaction |
| 12/Sept/2006<br>13/Sept/2006 | 22:00<br>9:30 | 23:55<br>11:00 | 205 m | 1 | Formatting the specification document and fixing some mistakes |

**Table 1 - LOG OF ACTIVITIES**

# 2. Glossary

**Complaint information**

A complaint information consists on these data: Date; Complaint data: description (mandatory) and observations (optional); Complainer data: name, street, complement, district, city, state/province, zip code, telephone number and e-mail (All these information are optional); Complaint situation (mandatory), which might be OPENED, SUSPENDED or CLOSED (in the event of a registration, its state must be OPENED);

After an analysis has been done, a complaint has also: technical analysis; analysis date; employee that made the analysis.

**Animal Complaint – DVA**

is related to: Animals apprehension; Control of vectors (rodents, scorpions, bats, etc.); Diseases related to mosquitos (dengue, filariose); Animals maltreatment.

In addition to complaint information an animal complaint has its specific ones. They are: Kind of animal (mandatory), amount of animals (mandatory), disturb date (mandatory); Disturb location data: street, complement, district, city, state/province, zip code and telephone number. All these information are optional.

**Food Complaint – DVISA**

is related to: cases where it is suspicious the ingestion of infected food.

In addition to complaint information an animal complaint has its specific ones. They are: Victim's name (mandatory); Victim's data: street, complement, district, city (or closest one), state/province, zip code and telephone number (All optional); Amount of people who ate the food, amount of sick people, amount of people who were sent to a hospital and amount of deceased people ( All mandatory); Location where the patients were treated, suspicious meal (All optional)

**Diverse Complaint - DVISA**

is related to: Cases related to several reasons, which are not mentioned above (restaurants with hygiene problems, leaking sewerage, suspicious water transporting trucks, etc.).

In addition to complaint information an animal complaint has its specific ones. They are: Age (mandatory), scholar level (optional), occupation (optional); Street, complement, district, city, state/province, zip code and telephone number of the closest location to the complaint location (All optional)

**Health unit**

consists on: unit code, unit description

**Specialty**

consists on: code and description

**Health unit / Specialty**

consists on: health unit and specialty

**Employee**

consists on: login, name and password

**Type of disease**

consists on: code, name, description, symptom and duration

**Symptom**

consists on: code and description

**Type of disease / Symptom**

consists on: type of disease and symptom

# 3. AOV-graph

This AOV-graph modeling consists on six goals models: Health watcher, Persistence, Realiability, Security, Usability and Performance. Some statistics about these goals models are in Table 2. Consider that the number of crosscutting relationships represents the number of specifications (of crosscutting relationship); in figures it is only possible see the number of

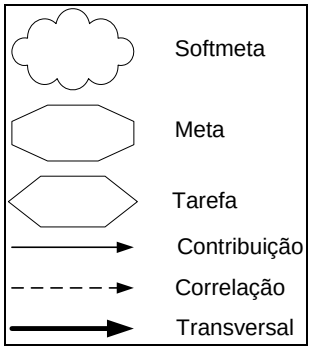
crosscutting links (before composition) and contributions (after composition). Therefore, 5 crosscutting relationships were represented by 11 links before composition, and after composition these 5 crosscutting relationships were represented by 38 contributions. Before composition, there were 84 relationships (crosscutting links + contributions) and after composition this quantity increased to 111 (contributions).

| Elements | Quantity before composition | Quantity after composition |
|---|---|---|
| Goals | 10 | 10 |
| Tasks | 70 | 77 |
| Softgoals | 11 | 11 |
| Crosscutting relationships | 5 | 5 |
| Links crosscutting | 11 | 38 |
| Contributions | 73+11=84 | 111 |
| Correlations | 11 | 11 |
| Types | 29 | 29 |
| Topics | 45 | 51 |
| Attributes | 0 | 0 |

**Table 2 - Statistics of AOV-graph modeling**

## 3.1 AOV-graph before composition

Figure 1 presents the AOV-graph modeling: each goal model is separated and the bold relationships represent the crosscutting links. Consider this legend to the models.



