

SINTEF IKT
SINTEF ICT

Address:
Postboks 124 Blindern
NO-0314 Oslo
NORWAY

Telephone:+47 73593000
Telefax:+47 22067350
postmottak.IKT@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

KEYWORDS:
User interface
specification languages;
Emergency response

Report

A Case-based Assessment of the FLUIDE Framework for Specifying Emergency Response User Interfaces

VERSION

Final

DATE

2016-12-28

AUTHOR(S)

Erik Gøsta Nilsson
Ketil Stølen

CLIENT(S)

The EMERGENCY project supported by the Research Council of Norway, p.nr. 187799/S10

CLIENT'S REF.

187799/S10

PROJECT NO.

90B261

NUMBER OF PAGES/APPENDICES:

100 incl. appendices

ABSTRACT

In this report, we present the results from assessing the FLUIDE Framework for model-based specification of user interfaces supporting emergency responders. First, we outline the special challenges faced when developing such user interfaces, and the approach used in the FLUIDE Framework to meet these challenges. Then we introduce the framework, including its two specification languages. Thereafter, we present highlights from the case addressing the specification of user interfaces for three existing emergency response applications. Based on these specifications, we discuss how well we succeeded, concluding that we were able to describe the applications in a comprehensive and understandable way taking similarities and difference between the applications into account. The language constructs function as intended, having two languages has proven valuable, and the specifications scale quite well. A more comprehensive presentation of the user interfaces in the three applications, and the complete FLUIDE specifications of these user interfaces are given in an appendix.

PREPARED BY

Erik G. Nilsson

CHECKED BY

Jan Håvard Skjetne

APPROVED BY

Bjørn Skjellaug

REPORT NO.
A26920

ISBN
9788214059014

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

SIGNATURE

SIGNATURE

SIGNATURE

Table of contents

1	Introduction	4
2	The FLUIDE Framework.....	6
2.1	The FLUIDE Specification Languages.....	6
3	Emergency Response Case Study.....	8
3.1	The MASTER Application.....	8
3.2	The Resource Manager Application.....	13
3.3	The eTriage Application	15
4	Discussion	18
4.1	To what extent were we able to successfully express the three applications?.....	18
4.2	To what extent are the specifications comprehensible to third parties?.....	19
4.3	Are there common patterns/differences between the three specifications?.....	20
4.4	Which constructs functioned well; which constructs functioned less so?	20
4.5	Are both languages needed or should they be integrated into one?.....	20
4.6	To what extent do the specifications scale?.....	21
5	Related work.....	21
6	Conclusions and Future Research	22
7	ACKNOWLEDGMENTS.....	22
8	REFERENCES	23
	Appendix A - Complete Specification of the Case	25
A.1	The MASTER Application.....	25
A.1.1	Incident category	27
A.1.1.1	FLUIDE-A specifications of the Incident category.....	27
A.1.1.2	FLUIDE-D specifications of the Incident category – Ribbon content.....	31
A.1.1.3	FLUIDE-D specifications of the Incident category – Map overlays	35
A.1.2	Response category.....	38
A.1.2.1	FLUIDE-A specifications of the Response category	38
A.1.2.2	FLUIDE-D specifications of the Response category – Ribbon content	40
A.1.2.3	FLUIDE-D specifications of the Response category – Map overlays.....	43
A.1.3	Resources category.....	44
A.1.3.1	FLUIDE-A specifications of the Resources category	45

A.1.3.2	FLUIDE-D specifications of the Resources category – Ribbon content	47
A.1.3.3	FLUIDE-D specifications of the Resources category – Tabular presentation	48
A.1.3.4	FLUIDE-D specifications of the Resources category – Map overlays.....	49
A.1.4	Victims category	50
A.1.4.1	FLUIDE-A specifications of the Victims category	51
A.1.4.2	FLUIDE-D specifications of the Victims category – Ribbon content	53
A.1.4.3	FLUIDE-D specifications of the Victims category – Tabular presentations.....	54
A.1.4.4	FLUIDE-D specifications of the Victims category – Map overlays	55
A.1.5	Risks category	56
A.1.5.1	FLUIDE-A specifications of the Risks category	57
A.1.5.2	FLUIDE-D specifications of the Risks category – Ribbon content	58
A.1.5.3	FLUIDE-D specifications of the Risks category – Map overlays	60
A.1.6	Top level of ribbon (ribbon buttons and ticker)	60
A.1.6.1	FLUIDE specifications the ribbon buttons	61
A.1.6.2	FLUIDE specifications the ribbon ticker	63
A.1.7	Coupling the different parts of the ribbon	66
A.1.7.1	FLUIDE-A specifications for coupling the different parts of the ribbon	66
A.1.7.2	FLUIDE-D specifications for coupling the different parts of the ribbon	68
A.1.8	Coupling the map overlays	70
A.1.9	Putting all parts of the MASTER together.....	71
A.1.9.1	FLUIDE-A specifications for putting all parts of the MASTER together	71
A.1.9.2	FLUIDE-D specifications for putting all parts of the MASTER together	72
A.2	The Resource Manager Application.....	73
A.2.1	Location tracking and overview of locations	73
A.2.1.1	FLUIDE-A specifications of location tracking and overview of locations.....	74
A.2.1.2	FLUIDE-D specifications of location tracking and overview of locations.....	77
A.2.2	Task overview and allocation	81
A.2.2.1	FLUIDE-A specifications of task overview and allocation	83
A.2.2.2	FLUIDE-D specifications of task overview and allocation	84
A.2.3	Putting all parts of the Resource Manager together.....	91
A.2.3.1	FLUIDE-A specification for putting all parts of the Resource Manager together	91
A.2.3.2	FLUIDE-D specifications for putting all parts of the Resource Manager together	92
A.3	The eTriage Application	93
A.3.1	FLUIDE-A specifications of the eTriage user interface.....	94
A.3.2	FLUIDE-D specifications of the eTriage user interface	96

1 Introduction

Emergency response operations are very varied, from simple everyday incidents to long-lasting serious catastrophes. More complex operations tend to have a fast changing nature, sometimes being almost unpredictable. Developing ICT solutions supporting such work is challenging. User interfaces (UIs) of ICT solutions need to adapt to and reflect these variations, and their design are therefore particularly challenging.

Support for user interfaces on equipment with different screen sizes is important to allow local leaders at the incident site employ the same applications in the same intuitive way on different kinds of equipment [16]. For field workers it is important to have non-intrusive ICT support, possibly offering non-visual modalities as an alternative to or in combination with visual presentation and interaction. One way the needs for flexibility is addressed in ICT solutions for emergency responders today is by providing generic, data-oriented ICT solutions. Such solutions are not tailored to specific tasks, and force the users to adapt to the solutions, and not the other way around.

There are on the other hand also many similarities and patterns between emergencies. These include types and occurrences of emergency response operations, as well as the actors involved in such operations. Furthermore, tasks and information needs have similar communalities across operation types, actual operations and agencies. In [17] we have argued that the similarities and pattern may be characterized by a limited number of categories of functionality.

The approach put forward in this report is to provide components supporting these categories of functionality, combined with means for composing end-user solutions from these components. The components need to be flexible and tailor-friendly, i.e. they need to combine being ready-to-use with being highly configurable. Composition is primarily done at design-time, while configuration (and certain types of composition) may also take place at run-time. Developing user interfaces for this kind of solutions using traditional programming languages with connected libraries is very challenging to the extent it is at all possible. It is extremely resource demanding because all imaginable combinations of functionality, compositions and configurations must be covered.

Model-based user interface development approaches [12] are well suited to meet the needs for cross-platform and cross-modality support, but existing model-based user interface development approaches are also seriously challenged by the requirements for flexibility. There is a need for building blocks that meet these five requirements:

- R1. Are at a sufficiently high level of abstraction to support development of user interfaces that work across platforms and modalities
- R2. Provide compound structures of simple elements and containers/dialogs to support common specifications between platforms and modalities
- R3. Have reflection mechanisms giving an awareness of model structures (including domain models) to support adaptation both at design- and run-time
- R4. Support development of user interfaces where the layout depends on the instances at run-time, typically using icons, maps, graphical elements, as well as alternative modalities like speech
- R5. Provide specific and explicit support for user interface patterns and styles that are particularly useful in the emergency response domain

In the following we discuss how well some languages and approaches supporting model-based user interface development [12], including some of the most influential ones, meet these requirements.

MARIA [22] meets R1 well, but fulfils R2 only partly. It offers building blocks that are abstractions of simple user interface elements (elements for entering data, presenting data, activation functions, etc.) as well as abstractions of containers for structuring these (including top level dialogs), but not any composed ones. With the chosen building blocks the composition structure of the user interfaces – which is different when the platforms have large differences – is reflected in the specifications, also at the abstract level. It meets R3 quite well by offering adaptation between platforms through a mitigation mechanism (including a run-time system), but this does not support other types of adaptation on single platforms. R4 is partly met by offering support for different modalities, and a possibility to show map-based user interfaces. The latter is though offered through composing external user interface services. Specifying such user interfaces does not seem to be directly supported. As MARIA is a general purpose language, R5 is not met.

Also UsiXML [9] meets R1 well, but fulfills R2 only partly for the same reasons as MARIA, although some compound components like tables are offered. It meets R3 to some extent. The connections to domain models are through transformation, but as these work both ways, a degree of traceability is achieved, and some models are available at run-time to facilitate adaptation. The adaptation is though restricted to fitting a user interface to similar devices with different form factors, not to platforms with large differences [13]. UsiXML is extensible, and there is an extensive family of related approaches providing various extensions [26]. Some of these extensions enhance the support for R3 [11], while other address R4 (including support for maps and 3D user interfaces), but there is no common, integrated support. Another extension supports development of a user interface for a flight cockpit [8], an application area having similarities to emergency response. Except for this, R5 is not met.

CAP3 [25] meets R1 and R2 in similar ways as MARIA and UsiXML, although the building blocks are slightly more abstract. According to [25], CAP3 supports adaptation to context, but this seems to be achieved through service integration and not involving any reflection mechanisms, showing limited or no support for R3. Except for supporting maps and live content like movies, R4 is not met. Only visual modalities seem to be supported. R5 is not met.

ICOs [14] employs a different approach by focusing on the user interface behaviour, abstracted using Petri nets. It does not support cross-platform specifications, and does therefore not meet R1, but it offers an abstract iWidget construct to embed the presentation part of user interfaces specified by other means. R2 is not met, neither. R3 is partly met by providing a run-time system. Reflection is provided through such mechanisms in Java. Both seem to be used primarily to provide interactive development support. ICOs puts emphasis on supporting development of post-WIMP user interfaces, so R4 is met. R5 is partly met, as ICOs have proven useful in closely related domains like command and control, air traffic control and cockpit systems.

The recent OMG standard IFML (Interaction Flow Modeling Language¹) [3] meets R1 only partly, as the building blocks are on a lower abstraction level than MARIA, UsiXML and CAP3, and only forms-based user interfaces are supported. It meets R2 to the same degree as MARIA, UsiXML and CAP3, but supports neither R3, R4 nor R5.

We have developed the FLUIDE Framework offering building blocks fulfilling R1-R5. How this is done is explained in the next section. In this report, which is an extended version of [18], we present the framework in an example-based manner, and report results from a case-based assessment of the framework.

¹ Available at www.ifml.org

2 The FLUIDE Framework

The FLUIDE Framework contains:

- A collection of ready-to-use and highly configurable components supporting flexible composition of end-user solutions for emergency responders
- Composition and configuration approaches
- The FLUIDE Specification Languages
- A generic mechanism transforming specifications to components or applications
- The FLUIDE Method supporting the use of the framework

The current version of the FLUIDE Framework is a first prototype, and thus the maturity of the different parts vary. The most mature part, and the subject of this assessment, is the FLUIDE Specification Languages: FLUIDE-A which is used for expressing abstract user interface (AUI), and FLUIDE-D which is used for expressing concrete designs, usually denoted concrete user interface (CUI). FLUIDE-D provides specific support for the emergency response domain through a library of user interface patterns that are particularly useful for this domain, and may be automatically transformed to a final user interface (FUI).

2.1 The FLUIDE Specification Languages

The user interface of an emergency response application must support the work performed by emergency responders. The four main language constructs in FLUIDE, presented as rounded rectangles in Figure 2.1, support a natural breakdown of such work (rectangles). Emergency response work can be categorized with respect to responder types, responder roles and high level tasks, as well as combinations of these. The categories of functionality [17] support categories of work and the task structures these categories contain.

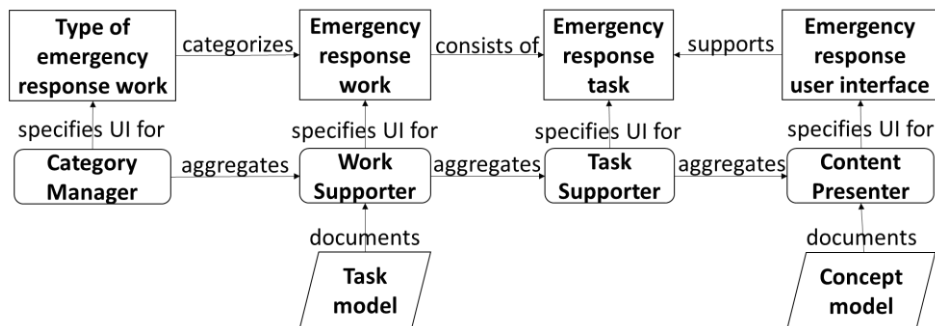


Figure 2.1 – Overview of the main constructs in FLUIDE-A

In FLUIDE-A, the *Category Manager* construct facilitates the specification of a whole application, or some part of it. A category of functionality supports certain work performed by emergency responders. Such work can be divided into tasks on different levels.

These tasks may be categorized both in a hierarchical goals/means structure and through temporal constraints between sets of tasks. Such task structures are specified using the *Work Supporter* construct, which includes a task model to specify hierarchical and temporal structures. The graphical syntax of FLUIDE-A uses a neutral hierarchical task model syntax to express the hierarchical structure. The temporal structure is expressed using operators. A special kind of Work Supporter aggregates other Work Supporters recursively.

A user interface supporting one task is required to manage information content relevant for solving the task. The information needs of individual tasks are specified using the *Task Supporter* construct. How the information content used in a Task Supporter is further broken down and structured in (part of) a user

interface is specified using the *Content Presenter* construct. The information to be presented by a Content Presenter is specified by a concept model where all entities are connected through relations. The concept model, together with the specification of an *anchor* (the root entity of the model), is sufficient for determining which information is to be presented in a FUI at run-time. FLUIDE-A employs a subset of the UML class model notation (extended with the anchor) to express the models. Additional platform-independent visual properties are expressed using annotations. As the same information may be useful when solving different tasks, and because other tasks may only require a subset of the same information, Content Presenters may be specified hierarchically.

In the CAMELEON glossary², one of the definitions of *interactor* is: "A computational abstraction that allows the rendering and manipulation of entities (domain concepts and/or tasks) that require input and output resources." The four main constructs in FLUIDE-A may be understood as interactors in this sense. We use *interactor construct* as a common term for these constructs, and the term *interactor instance* to refer to an occurrence of an interactor construct in a specification.

FLUIDE-D is used for specifying designs for FLUIDE-A specifications, and contains variants of the four main constructs in FLUIDE-A, using the same names with the suffix *design*. The interactor design constructs in FLUIDE-D are used to specify which parts of the domain and task models that are to be included in a FUI. FLUIDE-D's core is the library of user interface patterns, operationalized in the *view* constructs, including *content views* for presenting the instances corresponding to the concept models. Views are used to specify how some part a FLUIDE-A specification is to be presented on a given user interface platform using certain modalities and user interface styles. The view constructs in FLUIDE-D make it possible to specify designs for a given FLUIDE-A specification for different target platforms through adding minimal amounts of platform specific specifications.

