Systems Engineering using Modelio

INCOSE 2012 Tool Vendor Challenge Case Study

Dr. Imran Rafiq Quadri

SOFTEAM
Think Object

Contents

Table of Figures

1 Introduction

The aim of this white paper is to provide the readers with a detailed description regarding Softeam's solution for the INCOSE 2012 Tool Vendor Challenge (TVC). INCOSE or International Council on Systems Engineering is a non-profit organization that is dedicated to the advancement of **Systems engineering**. INCOSE aims to advance and harmonize systems engineering standards used worldwide. Each year at the annual INCOSE international conference, a Tool Vendor Challenge (TVC) is proposed to participating systems engineering tool vendor exhibitors, in order to provide an innovative solution to the issued challenge, using the capabilities of their tools, technologies and environments. Similarly, the INCOSE 2012 conference also issued a challenge related to a national incident management system. The idea was not to create an in-depth and perfect solution for the given challenge, but to illustrate the features present in a specific tool/environment such as related to requirements engineering.

Softeam has taken part in that particular TVC, and presented its solution based on the **Modelio environment**.

This white paper compliments the actual modeled solution and the dissemination video (available on Softeam's Modeliosoft website) in order to help the readers gain an insight into the proposed solution.

2 INCOSE 2012 Tool Vendor Challenge

The original statement of the INCOSE 2012 Tool Vendor Challenge [1] is as follows:

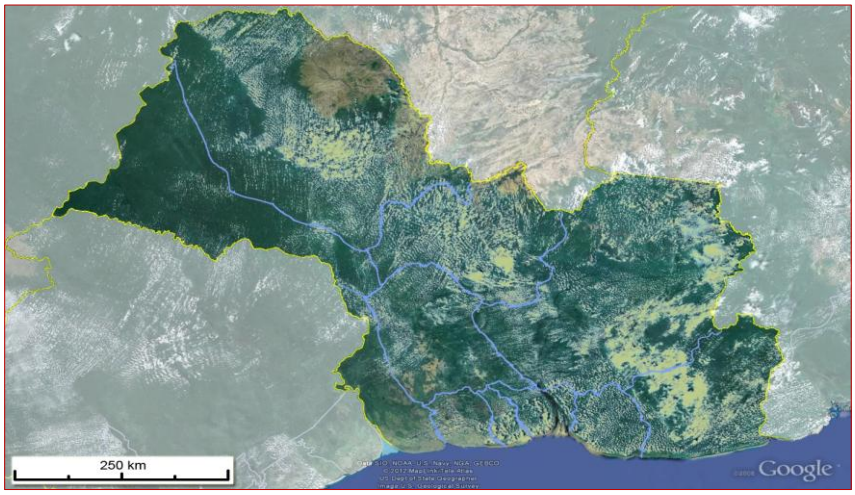

Figure 1 - Overview of Forestia

"Forestia (as seen in Figure 1) is a country that wishes to improve both the effectiveness and efficiency of its firefighting capacity. The current rapid response organization relies on 11 departmental operation centers that manage, command and control local assets independently from each other. It is acknowledged that this organization leads to inefficiency; for example one region is lacking capacity, while another had no immediate use of its own assets.

The Ministry of Interior has asked the solution providers to design, build and deliver a *Permanent Emergency Response Coordination Center (PERCC)* for wildfires, in charge of coordinating the 11 departmental operation centers. The two main expected improvement areas are:

- Decrease response time;
- Optimize resources allocation.

PERCC shall cover situational awareness, incident, and resource management. It maintains a real-time status of available firefighting assets. PERCC centralizes the alerts, assesses them, and allocates appropriate assets to the departmental operation center for recovery. In order to optimize the use of resources, it is required that PERCC has real-time visibility of firefighting assets availability, that might be affected by maintenance, holidays, illness, etc.

PERCC shall cooperate with the following existing systems:

1. The departmental operation centers (DOC), that currently manages local assets, will now focus on command and control of the firefighting assets that are allocated to fight the wildfire, and no longer be responsible for resource management;
2. The fire fighting assets that report status and availability:
    a. The ground firefighter squads;
    b. The water bombers;
3. The civil and military air traffic management for coordination of the airspace management in the area of the wildfire;
4. The weather forecast administration that shall give useful information for the evaluation of the crisis and identification of the most effective fire fighting means.

PERCC shall have three distinct deployment options corresponding to three operational modes:

- Green : low risk (e.g. winter time);
- Amber : high risk (e.g. summer time);
- Red : crisis (during fire fighting).

Additionally, it was stressed upon to the tool vendor providers that the initial time period for the system design was to start from beginning of July 2012, and the government wanted to have a first level of capability before end of summer 2012. Therefore, a condition was that the update of departmental operation centers must be initiated before the PERCC was fully defined, in other words before the interface between PERCC and DOC was fully defined in the specifications.

The tool vendors were asked to:

a. Compose the system specification;
b. Demonstrate requirements handling;
c. Define the system in its environment;
d. Define logical sub-systems, flows from sub-systems to the environment and flows between sub-systems;
e. Define sub-system activities;
f. Describe the functional modes of the system and transfers between them;
g. Describe the interactions between the PERCC and the water bomber provider to implement the required report status;
h. Demonstrate the handling of concurrent engineering. They were asked to deliver preliminary interface requirements to the company in charge of the DOC development in a couple of weeks. Those requirements were to be later refined and or amended by a third party. It was envisaged that the tool vendors would need to re-inject the refinements in their initial designs;
i. Draw a timeline for the development of the system;
j. Show how the vendor tool supports the System Engineering activities.

It was also emphasized that tool vendors should include the particularities of the human sub-systems (or humans) that are part of the total system. The tool vendors were allowed to focus on one particular system function (e.g. optimization resource of allocation) to demonstrate the capabilities of their tool rather than covering all system's features**"**.

3 Summary of proposed solution

We now provide a brief summary of our proposed solution. As seen in Figure 2, our proposed solution integrates several aspects of the solution space related to systems engineering; such as requirements management, structural/behavioral modeling, concurrent engineering, traceability, impact analysis, among others. The idea is to showcase the different capabilities of our modeling environment while in parallel illustrating an effective solution to the proposed challenge. We refer the reader to the modeled project, available on Modeliosoft website[1], along with the generated documentation and web model, which in turn complement this document.

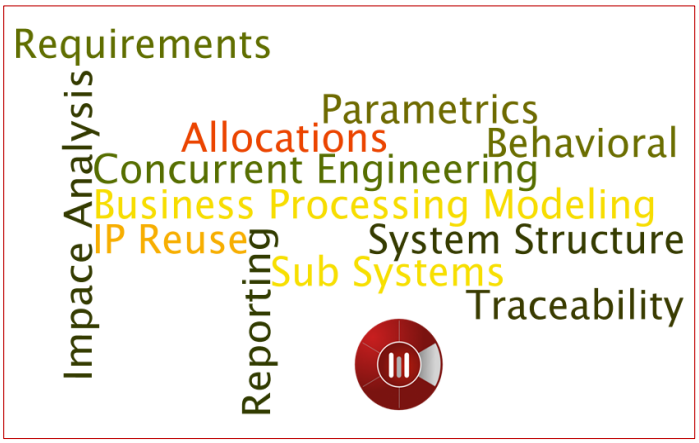

Figure 2 - The features to be integrated in Softeam's INCOSE TVC Solution

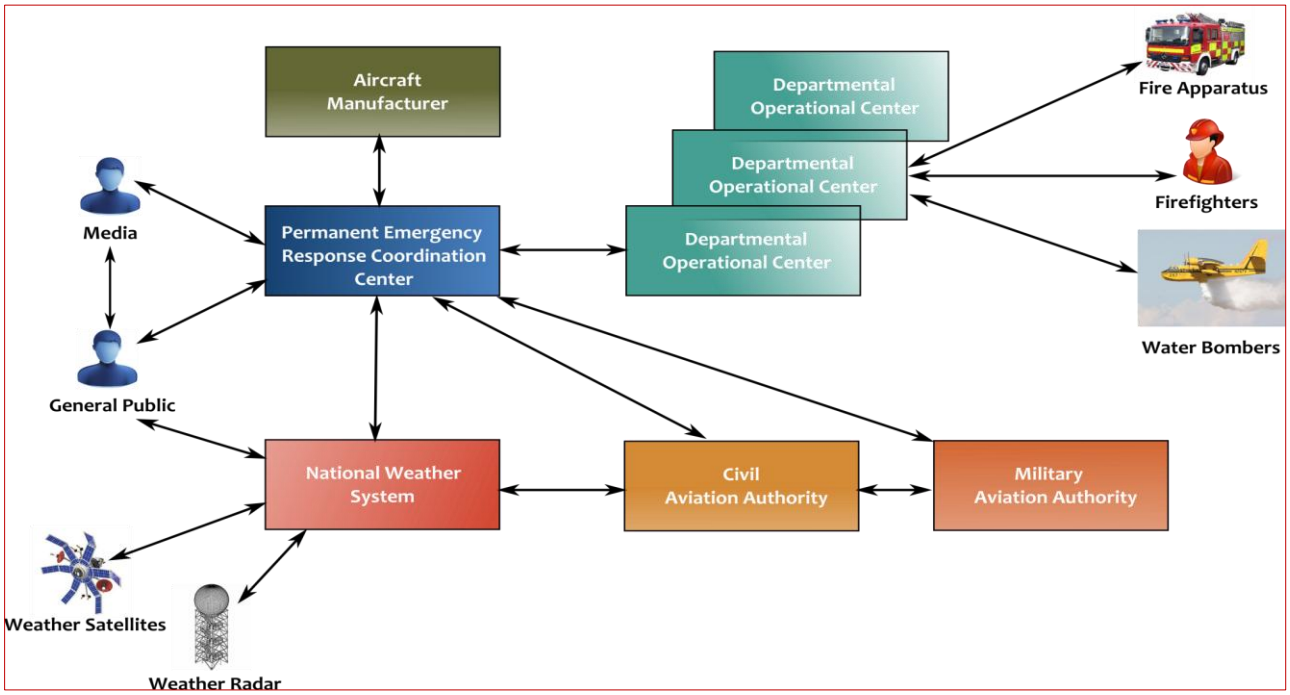

Figure 3 - An overview of the global scenario

Figure 3 illustrates how we comprehend the global scenario. As indicated in the initial requirements, the *PERCC* is communicating with 11 *DOCs* and several other entities that comprise the external environment. Some of these

[1] Modelio White Papers, *http://www.modeliosoft.com/en/resources/white-papers.html*, 2012

entities such as the *National Weather System (NWS)* and the *Civil Aviation Authority (CAA)* provide feedbacks to the *PERCC*, which is then sent to other entities such as the *DOCs*, the *Firefighters* and the *Water bombers* that are directly related to firefighting. The *National Weather System* gets weather related information (such as temperature, wind, visibility, pressure, etc.) from the weather satellites and the weather radars. Information such as related to possible incidents (floods, wildfires, thunderstorms, etc.) is also measure by the weather system. The *NWS* calculates possibilities of these incidents, such as occurrence of a wildfire in case of thunderstorms and may issue a red flag warning/alert to the *PERCC* to take necessary actions, in the case of a possible or eventual wildfire occurrence. The *NWS* may also inform the general public about these weather conditions either by the media or via radio, television or by means of its website.

The *NWS* also sends that information to the *CAA* and the *Military Aviation Authority (MAA)* to make any flight plan modifications to their civil/military aircrafts flight routes. The CAA also coordinates with the *MAA* and sends consolidated information to the *PERCC* to synchronize the different flight plans/routes of the civil/military aircrafts and the firefighting water bombers to avoid any conflicting issues that could arise.

The *PERCC* thus receives information from the *NWS* and *CAA/MAA* regarding the weather conditions and flight routes respectively. In case of any possible national incident that could endanger lives and property, the *PERCC* is able to inform the general public by means of the media and its official website. The *PERCC* also communicates with the air/rotorcraft manufacturer to address the aircraft requirements needed to carry out effective firefighting, such as salvo capacity, cruising speed, etc. This exchange enables *PERCC* to have efficient firefighting aircrafts that are constructed due to feedback obtained from previous firefighting incidents.

The *PERCC* also communicates with the *DOCs* to carry out global resource management and administration. In case of a firefighting scenario, the *PERCC* manages the resources available between the different *DOCs*, and then tells the *DOCs* to carry out the firefighting operations. The *DOCs* being in charge of command and control operations, then carry out the necessary firefighting activities by deploying the related resources (firefighters, aircrafts, vehicles, equipment, etc.)

Our proposed solution makes use of **Systems Modeling Language (SysML)**[2], an industrial standard for carrying out systems engineering [2]. We make use of several SysML design phases, such as requirements specification, system structural (block and internal blocks) and related behavioral (activity, state machine and sequences) specifications, parametrics (for expressing mathematical equations in the high-level models), among others. Additionally, the **Business Processing Modeling Notation (BPMN)** [3] has also been utilized in the project, to illustrate Modelio's capability to handle multiple standards at the same time.

Moreover, we make use of added features in Modelio, such as those related to traceability and automatic diagram generation (for example Impact Analysis and Dependency diagrams) to help a system designer become more productive, with the intended goal being to decrease system design time and improve the overall productivity.

Finally, using Modelio's automatic document generation capabilities, we are able to create either classic documents (such as HTML or MS Office based) or Web model based ones. The latter help in reviewing a complete project and its aspects (such as internal hierarchy, composition of system, etc.) without requiring a modeling environment such as Modelio.

[2] It should be noted that the solution has been based on the SysML 1.2 standard, during the development and presentation of the case study, a new 1.3 version of SysML standard was made available that introduces and modify some of the existing concepts present in this solution, such as the usage of flow ports. Henceforth, updated versions of the solution and Modelio SysML module are being developed to be compatible with the new SysML 1.3 standard.

4 Specification of proposed solution

4.1 Requirements specification

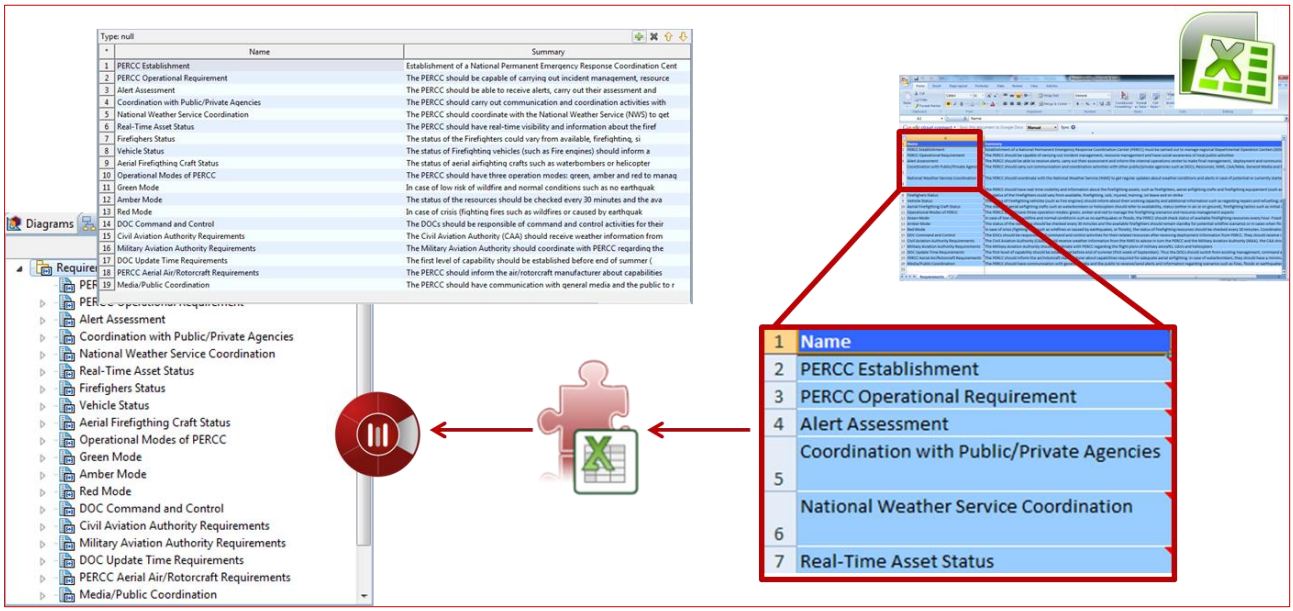

Figure 4 - Creating and importing requirements in Modelio

The first phase of any effective system specification is to determine the global requirements of the system. These requirements are crucial for determining the functional as well as non functional aspects of the system, and help in discovering and documenting system properties. The requirements can be written in various manners: for example they can be depicted by means of software requirement specification documents or can be graphically illustrated via modeling languages such as UML [4] or SysML.