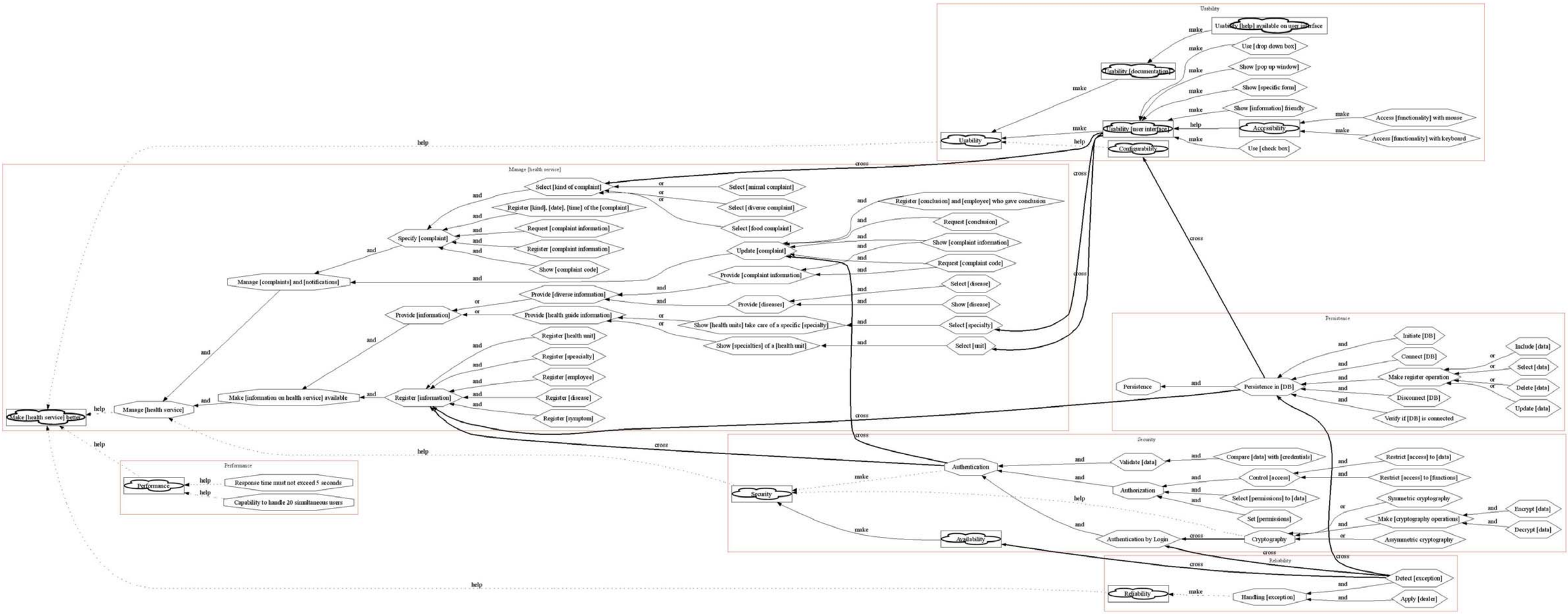



**Figure 1 – AOV-graph of the Health Watcher System before composition**

### 3.1.1 Descriptions of Crosscutting Relationships

**1 - Source: Persistence in [DB]**

```
crosscutting {
        source = Persistence in [DB]
        pointcut P15.1: include(Register.*; task; name)  and not include(Register [information])
        pointcut P15.2: include(Configurability)
        advice (around): P15.1 { (Make register operation; and)}
        intertype declaration (element): P15.2 {
                task = (Set [DB]; make) {
                        task = (Select [Microsoft Access]; or) {}
                        task = (Select [Oracle]; or) {}
                        task = (Select [MySQL]; or) {} } }

}
```

This crosscutting relationship is defined by two pointcuts, an advice and an intertype: the first pointcut uses regular expression, matching all tasks with Register (they are: (Register [kind], [date], [time] of the [complaint]) (Register [complaint information]) (Register [conclusion] and [employee] who gave conclusion) (Register [health unit]) (Register [speacialty]) (Register [employee]) (Register [disease]) (Register [symptom])), but not ( Register [information] ) because in this case Register information is only a way of group tasks; the second pointcut is only defined by Configurability; in first pointcut is added the Make register operation task defined in advice, this task is a task of Persistence in [DB]; the second pointcut is affected by a new task (Set [DB]), using intertype this task is added in source point (persistence in DB) and in the pointcut (as a decomposition).