In the FLUIDE Specification Languages, R1 is met by having building blocks both for the abstract and concrete user interfaces that are at a higher abstraction level than corresponding building blocks in the approaches discussed above. R2 is met by providing compound building block as constructs in the languages. The difference compared to other approaches is most evident for the *content views* used as part of the FLUIDE-D specifications. Such views specify how the instances corresponding to a concept model fragment is presented. They provide means for specifying quite advanced designs in a very compact way through exploiting pairs of user interface patterns and model patterns. These views support user interface patterns that are particularly useful in the emergency response domain, and in this way FLUIDE meets R5. We use the term *user interface pattern* to denote a user interface design pattern, i.e. a pattern focusing on reoccurring visual and structural aspects as well as generic behaviour of user interfaces. Compound building blocks are in their nature more specialized than simple ones. To counter for this, the views provide versatility through being based on model patterns. We use the term *model pattern* to denote patterns of the same type as Gamma et al. [7], i.e. expressed in terms of a concept model. This means that the views may be used to specify advanced user interfaces managing a wide variety of information as long as the information to be presented has a structure that matches the model patterns used in the view. For example the *Map Icons View* provides means for specifying an icon-based presentation of any type of information in a map user interface as long as the model follows a given structure (including providing locations) – working just as well for presenting incident objects, resources, victims, important locations or risks. Thus, such views combine being specialized and powerful with regards to emergency response need with being versatile with regards to the actual information they present. The compound view types in FLUIDE-D also support specification of user interfaces where the layout is depending on the instances at run-time, thus meeting R4. R3 is supported in FLUIDE through enabling reflection mechanisms by embedding domain models as part of the specifications. This includes the concept models used in the content views just discussed, as well as task models. This also provides traceability, which

² <http://giove.isti.cnr.it/projects/cameleon/glossary.html>

reduces the challenges connected to roundtrip engineering which is inherent in model-based systems development.

3 Emergency Response Case Study

In order to assess the FLUIDE Framework we retrospectively specified the user interface of three existing emergency applications (without any connections to FLUIDE), all three of which were developed as part of the research project BRIDGE³. The three applications are: MASTER, eTriage and Resource Manager. We denote this the target user interface. The advantages of specifying already existing applications are realism and that we do not need to design the user interface from scratch. Using three applications in the case provides variation with regards to users, tasks, managed information, platform and style. The disadvantage on the other hand is that we mainly assess the suitability of the FLUIDE Specification Languages. The full description of the three applications and the corresponding FLUIDE specifications are available in Appendix A. In the following, we present excerpts from these specifications to provide more background on the FLUIDE languages and their usage.

3.1 The MASTER Application

The MASTER application consists of a large map display showing information overlays (icons and other visual representations) with relevant information for the emergency response at hand (shown in Figure A.2). All information overlays on the map belong to one of five categories. The application also includes a *ribbon* showing the information elements in the overlays grouped by these categories, which are further divided into sub categories. On the top level, the ribbon contains a set of buttons for accessing the information in each category, as well as a ticker showing summary information – as illustrated in Figure 3.1.



Figure 3.1 – Ribbon showing the ticker and the top level buttons

The ribbons for each of the categories are similar to each other, but present different types of information. We only show the ribbon for the *victims* category (Figure 3.2).

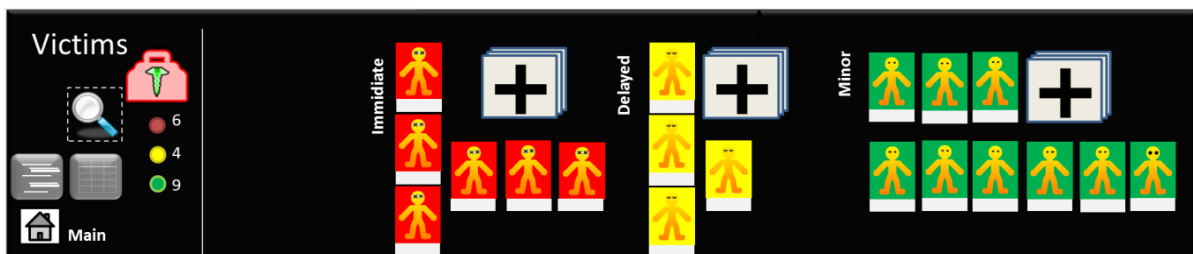


Figure 3.2 – Ribbon content for the victims category

³ www.bridgeproject.eu

The ribbon contains some overview information at the left and icons for each of the sub categories in a horizontal scrollable view at the right. Each victim is represented by an icon in the ribbon and on the map. The plus icons represent functionality for adding elements of the given sub category. The information in the victims category may also be presented in a tabular form, as shown in Figure 3.3.




Patients						
	Status	Description	BP	Temp.	Name	...
1	 	Unconscious	139/90	38	Unknown	...
8	 	Seriously bleeding	80/40	35	Unknown	...
3	 	Deliriously	200/140	40	Per Olsen 	...
5	 	Broken leg	Jon Tronstad 	...
2	
11	 	Scratches
15	 	Dead

Figure 3.3 – Tabular presentation of victim information

The *Victims Presenter* (Figure 3.4) specifies both the right hand part of the ribbon for the victims category and the tabular presentation of victim information in FLUIDE-A, using the Basic Content Presenter construct.

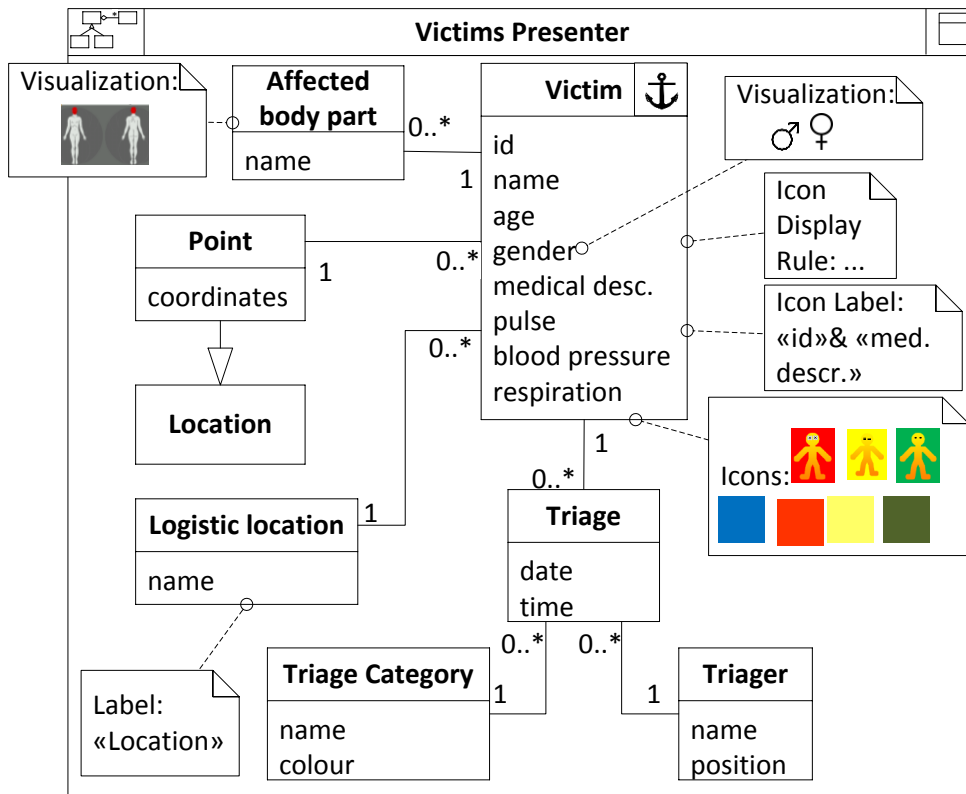


Figure 3.4 – FLUIDE-A specification of all the Victims sub categories

The outer border with a name is used for all FLUIDE-A interactor constructs. Decorations on the border determine the construct and whether it is basic or aggregated. The user interface annotations specified in Figure 3.4 express which icon that should be used to visualize instances of the entity *Victim*, rules for displaying icons, labels, as well as visualization of entities or attributes. The left side of the ribbon for the victims category is specified by a Basic Content Presenter called *Victim Summary Presenter* (shown in Figure A.51). This presenter, together with the *Victims Presenter* (Figure 3.4), contain the needed information for specifying the ribbon category in Figure 3.2. The specification of this in FLUIDE-A is shown in Figure 3.5 using an Aggregated Content Presenter.

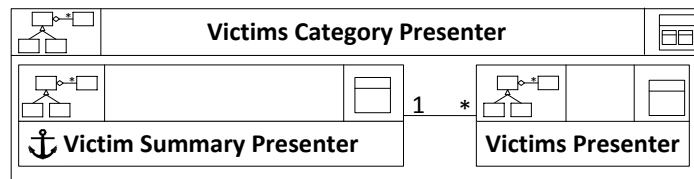


Figure 3.5 – FLUIDE-A specification of the Victims Category (Figure 3.2)

Aggregated presenters include only the border parts of their members, and the members' names are shown inside the presenter instead of in the heading.

Figure 3.6 shows a FLUIDE-D Basic Content Presenter Design specifying the right hand part of the ribbon for the victims category (Figure 3.2).

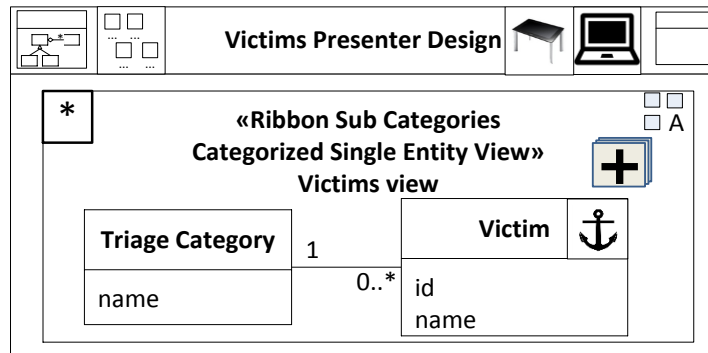


Figure 3.6 – FLUIDE-D specification of all the Victims sub categories

The outer border of a FLUIDE-D specification resembles the FLUIDE-A border, but it also specifies user interface style(s) and modalities/platform(s) the design is targeted at. Presenter designs contain one or more views. There are four main types of views, i.e. layout manager view, decorative view, content view and content integration view. The view shown in Figure 3.6 is a content view, i.e. a view that presents instances of one or more entities. Content and content integration views use UML stereotype notation to denote the view type before its name. A *1* or *** on the top left of a content view denotes whether the view presents one or a number of instances of the anchor entity. The available content and content integration views make up the FLUIDE library of emergency response user interface patterns.

All content views impose restrictions on the model fragment they may present, expressed in FLUIDE-D as a model pattern fitting the user interface pattern supported by the content view. The model pattern for the domain-specific content view used in Figure 3.6 must contain the entity to be presented (*Victim*), and a

categorizing entity (*Triage Category*). The view presents a number of instances of the presented entity as a set of grids or tables of icons. The number of grids is determined by the number of instances of the categorizing entity. These instances also determine the labels for each sub type. The plus icon on the right hand side of the view is specified using a property setting for the view. The icons to use in the grid are obtained from the icon annotation on the *Victim* entity in the corresponding FLUIDE-A specification (Figure 3.4).

Figure 3.7 shows a Basic Content Presenter Design specifying the Tabular presentation of victim information (Figure 3.3).

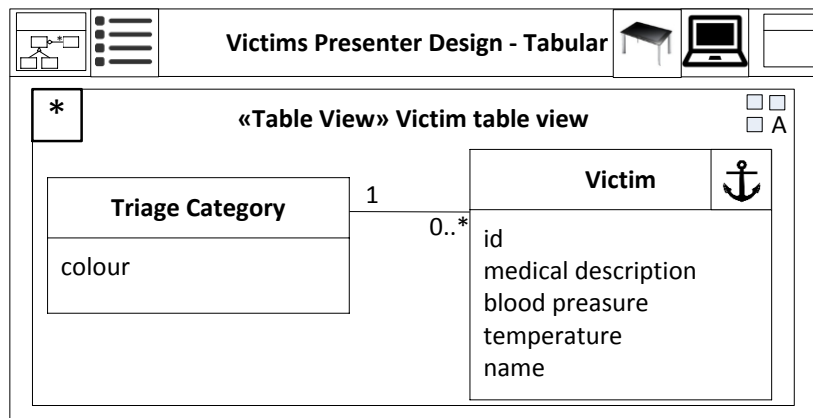


Figure 3.7 – FLUIDE-D specification of the Tabular user interface in Figure 3.3

It contains one generic content view. Its model pattern must contain one entity (possibly with subtypes) that determines the rows in the table (*Victim*). It may also include related entities, as long as the cardinality on the side of the related entity is one.

Figure 3.8 shows an Aggregated Content Presenter Design specifying the Ribbon content for the victims category (Figure 3.2).

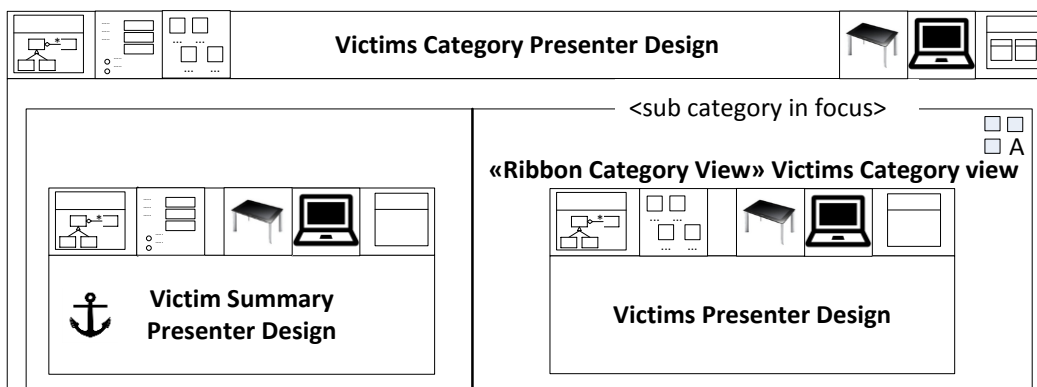


Figure 3.8 – FLUIDE-D specification of the Victims Category

It only includes the border parts of the member presenter designs. The anchor symbol indicates which of the member presenter designs the aggregated one inherits the anchor from. The presenter design in Figure 3.8 contains a content integration view. Content integration views integrate related content from different presenter designs. Content integration views require that their member presenter designs use specific content

or content integration views. The Ribbon Category View used in Figure 3.8 has two slots. The part to the left of the vertical line may only aggregate exactly one presenter design containing a Ribbon Category Overview View. The right hand slot may aggregate different view types, including the one used in Figure 3.6

To specify the coupling of the 5 ribbon categories (of which one is specified in Figure 3.5 and Figure 3.8) in FLUIDE-A, we use a Task Supporter with 5 children, shown in Figure 3.9.

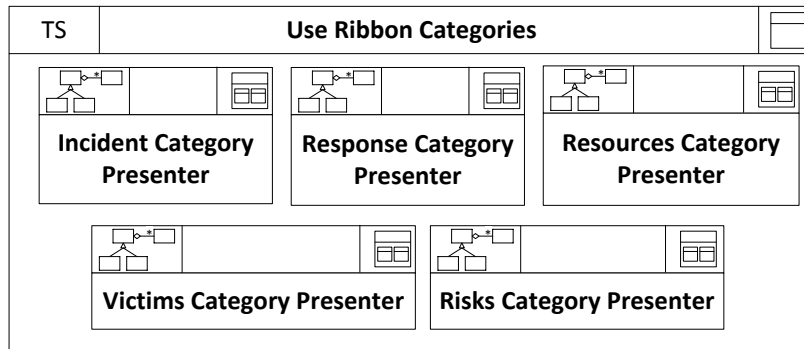


Figure 3.9 – FLUIDE-A specification of a Task Supporter connecting the five ribbon categories

The corresponding Task Supporter Design is shown in Figure A.87. To specify the coupling of the ribbon buttons (specified in the Task Supporter *Use ribbon buttons* which is shown in Figure A.70) and the ribbon categories (Figure 3.9), the Basic Work Supporter in Figure 3.10 is used.