As seen in the previous section, the TVC provides a global scenario as well as some requirements that must be met when the design of the PERCC system is to be carried out. We take these requirements as the initial top level requirements, and then integrate some additional requirements to these top level requirements to create the *PERCC* system. We initially created the requirements in a Microsoft Excel document, as seen in Figure 4, and then imported these requirements in Modelio using the Excel Exchange module[3]. The Excel Exchange module is used to import/export modeling elements (such as requirements) and related properties between Modelio and Excel. Thus designers can export their specified requirements, business rules, goals and use cases from Modelio to Excel. The Excel Exchange module also enables synchronization between Excel and Modelio. Therefore, it is possible for designers to modify their initial requirements in Excel, and then synchronize with the previously exported requirements present in Modelio. The goal being to give the designer freedom to select a editing tool per his preference criteria, either Modelio or Excel, facilitating the overall design process.

Once the requirements are successfully imported in Modelio, the Analyst[4] module can be used that enables graphical requirement modeling, complaint with SysML and integrates features such as Impact Analysis among others. Details regarding these features are provided later on in the white paper.

Figure 5 illustrates an extract of the requirements imported in Modelio and respecting SysML specifications. Each of the requirement has a description that illustrates its functional aspect. Moreover, additional information (such as risk factor, benefit, scope, stability, etc.) can be added to these requirements, however this step is not shown in the figure.

---

[3] Modelio Excel Exchange Module, *http://www.modeliosoft.com/en/modules/excel-exchange.html*, 2012
[4] Modelio Analyst Module, *http://www.modeliosoft.com/en/modules/analyst.html*, 2012

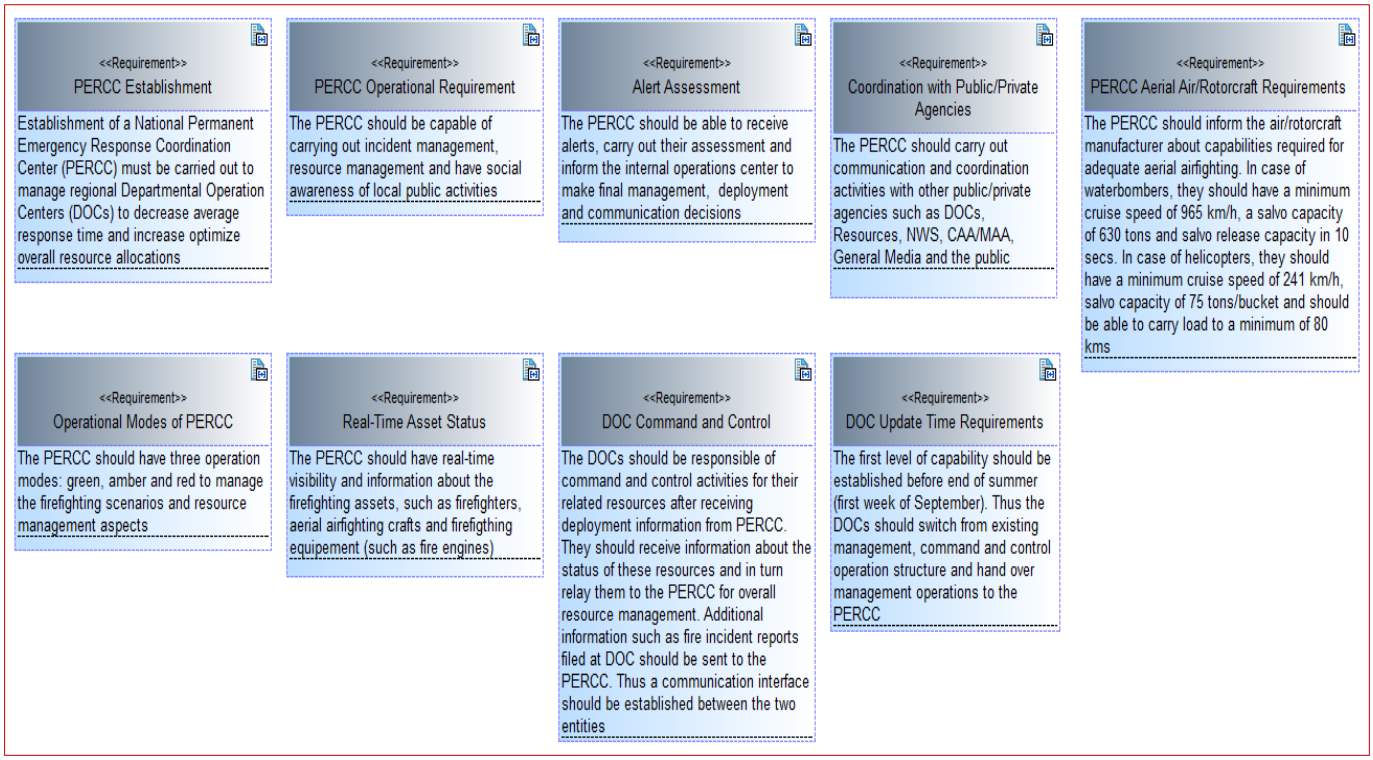

Figure 5 - Extract of requirements related to the PERCC

The figure shows some requirements directly taken from the TVC while others are expanded or enriched to provide a viable solution with concrete details. For example, the *Operational Modes of PERCC* requirement describes the different working modes of the *PERCC* as stated initially in the TVC, while the *PERCC Aerial Air/Rotorcraft Requirements* enrich the initial requirements: by providing more specific details needed by the *PERCC* for effective firefighting air/rotorcrafts, such as related to effective aircraft cruising speed and fire retardant salvo capacity.

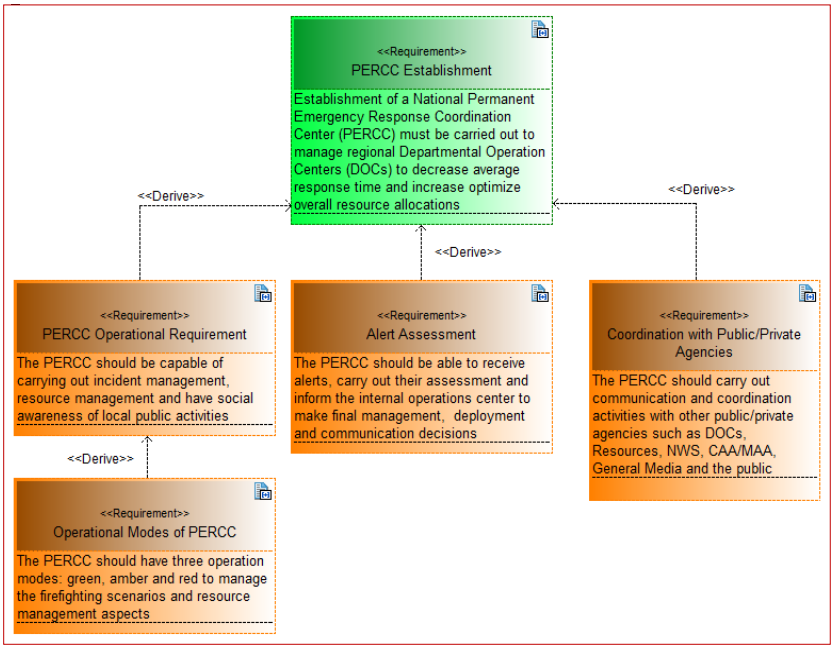

Figure 6 - Structuring the requirements

In Figure 6 we present an extract of the organization of the specified requirements. The primary level requirement is the *PERCC Establishment* (highlighted with green color), and the secondary level requirements like *Alert Assessment* derive from the primary requirement and are highlighted with orange color. This organization provides a coherent structure to the imported requirements, and also enabling the designers to determine the priority of the requirements.

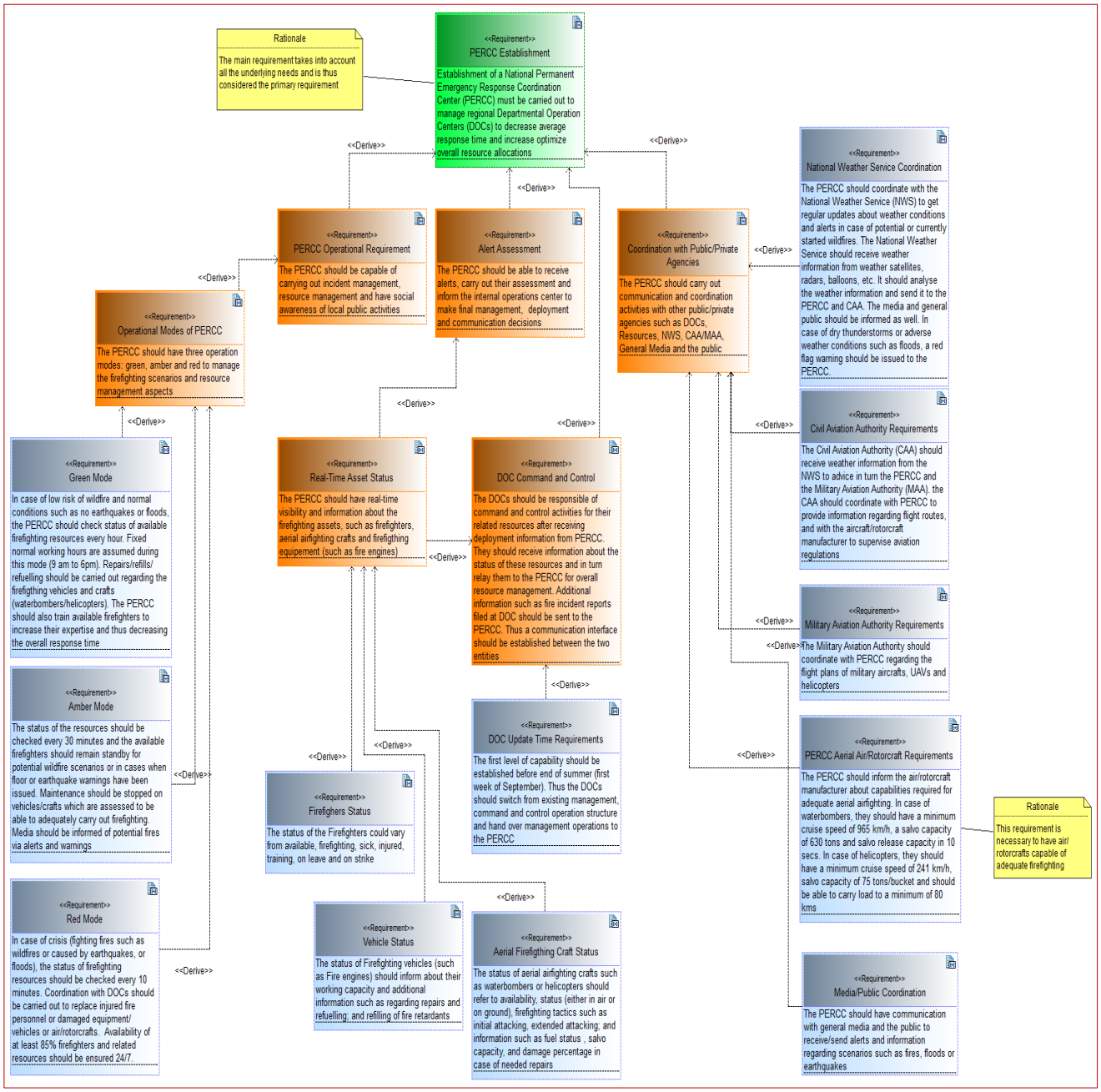

Figure 7 - PERCC requirements: the big picture

In Figure 7, the global overview of the PERCC requirements is shown. Here in addition to the primary and secondary level requirements, additional tertiary requirements are also present to enrich the high level requirements. Additionally, we provide some *Rationales* to the requirements (as per the SysML standard) for the subsequent documentation purposes.

4.2 Initial behavioral specification using Use cases

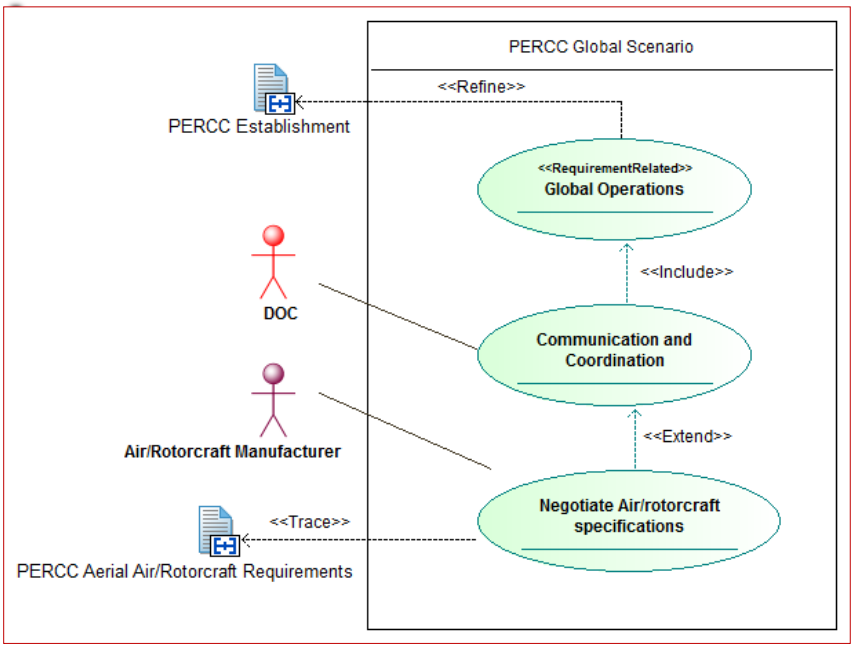

Figure 8 - Behavioral analysis using use cases

Once the requirements phase is completed, it is possible to carry out the behavioral analysis phase. In the particular case of the TVC solution, we utilize use cases to determine the different *PERCC* related scenarios. These use cases can be automatically generated to link with the specified requirements via the Modelio Analyst module or can be explicitly created by a system designer. In Figure 8, we provide an extract of the use cases related to the *PERCC*. Here as seen in the figure, the use cases are interacting with external entities such as the *DOC* and the *Air/Rotorcraft Manufacturer* to achieve certain goals, such as negotiation and coordination. The use cases can refine the initial requirements, such as the *Global Operations* use case refining the *PERCC Establishment* primary requirement. Additionally, via Modelio, it is possible to create trace dependencies between different modeling elements; such as the trace dependency present between the *Negotiate Air/rotorcraft specifications* and the *PERCC Aerial Air/Rotorcraft Requirements*.

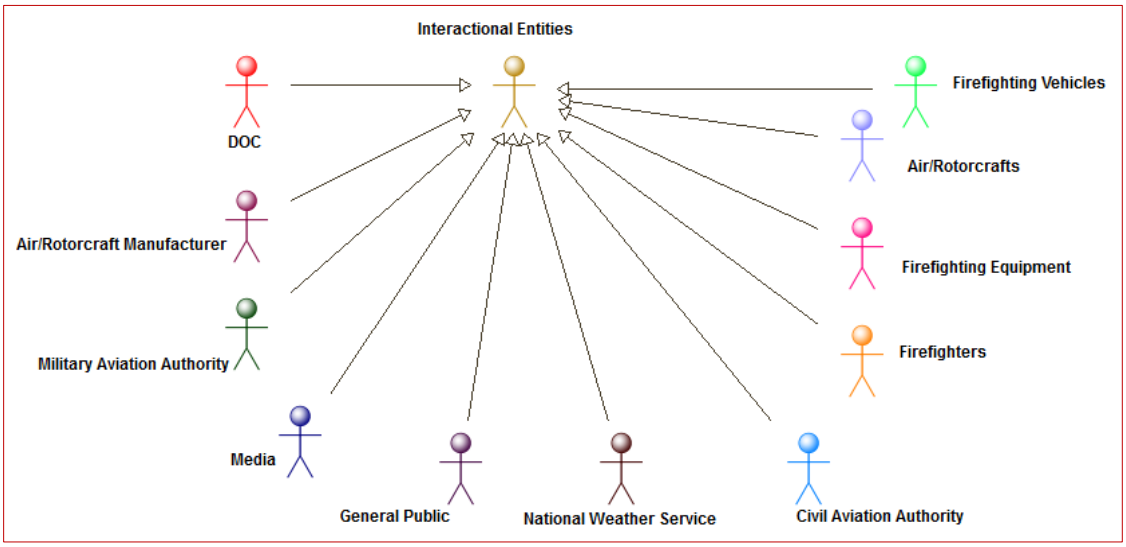

Figure 9 - The external environment related to the PERCC

In Figure 9, we specify the external environment interacting with the *PERCC* by means of actors, as initially defined in Figure 3. Here different entities are described, such as the *DOC*, the *Firefighters*, the *Civil Aviation Authority*, among others, all being derived from *Interactional Entities*.