**2 - Source: Detect [exception]**

```
crosscutting {
        source = Detect [exception]
        pointcut PC5.3: include(Persistence in [DB])
        pointcut PC5.4: include(Authentication by Login)
        pointcut PC5.5: include(Availability)
        intertype declaration (element): PC5.3 {task = (Detect [persistence exception]; and) {} }
        intertype declaration (element): PC5.4 {task = (Detect [authentication exception]; and) {} }
        intertype declaration (element): PC5.5 {task = (Detect [availability exception]; and) {} }
}
```

This crosscutting relationship has three pointcuts and one intertype for each one of them. In this case is the new tasks Detect [persistence exception], Detect [authentication exception] and Detect [availability exception] are added in Detect [exception], and in each pointcut is added the associated task.

**3 - Source: Authentication**

```
crosscutting {
        source = Authentication
        pointcut P9.3.1: include(Update [complaint]) and include(Register [information])
        advice (around): P9.3.1 { (Authentication by Login; and)}
}
```

Authentication by login affect two points (Update [complaint] and (Register [information]). The sense of around is, for example: Authentication by login is necessary to satisfy Update [complaint].

**4 - Source: Cryptography**

```
crosscutting {
        source = Cryptography
        pointcut PC9.3.5.1: include(Authentication by Login)
        advice (around): PC9.3.5.1 {(Symmetric cryptography; and)}
}
```

Cryptography only affects one point (Authentication by login). This crosscutting relationship was defined because other points can easily appear, but it can be excluded.

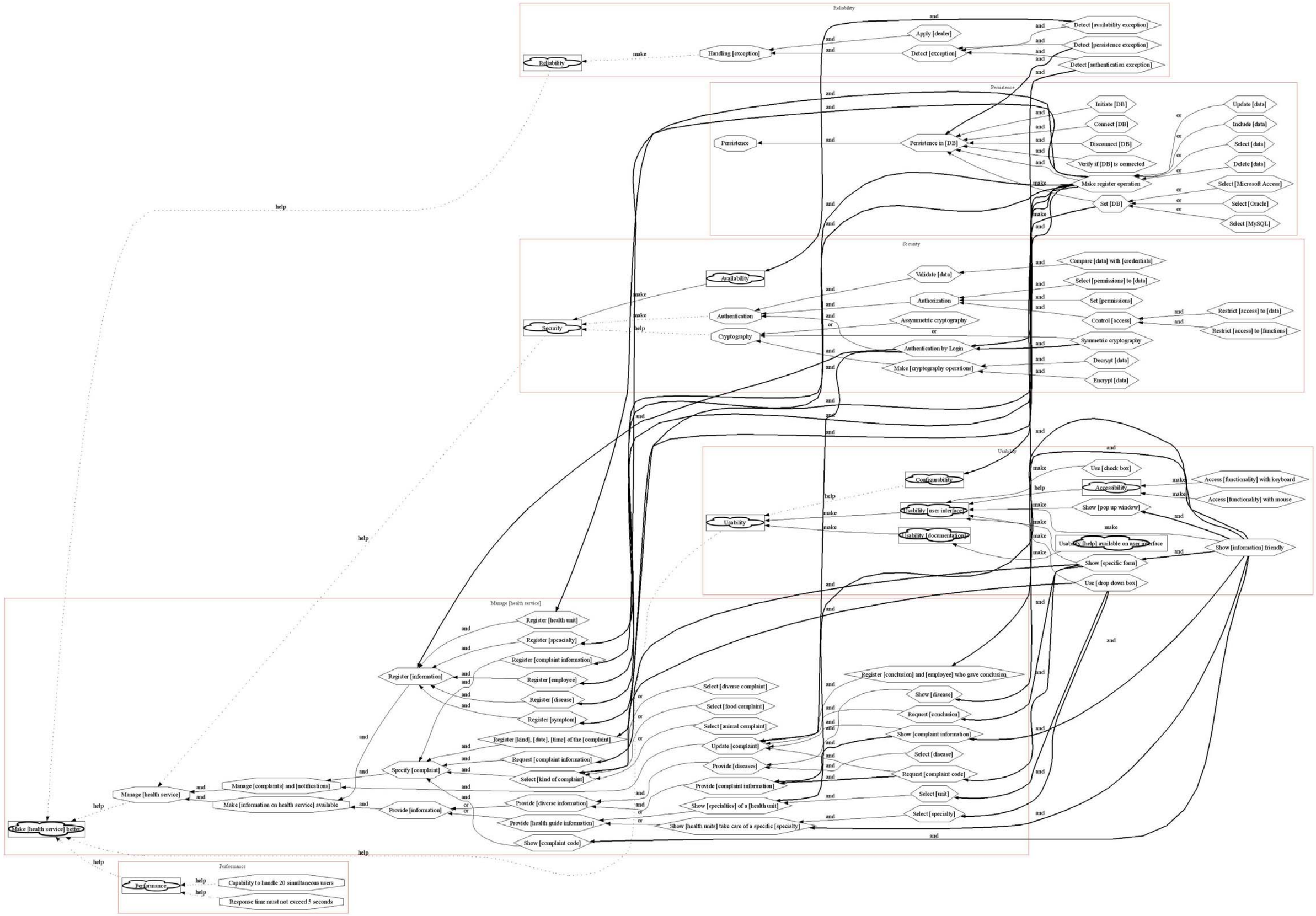
**5 - Source: Usability [user interface]**

```
crosscutting {
        source = Usability [user interface]
        pointcut PC3.2:  include(Select [kind of complaint]) and include(Select [specialty]) and
include(Select [unit])
        pointcut PC3.3: include(Request.*; task; name)
        pointcut PC3.4: include(Show.*; task; name)
        advice (around): PC3.2 {(Use [drop down box]; and)}
        advice (around): PC3.3 {(Show [specific form]; and)}
        advice (around): PC3.4 {(Show [information] friendly; and)}
}
```