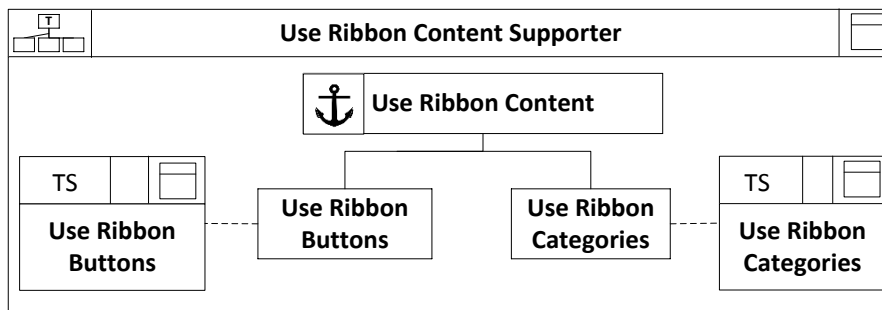


Figure 3.10 – FLUIDE-A specification of a Basic Work Supporter coupling the ribbon categories with the top level set of ribbon buttons

Each task in a Work Supporter may have a connected Task Supporter. In the Work Supporter in Figure 3.10, there are three tasks, of which two have Task Supporters. The corresponding design for the Work Supporter is shown in Figure A.87.

To specify the entire ribbon (buttons, ticker and the individual ribbon categories), the Aggregated Work Supporter in Figure 3.11 is used.

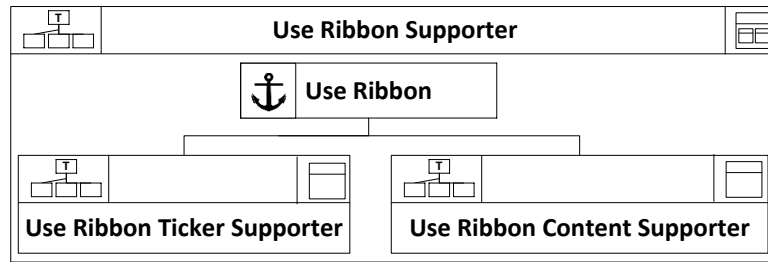


Figure 3.11 – FLUIDE-A specification of an Aggregated Work Supporter for the Use Ribbon task

An Aggregated Work Supporter must add exactly one task (possibly with a Task Supporter) on the level above the member supporters. The supporter in Figure 3.11 couples the ribbon contents (Figure 3.10) with the Work Supporter for the ribbon ticker. The corresponding design is shown in Figure A.90.

The map part of MASTER in FLUIDE-A is specified using a number of Basic and Aggregated Content Presenters, a Task Supporter, and a Basic Work Supporter (presented in Appendix A). To specify the coupling of the ribbon and the map, a Category Manager is used. A Category Manager may aggregate both Content Presenters and Work Supporters. The *Master Category Manager* (shown in Figure A.94) aggregates a Basic and an Aggregated Work Supporter. The corresponding design is shown in Figure 3.12.

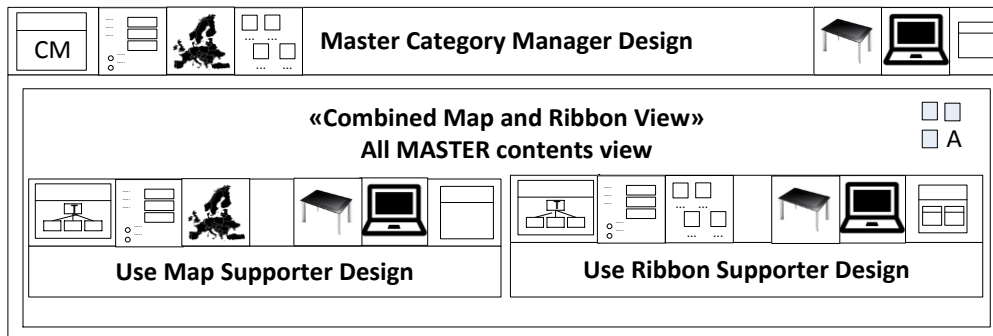


Figure 3.12 – FLUIDE-D specification of the MASTER user interface

Figure 3.12 uses a domain-specific content integration view which puts together one design containing a Map View with another design containing a Ribbon View, together making up a complete user interface with the map and the ribbon working together. All designs having design children exploit the sum of styles and modality/platforms of their children (and their children recursively).

3.2 The Resource Manager Application

The Resource Manager is a smartphone application supporting personnel in the field by managing locations as well as receiving and responding to task allocations. Figure 3.13 shows two of the user interfaces it consists of.

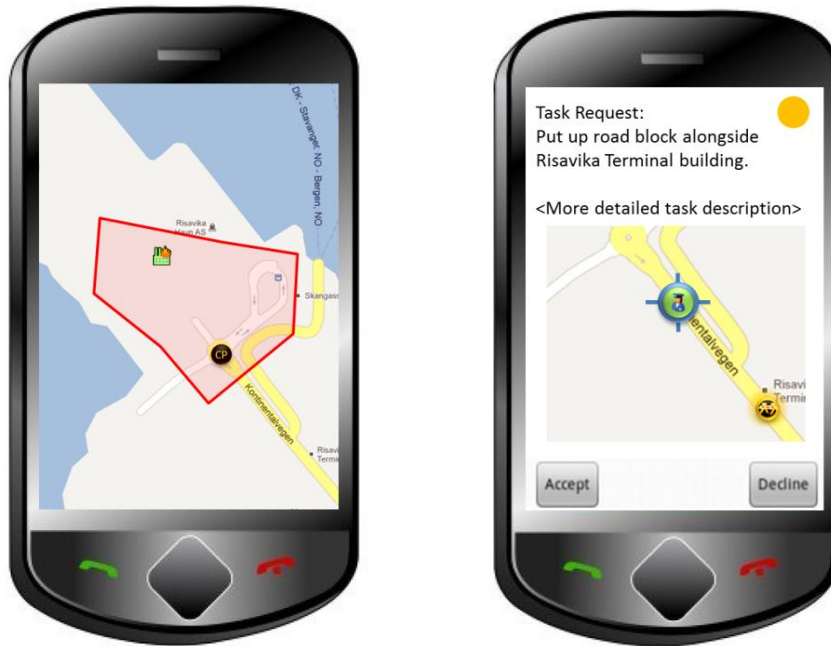


Figure 3.13 – Two example Resource Manager user interfaces

The Content Presenters or Task Supporters specifying the user interfaces of the Resource Manager are presented in Appendix A. In this section, we only show the Work Supporter in Figure 3.14.

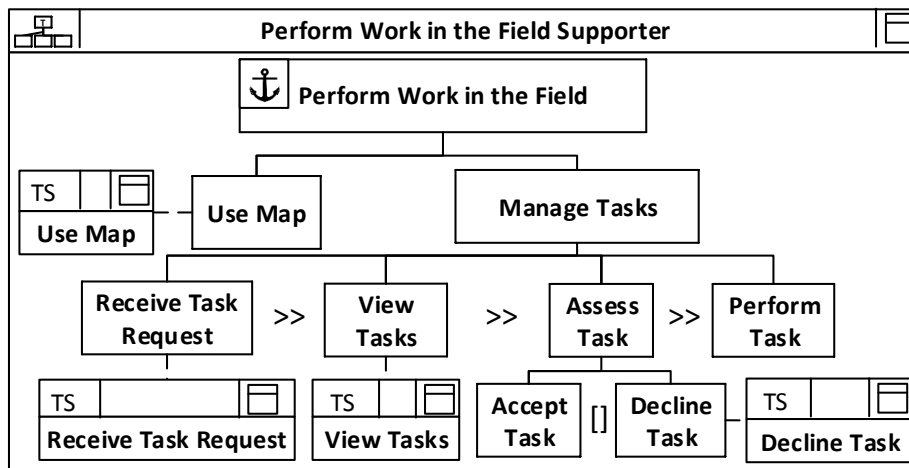


Figure 3.14 – FLUIDE-A specification of the Basic Work Supporter *Perform Work in the Field Supporter*

It uses a more complex task model than the Work Supporter shown for the MASTER application (Figure 3.10). There are four levels in the task model in the Work Supporter, and four of the nine tasks have connected Task Supporters. The task model contains some operators (using the same symbols and precedence rules as CTT [21]): ">>" indicates sequence in task performance, while "[]" indicates a choice between tasks. No operator means that the tasks may be performed in an arbitrary sequence, including in parallel. The corresponding design is shown in Figure 3.15.

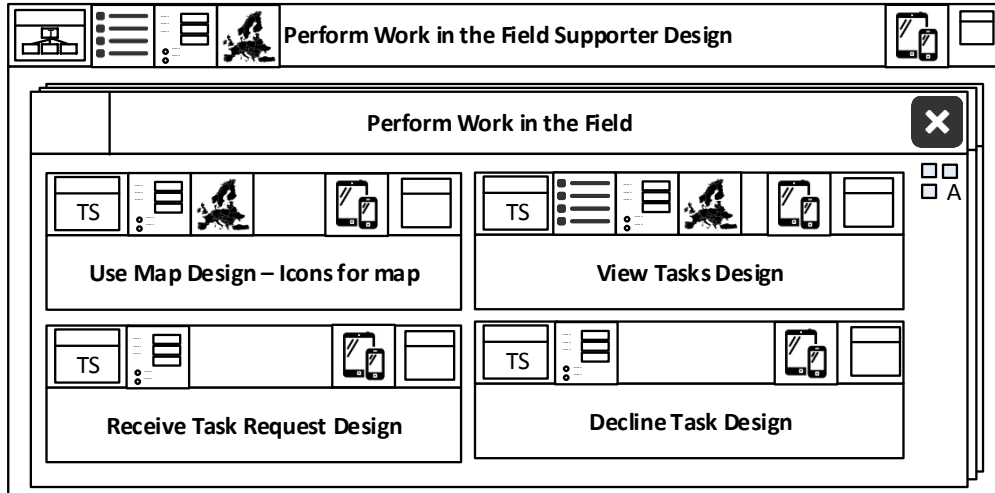


Figure 3.15 – FLUIDE-D specification of a Basic Work Supporter Design for the Basic Work Supporter in Figure 3.14

This Work Supporter Design utilizes a decorative view specifying that the child designs are a set of loosely connected windows (or full screen dialogs on a mobile device). The close icon indicates that windows are used, while the stacking look is used to indicate that the view contains a number of windows. The child Task Supporter Designs correspond to the children of the Work Supporter shown in Figure 3.14. Note also that the design uses touch-based mobile device modality/platform.

3.3 The eTriage Application

The eTriage application supports personnel in the field triaging⁴ victims. The user interface giving an overview of victims already triaged is shown in Figure 3.16.

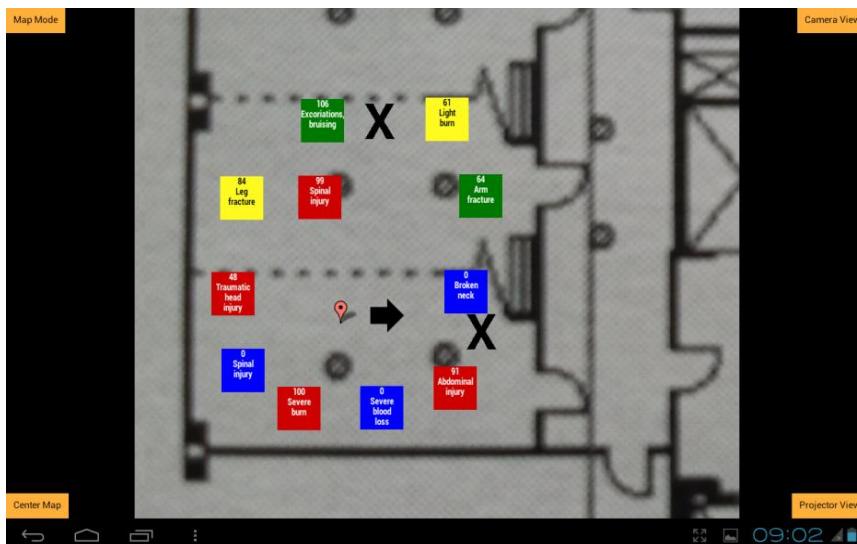


Figure 3.16 – Map based user interface giving an overview of triaged victims

⁴ Triage is the process of determining the priority of patients' treatments based on the severity of their condition.

The user interface employed when triaging a single victim or looking at the details for a victim already triaged is shown in Figure 3.17.

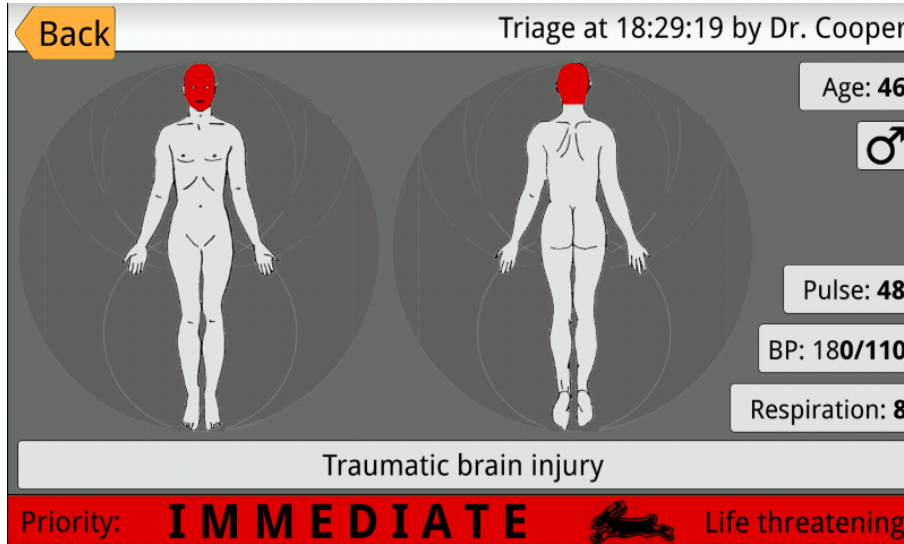


Figure 3.17 – User interface for managing details about a triaged victim

The content managed by these user interfaces are covered by the FLUIDE-A Basic Content Presenter *Victims Presenter* shown in Figure 3.4. The FLUIDE-D specification of Figure 3.16 is shown in Figure 3.18.

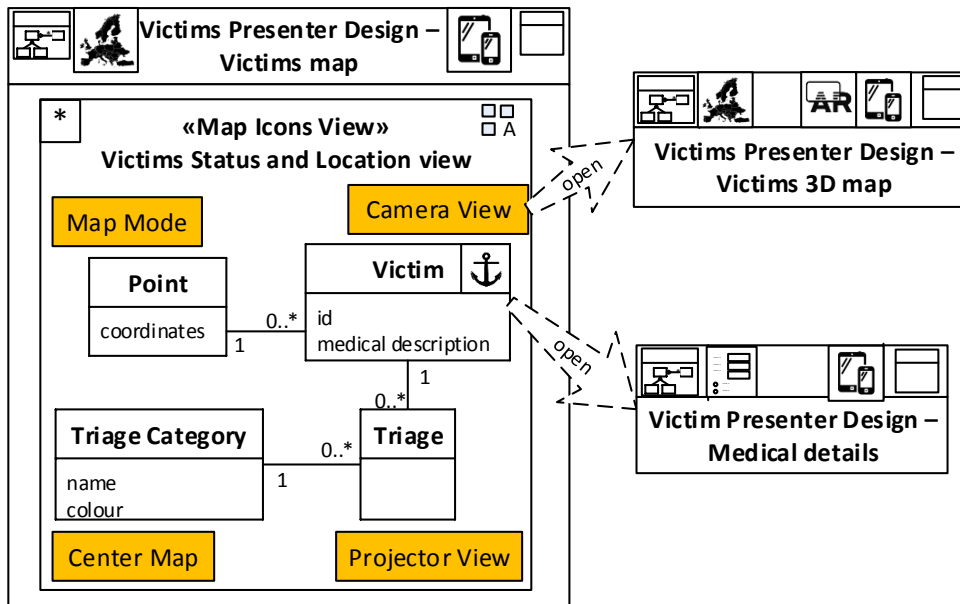


Figure 3.18 – FLUIDE-D specification of the user interface in Figure 3.16

The design in Figure 3.18 uses a map-based user interface style, shown by the icon on the left side in the header. It contains one Map Icons View, which is a domain-specific content view type. The model pattern must contain one entity (possibly with subtypes) that has a relation to a location entity (providing a point). It may also include related entities, as long as the cardinality on the side of the related entity which is presented is one. It presents a number of instances of the presented entities as icons on a map. If the presenter design

using this type of view is member of a Map View (a domain-specific content integration view), either directly or one or more times among its parents, the icons are shown on the map provided by the Map View highest up in the hierarchy. If a Map Icons View does not have a parent providing a Map View, it will provide its own map. Three of the buttons inside the views are property values of the view, the fourth (the button labelled "Camera View") is a button providing navigation. The presenter design shown in Figure 3.18 also specifies the intended dialog navigation to take place when an icon on the map is tapped. It is shown as a dashed-lined arrow with a growing size. The type of dialog navigation (in this case *open*) is shown as text on the arrow. The small end indicates which element of the user interface that triggers the dialog navigation. The point of the arrow identifies the target. The target of one of the dialog navigation specifications is a representation of the presenter design shown in Figure 3.19.