4.3 Fulfillment of specified requirements

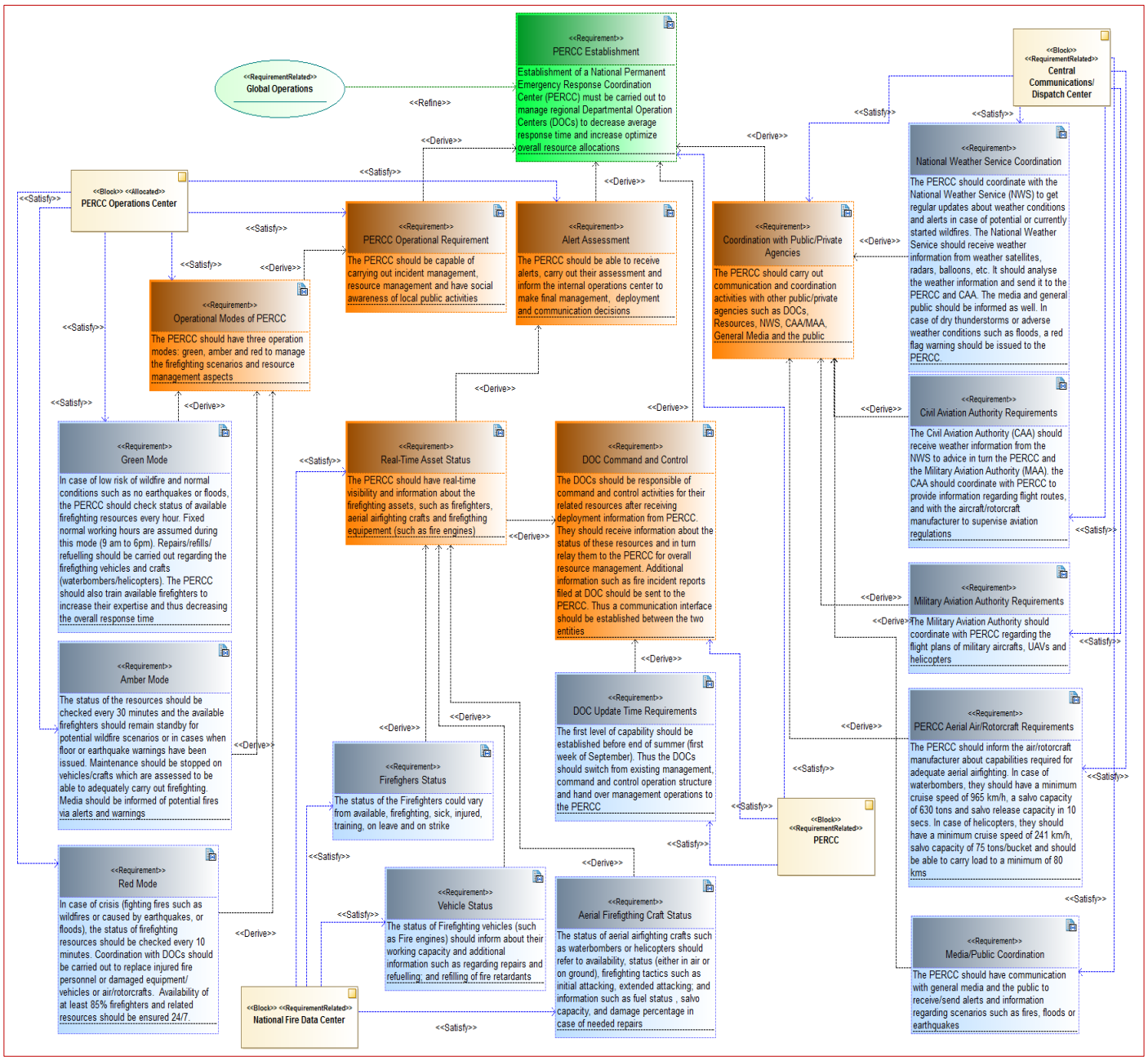

Figure 10 - Completing the initial PERCC requirements

Afterwards, Figure 10 illustrates how the defined requirements can be satisfied or refined. The requirements are linked to modeled SysML blocks (defined in the subsequent section) by means of a *satisfy* dependency. This dependency illustrates that a requirement is being satisfied by a system component. It is possible for one SysML block to satisfy several specified requirements: for example the Central *Communications Dispatch Center* block, responsible for the communication between the *PERCC* and external environment, is satisfying several defined requirements as seen in the figure. Moreover, we also see that the use case depicted in Figure 8 is also present here, to refine the primary system requirement.

We now focus on the *PERCC* system structure and internal behavior specification. We first describe the *PERCC* system and its internal structure which has helped to complete the requirements design phase.

4.4 PERCC structure specification

4.4.1 PERCC block structure

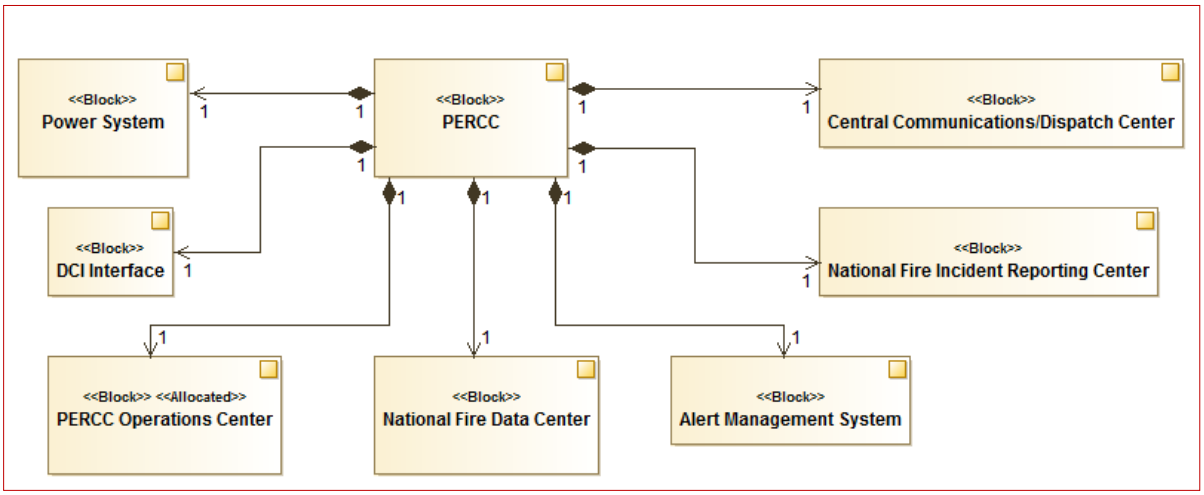

Figure 11 - PERCC block structure diagram

We now describe the structure of the PERCC in Figure 11. Using the SysML block diagram, we specify the different components/sub-systems (or SysML blocks) present inside the main *PERCC* block. The PERCC contains several sub-systems such as:

- *Power System*: To provide power to the *PERCC*;
- *DCI Interface*: The communication interface between the *PERCC* and the *DOCs*. The *DCI Interface* is created by an external third party company and could be modified later on per the TVC requirements related to concurrent engineering;
- *PERCC Operations Center*: The central system of the *PERCC*. It receives information from other PERCC sub-systems (such as related to alerts, firefighting resources) and is responsible for effective resource allocations between the different *DOCs*;
- *National Fire Data Center*: The national data center for the *PERCC*. It is responsible for storing the data related to firefighting incidents, reports as well as the real-time status related to the firefighting resources (firefighters, water bombers, etc.). The status could include readiness level and availability of firefighters/water bombers and additional information such as that related to the health/expertise level of firefighters;
- *Alert Management System*: It receives alerts from the *National Weather Service* and carries out alert assessment and prioritization (for example based on the potential damage due to possible wildfires);
- *National Fire Incident Reporting Center*: The system is responsible for getting feedbacks related to firefighting incidents. This feedbacks include firefighting personnel filling fire reports after putting out fires and indicating damages (related to human/animal lives, infrastructure), costs and spent resources;
- *Central Communications/Dispatch Center*: It is responsible for communication between the different sub-systems of the *PERCC*. It also enables the *PERCC* to communicate with the external environment.

4.4.2 PERCC internal block structure

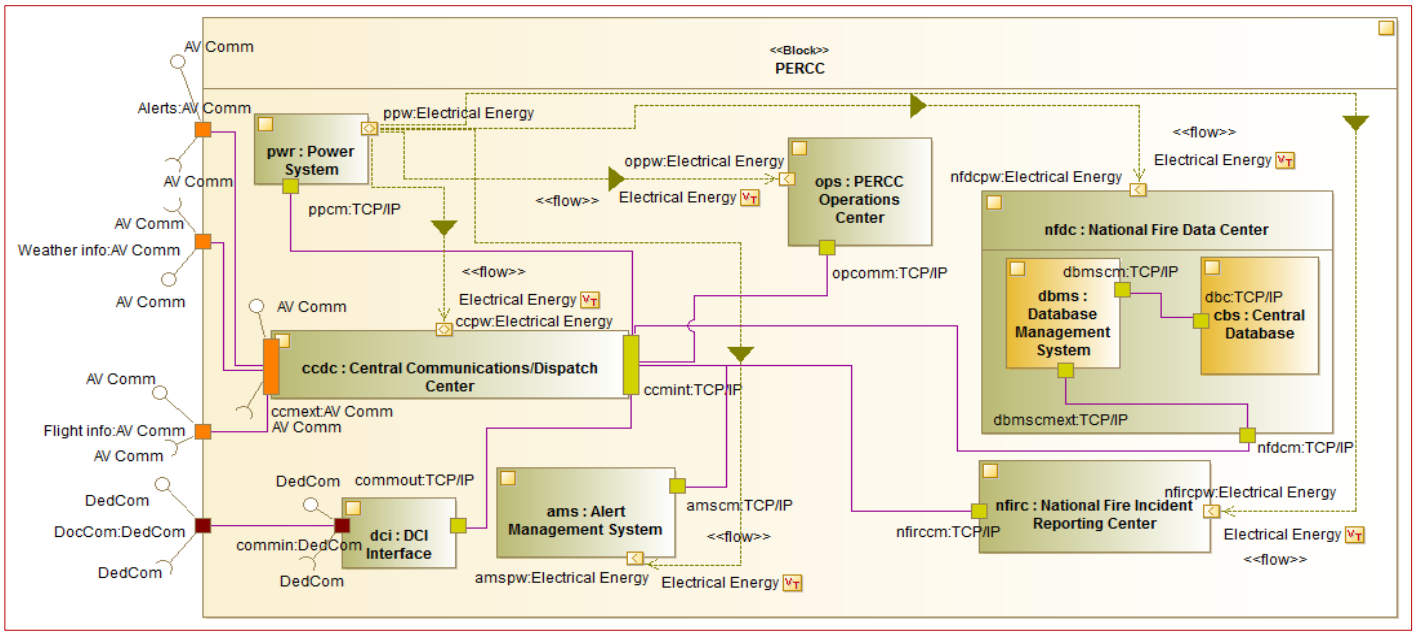

Figure 12 - PERCC internal block structure diagram

Once we define the different sub-systems of the *PERCC*, we create a SysML internal block diagram of the *PERCC*. This step is carried out because while a SysML block diagram enables to determine the composition of a system (*PERCC* in this particular case), it does not specify how the different sub-systems are connected to each other. As seen in Figure 12, the different sub-systems are communicating with each other by means of ports and connectors. The Power System *pwr* instance sends power to the other sub-systems by means of its *ppw* SysML flow port, which is in turn received by the respective flow ports of the other sub-systems. The different source/target flow ports and the information flows between them are typed with the *Electrical Energy* value type, specified later on in Section 4.4.3.

The Central Communications Dispatch Center *ccdc* instance provides communication links between the different *PERCC* sub-systems and with the external environment. The communication between the different sub-systems is assured by their relative SysML ports typed by the *TCP/IP* interface and the communication links between them, by means of connectors between the source/target ports. It should be observed that the *TCIP/IP* interface is used only for inter communication by the *PERCC* sub-subsystems, and two specific interfaces are used for *PERCC*/external environment communication: The *DedCom* and *AV Comm* interfaces. The first one is used only for the dedicated communication between the PERCC and the *DOCs*, while the second one is used for communication between the other external entities (such as media and public) and the *PERCC*. Thus, the intra communication between the *PERCC* and these external entities is assured by the ports of the *ccdc* instance and the *PERCC* which are typed with the *AV Comm* interface (which signifies audio/video communication). The PERCC receives information such as alerts, weather and flight plan/route information that is conveyed to the *ccdc*, which in turn sends the information to the respective sub-systems. Alternatively, the ports of the DCI Interface *dci* instance and *PERCC,* delegated to communicating with the *DOCs* are typed with the *DedCom* interface. The two distinct external communication interfaces have been created so that any eventual changes in the communication interface between the *PERCC* and the *DOCs* do not cause any modifications in rest of the communication network.

Finally, we see the internal structure of *National Fire Data Center*. The *nfdc* instance itself contains two sub-systems: a *Database Management* System and a *Database*, communicating with each other and other sub-systems of the PERCC.

4.4.3 User defined interfaces and value types

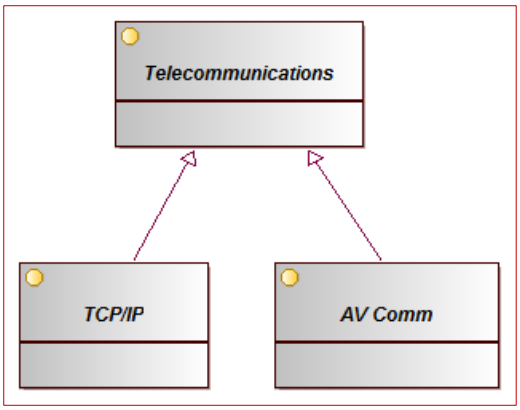

Figure 13 - Communication interfaces for the PERCC

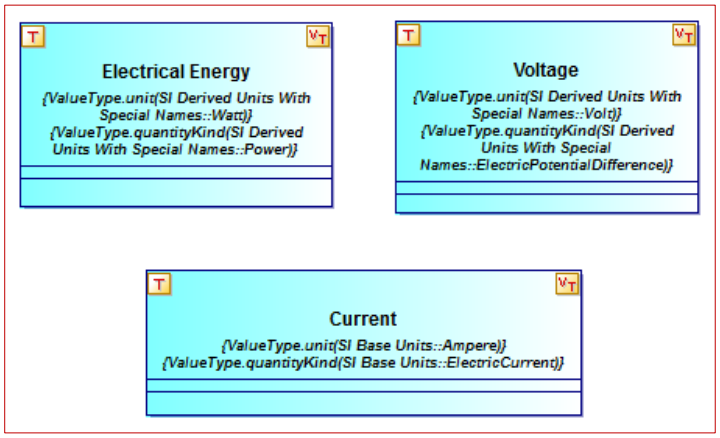

Figure 14 - Value types for the PERCC

Figure 13 and Figure 14 illustrate the user defined interfaces and value types created for the INCOSE TVC solution. The *TCP/IP* and *AV Comm* interfaces described previously both inherit from a generic *Telecommunications* interface. The operations of these interfaces, while present, are not shown here to avoid complexity issues related to system specifications.