This crosscutting relationship has two pointcuts defined by regular expression (Request.* ={ (Request [complaint information]) (Request [complaint code]) (Request [conclusion])} and Show.*={ (Show [complaint code]) (Show [complaint information]) (Show [health units] take care of a specific [specialty]) (Show [specialties] of a [health unit]) (Show [disease]) (Show [pop up window]) (Show [specific form]) }). Respectively, this pointcuts are affected by Show [specific form] and Show [information] friendly. The pointcut PC3.2 is affected by Use [drop down box] task.

## 3.2 AOV-graph after composition

**Figure 2 – AOV-graph of the Health Watcher System after composition**


Reliability
Reliability
make
Handling [exception]
and
Apply [dealer]
and
Detect [exception]
and
Detect [availability exception]
and
Detect [persistence exception]
and
Detect [authentication exception]
Persistence
Persistence
and
Persistence in [DB]
and
Initiate [DB]
and
Connect [DB]
and
Disconnect [DB]
and
Verify if [DB] is connected
and
Make register operation
or
Update [data]
or
Include [data]
or
Select [data]
or
Delete [data]
make
Set [DB]
or
Select [Microsoft Access]
or
Select [Oracle]
or
Select [MySQL]
Security
Security
make
Availability
make
Authentication
help
Cryptography
and
Validate [data]
and
Authorization
or
Assymetric cryptography
or
Symmetric cryptography
and
Compare [data] with [credentials]
and
Select [permissions] to [data]
and
Set [permissions]
and
Control [access]
and
Restrict [access] to [data]
and
Restrict [access] to [functions]
and
Authentication by Login
and
Make [cryptography operations]
and
Decrypt [data]
and
Encrypt [data]
Usability
Configurability
Usability
make
Usability [user interface]
make
Usability [documentation]
help
Accessibility
make
Use [check box]
make
Access [functionality] with keyboard
make
Access [functionality] with mouse
make
Show [pop up window]
make
Usability [help] available on user interface
make
Show [specific form]
make
Use [drop down box]
and
Show [information] friendly
Manage [health service]
Manage [health service]
and
Manage [complaints] and [notifications]
and
Make [information on health service] available
and
Provide [information]
and
Specify [complaint]
and
Register [information]
and
Register [health unit]
and
Register [speacialty]
and
Register [complaint information]
and
Register [employee]
and
Register [disease]
and
Register [symptom]
and
Register [kind], [date], [time] of the [complaint]
and
Request [complaint information]
and
Select [kind of complaint]
or
Select [diverse complaint]
or
Select [food complaint]
or
Select [animal complaint]
and
Update [complaint]
and
Provide [diseases]
and
Provide [complaint information]
and
Provide [diverse information]
or
Provide [health guide information]
and
Show [specialties] of a [health unit]
and
Show [health units] take care of a specific [specialty]
and
Show [complaint code]
and
Register [conclusion] and [employee] who gave conclusion
and
Show [disease]
and
Request [conclusion]
and
Show [complaint information]
and
Select [disease]
and
Request [complaint code]
and
Select [unit]
and
Select [specialty]
and
Make [health service] better
Performance
Performance
help
Capability to handle 20 simultaneous users
help
Response time must not exceed 5 seconds