A FLUIDE-D specification of Figure 3.17 is shown in Figure 3.19.

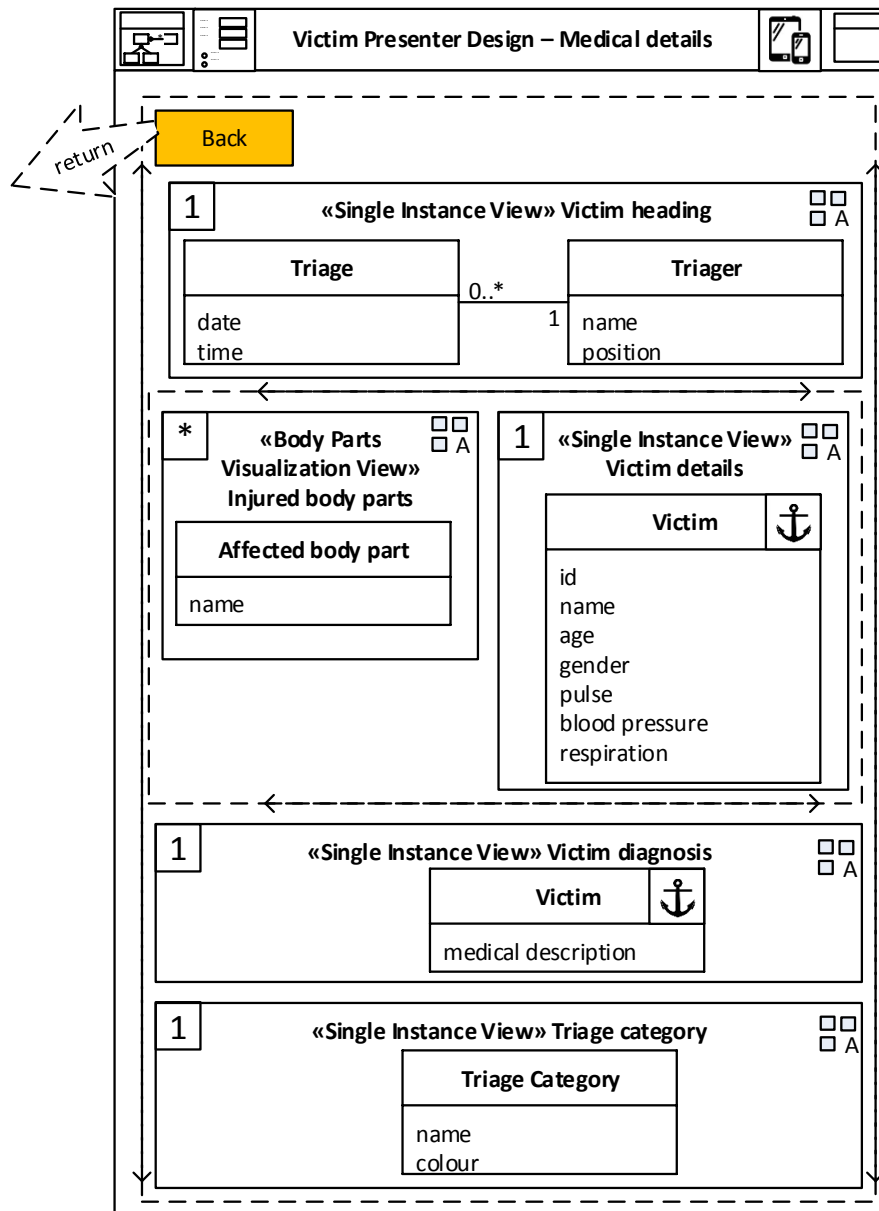


Figure 3.19 – FLUIDE-D specification of user interface in Figure 3.17

The design in Figure 3.19 puts together five content views using a number of layout manager views (the designs used in MASTER leave most of the layout to the content and content integration views). Layout manager views are not given names, and are shown using dashed lines (to indicate that they are usually not visible). The arrows on the dashed line specify whether the children are organized horizontally or vertically. Four of the content views used are Single Instance Views, a generic content view type for presenting one instance at a time of a single entity. The presenter design also uses the domain-specific Body Parts Visualization View, which presents multiple instances together graphically. The presenter design also contains a button for navigating back to the dialog from which the *Medical details* was opened using the "return" type of dialog navigation.

4 Discussion

As a result of the case study we have obtained three specifications – one for each application. In the following, we discuss the suitability of the FLUIDE languages by a careful inspection of these specifications as available in Appendix A. To structure the discussion we have formulated the following six research questions:

1. To what extent were we able to successfully express the three applications?
2. To what extent are the specifications comprehensible to third parties?
3. Are there common patterns/differences between the three specifications?
4. Which constructs functioned well; which constructs functioned less so?
5. Are both languages needed or should they be integrated into one?
6. To what extent do the specifications scale?

When conducting the assessment, we have put emphasis on dealing with the challenges related to the fact that the assessment was performed by the researchers that have developed the FLUIDE Framework. Firstly, these challenges were addressed by formulating research questions that were possible to address solely by assessing the specifications and the corresponding user interfaces. Secondly, the discussions addressing the research questions are to a large extent based on quantitative data obtained by careful analyses of the specifications and the user interfaces they specify. We claim that both these measures have contributed to an objective assessment with comparable conclusions to a possible outcome of an assessment performed by other researchers.

4.1 To what extent were we able to successfully express the three applications?

We were able to fully describe all three applications. In doing this, we met no major obstacles. We claim that the specifications of the three applications contain sufficient information for the target user interfaces to be schematically deducible from them. To support this claim, we use two examples to show how all the different parts of a target user interface are reflected in the corresponding specification: In Figure 4.1 we show where the different parts of the user interface in Figure 3.2 are specified.

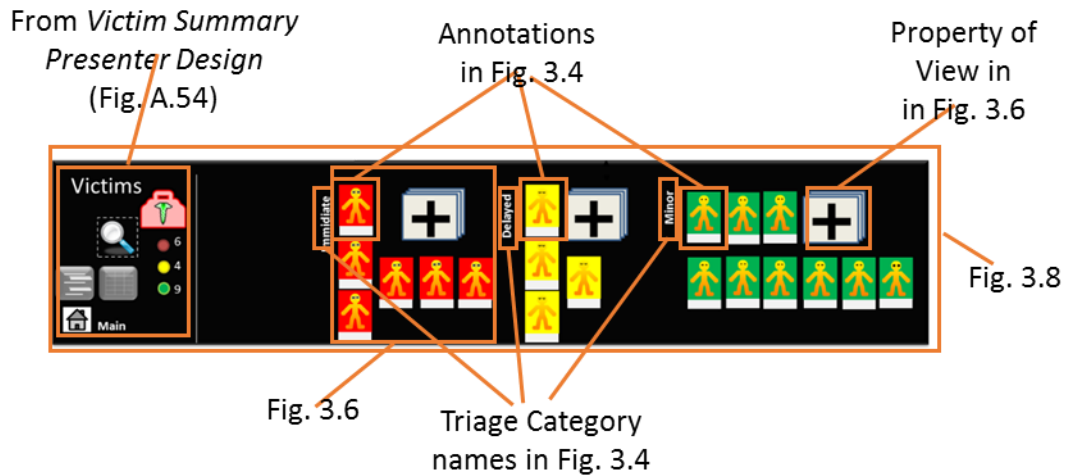


Figure 4.1 – Where the parts of the user interface in Figure 3.2 are specified

In Figure 4.2 we show the correspondence between the elements in the user interface in Figure 3.17 and elements in its FLUIDE-D specification (Figure 3.19).

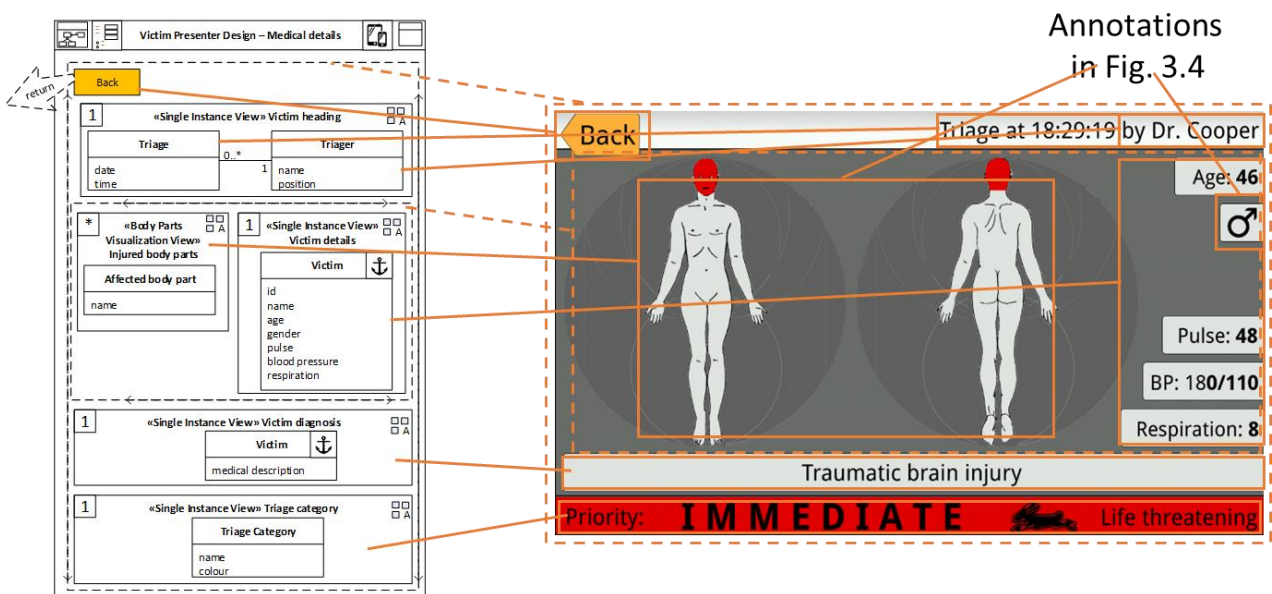


Figure 4.2 – Connections between the user interface in Figure 3.17 and its specification

4.2 To what extent are the specifications comprehensible to third parties?

The fact that the model fragments are expressed using known modelling languages gives systems developers (and end users) knowing UML class models and task models a head start. Also, as can be seen in the presentation of the case, the powerful constructs in the languages – particularly the view constructs – make the specifications fairly simple.

The naming of the different content and content integration views are made to highlight which kind of user interface pattern they support and how they are used. To add to this, the decorations on the FLUIDE-D specifications give comprehensive and understandable information about style, modality and platform in a compact manner. These observations are supported by an experiment using another case [19].

4.3 Are there common patterns/differences between the three specifications?

The three applications all support emergency responders, and they manage overlapping information. They have many commonalities, but they also vary regarding number of user interfaces, their complexity, style, platform, modality, and whether they are data or task oriented. The number of user interfaces is reflected in which interactor construct that are used. MASTER and Resource Manager use all constructs, while eTriage having few user interfaces only use the Content Presenter and Category Manager (Design) constructs. As Resource Manager and eTriage only manage one category of information each, while MASTER manages five, there is only one Content Presenter instance in the FLUIDE-A specifications of Resource Manager and eTriage respectively, while there are 20 such instances in the FLUIDE-A specification of MASTER.

MASTER is data-oriented and puts few restrictions on the sequence in which the different parts are to be used. Resource Manager is task oriented with a natural sequences in which the different user interfaces are used to solve specific tasks. This difference is reflected in the task models in the Work Supporters in their specifications. The task models in the Work Supporters in MASTER (like the one in Figure 3.10) are quite simple, while the task model in the Work Supporter in Resource Manager (Figure 3.14) is more complex, containing operators to specify the expected sequence the user interfaces are to be used.

4.4 Which constructs functioned well; which constructs functioned less so?

All the interactor constructs in FLUIDE-A and all interactor design constructs in FLUIDE-D are used in the case. In the case description, examples are given for all the interactor (design) constructs, either the FLUIDE-A, the FLUIDE-D, or both variants. Based on this, we may conclude that the interactor (design) constructs match the needs of the case. All sub constructs (construct that may only be used as part of an interactor instance) except one match the needs of the case. The annotation sub construct is not used in any of the Work Supporters. This may be an indication that annotations are not very useful for this construct.

All the four view types in FLUIDE-D are used in the case, and the distribution between them reflects characteristics of the case. Content views may be further divided into domain-specific and generic ones. The domain-specific ones are more used than the generic ones, although one of the generic ones (Single Instance View) is among the individual view constructs used most often. These findings indicate that having a library combining domain-specific and generic view types is useful in an emergency response case.

4.5 Are both languages needed or should they be integrated into one?

The main rationale for having the traditional split AUI and CUI [4] in the FLUIDE languages is to support development across platforms, styles, modalities and even applications, by having the common parts of the specifications expressed in FLUIDE-A and the more specific parts expressed in FLUIDE-D, among other to avoid redundant specifications. This split works well for the Content Presenters. The quantitative analysis shows that 57% of the interactor instances in the specification are Content Presenters. All these instances are used as basis for at least two designs, and there are on average 3.1 designs for each instance of Basic Content Presenter. The most versatile (among them the *Victims Presenter* in Figure 3.4) are used as a basis for as much as six designs. Approximately a third of the Basic Content Presenters are used as the basis for presenter designs in two different applications, showing that our aim of using FLUIDE-A specifications across applications is achievable. Furthermore, there is one Basic Content Presenter (the *Victims Presenter* in Figure 3.4) that is used as basis for presenter designs exploiting all four styles used in the case (ribbons, tabular, maps and forms).

All the instances of the other interactor constructs are used by exactly one interactor design instance. The main explanation for this finding is that the tasks and task models tend to be more specialized than the concept models, indicating that the Task Supporters and Work Supporters in the FLUIDE-A specifications, although intended to be abstract, will naturally be specific for one (part of) a user interface on a given

platform. This is in line with the findings of for example Clerckx et al. [5]. This shows that the split between FLUIDE-A and FLUIDE-D is important and successful for Content Presenters, but not important for the other constructs. As the Content Presenters represent more than half the interactor instances in the case, keeping the split for this construct seems advisable. An option would be to merge the languages for the other constructs, but this would violate the symmetry between the languages. Thus, our conclusion is that although the case has shown that the split is not important for all the constructs, we find the benefits from keeping the split higher than the disadvantages of having the split only for some constructs.

4.6 To what extent do the specifications scale?

The main means for specifying user interfaces of different size and complexity in FLUIDE is the aggregation mechanisms in the languages. They enable splitting the specification of large and/or complex user interfaces into smaller, manageable pieces through reusing lower level interactor (design) instances in higher level ones. Such mechanisms are available both in FLUIDE-A and FLUIDE-D. Through these mechanisms, it is possible to keep each interactor instance on a reasonable size. The specifications show that all the main aggregation mechanisms are used, and that each time an aggregation mechanism is used, a limited number of instances are aggregated (but usually more than one). This makes the specifications simple and manageable, and is a natural consequence of having four levels of interactor constructs, of which two may be used recursively. The most complex interactor (design) instances are included in the case description (Figure 3.4 for FLUIDE-A and Figure 3.19 for FLUIDE-D). Both these are of reasonable size.