The different user defined value types such as *Electrical Energy*, *Voltage* and *Current* are created in combination with the SysML Model library for Quantity Kinds and Units[5] defined in Modelio. Each of the user defined value type has two tagged values: *Quantity Kind* and *Unit*, corresponding to predefined SysML library concepts. For example, the *Electrical Energy* value type has the Quantity Kind set to *Power*, and the Unit set to *Watt* appropriately.

It should be noted that the *DedCom* interface is not specified by the designer itself, but is defined by the third party responsible for the *DCI Interface*. Details about this aspect are covered in Section 4.4.4 subsequently.

[5] Modelio SysML Model Library, *http://forge.modelio.org/attachments/download/4825/SIDefinition.ramc*, 2012

4.4.4 Concurrent engineering in Modelio

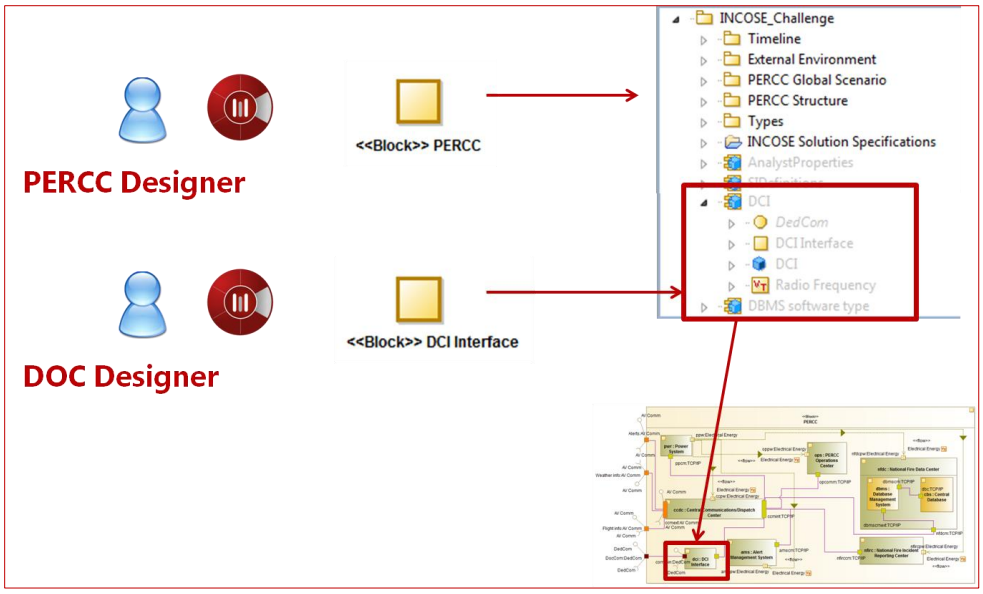

Figure 15 - Example of concurrent engineering in Modelio: user defined/third party components can be integrated in an existing design specification

We now describe how we can carry out concurrent engineering with Modelio, as this was one of the initial requirements in the INCOSE 2012 TVC related to the interface between the *DOCs* and the *PERCC*.

Modelio provides the notion of 'model components'[6] which are defined as either user defined or third party independent and identifiable components that can form parts of a larger model, and are packaged into a single file. Sometimes referred also as *Reusable Autonomous Model* (RAM) components, these components can be used in a collaborative teamwork context, to allow different system designers, developers or development teams working on the same project to work only on a limited part of the complete project model. They also allow components developed by a third party to be integrated in a system model, or to provide a new implementation of an existing system component to be successfully become part of the modeled specifications. Thus RAM components can be viewed as promoting *Intellectual Property* (IP) re-use, one of the most significant aspects of systems engineering.

In the particular case of the TVC solution, the *DCI Interface* block was defined in another instance of Modelio and then integrated as a RAM component in the Modelio project containing the solution model, as seen in Figure 15. The *DCI Interface* block and related modeling elements are packaged into a single RAM component, termed as *DCI*. This RAM includes the *DedCom* interface needed for typing the ports of *DCI Interface* as well as the value type *Radio Frequency*, needed to determine the *band* attribute of the *DedCom* interface.

The DCI RAM component after a successful integration in the original project was instrumental in the design specification of the *PERCC*, as seen in and Figure 11 and Figure 12.

In the case that the *DCI Interface* needs to be modified later on, the modifications can be done in a parallel and separate manner, and then can be integrated in the main project via a RAM component. It is evident that while some modifications may need to be carried out, for example, in the case of modifications of the ports of the *DCI Interface* block, resulting in some changes in the communication links between the *PERCC* and the *DCI Interface*, the changes do not affect the whole system structure.

[6] Modelio Model Components, *http://forge.modelio.org/projects/modelio-user-manual-english-21/wiki*

4.4.5 Allocation aspects

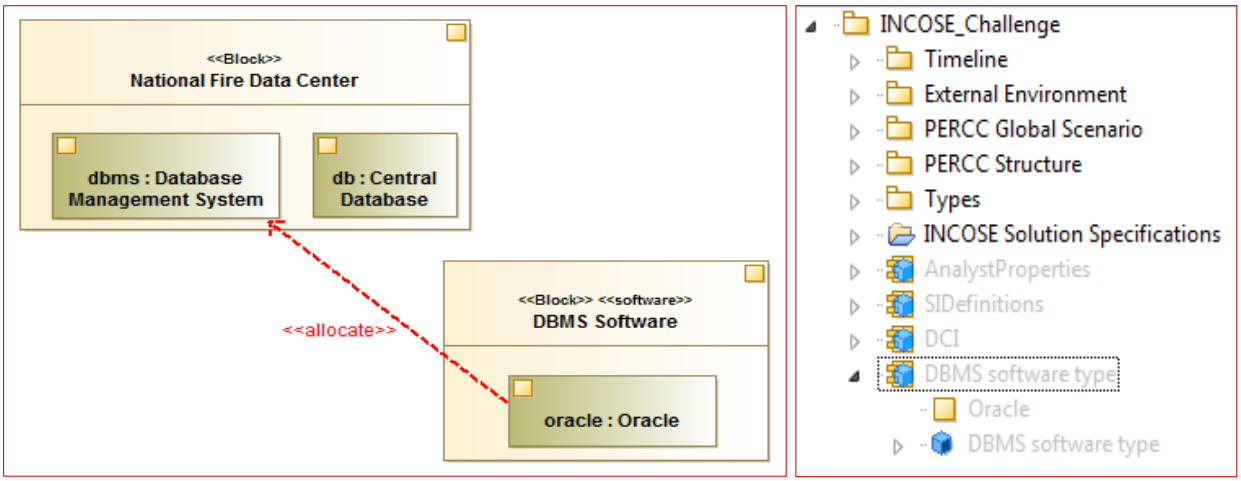

Figure 16 - Allocation of the database software onto the corresponding Database Management System

In Figure 16, we illustrate the SysML allocation mechanism that can be used to map or link source and target modelling elements, for example a software task onto a hardware processing element. Here as seen in right side of the figure, a *DMBS software type* RAM component is integrated into the main project that is meant to illustrate a real-life database such as SQL or Oracle.

Here in this particular case, the *Oracle* block inside the RAMC represents the database to be used in the *Database Management System* of the PERCC. For this purpose, the *Oracle* block is instantiated inside the *DBMS Software* (as the *oracle* instance) and then mapped to the *dbms* instance of the Database Management System via a SysML *allocate* dependency.

This dependency illustrates that the oracle database software has been mapped to the database management system and is being executed on it.

In case of an eventual modification, such as the need to change the current database software to an another; a new RAM component containing the new database software can be integrated in the main project and the existing allocation seen in Figure 16 can be modified to illustrate the new mapping between the database software and management system.

4.4.6 Constraint block and parametric diagrams for Power management

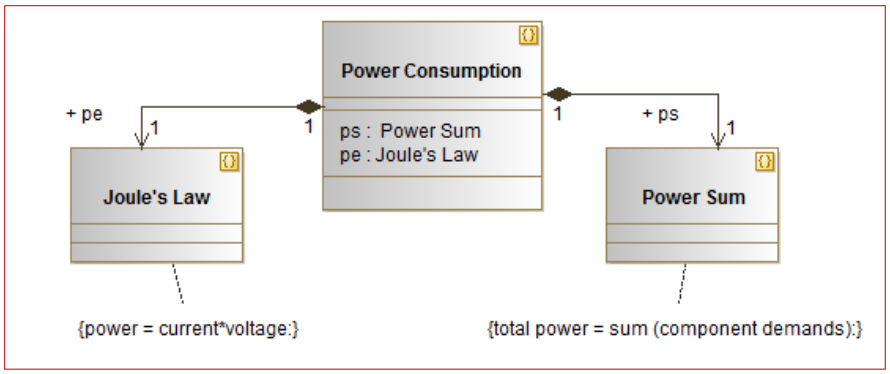

Figure 17 - Constraint Blocks for specifying PERCC power management aspects

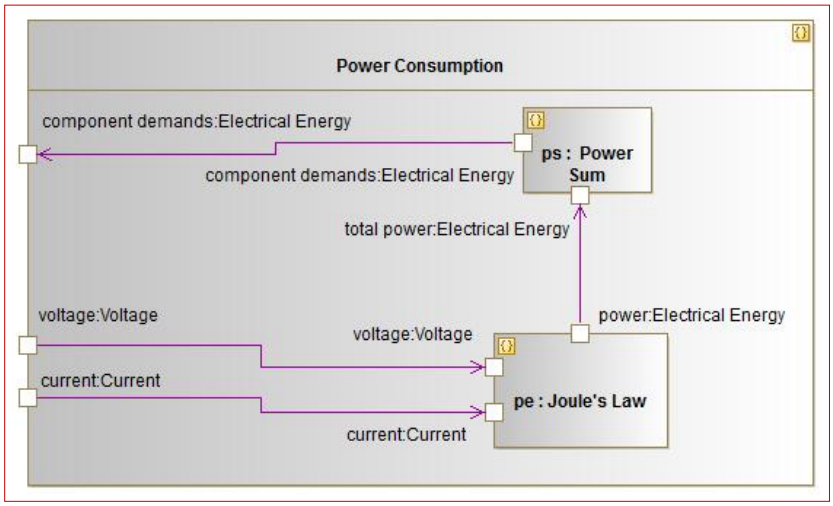

Figure 18 - Parametric diagram depicting the PERCC Power Consumption aspects

In Figure 17 and Figure 18, we describe a constraint block diagram and a parametric diagram to illustrate the power related aspects of the *PERCC*.

Figure 17 illustrates a SysML block diagram with constraint blocks, indicating that the constraint *Power Consumption* is composed of Joule's Law and Power Sum to build a more complex equation. While, Figure 18 shows the parametric diagram for the Power Consumption from the previous figure. The constraint properties *pe* and *ps* are in fact utilizations of the *Joule's Law* and *Power Sum* constraint blocks, respectively. Finally, Figure 19 illustrates how the *Power Consumption* block in used for the construction of the *Power System* block of the *PERCC*.

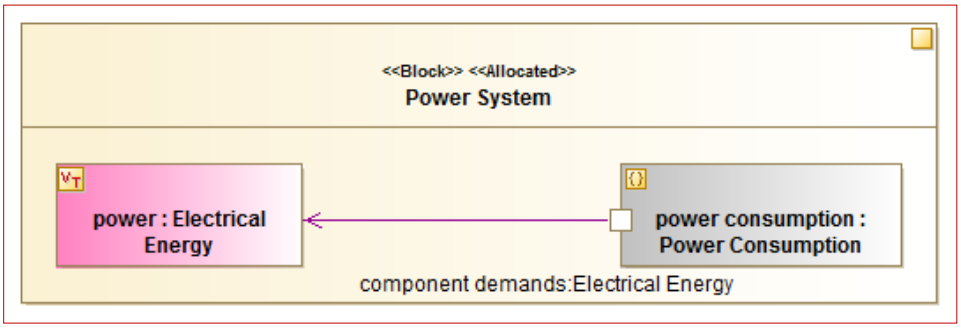

Figure 19 - Internal block diagram of the Power System

4.5 PERCC internal behavior

4.5.1 Global System Activity of the PERCC

We now turn towards the specification of the internal behavior of the PERCC. Figure 20 shows the global behavior of the PERCC via the *Global PERCC System Activity* that is depicted in a SysML activity diagram.

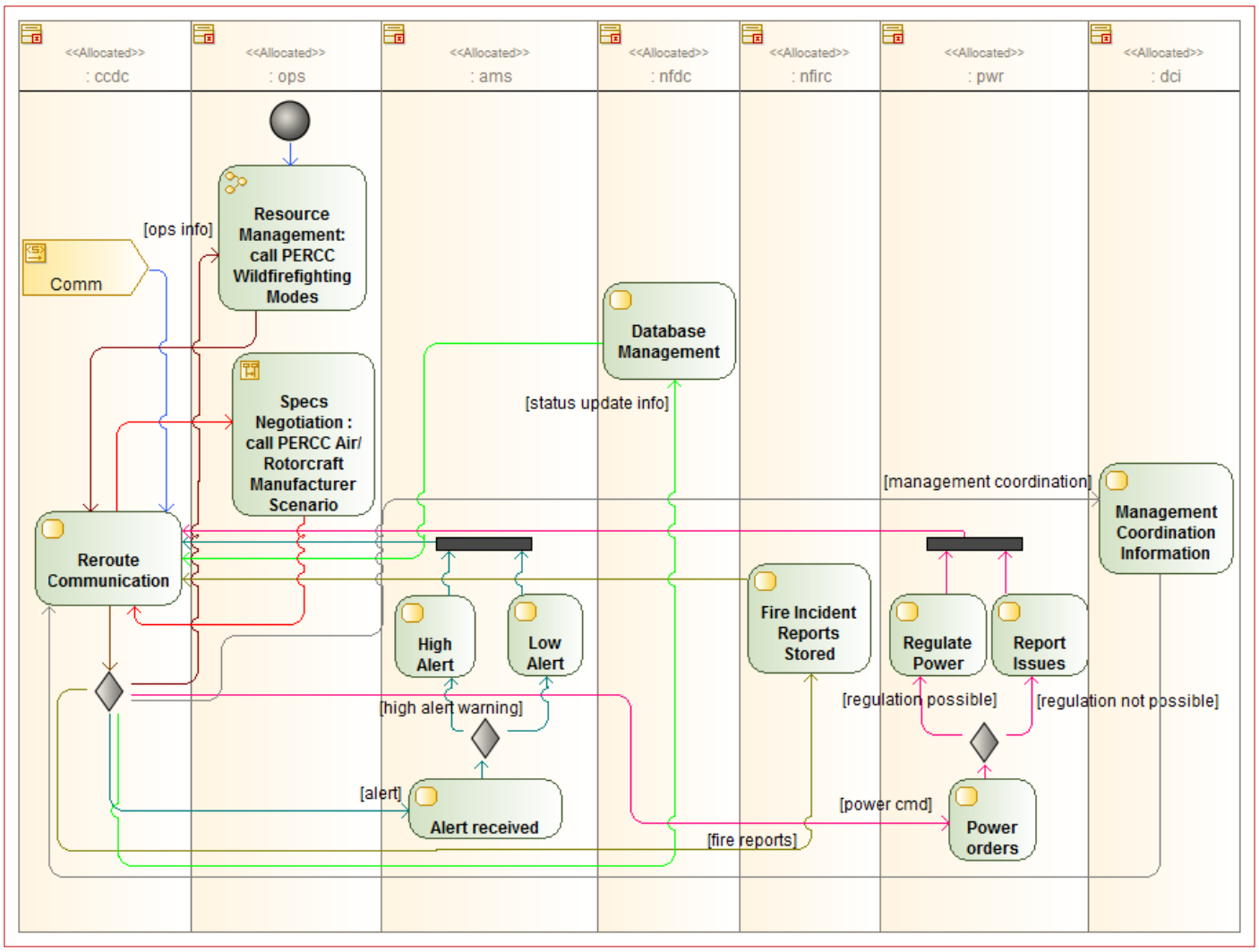

Figure 20 - Global PERCC System Activity

As seen in the figure, each of the vertical partitions of the activity represents a sub-system of the PERCC and is turn allocated to that particular sub-system instance via the traceability management features[7] present in Modelio, by means of the *link editor* window, as seen in Figure 21.