# 4 Views of the AOV-graph

## 4.1 Colored AOV-graph

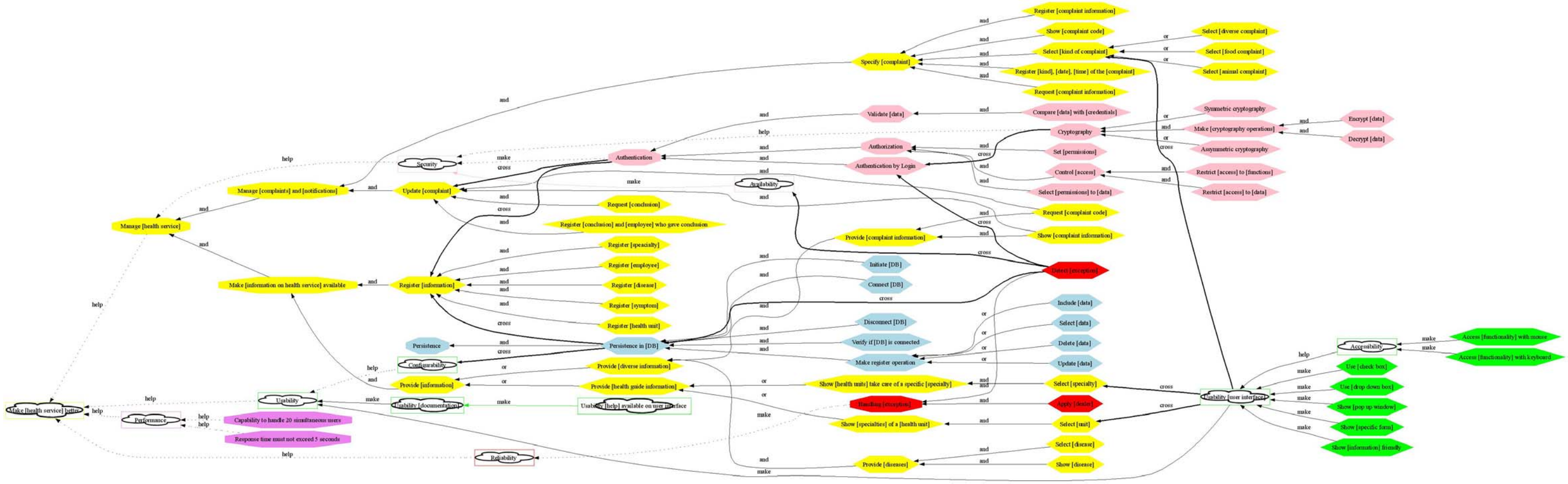



**Figure 3 – AOV-graph of the Health Watcher System before composition**

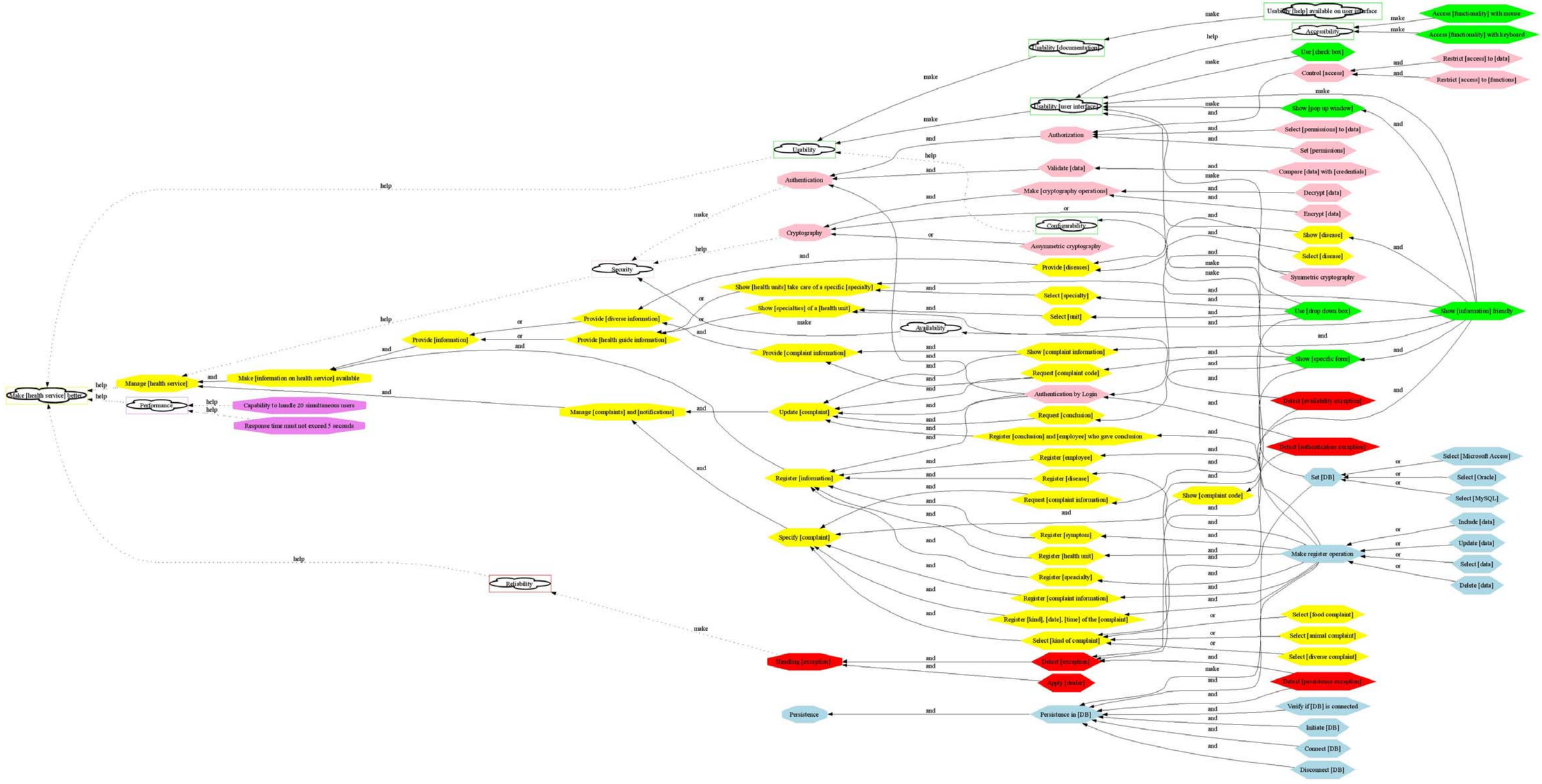



**Figure 4 – AOV-graph of the Health Watcher System after composition**

## 4.2 Hierarchy of softgoals, goals and tasks

In bold, we explicit new decompositions that were added by the composition.

### 4.2.1 Model: Manage [health service]

- Make [health service] better
- Manage [health service]
  - Manage [complaints] and [notifications]
    - Specify [complaint]
      - Select [kind of complaint]
        - Select [diverse complaint]
        - Select [food complaint]
        - Select [animal complaint]
        - **Use [drop down box]**
      - Register [kind], [date], [time] of the [complaint]
        - **Make register operation**
      - Request [complaint information]
        - **Show [specific form]**
      - Register [complaint information]
        - **Make register operation**
      - Show [complaint code]
        - **Show [information] friendly**
    - Update [complaint]
      - Request [complaint code]
        - **Show [specific form]**
      - Show [complaint information]
        - **Show [information] friendly**
      - Request [conclusion]
        - **Show [specific form]**
      - Register [conclusion] and [employee] who gave conclusion
        - **Make register operation**
      - **Authentication by Login**
  - Make [information on health service] available
    - Register [information]
      - Register [health unit]
        - **Make register operation**
      - Register [speacialty]
        - **Make register operation**
      - Register [employee]
        - **Make register operation**
      - Register [disease]
        - **Make register operation**
      - Register [symptom]
        - **Make register operation**
      - **Authentication by Login**
    - Provide [information]
      - Provide [health guide information]
        - Show [health units] take care of a specific [specialty]
          - Select [specialty]
            - **Use [drop down box]**
          - **Show [information] friendly**
        - Show [specialties] of a [health unit]
          - Select [unit]