5 Related work

There are quite a few languages and approaches supporting model-based user interface development [12]. As shown in the introduction, neither some of the most influential of these [9, 14, 22] nor OMG's IFML standard [3] meet all the requirement we have identified. In particular, none of the assessed approaches meet the requirement of having compound building blocks. UsiXML, MARIA, CAP3 [25] and RBUIS [2] build on or relates to the CAMELEON Framework [4], which depicts a close connection between AUIs and concept and task models. Despite this, neither of these embed such models, including their structure, as part of specifications. In UsiXML, the language supports specification of such models, but the connection to the AUI is through graph transformations. In MARIA, elements from concept models are referred in the AUI, as are operators from task models. CAP3 have explicit relations to task models in the AUIs, but use this mainly to specify behaviour, not for aggregation as we do. RBUIS keeps the connection to task models, even at run-time where these models are used as the basis for simplification of layout and features to roles, operationalized through adaptation of the CUIs. The AUIs are also used in this process, which enables fine-grained adaptations that are difficult to include in specifications. Our adaptations to roles are more coarse-grained, and based on the task models at design-time. Adaptation at run-time is partly focused on keeping the context of use in user interfaces on different devices, and partly on adaptations to changes in the external context. In realizing the latter, the approach used in RBUIS will be considered. We use the concept *interactor* in a similar way as it is used in the CAMELEON Framework, MARIA, CAP3 and Trættemberg's work [24].

The graphical syntax we use in FLUIDE-D is inspired by the way Canonical Abstract Prototypes [6] embed abstract interactors in an abstract layout, but it also differs significantly through our use of views and model element instead of containers and simple user interface elements. Our view constructs rely on combining and coupling user interface patterns and model patterns. Using user interface patterns in user interface development approaches is not uncommon. Ahmed and Ashraf [1] use patterns extensively, but focus on task and user interface patterns. Lin and Landay [10] also use user interface patterns in their cross-device development tool, but they rely on correspondence between CUI elements on different platforms rather than abstractions. In MyUI, Peissner et al. [23] make extensive use of patterns combined with state charts for their AUI. In addition to user interface patterns, they use patterns for categorizing devices, user groups, user interface elements, as well as adaptation to these. They do not apply model patterns. Vanderdonck and

Simarro [27] support patterns both for domain and user interface models, but do not combine them. Using model patterns as part of model-based user interface development is not very common. Trættestad [24] uses such patterns as part of his languages, but does not apply it to a view mechanism in the CUIs. The transformation mechanism presented in [15] also uses model patterns.

6 Conclusions and Future Research

We have presented the FLUIDE Framework supporting development of flexible user interfaces for emergency response applications through a component-based approach. The framework contains the FLUIDE Specification Languages with a unique combination of feature. The languages provide compound building blocks, which for AUIs enables common specifications across platforms and modalities with large differences, and for CUIs enables compact specifications of advanced user interfaces, including user interfaces where the layout is depending on instances at run-time. The compound building blocks are made versatile by supporting model patterns, and provides a library of user interface patterns that are particularly useful in the emergency response domain. The specifications embed domain model (concept and task models), including their structure, enabling reflection to support composition and adaptation, as well as traceability to support roundtrip engineering.

We have assessed the FLUIDE specification languages by specifying three existing emergency response applications. The experience from using the FLUIDE languages for specifying the case indicates that they are well suited for specifying user interfaces in applications supporting emergency responders working at the incident site. More precisely, we were able to fully describe all the three applications without meeting any major obstacles. Moreover, we have argued that the specifications contain sufficient information for the target user interfaces to be schematically deducible. We have tried to highlight that the specifications are comprehensible to third parties because they use known modelling style, have powerful constructs making the specifications fairly simple, yet carrying comprehensive information about the user interface being specified.

The commonalities and the variations between the three applications are well reflected in the three specifications, both with respect to which constructs that are used, and level of details in the specifications. The specifications contain occurrences of all the interactor and interactor design constructs. All these, as well as most of their sub constructs, including the view types, were used as expected and intended. The case indicates that annotations on Work Supporters are probably not needed. We experienced that having a library combining domain-specific and generic view types is useful when specifying emergency response user interfaces.

The specification includes a set of very versatile Content Presenters, working across both styles, modalities and applications. Even though the other construct proved less versatile, this shows that the split between FLUIDE-A and FLUIDE-D works, as the Content Presenters represent more than half the interactor instances in the case. The specifications scale very well because the reuse mechanisms are extensively used in the case, both on construct and instance level, contributing to keeping the specifications simple. The case also shows that the complexity of the target user interfaces is well reflected in the complexity of the FLUIDE specifications.

Among our planned future research is to complement the framework, including tool support and adaptation mechanisms.

7 ACKNOWLEDGMENTS

The work on which this report is based is supported by the EMERGENCY project (187799/S10), funded by the Norwegian Research Council and the following project partners: Locus AS, The Directorate for Civil Protection and Emergency Planning, Geodata AS, Norwegian Red Cross, and Oslo Police District. The work

is also supported by the ANYWHERE Project (grant: 700099) funded by the European Commission within the H2020 Framework Programme.

8 REFERENCES

1. S. Ahmed and G. Ashraf. 2007. Model-based user interface engineering with design patterns. *Elsevier Journal of Systems and Software* 80(8), 1408-1422.
2. P. A. Akiki, M. Keynes and A. K. Bandara. 2013. RBUIS: Simplifying Enterprise Application User Interfaces through Engineering Role-Based Adaptive Behavior. *Proceedings of EICS'13. ACM*
3. M. Brambilla and P. Fraternali. 2014. Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML. *Morgan Kaufmann*.
4. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. 2003. A Unifying Reference Framework for Multi-Target User Interfaces. *Oxford Journals Interacting with Computers* 15 (3), 289-308.
5. T. Clerckx, K. Luyten, and K. Coninx. 2004. Generating context-sensitive multiple device interfaces from design. *Proc. of CADUI'04. Springer*.
6. L. L. Constantine. 2003. Canonical Abstract Prototypes for abstract visual and interaction. *Proc. of DSV-IS'03. Springer*.
7. E. Gamma, R. Helm, R. Johnson and J. Vlissides. 1994. Design Patterns – Elements of Reusable Object-Oriented Software. *Addison-Wesley*.
8. J. Gonzalez-Calleros, J. Vanderdonckt, A. Lüdtke and J-P. Osterloh 2010. Towards Model-Based AHMI Development. *Proc. of HCI-Aero'10. ACM*.
9. Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, V. López-Jaquero. 2004. USIXML: a language supporting multi-path development of user interfaces. *Proc. of EHCI-DSVIS'04. Springer*.
10. J. Lin, J.A. Landay. 2008. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. *Proc. of CHI'08. ACM*.
11. V. López-Jaquero, J. Vanderdonckt, F. Montero and P. González. 2007. Towards an Extended Model of User Interface Adaptation: The ISATINE Framework. *Proc. of EIS'07, LNCS 4940, Springer*.
12. G. Meixner, F. Paternò and J. Vanderdonckt 2011. Past, Present, and Future of Model-Based User Interface Development. *De Gruyter i-com* 10(3), 2-11.
13. V. G. Motti and J. Vanderdonckt 2013. A Unified Model for Context-aware Adaptation of User Interfaces. *Revista Română de Interacțiune Om-Calculator* 6(3), 211-248.
14. D. Navarre, P. Palanque, J-F. Ladry, and E. Barboni. 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Trans. Comput.-Hum. Interact.* 16(4).
15. E.G. Nilsson, J. Floch, S. Hallsteinsen, and E. Stav. 2006. Model-based User Interface Adaptation. *Elsevier Computers & Graphics*, 30(5), 692-701.
16. E.G. Nilsson and K. Stølen. 2010. Ad Hoc Networks and Mobile Devices in Emergency Response – a Perfect Match? *Proc. Second International Conference on Ad Hoc Networks. Springer*.
17. E.G. Nilsson and K. Stølen. 2011. Generic functionality in user interfaces for emergency response. *Proc. OZCHI'11. ACM*.
18. E.G. Nilsson and K. Stølen. 2016. A Case-based Assessment of the FLUIDE Framework for Specifying Emergency Response User Interfaces. *Proc. of EICS'16. ACM*.
19. E.G. Nilsson and K. Stølen. 2016. The FLUIDE Framework for Specifying Emergency Response User Interfaces Employed to a Search and Rescue Case. *Proceedings of ISCRAM 2016*.
20. E.G. Nilsson and K. Stølen. 2016. The FLUIDE Specification Languages with an Accompanying Method. *Technical report A27972, SINTEF ICT*.
21. F. Paternò. 1999. Model-based Design and Evaluation of Interactive Applications, *Springer*.

22. F. Paternò, C. Santoro, and L.D. Spano. 2009. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. on Computer-Human Interaction* 16(4).
23. M. Peissner, D. Häbe, D. Janssen and T. Sellner. 2012. MyUI: generating accessible user interfaces from multimodal design patterns. *Proc. of EICS'12. ACM.*
24. H. Trættemberg. 2002. Model-based User Interface Design. *PhD thesis, NTNU.*
25. J. Van den Bergh, K. Luyten and K. Coninx. 2011. CAP3: context-sensitive abstract user interface specification. *Proc. of EICS'11. ACM.*
26. J. Vanderdonck 2008. Model-Driven Engineering of User Interfaces: Promises, Successes, and Failures. *Proc. of ROCHI'08. Matrix Rom.*
27. J. Vanderdonck and F. M. Simarro 2013. Generative Pattern-Based Design of User Interfaces. *Proc. of PEICS'10. ACM.*

Appendix A - Complete Specification of the Case

This appendix contains the full description of the three applications and the corresponding FLUIDE specifications.

A.1 The MASTER Application

The MASTER application consists of a large map display where the map is augmented with information overlays (icons and other visual representations) with relevant information for the emergency response at hand. This information may be divided into different categories [17]. Figure A.1 shows a picture of slightly different version of the MASTER application running on a PixelSense table.

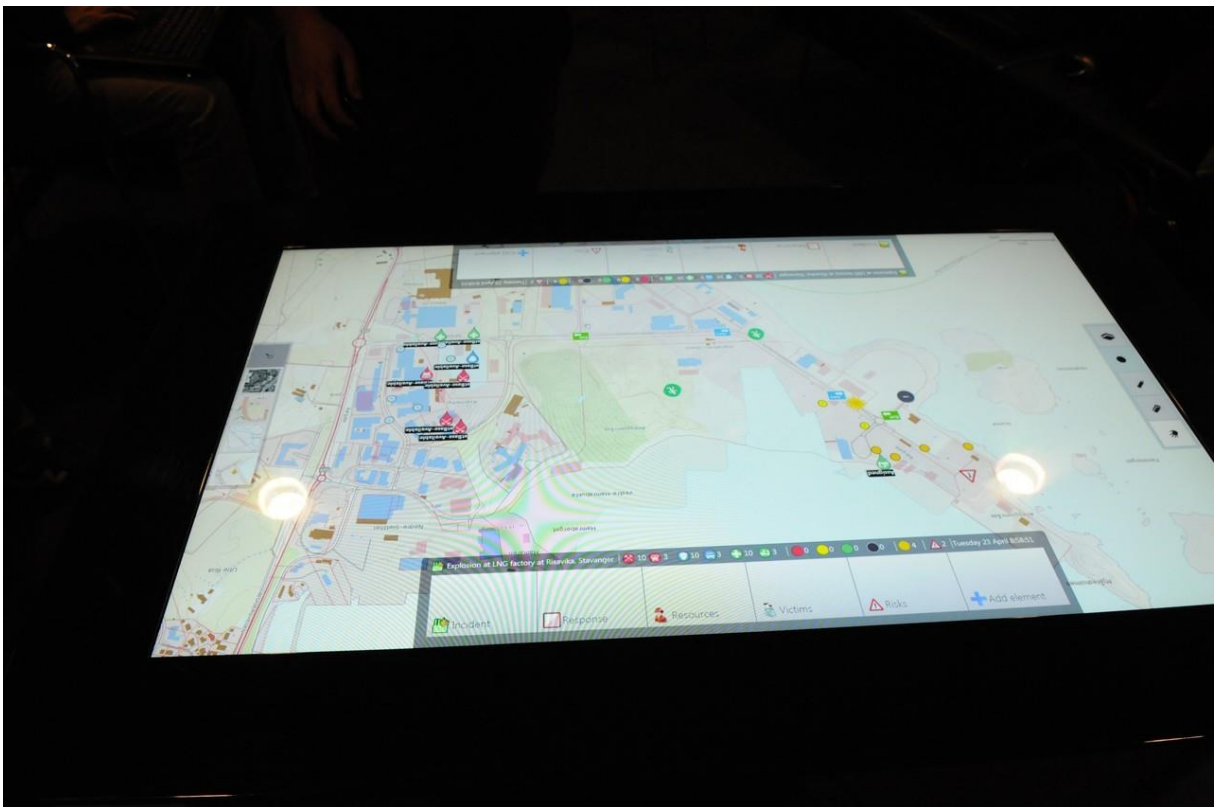


Figure A.1 – MASTER application running on the PixelSense table

Figure A.2 shows a schematic view the main map-based user interface.

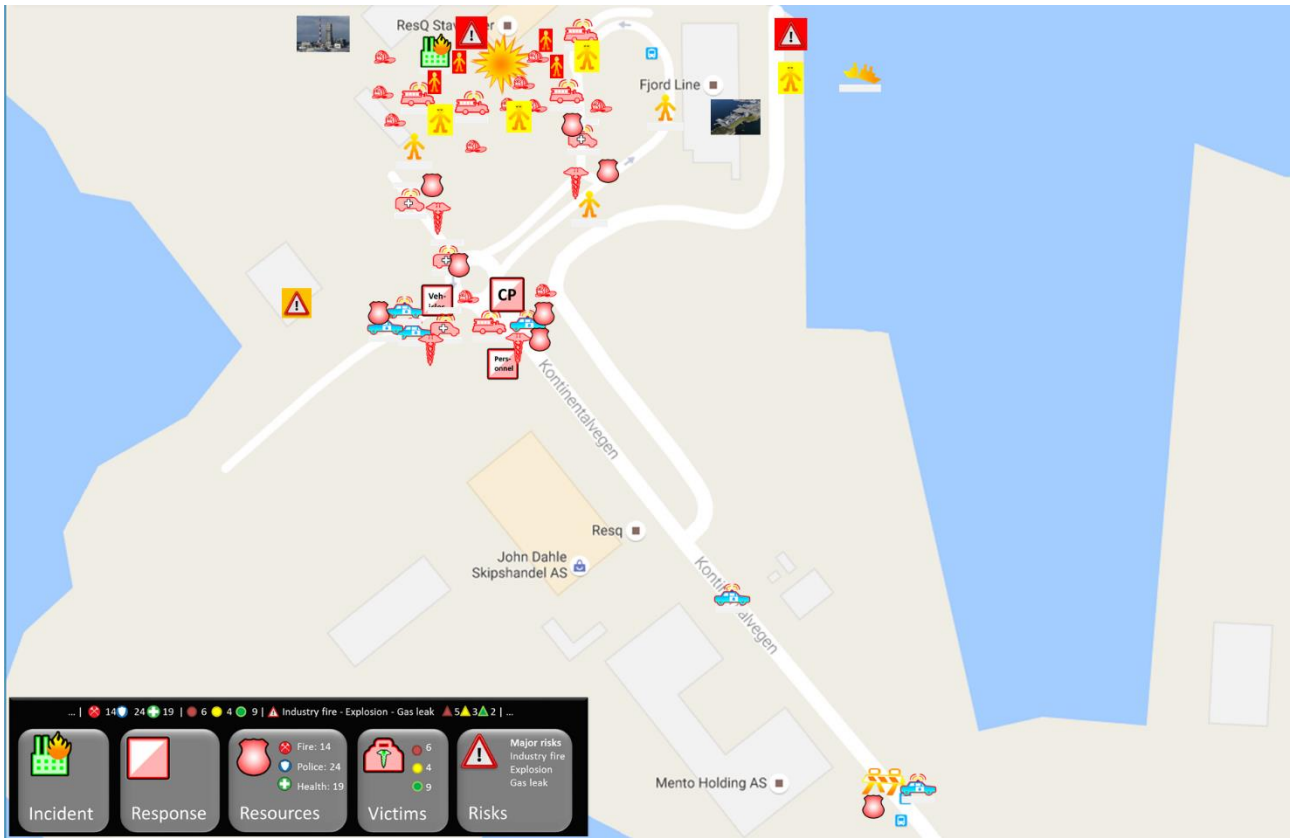


Figure A.2 – Main map user interface for the MASTER application

All information overlays on the map belong to one of five categories. In addition to the information overlays on the map, the user interface includes a *ribbon* (at the bottom left in Figure A.2) showing the information elements in the overlays grouped by these categories, which are further divided into sub categories. On the top level, the ribbon contains a set of buttons for accessing the information in each category, as well as a ticker showing summary information – as illustrated in Figure A.3.