The system activity describes the following actions of the PERCC:

- A communication received by the *Central Communications Dispatch Center* is rerouted to different sub-systems of the *PERCC*, depending upon the type of the signal/communication received. For example, an alert is sent to the alert system while fire reports are sent to the *National Fire Incident Reporting Center*;
- The central *Resource Management* action which in turn calls the operational modes state machine (*WildFire Fighting Modes*) of the *PERCC*. This action is allocated to the *Operations Center* that manages the central

[7] Modelio Traceability Management Features, *http://www.modeliosoft.com/en/features/integrated-requirements-and-vision-support.html*, 2012

resource management actions. Depending upon the information received (alert information from the *Alert Assessment System*, or status information about current resources from the *Database Management System*), the operating mode of the *PERCC* is chosen via the called state machine, detailed later in Section 4.5.2;

- The *Operations Center* also negotiates with the *Air/Rotorcraft manufacturer* via the *Specs Negotiation* action. This action subsequently calls the *PERCC Air/Rotorcraft Manufacturer Scenario* defined in Section 4.6;
- *The Alert Assessment System* after receiving an alert determines the alert priority and sends the information to the *Operations Center*. A high alert means that a wildfire is currently occurring, or there is a strong risk/possibility of a fire that could be started in a short time period (for example, in several hours, the next day, etc.). This could result in the *PERCC* going in either amber or red mode. Alternatively, a low alert signifies that while there may be some adverse weather conditions (for example, a slight increase in weather temperature), they do not indicate a probability of fire, at the instance of the issuance of the alert. The situation may reverse later, resulting in a high alert. In a low alert, the *PERCC* functions at the green operating mode;
- The *National Fire Incident Center* receives fire reports from the firefighting personnel and then sends a consolidated report to the *National Fire Data Center*, which in turn updates its database with the new information;
- The *National Fire Data Center* keeps current real-time information about the firefighting resources and sends them to the Operations Center at pre determined regular time intervals (the time interval set by default is 1 minute);
- The *Operations Center* makes a decision about the resource management for the different *DOCs*, after receiving all the necessary information (such as related to firefighting resources) and sends that information to the *DCI Interface* via the communication center;
- The resource management information received by the *DCI Interface* is then sent to the *DOCs;*
- The *Power System* upon receiving commands from the *Operations Center* regulates the power management operations. If any arriving issues make it difficult or impossible to carry out power regulation (such as fault in power turbines), then the power system informs the *Operations Center* in order for the latter to carry subsequent necessary actions.

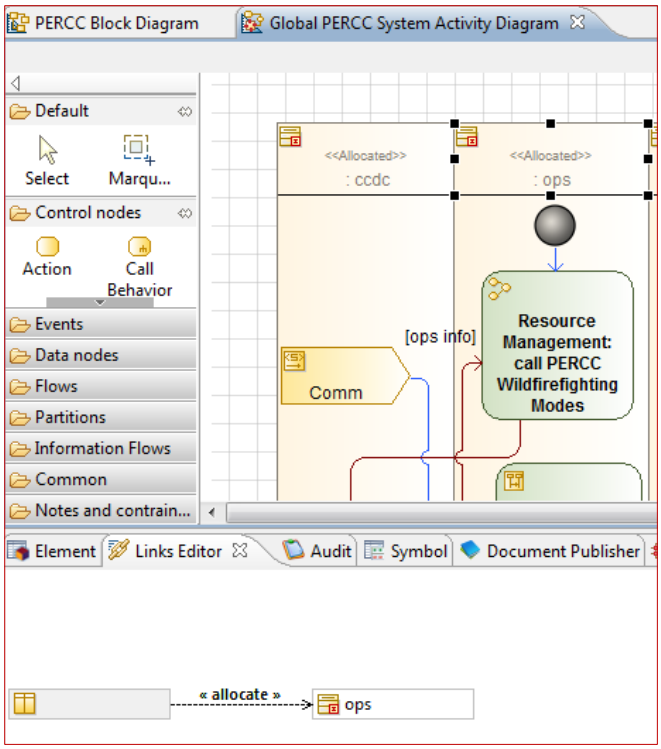

Figure 21 - A screenshot of Modelio traceability management features

4.5.2 Operational modes of the PERCC

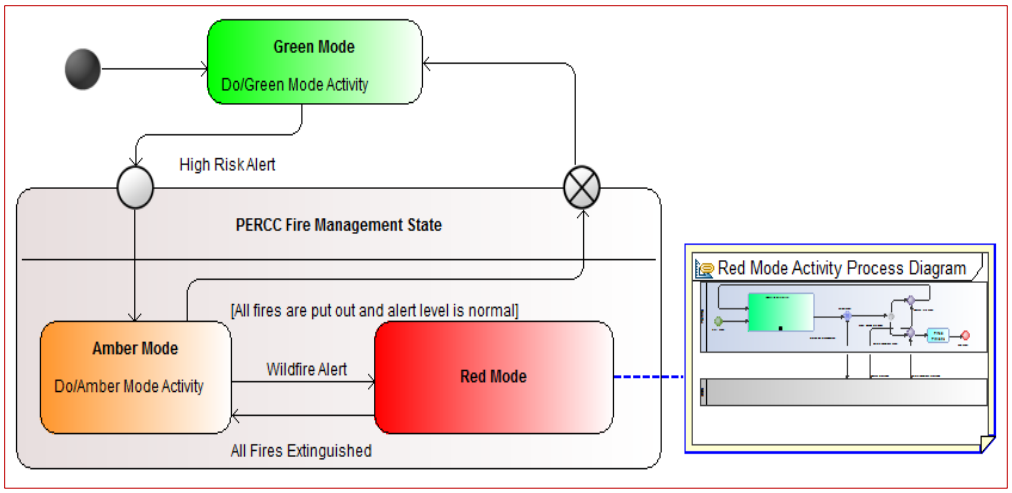

Figure 22 - Mode management state machine of the PERCC

We now describe the operational modes of the *PERCC*. As seen in Figure 22. The operation modes state machine determines the running mode of the *PERCC* at a particular time instance, depending upon the alert received by the *Alert Assessment System*.

As seen in the figure, the default operation mode/state of the *PERCC* is the *Green Mode* and it carries out certain operations when placed in that particular mode, as indicated by the related *do activity* termed as *Green Mode Activity*. The behavior of that do activity is described subsequently in Section 4.5.3.

If the *PERCC* receives a high risk alert, then it switches from the *Green Mode* to the *Amber Mode* and carries out operations related to that particular state. If a wildfire situation arises and the high alert received by the *PERCC* contains information that a wildfire has been started, then the PERCC switches from the *Amber Mode* to the *Red Mode* and commences firefighting coordination between the *DOCs*.

The *PERCC* remains in the *Red Mode* until all fires have been extinguished and then reverts back to the *Amber Mode*. The *PERCC* remains in that mode until the risk of wildfires has passed and the alert has returned back to a normal/low level, prompting a switch back to *Green Mode*.

It should be noted that *Red Mode* has a '*related diagram*' notation attached to it, which expresses the related behavior for this particular state.

This feature enables designers in Modelio to link a source modeling element to a related diagram and helps to increases easiness with project navigation.

4.5.3 Green mode of the PERCC

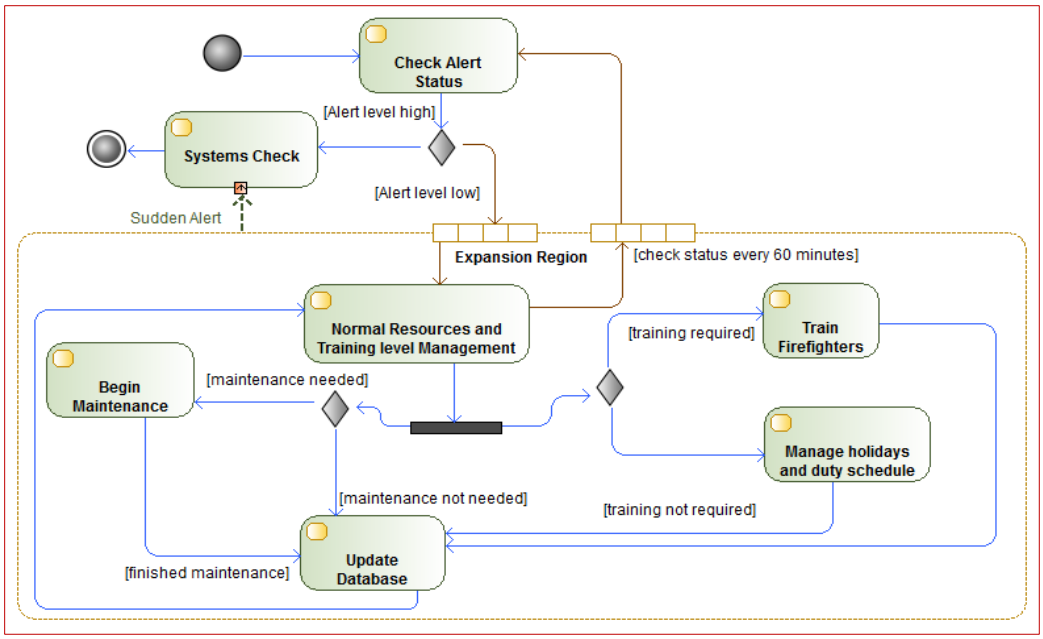

Figure 23 - Green Mode Activity of the PERCC

We now look at the underlying green mode of the *PERCC*, as seen in Figure 23. The *Green Mode Activity* determines the internal operations when the *PERCC* is operating in that particular mode. As stated in the requirement, the system initially remains in the green mode until a high/risk alert is issue, necessitating a mode switch. This aspect is represented by the decision-merge node between the *Check Alert Status* action and the *Expansion Region*.

In case of a normal condition, we enter into the expansion region and carry out management activities related to the resources, alternatively, we carry out a systems check action before switching to amber mode.

In the expansive region, we carry out resource management related to the firefighting personnel and the related equipment, vehicles and air/rotorcrafts. Fixed normal working hours for firefighting personnel are assumed during this mode, such as from a time duration of 9 am to 6pm.

Training may be carried out to increase expertise of personnel which are currently available (as compared to personnel on holidays, sick leaves, etc.). Equally, holidays and rest durations are assigned to firefighting personnel, and the information is regularly updated and sent to the database of the *National Fire Data Center*.

In addition to checking the status of the firefighting personnel, the status of the equipment, vehicles and air/rotorcrafts is also monitored, as stated before. In case of any required maintenance (such as refueling or repairing), the maintenance operation is carried out and information is sent to the database to apprise the *PERCC* of the change in resource status.

Finally, as seen from the initial requirements, the *PERCC* remains in the green mode until it receives an urgent alert, requiring a mode switch. Additionally, the possible presence of an alert is checked every hour at regular intervals, in order to ensure effective alert monitoring.

4.5.4 Red Mode of the PERCC

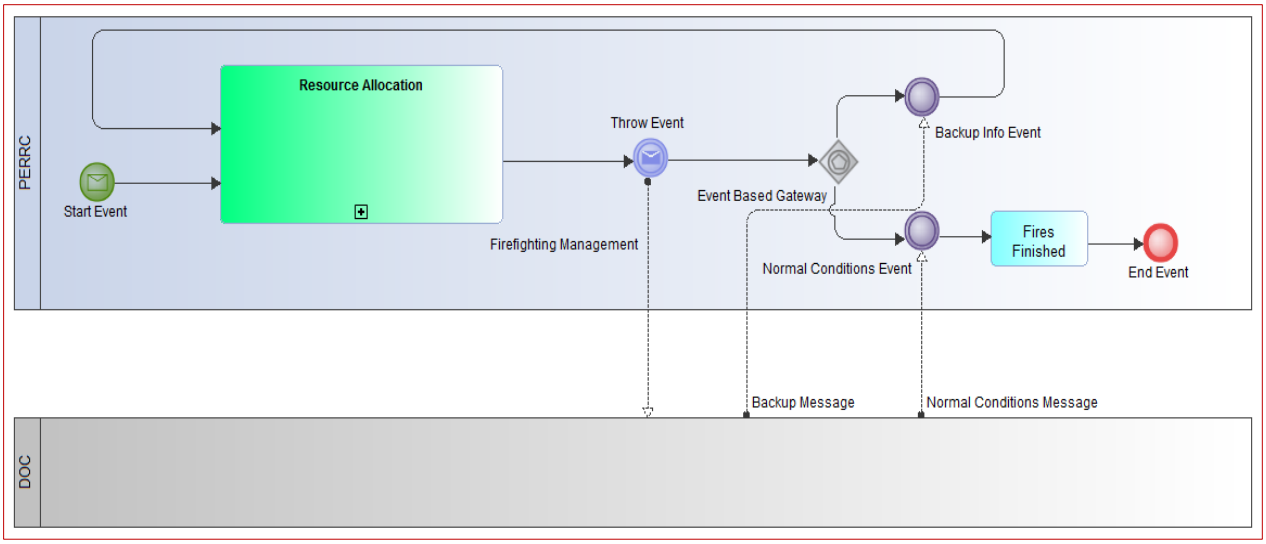

Figure 24 - Red Mode (BPMN) Activity of the PERCC

For the particular case of the INCOSE 2012 TVC, we specify the *Red Mode Activity* by means of a BPMN Activity Diagram. The purpose being to illustrate that Modelio can be used to help designer bridge several standards, when creating their system specifications.

Thus, different designers can express their ideas by using distinct norms they are familiar with, in order to conceive the global system. For example, designers can describe the business processes like change management with standards such as BPMN; information architecture with TOGAF [5]; and hardware/software aspects of the system with standards such as SysML and MARTE [6]. Modelio enables to link the different system aspects and enables to increase the overall system comprehensibility.

As seen in Figure 24, semantics of BPMN are utilized to describe the operations of the red mode. The two *PERCC* and *DOC* pools show the related entities.

When the *PERCC* receives a high alert, it carries out a resource allocation algorithm (as indicated by the *Resource Allocation* sub-process) and sends necessary resource coordination information to the different *DOCs*. A *DOC* after receiving the necessary information, carries out command and control of its respective firefighting resources, such as personnel, vehicles and aircrafts.

If a *DOC* is able to put out all fire successfully for which it was responsible , then the *DOC* sends a message to the *PERCC* which in turns exits the red mode (after receiving such messages from all the 11 *DOCs*).

In case a *DOC* needs backup, such as additional firefighting resources, it signals the *PERCC*, which then could allocate the additional resources to the particular *DOC*.