            - **Use [drop down box]**
        - **Show [information] friendly**
    - Provide [diverse information]
        - Provide [complaint information]
            - **Request [complaint code]**
            - **Show [complaint information]**
        - Provide [diseases]
            - Select [disease]
            - Show [disease]
                - **Show [information] friendly**

### 4.2.2 Model: Persistence

- Persistence
    - Persistence in [DB]
        - Verify if [DB] is connected
        - Initiate [DB]
        - Connect [DB]
        - Make register operation
            - Include [data]
            - Select [data]
            - Delete [data]
            - Update [data]
        - Disconnect [DB]
        - **Set [DB]**
            - **Select [Microsoft Access]**
            - **Select [Oracle]**
            - **Select [MySQL]**
        - **Detect [persistence exception]**

### 4.2.3 Model: Reliability

- Reliability
- Handling [exception]
    - Detect [exception]
        - **Detect [persistence exception]**
        - **Detect [authentication exception]**
        - **Detect [availability exception]**
    - Apply [dealer]

### 4.2.4 Model: Security

- Security
    - Availability
        - **Detect [availability exception]**
- Authentication
    - Authentication by Login
        - **Detect [authentication exception]**
        - **Symmetric cryptography**
    - Validate [data]
        - Compare [data] with [credentials]
    - Authorization
        - Select [permissions] to [data]

        - Set [permissions]
        - Control [access]
            - Restrict [access] to [functions]
            - Restrict [access] to [data]
- Cryptography
    - Assymmetric cryptography
    - Symmetric cryptography
    - Make [cryptography operations]
        - Encrypt [data]
        - Decrypt [data]

### 4.2.5 Model: Performance

- Performance
- Response time must not exceed 5 seconds
- Capability to handle 20 simultaneous users

### 4.2.6 Model: Usability

- Usability
    - Usability [documentation]
        - Usability [help] available on user interface
    - Usability [user interface]
        - Accessibility
            - Access [functionality] with mouse
            - Access [functionality] with keyboard
        - Use [check box]
        - Use [drop down box]
        - Show [pop up window]
            - **Show [information] friendly**
        - Show [specific form]
            - **Show [information] friendly**
        - Show [information] friendly
- Configurability
    - **Set [DB]**

# 5. Evaluating the HealthWatcher and the AOV-graph

We think the HealthWatcher is a good example, but it would be better if other functionalities were added in order to provide flexibility for exploring crosscutting concerns come from functional requirements (specific from the health domain). We only modeled HealthWatcher's AOV-graph based on the functionalities presented in Use Case Specification, and we only found the traditional crosscutting concerns, those come from Persistence, Usability, Security, Realiability.

With the AOV-graph was possible to model the HealthWatcher system, but some information is not properly captured, such as data types, actors for each goal/task and sequence of tasks:

- It is possible to refine the modeling in order to add more information about data types, but it can make the model very big and unmanageable, thus we prefer to use a glossary in order to define the information about the data types.

- It is possible to add the actors to tasks and goals by using intertype declarations (with attribute type) and would be necessary to define a new view showing these actors.
- It is possible to generate a scenarios view in order to show sequences of activities, but this view can only be generated by using the sequence of tasks defined in XML language.

Some advantages on using AOV-graph:

- The AOV-graph represents very well the interactions between functional and non-functional requirements.
- Using AOV-graph it is possible to generate different views, such as: partial views of the AOV-graph, scenarios, class diagram, entity-relationship, list of requirements.

Some drawbacks on using AOV-graph:

- The AOV-graph models tend to increase a lot (graphically), making the visualization (and printing) difficult. In these cases it is essential to have partial views of AOV-graph.
- There are no tools available for edition of AOV-graphs (XML) and for edition of the created views. Although, there are a set of XSLTs to generate views, they are not integrated and the transformations rules need to be improved. It is also necessary a semantic verification tool.