Figure A.3 – Ribbon showing the ticker and the top level buttons with summary information

Below we present the each of these categories, including the target user interface and the corresponding FLUIDE-A and FLUIDE-D specifications. After that, we present the FLUIDE-A and FLUIDE-D specifications for the ribbon itself (Figure A.3), the map part (Figure A.2), as well as how the ribbon and the map parts of the user interface are coupled.

A.1.1 Incident category

The ribbon for the incident category is shown in Figure A.4. As for the other ribbon category presentations, it contains some overview information at the left and icons for each of the sub categories in a horizontal scrollable view on the right. Each icon in the ribbon represents some information element from the incident overlay on the map. Each icon in a sub category in the ribbon is also shown on the map (although it may not be visible in the current map extent). The white areas under the icons are placeholders for labels on the icons. The plus icons represent functionality for adding elements of the given sub category (by dragging the plus icon and dropping it on the map).

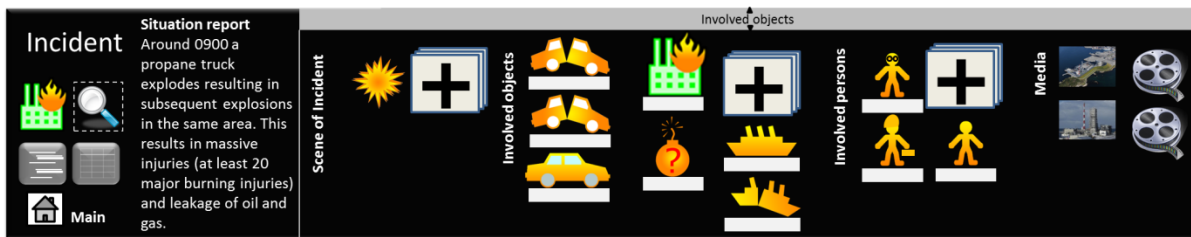


Figure A.4 – Ribbon content for the incident category

A.1.1.1 FLUIDE-A specifications of the Incident category

Figure A.5 to Figure A.10 show how the user interface in Figure A.4 may be specified in FLUIDE-A. Except for the *Incident Presenter* (Figure A.5) and *Incident Category Presenter* (Figure A.10), all the FLUIDE-A specifications also apply to the map overlay for this category.

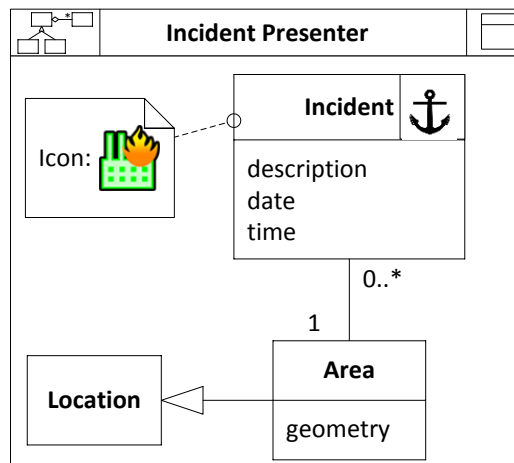


Figure A.5 - FLUIDE-A specification of the overview information shown at the left in Figure A.4

The outer border of a FLUIDE-A specification indicates an instance of one of the main constructs in the language. Which construct is determined by the symbol in the top left corner. The symbol shown in Figure A.5 denotes that a Content Presenter is being specified. The symbol in the top right corner indicates whether the presenter is basic or aggregated. The symbol shown in Figure A.5 denotes a basic presenter. The symbol for aggregated presenter is shown in Figure A.10. The name of the interactor instance is given in the heading of the specification.

The content part of the specification contains the connected concept model fragment associated with the presenter. This part of the specification is expressed in standard UML class model notation, except for the anchor symbol, which indicate which of the entities in the model fragment that act as the anchor for the model. The purpose and role of the anchors are explained in [20]. The UML class model comment construct is used to express user interface annotations to different parts of the associated model. The position in which the comment is attached denotes which part of the model the annotation applies to. The annotation specified in Figure A.5 expresses which icon that should be used to visualize instances of the entity Incident. Annotations may also be attached to attributes and relations.

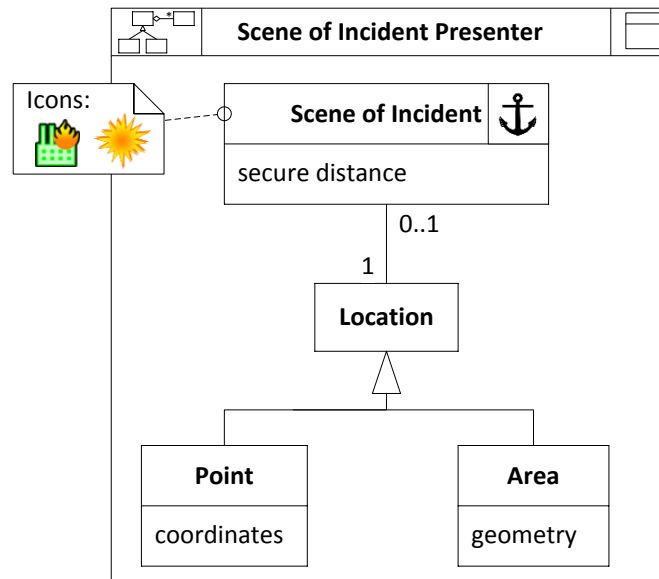


Figure A.6 - FLUIDE-A specification of the Scene of Incident sub category

The annotation on the *Scene of Incident* entity that is part of the *Scene of Incident Presenter* in Figure A.6 indicates that two different icons may be used to visualize instances of this entity.

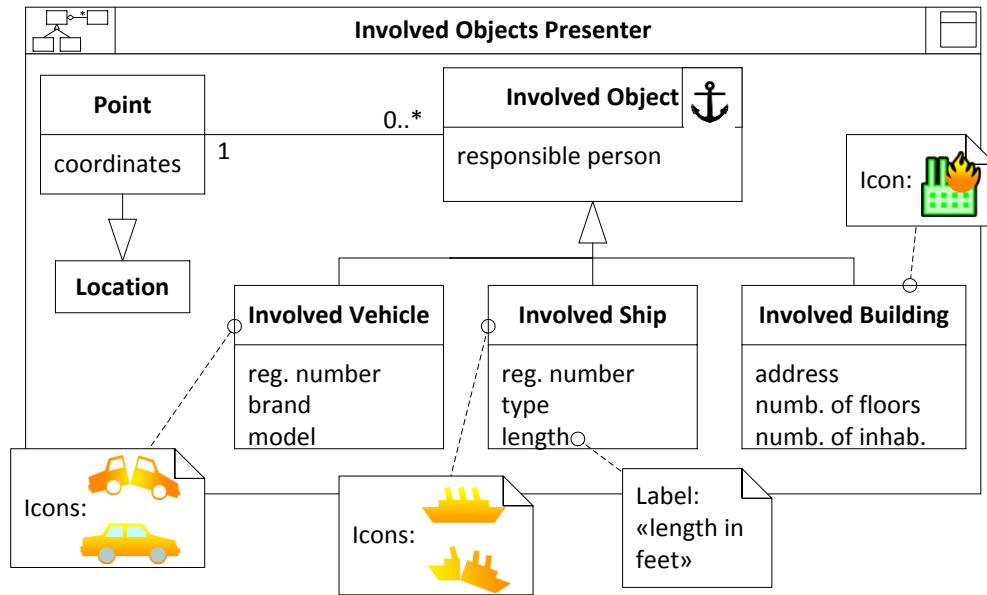


Figure A.7 - FLUIDE-A specification of the Involved Objects sub category

The *Involved Objects Presenter* in Figure A.7 includes an annotation with the content 'Label: "length in feet"'. This annotation applies to the length attribute of the *Involved Ship* entity. Also note that each of the sub types of the entity *Involved Object* are annotated with different sets of icons.

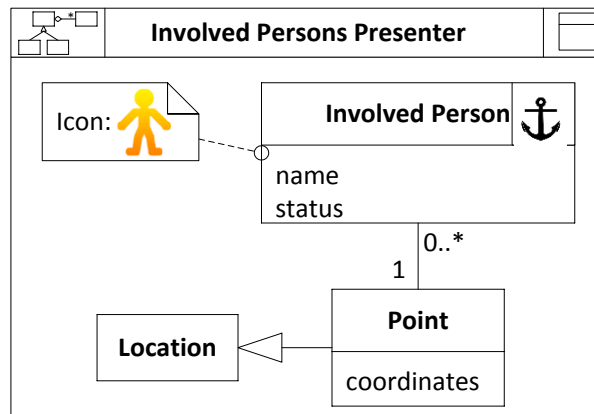


Figure A.8 - FLUIDE-A specification of the Involved Persons sub category

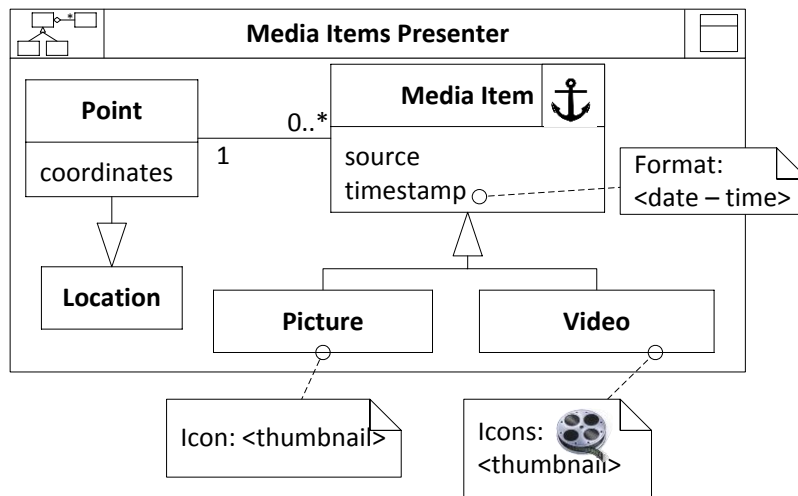


Figure A.9 - FLUIDE-A specification of the Media sub category

The *Media Items Presenter* shown in Figure A.9 includes three annotations. Two of these use the key word *<thumbnail>*, which denotes that instead of a fixed icon, a thumbnail of the *Picture* or *Video* should be used instead. For the entity *Video*, the annotation also includes a default icon which should be used in case a thumbnail is not available. The annotation on the attribute *timestamp* of the entity *Media Item* specifies that the date part of the timestamp should be presented before the time part. How the actual date and time parts should be formatted is not specified.

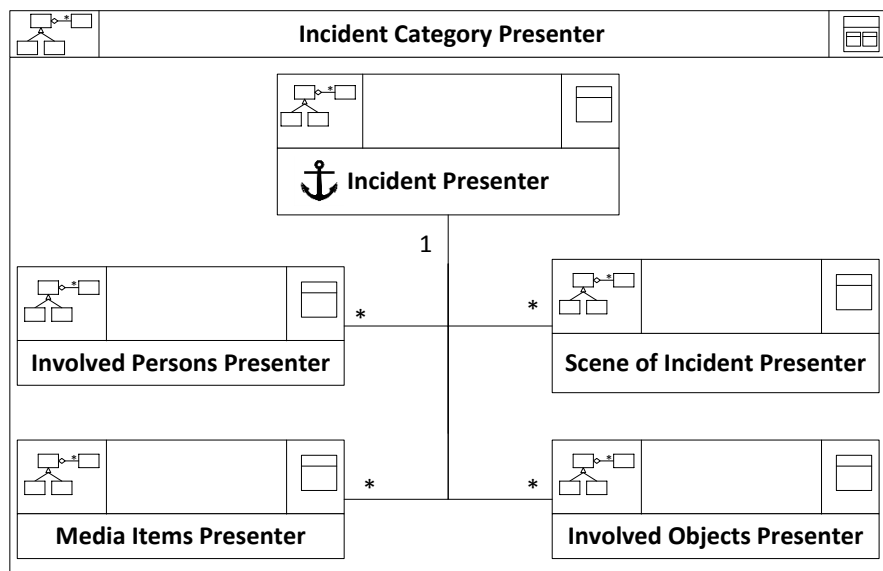


Figure A.10 - FLUIDE-A specification of the Incident Category (i.e. the whole user interface shown in Figure A.4)

Unlike the specifications in Figure A.5 - Figure A.9 (which all are Basic Content Presenters), the *Incident Category Presenter* specified in Figure A.10 is an Aggregated Content Presenter. This is indicated by the symbol in the top right corner.

The content part of an aggregated presenter does not contain a UML class model, but rather references to other content presenters. The specifications include only the border parts of the presenters that are part of the aggregated one. The names are shown inside the borders instead of in the heading to pinpoint that they represent other presenters. The anchor symbol indicates which of the child presenter the aggregated one inherits the anchor from. The relations between the child presenters specify presenter relations [20]. Aggregated presenters may not contain additional annotations.

A.1.1.2 FLUIDE-D specifications of the Incident category – Ribbon content

Figure A.11 to Figure A.16 show how the user interface for the ribbon content for the incident category (Figure A.4) may be specified in FLUIDE-D. These specifications only apply to the ribbon part of the user interface.

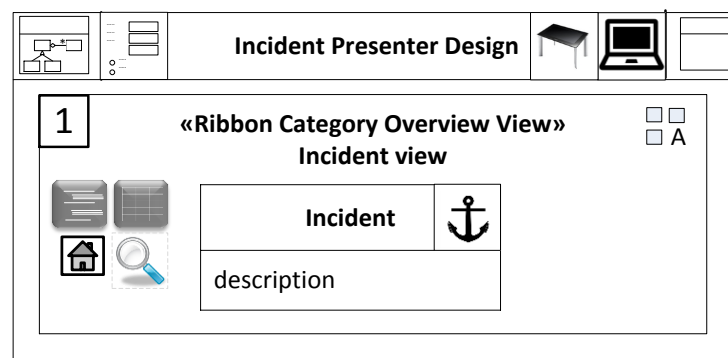


Figure A.11 - FLUIDE-D specification of the overview information shown at the left in Figure A.4 (corresponding FLUIDE-A specification is shown in Figure A.5)

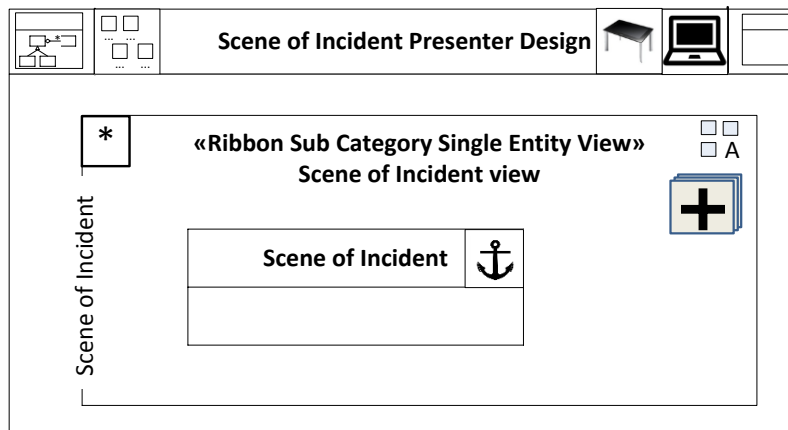
The outer border of a FLUIDE-D specification is built along the same lines as the FLUIDE-A specifications, but contains additional information. The symbols in the top left corner are the same as in FLUIDE-A except that a “window” frame is added around the FLUIDE-A symbols to indicate that the specification is closer to a final user interface. The symbol shown in Figure A.11 denotes that a Content Presenter Design is being specified. The symbols in the top right corner are the same as in FLUIDE-A.