4.6 Communication with external environment (the case of Air/Rotorcraft manufacturer)

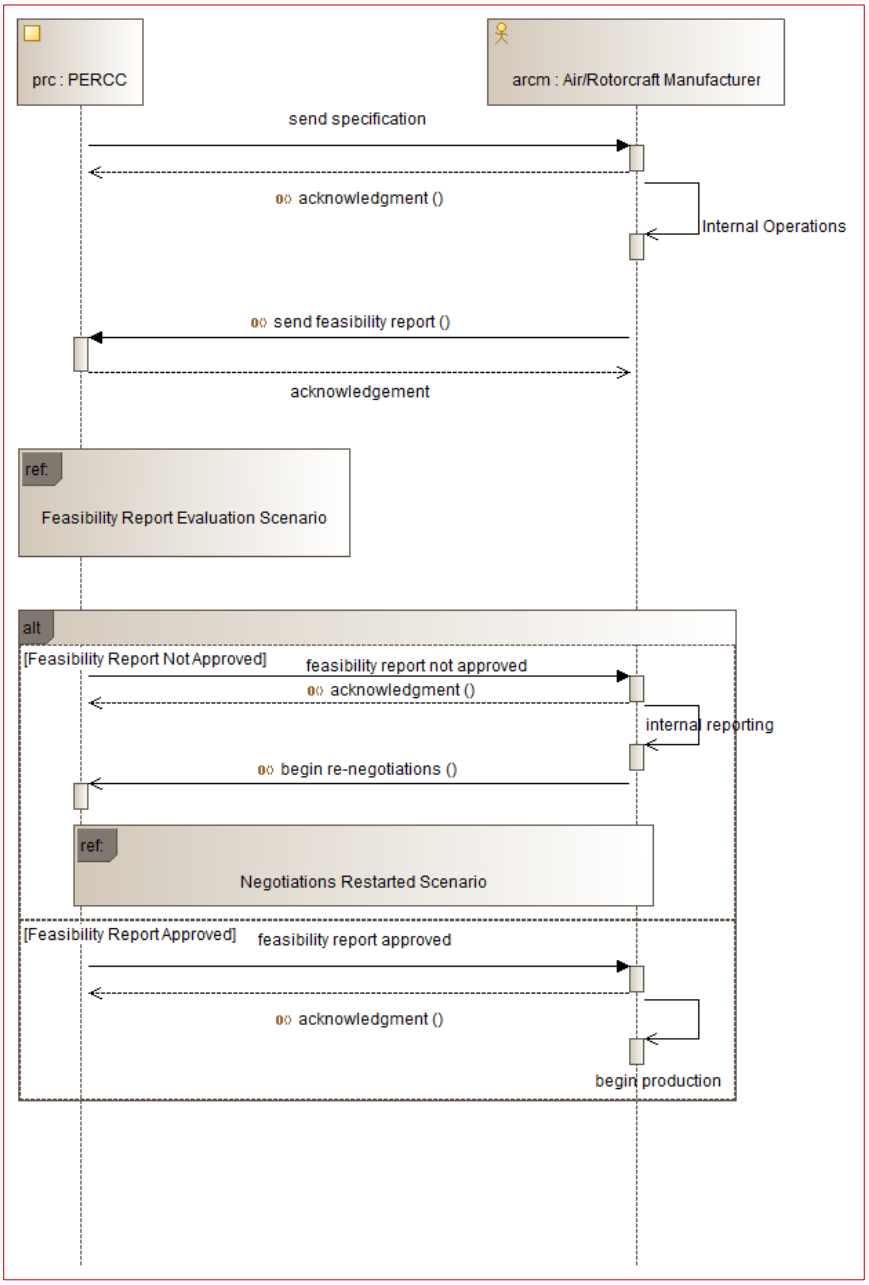

Figure 25 - PERCC negotiating with the Air/Rotorcraft Manufacturer

Figure 25 depicts the interaction between the *PERCC* and the *Air/Rotorcraft Manufacturer*. The *PERCC* initially sends the necessary specifications to the manufacturer (as indicated by the specified requirements), which in turn carries out an initial assessment and sends a feasibility report to the *PERCC*. The *PERCC* in turn evaluates the feasibility report regarding details such as costs, specifications etc.

If the *PERCC* accepts the report, it sends an approval to the manufacturer to begin the air/rotorcraft manufacturing. Alternatively, the manufacturer may need to make some changes (such as making comprises on the manufacturing costs), and then re-initialize the negotiations.

4.7 Timeline of the system

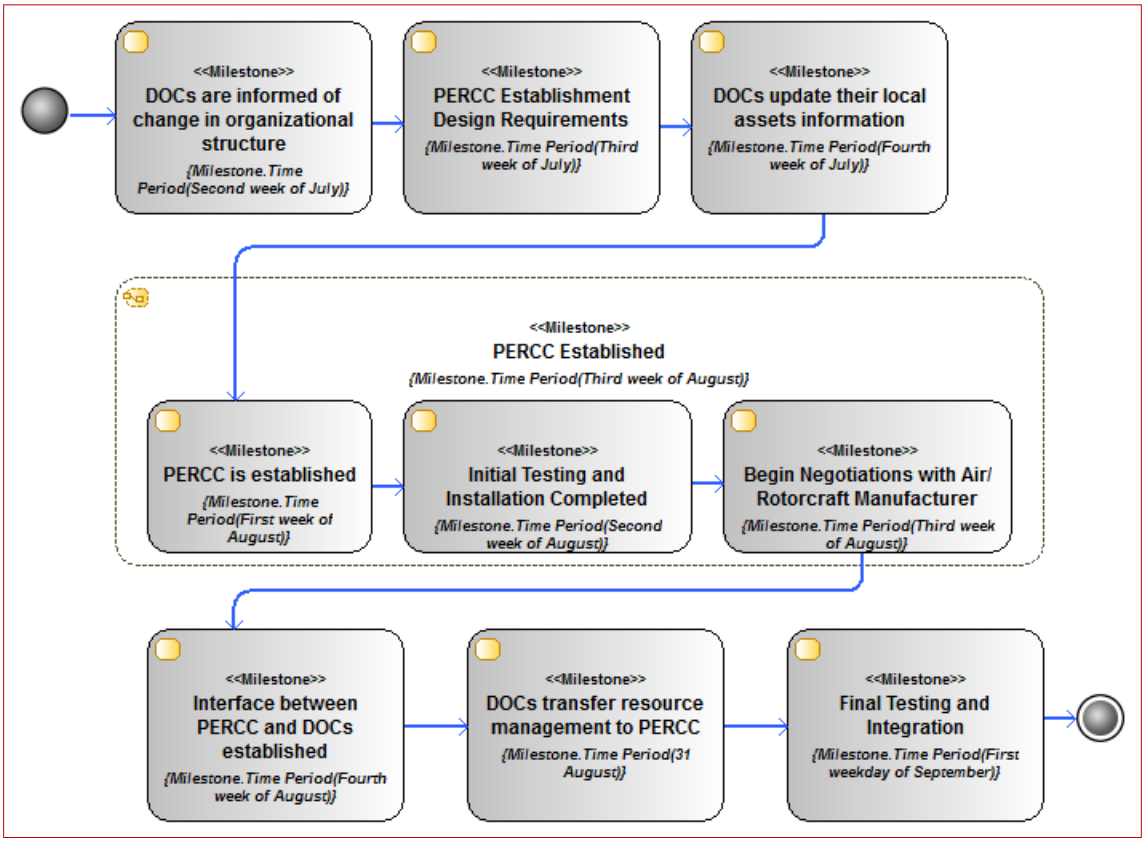

Figure 26 - System timeline

Figure 26 illustrates the timeline for the specification, establishment and deployment of the *PERCC*. For this purpose, a user defined *Milestone* stereotype (with a *Time Period* tagged value) was specified to determine the system life cycle.

As can be seen in the figure, initially the *DOCs* are informed of the change in the organizational structure (such as them giving up the resource allocation management).

This is followed by specification of the *PERCC* design requirements and an update of the *DOC* local asset information. Once the *PERCC* is established, then initial testing is carried out (such as testing integration of the different sub-systems) followed by a negotiation with the *Air/Rotorcraft Manufacturer* to produce the water bombers.

This step is followed by construction of a dedicated communication interface between the *PERCC* and the *DOCs*, resulting in the *DOCs* sending their local asset information to the *PERCC*, in order to create a global consolidated resource management database.

Afterwards, final testing and integration between the *PERCC* and the *DOCs* is checked before the *PERCC* is ready to be deployed.

4.8 Updated requirements for the PERCC

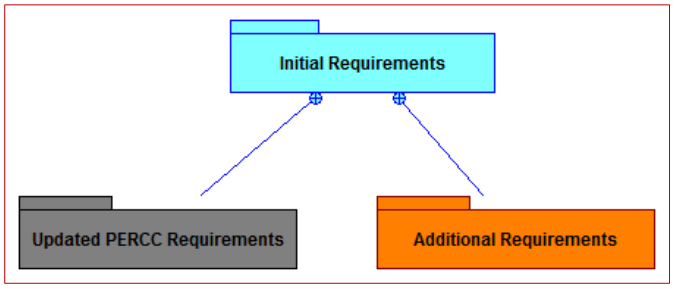

Figure 27 - PERCC updated requirements

The INCOSE 2012 TVC provided a challenging twist, as the initial requirements were ultimately modified and the solution providers were asked to accommodate the new requirements in their solutions. This is like any real-life scenario, where designers may need to change/modify the initial requirements to better suit their desired objectives.

The official statement regarding the updated requirements is as follows:

"The Ministry of Interior wish to apply the same operational concept of *PERCC* for other disaster recovery activities, for example: floods, earthquakes, industrial or nuclear accidents. The tool vendors were asked to:

1. Identify system requirements impacted by the decision and update them;
2. Propose an architecture for an "*Earthquake PERCC*" and for a "*Flood PERCC*", identifying commonalities and specificities;
3. Perform an architecture tradeoff analysis between one unique *PERCC* and one for each type of threat.

They were also asked to demonstrate the management of common and specific requirements as well as sub-system architecture, and how change management could be performed in the coming years."

As seen in Figure 27, the updated requirements were integrated in the requirements phase. Some modifications were carried out for the successful integration, as indicated in the left side of Figure 28. Impact of these new modifications can be evaluated by means of traceability management features, as described later on in Section 4.9.4.

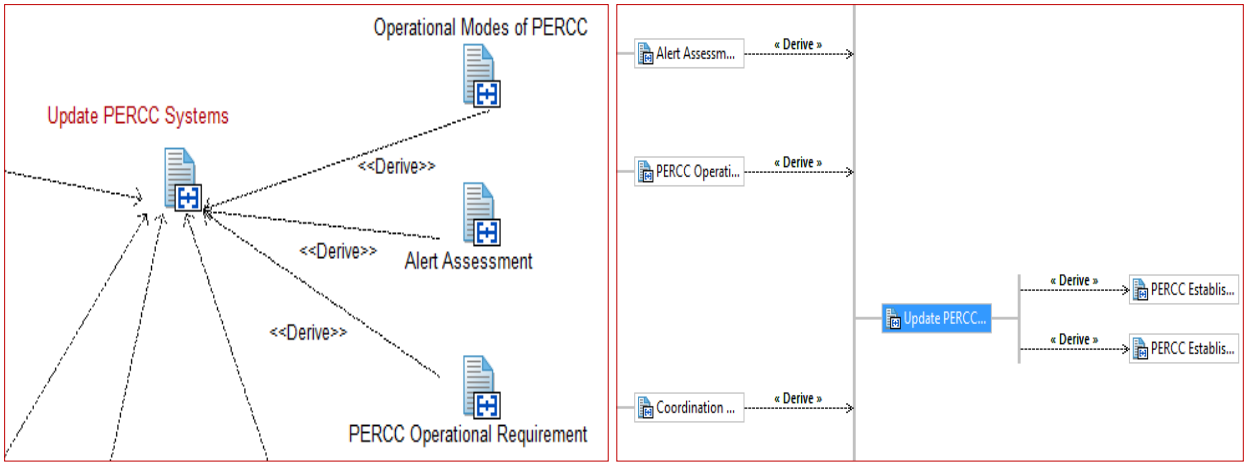

Figure 28 - Extract of the updated traceability view and system requirements view

The initial PERCC system specification was designed as a generic national emergency response center, in such a manner to be compatible with any eventual system modifications. For this reason, we were not required to make any major drastic changes with the inclusion of scenarios such as related to earthquakes, floods, etc. The changes to the system specifications due to the updated requirements are as follows:

- The PERCC Block Structure is modified to include three additional sub-systems: *WildFire System Implementation*, *Flood Management System Implementation* and *Earthquake Management Systems Implementation* blocks, as seen in Figure 29. Each block and its underlying behavior can be seen as an implementing a distinct course of action, when dealing with a particular scenario. For example, in case of wildfires, water bombers could be given priority and utilized to put out fires, while in case of a flood, they can be used for scouting purposes. Each block has an underlying behavior similar to the operational modes state machine seen in Figure 22.

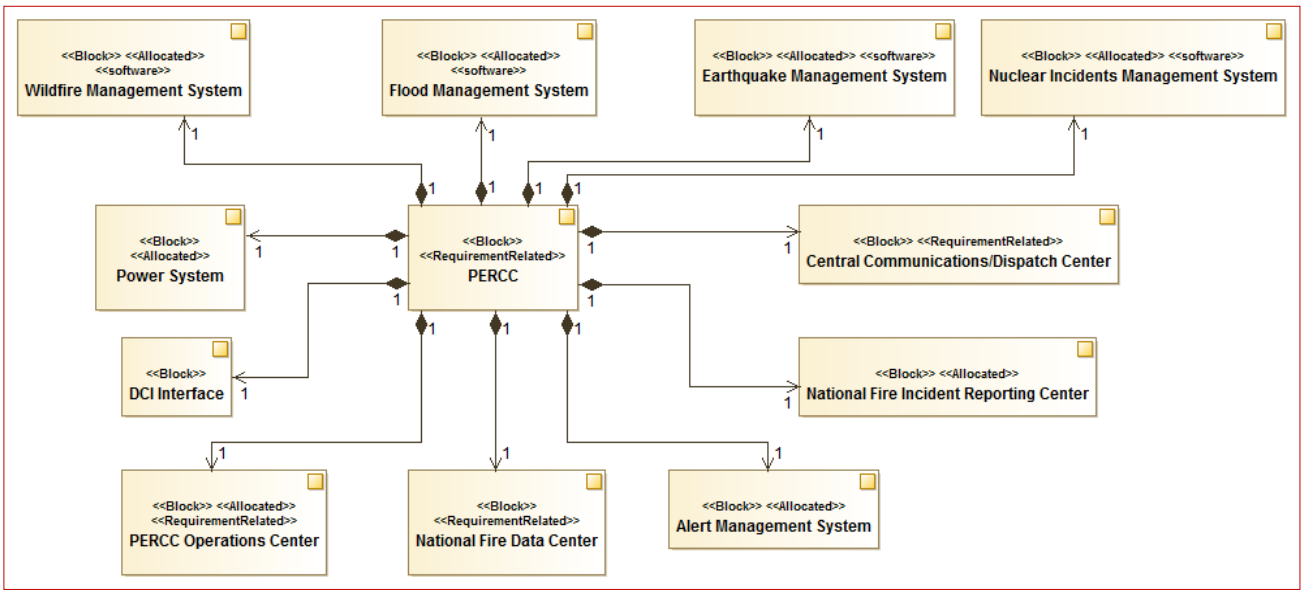

Figure 29 - Updated PERCC block structure diagram

- The *Resource Management* action of the *Global PERCC System Activity* now calls a *PERCC Operations Activity*, displayed in Figure 30. This activity, as seen in Figure 31, enables executing a particular firefighting scenario based implementation. Each implementation in turn, can call its particular mode state machine of the *PERCC* for managing the global system modes. Moreover, as seen in the related traceability 'link editor' window, each of the 'implementation' actions is allocated to a particular sub-system (for example, the *WildFire System Implementation* action is allocated to the *Wildfire Management System*). This allocation creates link between the previously mentioned blocks and the different actions.

Hence, the different sub-systems/behaviors are just used to determine the firefighting scenarios and the optimal resource allocation techniques. This information in turn is then sent to the respective mode state machine for effecting a mode change. The mode state machines for the different additional scenario-related blocks (floods, earthquake, etc.) are similar to the initial mode state machine depicted in Figure 22, albeit with minute changes: for example, in the state machine for managing modes for combating floods, the transition to switch from amber to red mode has the *Flood alert* condition.

These minute modifications in turn illustrate that our initial design specification was robust enough to accommodate changes carried out later.

Unfortunately, up to the writing of this white paper, Modelio does not support architecture tradeoff analysis aspects. For this reason, this aspect has not been elaborated upon in this document.