The heading part of a FLUIDE-D specification has two additional symbol types. To the right of the construct symbol, one or more symbols are used to indicate which user interface style(s) that the design exploits. The symbol shown in Figure A.11 denotes that the design is forms-based. To the left of the basic/aggregated symbol, one or more symbols are used to indicate which modalities and/or platform(s) the design is targeted at. The symbols shown in Figure A.11 denote that the design is for a standard PC and table top using a visual presentation and mouse or touch interaction. The name in header part of the outer border specifies the name of the interactor design instance being specified (in this case the name of the Basic Content Presenter Design). The name is also used to connect the FLUIDE-D specification to the FLUIDE-A specification it is a refinement of. The name of a FLUIDE-D specification should be the same as the FLUIDE-A specification with the word “Design” added to it. As there may be more than one FLUIDE-D specification for each FLUIDE-A specification, an additional name may be added, concatenated with a dash or separated by a line break. The design shown in Figure A.11 does not exploit this option.

The content part contains the views that constitute the design. There are four main types of views, i.e. layout manager view, decorative view, content view and content integration view. The view shown in Figure A.11 is a content view, i.e. a view that presents instances of one or more entities. Content views are shown as a solid rectangle with the UML stereotype notation to denote the view type and the name of the view instance.

At the top left of a content view, a “1” or “*” is used to denote whether the view presents one or a number of instances of the anchor entity. This is usually given by the content view type, but is anyway included to make the semantics of the content view type explicit, and thus enhance the readability of the specifications. At the top right of all view types except layout manager views, the layout method of the view instance is shown. The view shown in Figure A.11 uses automatic layout (given by the “A” in the layout symbol).

The content view type Ribbon Category Overview View used in Figure A.11 is a domain-specific content view type for presenting the left side of a ribbon category. As all content views, it imposes restrictions on the model fragment it may present, expressed in FLUIDE-D as a model pattern. The view shown in Figure A.11 requires a single entity (*Incident*) as its model pattern. The icons shown on the left hand side of the view are property setting for the view. In the view in Figure A.11, the icons specify that it should be possible to open lists and table presentations, to perform search, and to navigate to the top level of the ribbon (the home button). The label is automatically set from the name of the entity anchor, and the incident icon is obtain from the icon annotation on the Incident entity in the FLUIDE-A specification (Figure A.5).



**Figure A.12 - FLUIDE-D specification of the Scene of Incident sub category
(corresponding FLUIDE-A specification is shown in Figure A.6)**

The presenter design in Figure A.12 is icon based, which is indicated in the user interface style icon in the heading. It contains one Ribbon Sub Category Single Entity View. It is a “sibling type” to Ribbon Category Overview View used in the specification in Figure A.11. It is one of two domain-specific content view type for presenting one sub category of a ribbon category. It presents a number of instances (indicated by the “*” in the top left of the view) of one entity as grid or table of icons. The text on the left border of the view (*Scene of Incident*) is a property specifying the label to be used for the sub category. For the specification in Figure A.12, this setting is redundant, as the default value for the label is the name of the entity anchor.

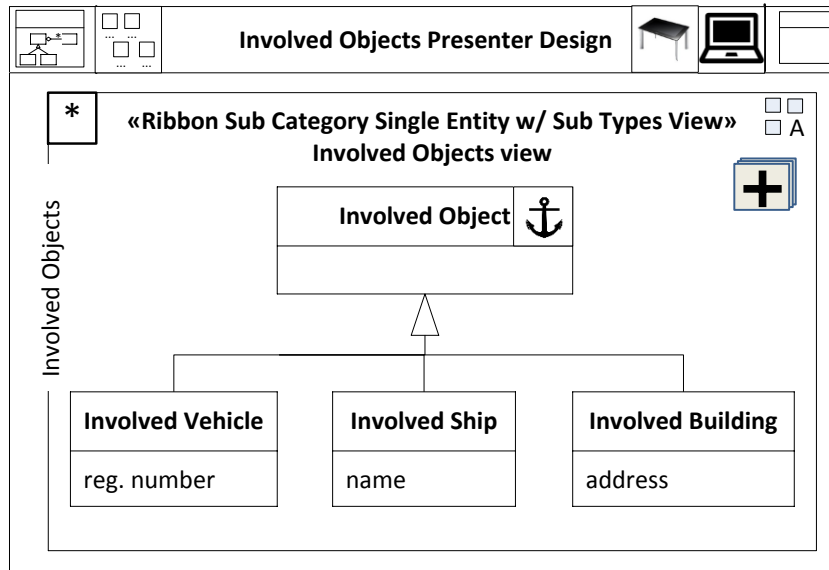


Figure A.13 - FLUIDE-D specification of the Involved Objects sub category (corresponding FLUIDE-A specification is shown in Figure A.7)

The presenter design in Figure A.13 contains one Ribbon Sub Category Single Entity with Sub Types View. It is a “sibling type” to Ribbon Sub Category Single Entity View and Ribbon Category Overview View used in the specification in Figure A.12 and Figure A.11. It is the second domain-specific content view type for presenting one sub category of a ribbon category. It presents a number of instances of one entity as grid or table of icons. The difference compared to Ribbon Sub Category Single Entity View is that the entity may have sub types displayed using different icons (typically one column per sub type). The text on the left border of the view (*Involved Objects*) is needed as the entity anchor has a different name.

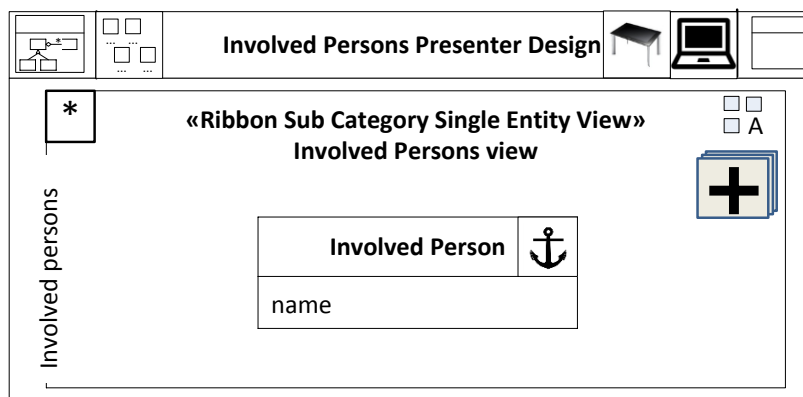


Figure A.14 - FLUIDE-D specification of the Involved Persons sub category (corresponding FLUIDE-A specification is shown in Figure A.8)

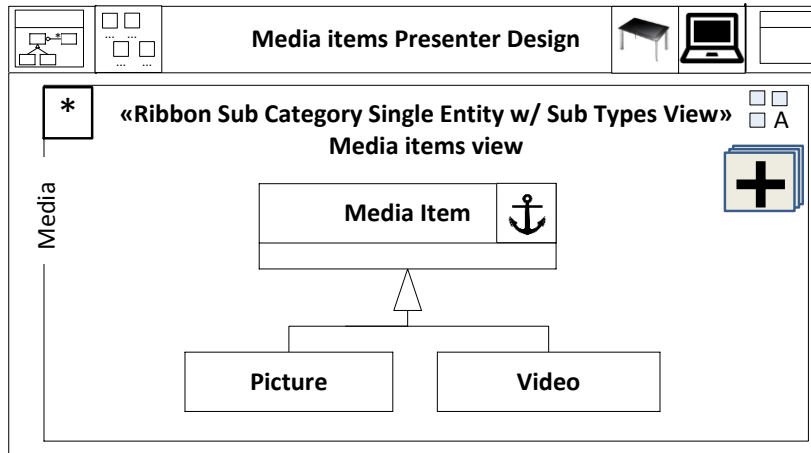


Figure A.15 - FLUIDE-D specification of the Media Items sub category (corresponding FLUIDE-A specification is shown in Figure A.9)

The specifications in Figure A.14 and Figure A.15 do not exploit any additional parts of FLUIDE-D.

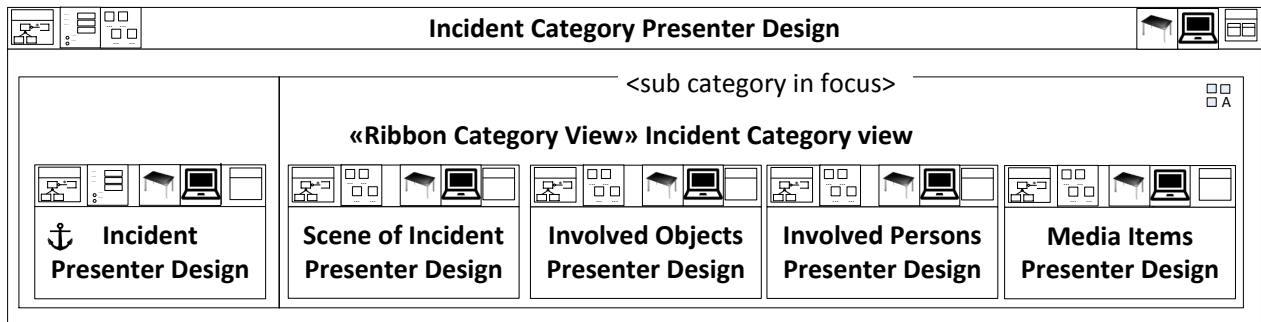


Figure A.16 - FLUIDE-D specification of the Incident Category (i.e. the whole user interface in Figure A.4) (corresponding FLUIDE-A specification is shown in Figure A.10)

Unlike the specifications in Figure A.11 - Figure A.15 (which all are Basic Content Presenter Designs), the *Incident Category Presenter Design* specified in Figure A.16 is an Aggregated Content Presenter Design. As for Aggregated Content Presenters, this is indicated by the symbol in the top right corner. The specification includes only the border parts of the child presenter designs. To pinpoint that the symbols represent another presenter design (more precisely, that it is a presenter design view), the names are shown inside the borders instead of in the heading. The anchor symbol indicates which of the member presenter design the aggregated one inherits the anchor from. As the presenter designs that are member of the aggregated presenter use different user interface styles, the aggregated presenter supports the sum of these. Thus, there are two user interface style icons in the left part of the heading (forms and icon based).

The presenter design in Figure A.16 contains one Ribbon Category View, which is a domain-specific content integration view. Content integration views integrate related content from different presenter designs. Content integration views require that their member presenter designs use specific content or content integration views. Ribbon Category Views have two slots. The slot to the left of the vertical line may only aggregate exactly one presenter design containing a Ribbon Category Overview View. The right hand slot may aggregate any number of Ribbon Sub Category Single Entity Views and/or Ribbon Sub Category Single

Entity with Sub Types Views, or one single Ribbon Sub Categories Categorized Single Entity View or Ribbon Sub Categories Categorized Single Entity with Sub Types View. The label property ("*<sub category in focus>*") uses a dynamic value.

A.1.1.3 FLUIDE-D specifications of the Incident category – Map overlays

Figure A.17 to Figure A.21 show how the user interface for map overlays (Figure A.2) for the incident category may be specified in FLUIDE-D.

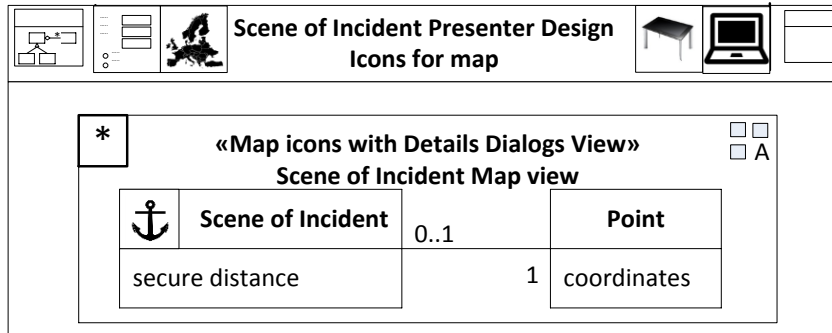


Figure A.17 - FLUIDE-D specification of map overlays for the Scene of Incident sub category (corresponding FLUIDE-A specification is shown in Figure A.6)

The presenter design in Figure A.17 applies a combined forms and map based user interface style, shown by the icons on the left side in header. It contains one Map Icons with Details Dialog View. This is a domain-specific content view type. The model pattern must contain one entity (possibly with subtypes) that has an association to a location entity (providing a point). It may also include related entities, as long as the cardinality on the side of the related entity is one. It presents a number of instances (specified by the "*" in the top left of the view) of the presented entities as icons on a map. In addition to providing the icons, the view type also provides forms based presentations of the instances represented by the icons in pop-up windows. These windows use automatic layout and present the attributes provided in the entity presented, as well as any one-related entities that are included in the model fragment (except for the entity providing the location).

If the presenter design using this type of view is member of a Map View content integration view (either directly or one or more times among its parents), the icons are shown on the map provided by the Map View highest up in the hierarchy. If a Map Icons with Details Dialog View does not have a parent providing a Map View, it will provide its own map.

The name of the presenter design is the name of the presenter it refines with an additional name added (*Icons for map*). This is needed to distinguish it from the presenter design that is specified for the same presenter as part of the ribbon specification (Figure A.12).

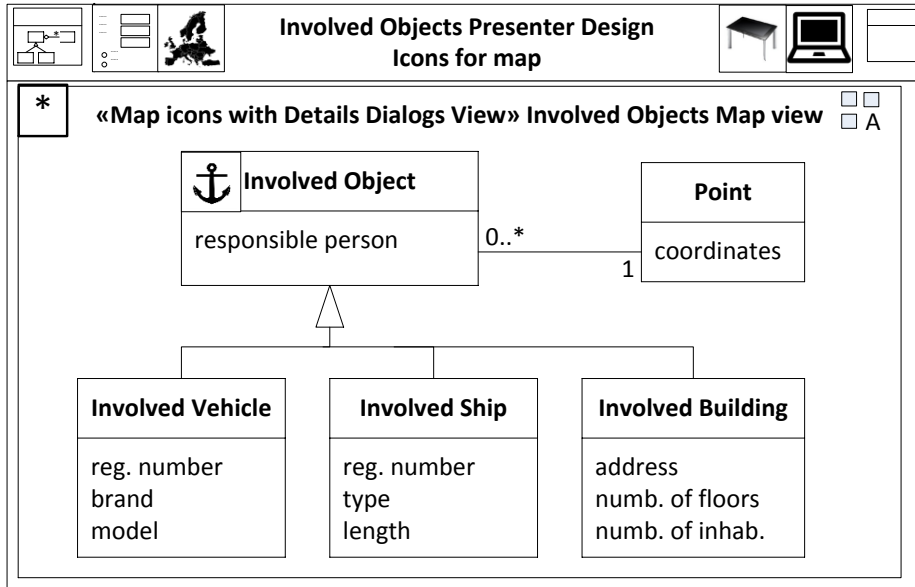


Figure A.18 - FLUIDE-D specification of map overlays for the Involved Objects sub category (corresponding FLUIDE-A specification is shown in Figure A.7)

In the presenter design in Figure A.18, the entity to show as icons on the map has sub classes, visualized using different icons.