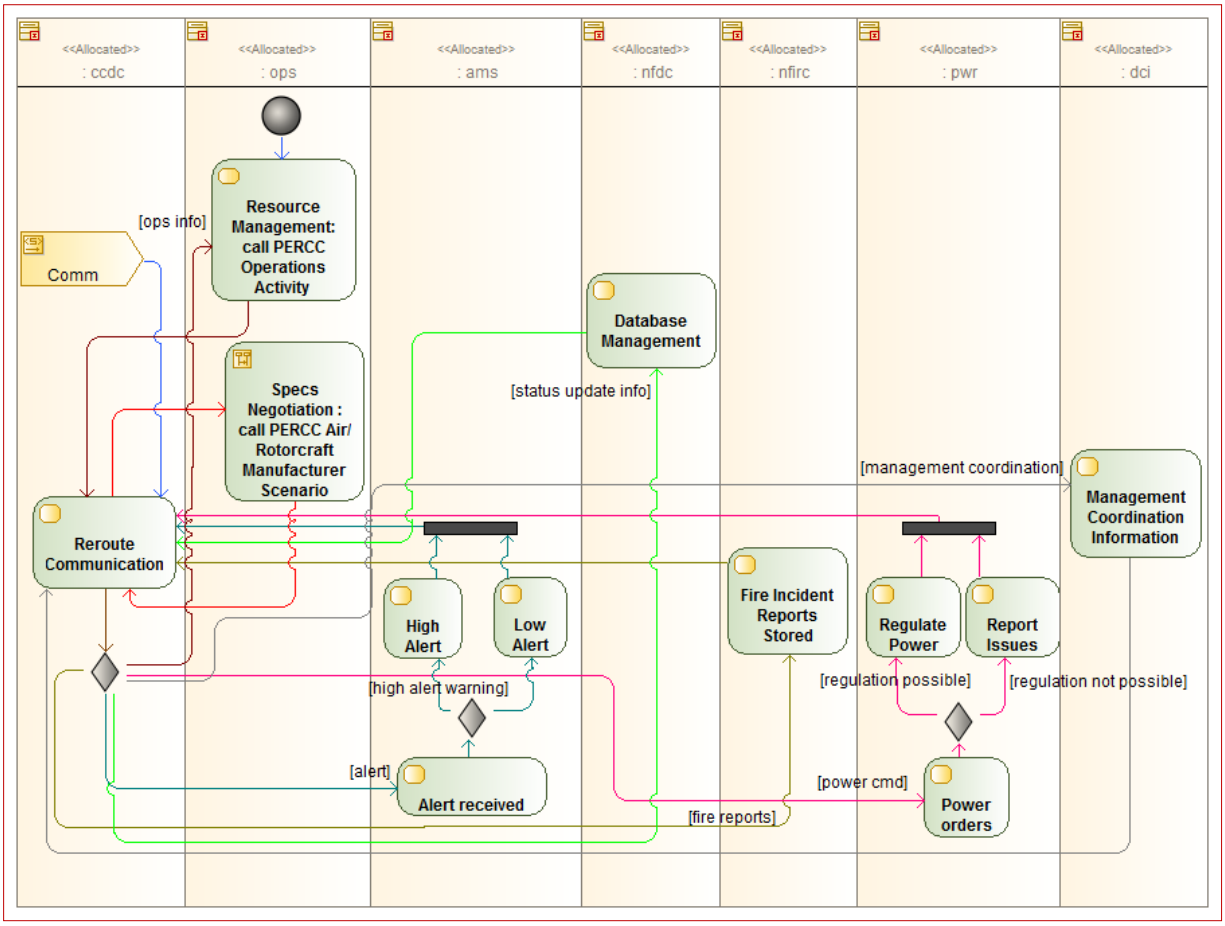

Figure 30 - Updated Global PERCC System Activity

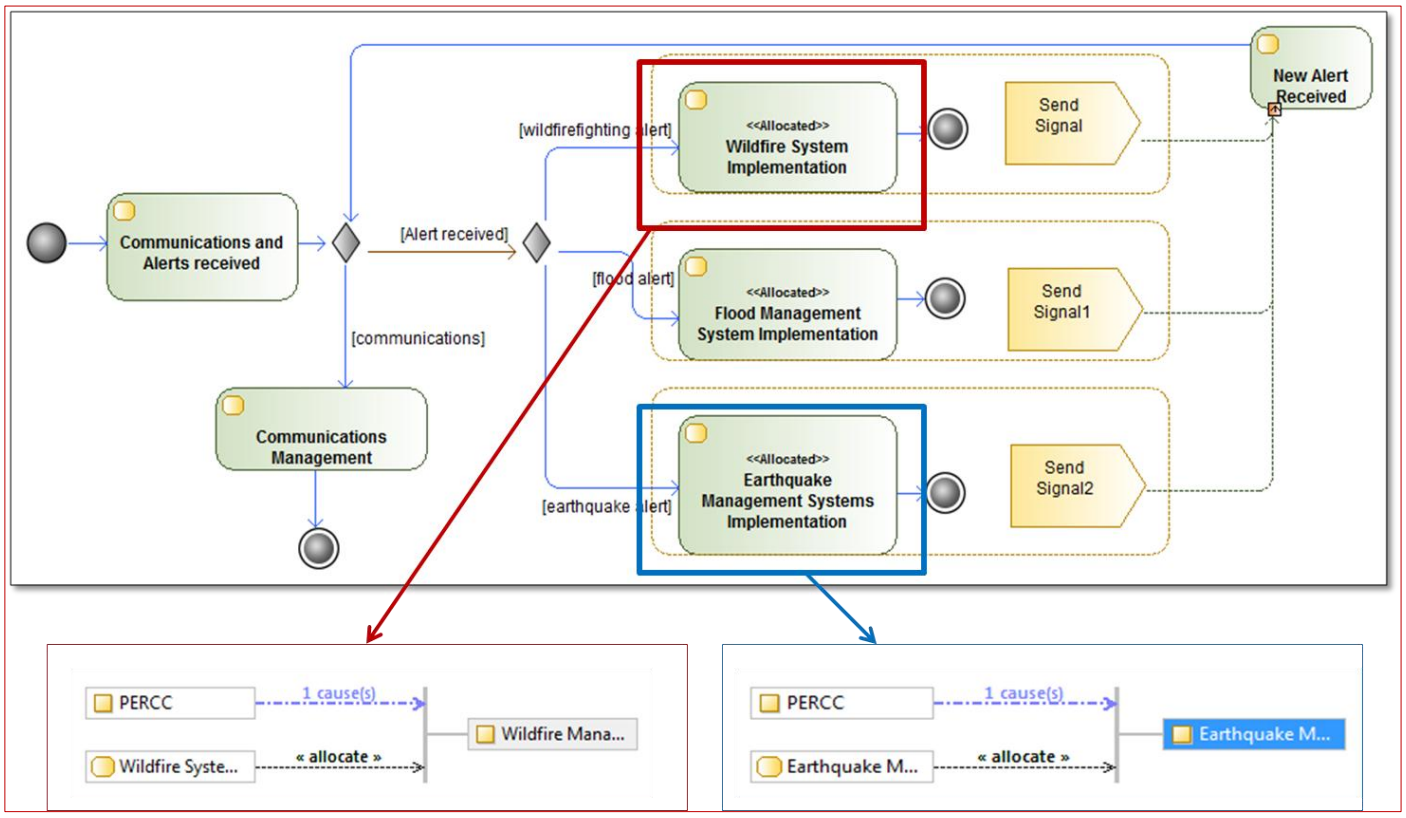

Figure 31 - PERCC Operations Activity

4.9 Modelio added features for increasing design productivity

4.9.1 Automatic Impact Analysis

We now describe the novel features in Modelio, which can be used to increase design productivity of system designers, such as automatic Impact Analysis of modeling elements, generation of diagrams and documentation among others.

We first describe the Impact Analysis feature of Modelio. This feature enables identifying of consequences of a change to a requirement, goal, business rule or dictionary. For example, in the particular case of our TVC solution, we selected the *National Fire Data Center* for impact analysis, in order to see possible consequences if that particular block is modified or removed.

As shown in Figure 32, the Impact Analysis results illustrate that the *PERCC* depends on various modeling aspects of the *National Fire Data Center* block for its own composition. This summary can help designers to deduce any possible consequences/effects that could arise by changing either a sub-system or the system itself.

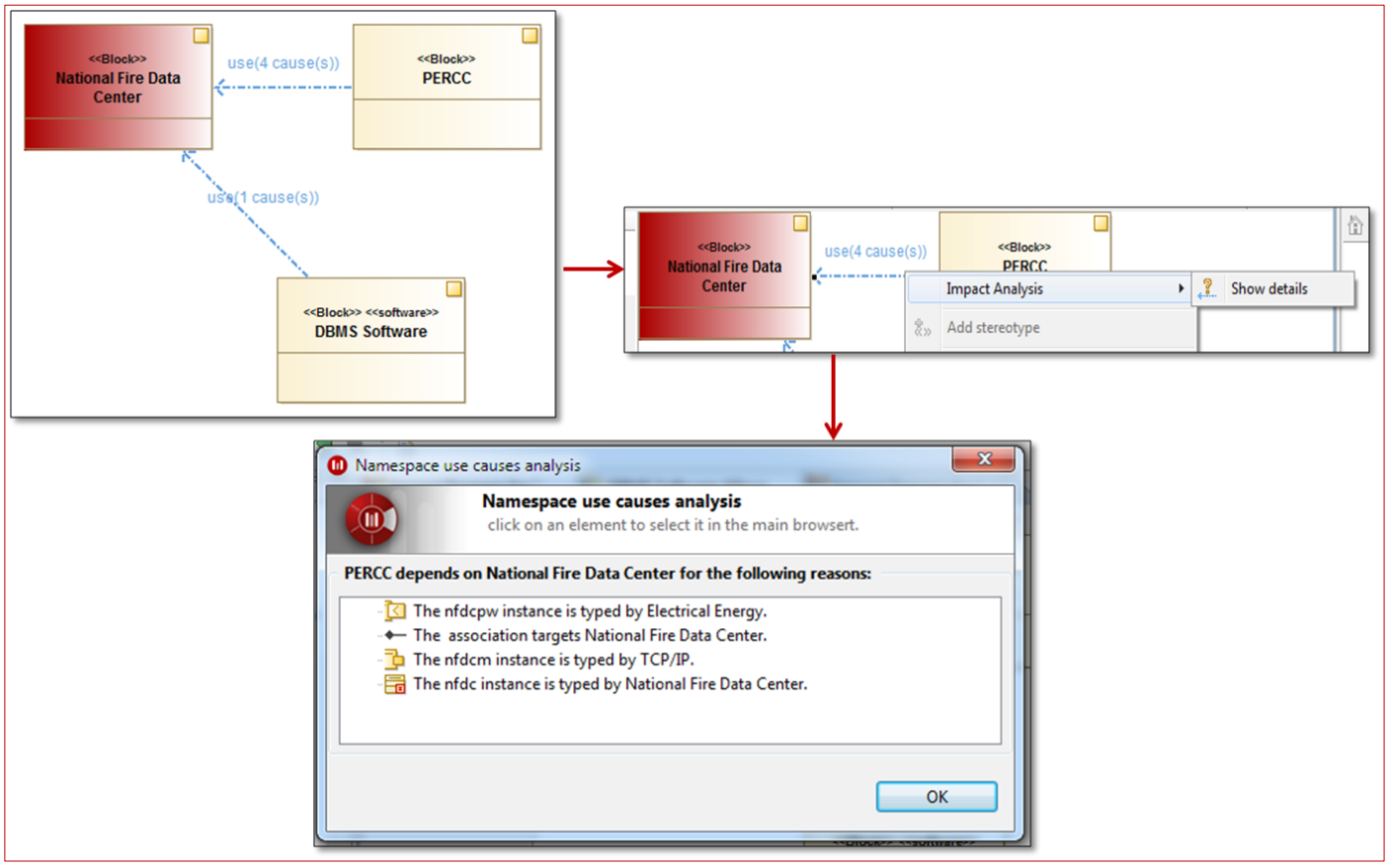

Figure 32 - Impact Analysis of the National Fire Data Center block

4.9.2 Automatic Dependency diagram creation

Automatic diagram generation is an extremely useful function of Modelio, notably if massive code reverse operations are run, during which existing code can be transformed into a model in order to analyze its design or documentation production. Modelio proposes a pragmatic approach to automatic diagram creation, as normally designers may find themselves asking the question of what modeling aspect of the system to show and how to show it effectively.

Modelio thus enables a limited number of generated diagram types, whose contents and presentation are precisely defined, and whose usefulness and use can be predicted. The needs of users on this type of element can be split into two categories:

1. The need to analyze the structure of the element, in other words, what is it made up of and what does it contain;
2. The need to analyze the position of the element and its role within the system. In this case, it is a Dependency/Impact Analysis.

Modelio enables automatic generation of several diagram types, which are as follows:

- Dependency diagram;
- Inheritance diagram;
- Class structure diagram;
- Package structure diagram;
- Sub-package diagram;

We now describe the functionality of a Dependency diagram, as seen in Figure 33. The Dependency diagram can be seen as an inverse of the Impact Analysis diagram, as it indicates how a modeling aspect (such as a system block) is dependent on other parts of the system in question. It shows the outgoing and incoming dependencies of a modeling element.

In the figure, the *PERCC* block depends on several sub-systems and interfaces for its composition. It should be noted here that modeling aspects (sub-systems/interfaces, etc.) defined by the designer are visualized differently, for example here, they are colored in pink, while those defined by a third party are colored in yellow. This enables to create a clear distinction between the internally/externally defined system aspects.

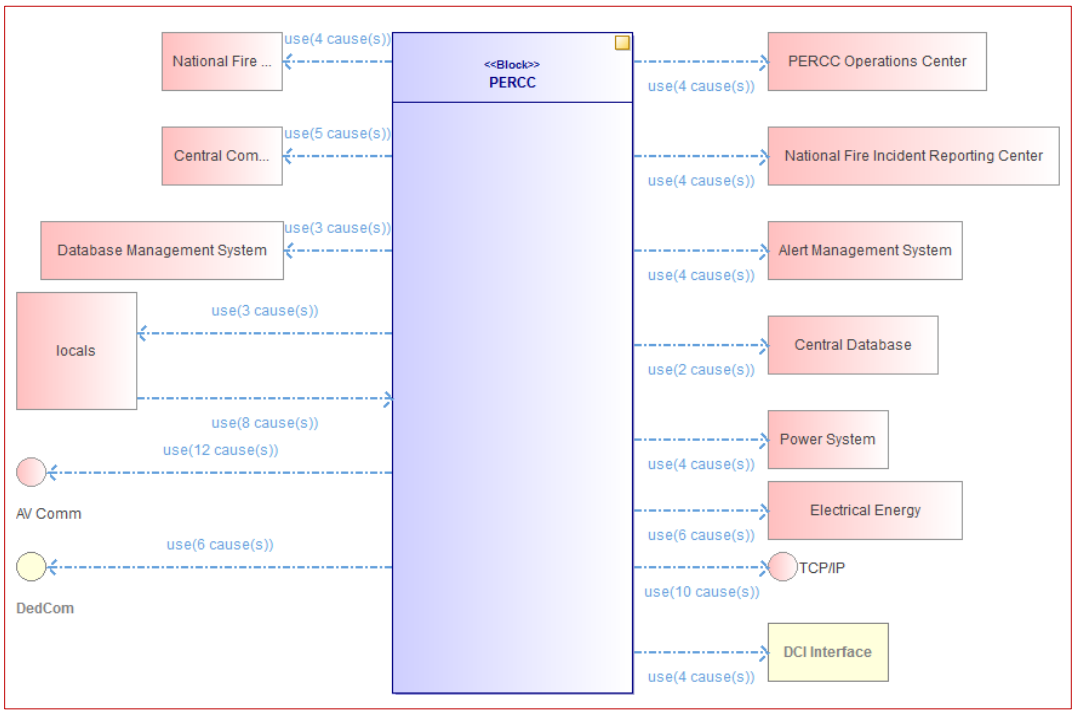

Figure 33 - Dependency diagram for the PERCC

4.9.3 Automatic Package Structure Diagram Creation

Similarly, we illustrate the automatic generation of a Package Structure diagram, as seen in Figure 34. The aim of this automatic diagram is to show the internal structure of a package, in other words the contents of the package, along with the links that exist between its sub-packages. It is often these links which establish the package's encapsulation function.

Here, in the figure, we generate the package structure of the INCOSE Challenge root package of our modeled project, which contains a hierarchical organization formed by the different underlying sub-packages.

The different sub-packages are:

- External Environment: A sub-package containing the external actors;
- PERCC Global Scenario: containing the use cases related to the PERCC;
- PERCC Structure: containing the PERCC Block, its sub-systems and underlying behavior;
- Timeline: detailing the timeline of the PERCC;
- Types: depicting the user define value types and interfaces, necessary for construction of the PERCC.

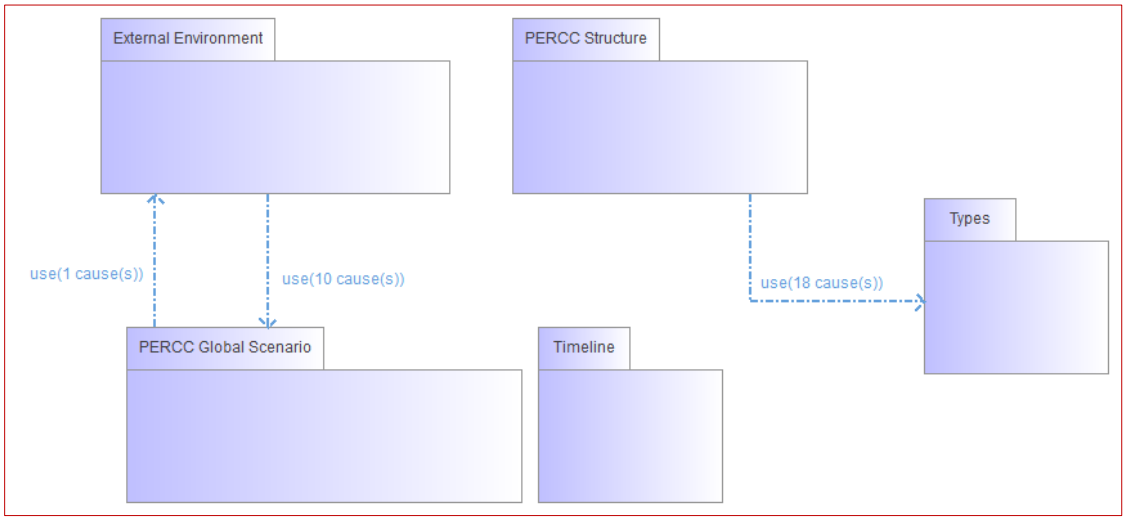

Figure 34 - Package Structure diagram

4.9.4 Traceability Management features in Modelio

We now illustrate the traceability management features in Modelio, as seen in Figure 35 and Figure 36. This feature enables creation of traceability links between different modeling elements (for example, between requirements and system blocks).

The orientation of the created link (whether it is from the central node's element to the dropped element or vice versa) is defined by the point at which a designer drops the element, with relation to the central node. The links editor tries to determine the type of the created link based on the types of link visible at the time of the drop (such as (generalizations, associations, dependencies, etc), If several types among those visible are valid, a dialog will open, asking the designer to choose the exact type to use for the creation.

Additionally, different upstream/downstream levels can be displayed, as indicated by a one-level upstream level in Figure 35, and two up-stream levels in Figure 36, with *PERCC* as the central node. Finally, usage guidelines about the traceability management features are can be found on the Modelio wiki[8].

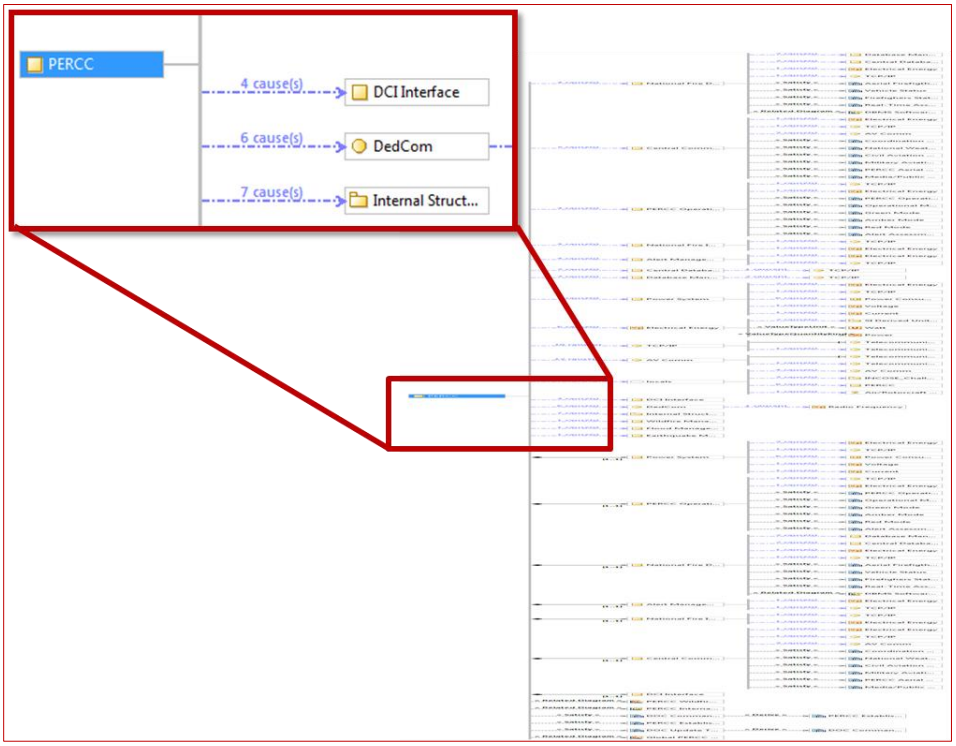

Figure 35 - Traceability features in Modelio (1 up-stream level view)

[8] Modelio 2.2 User manual, *http://forge.modelio.org/projects/modelio-user-manual-english-22/wiki/Modeler-_modeler_interface_linkeditor_view*, 2012

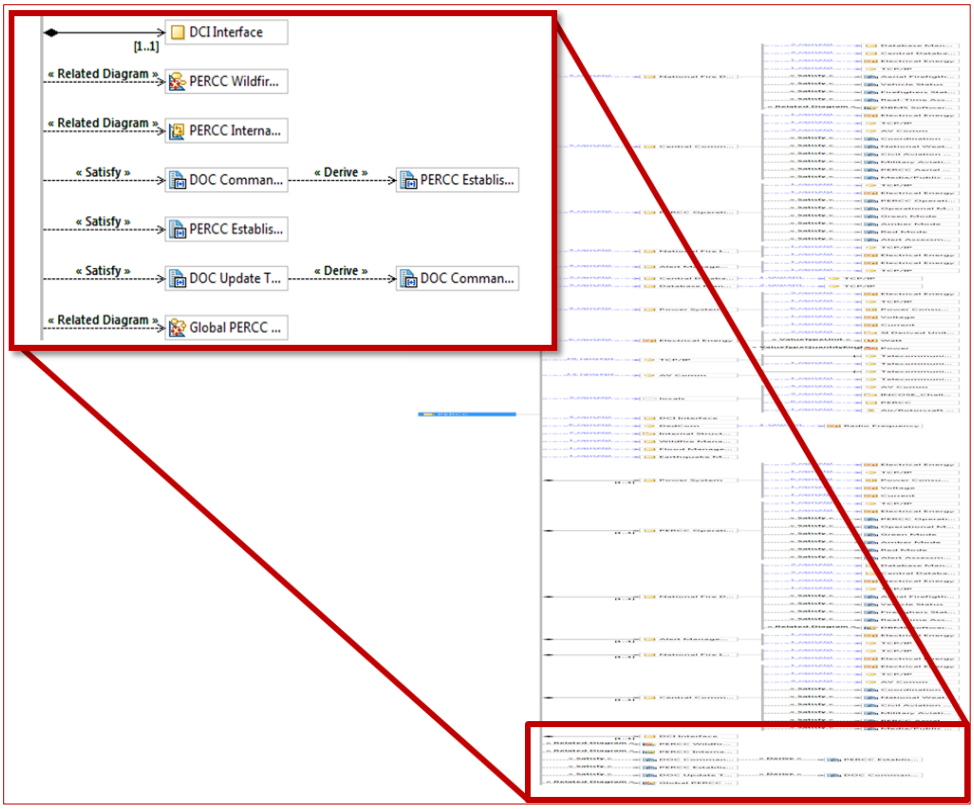

Figure 36 - Traceability features in Modelio (2 up-stream levels view)

4.10 Document generation capabilities

4.10.1 Static document generation

Modelio Document Publisher makes it easy to write and produce quality documentation, simply by making the most of the models built by a systems designer. By providing assistance in what he should write, this module ensures that a designer needs only write what is truly necessary, before automatically producing documents of a quality far superior to those created manually.

For example, a designer can generate detailed documentation that includes a glossary, requirements, a section dedicated to use cases, a section dedicated to class diagrams and a traceability matrix, with hypertext links systematically included, as seen in Figure 37. Modelio also permits designers to define their own document types with ease, by means of a document template editor. The template editor uses a dedicated graphical editor and is utilized for the creation of *document templates*: A document template can be viewed as a detailed table of contents that describes the form of the document, which the designer wishes to generate from the modeled specifications.

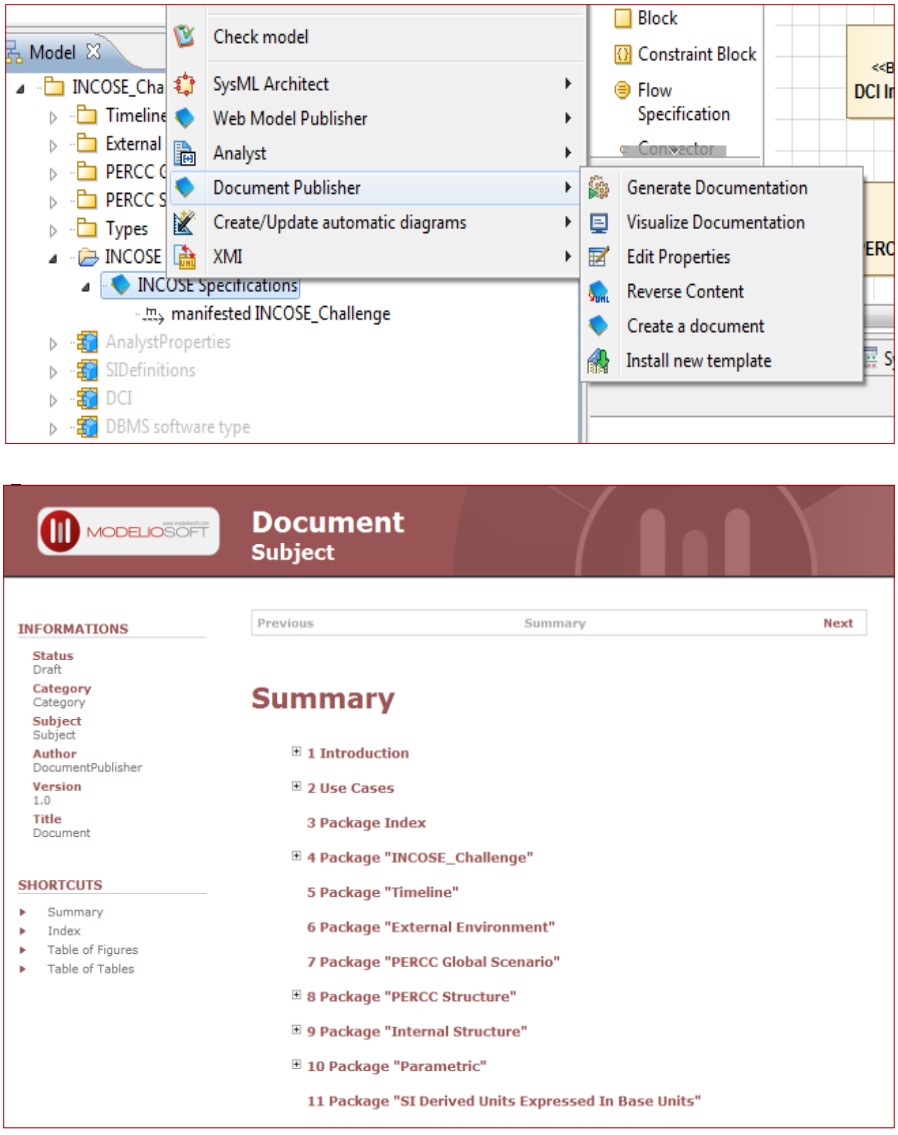

Figure 37 - Document generation features in Modelio

4.10.2 Web Model generation

The Web Model Publisher is an interesting feature of Modelio, as it enables designer to review their complete project (present with the different hierarchical levels, modeling elements, description, etc.) without the need of a Modelio installation. This can be useful for collaborative reviewing purposes, as a designer can send his design to a senior systems architect who in turn can review the whole project and send his reviews to the original designer for eventual design modifications.

The Web Model Publisher module browses a modeled specification and produces a complete and comprehensive documentation in HTML. Any description notes provided by the designer are displayed within the diagrams. Additionally, hyperlinks are generated from the model, within diagrams, chapters and texts, as seen in Figure 38.

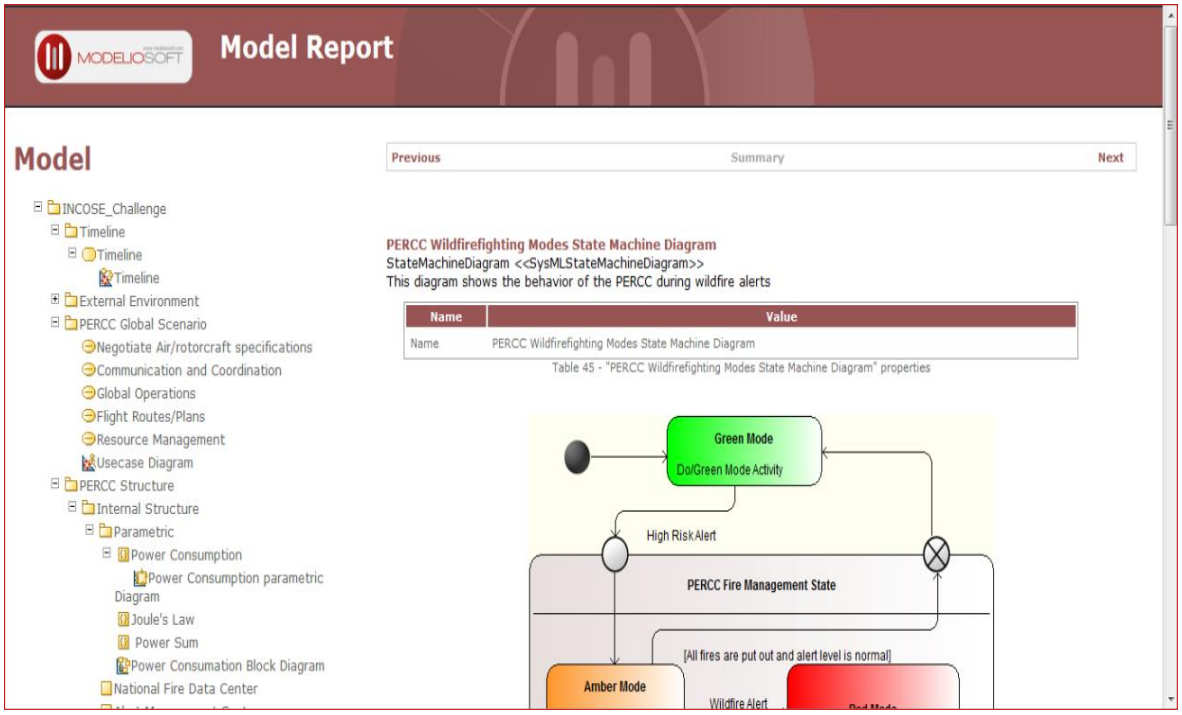

Figure 38 - Web Model creation in Modelio

5 Conclusion

This white paper provides the solution for the INCOSE 2012 TVC specified in Modelio. We illustrated how different systems engineering design phases can be easily carried out via our modeling environment, such as requirements engineering, structural/behavioral modeling, hardware/software allocation along with additional features such as impact analysis, traceability management, automatic diagram and document generation. These capabilities of Modelio can in turn help to increase the design productivity of system designers, resulting in decreases costs and overaall system design life cycles.

6 References

[1] INCOSE, INCOSE 2012 Symposium, *http://incose.org/symp2012/*, 2012

[2] Object Management Group Inc (OMG), Final Adopted OMG SysML Specification (v1.3), *http://www.omg.org/spec/SysML/1.3/*, *2012*

[3] Object Management Group Inc (OMG), Business Process Model And Notation (BPMN) Version 2.0, *http://www.omg.org/spec/BPMN/2.0/PDF/*, 2012

[4] Object Management Group Inc (OMG), OMG Unified Modeling Language superstructure, v2.4.1, *http://www.omg.org/spec/UML/2.4.1*, 2011

[5] The Open Group Architecture Framework (TOGAF), *http://www.togaf.info/*, 2012

[6] Object Management Group Inc (OMG), Modeling and Analysis of Real-time and Embedded systems (MARTE), v1.1, *http://www.omg.org/spec/MARTE/1.1/PDF*, 2011