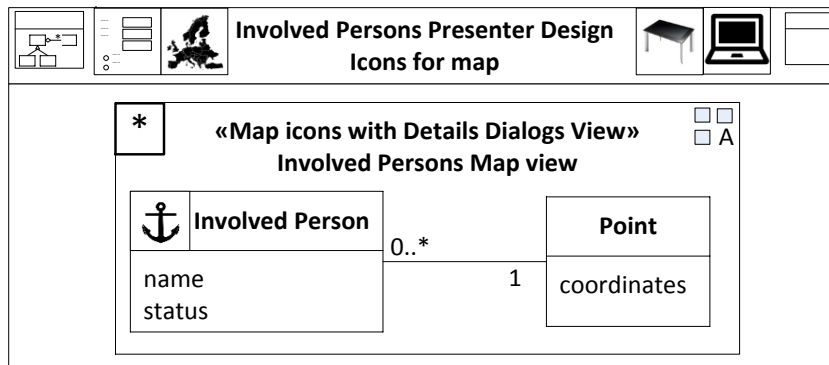


Figure A.19 - FLUIDE-D specification of map overlays for the Involved Persons sub category (corresponding FLUIDE-A specification is shown in Figure A.8)

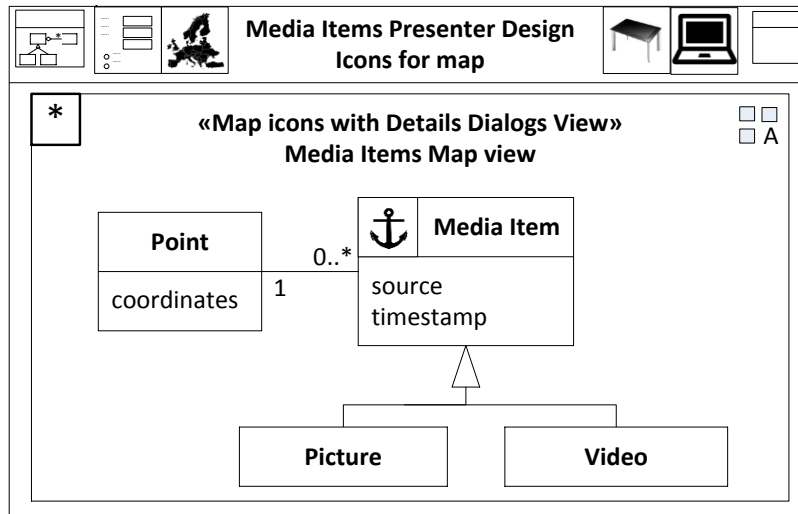


Figure A.20 - FLUIDE-D specification of map overlays for the Media Items sub category (corresponding FLUIDE-A specification is shown in Figure A.9)

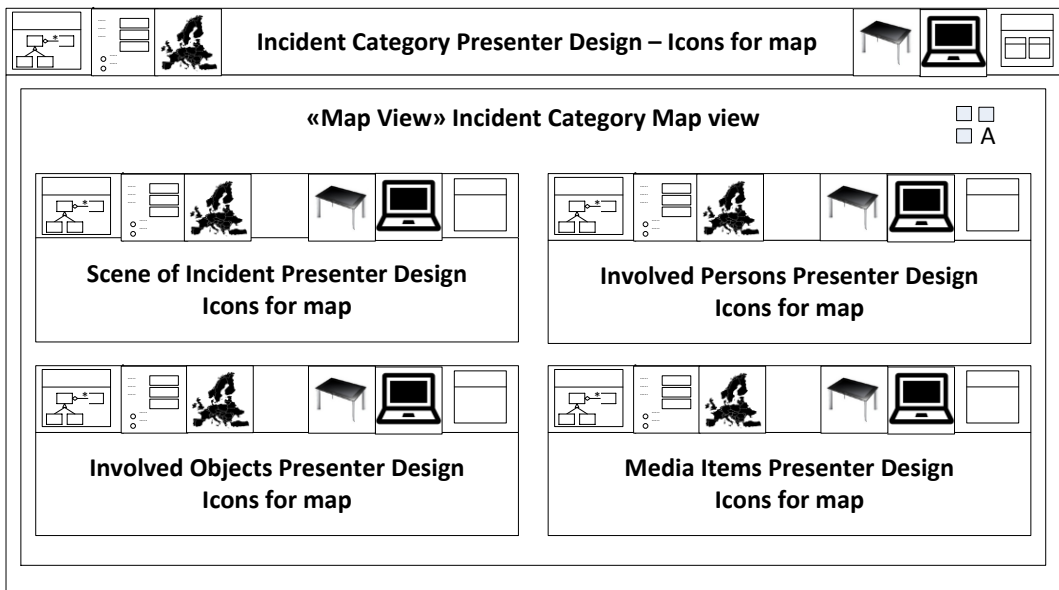


Figure A.21 - FLUIDE-D specification of map overlays for the Incident Category (corresponding FLUIDE-A specification is shown in Figure A.10)

The presenter design in Figure A.21 is an aggregated presenter design. It contains one view, which is a Map View type content integration view. It may have any number of presenter designs as members as long as these exploit other Map Views, Map Icons View, Map Icons with Details Dialog View or Map Outline View. A presenter design using a Map View may itself be member of a Map View (in any number of levels). As explained above, the presenter design having a Map View highest in the hierarchy provides the actual map on which all the content is presented.

It should also be noted that none of the member presenter designs contains the anchor of the associated model fragment (shown in the corresponding FLUIDE-A specification). This cause no problems, as the information about the anchor, which is used to determine which instances that will be shown on the map, is part of the corresponding FLUIDE-A specification which is available when needed.

A.1.2 Response category

The ribbon for the response category is shown in Figure A.22.

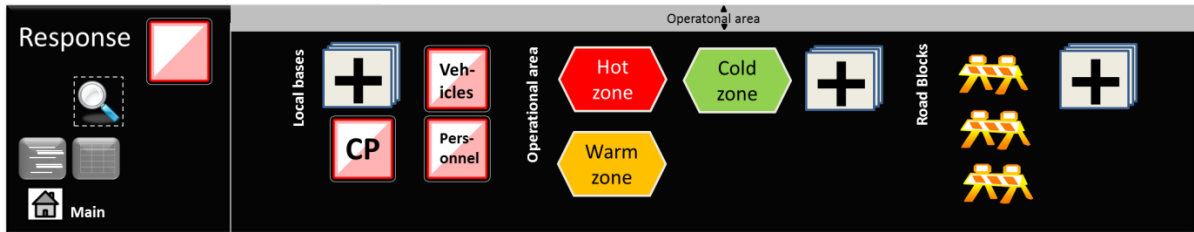


Figure A.22 – Ribbon content for the response category

As both the FLUIDE-A and FLUIDE-D specifications are similar to the corresponding specifications for the Incident category, there are few or no comments and explanations to the specifications.

A.1.2.1 FLUIDE-A specifications of the Response category

Figure A.23 to Figure A.27 show how this user interface may be specified in FLUIDE-A. Except for the *Response Presenter* (Figure A.23) and *Response Category Presenter* (Figure A.27), all these FLUIDE-A specification also apply to the map overlay for this category.

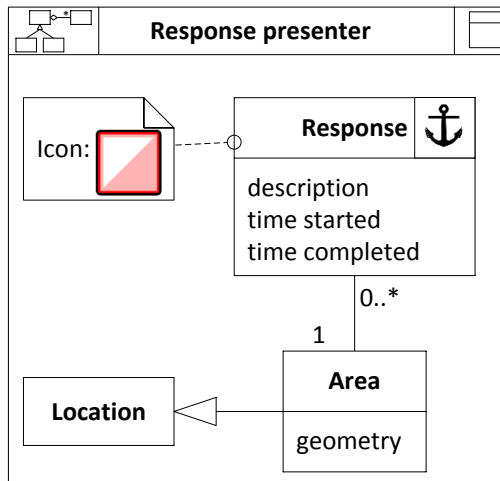


Figure A.23 - FLUIDE-A specification of the overview information shown at the left in Figure A.22

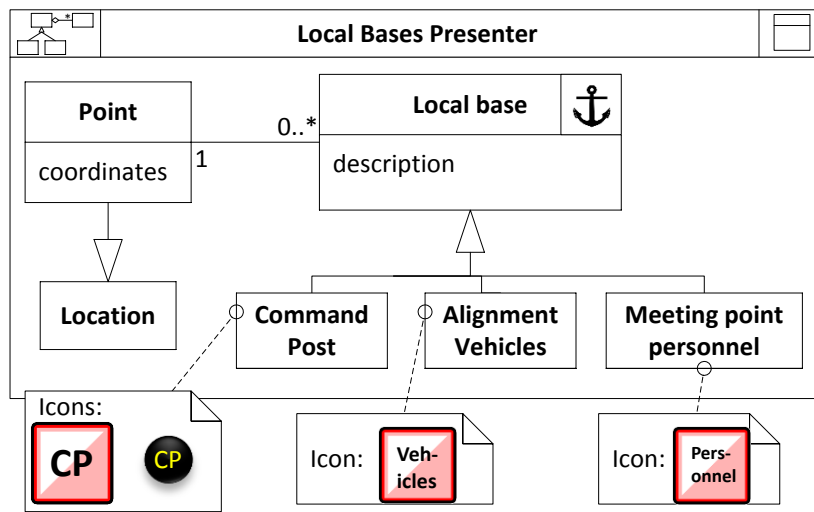


Figure A.24 - FLUIDE-A specification of the Local Bases sub category

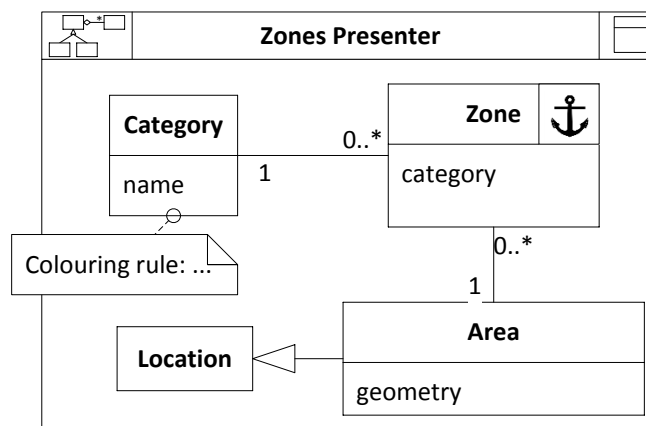


Figure A.25 - FLUIDE-A specification of the Operational Area (zones) sub category

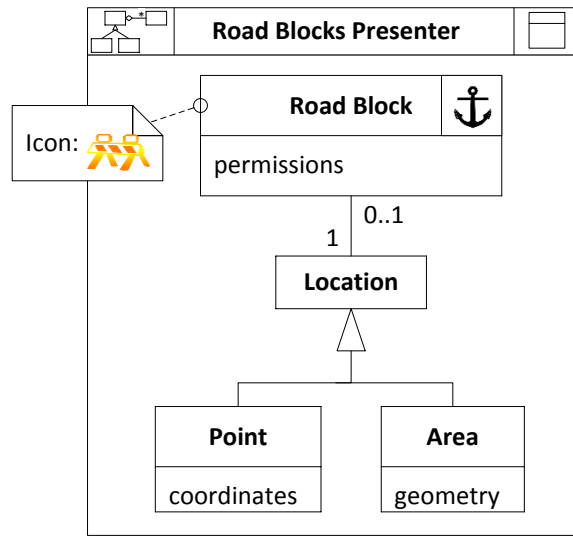


Figure A.26 - FLUIDE-A specification of the Road Blocks sub category

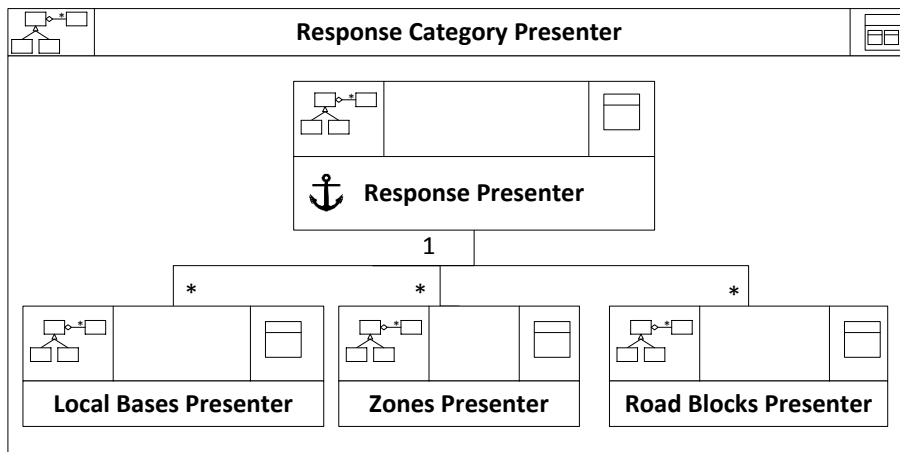


Figure A.27 - FLUIDE-A specification of the Response Category (i.e. the whole user interface in Figure A.22)

A.1.2.2 FLUIDE-D specifications of the Response category – Ribbon content

Figure A.28 to Figure A.32 show how the user interface for the ribbon content for the response category (Figure A.22) may be specified in FLUIDE-D. These specifications only apply to the ribbon part of the user interface.

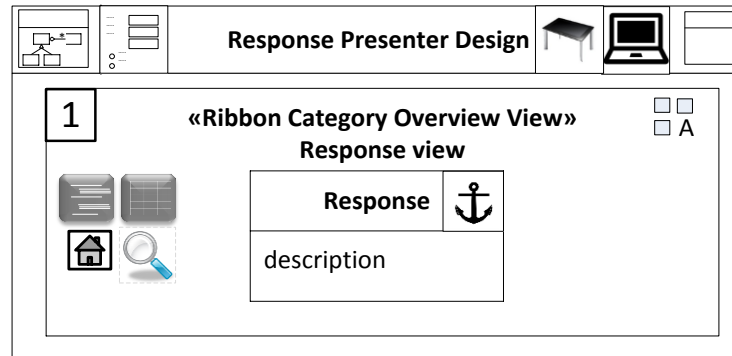


Figure A.28 - FLUIDE-D specification of the overview information shown at the left in Figure A.22 (corresponding FLUIDE-A specification is shown in Figure A.23)

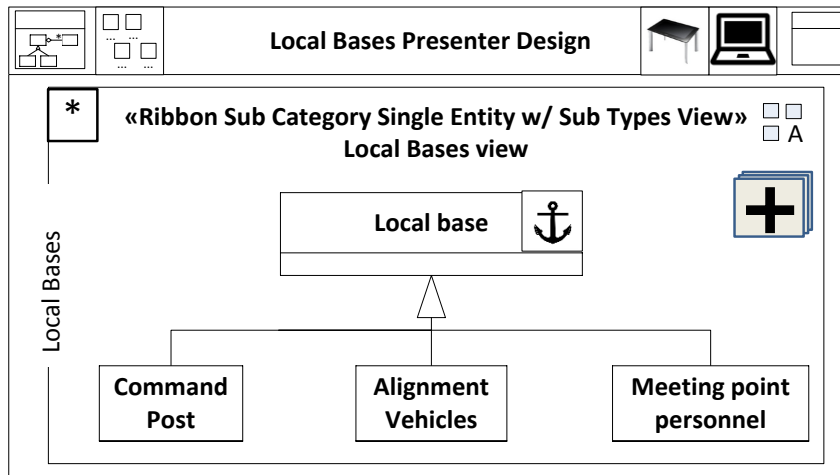


Figure A.29 - FLUIDE-D specification of the Local Bases sub category (corresponding FLUIDE-A specification is shown in Figure A.24)

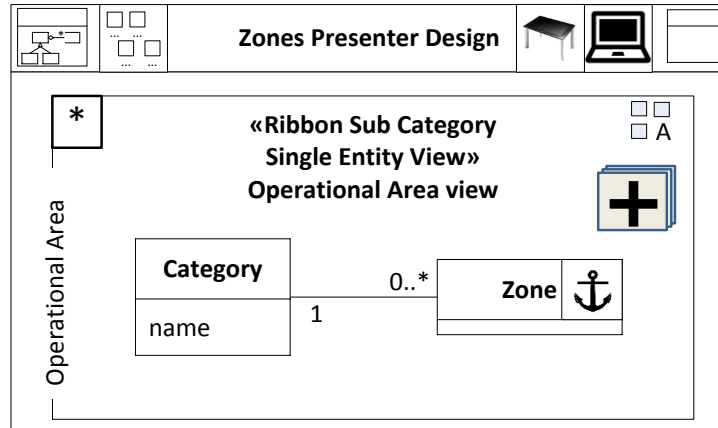


Figure A.30 - FLUIDE-D specification of the Operational Area (zones) sub category (corresponding FLUIDE-A specification is shown in Figure A.25)

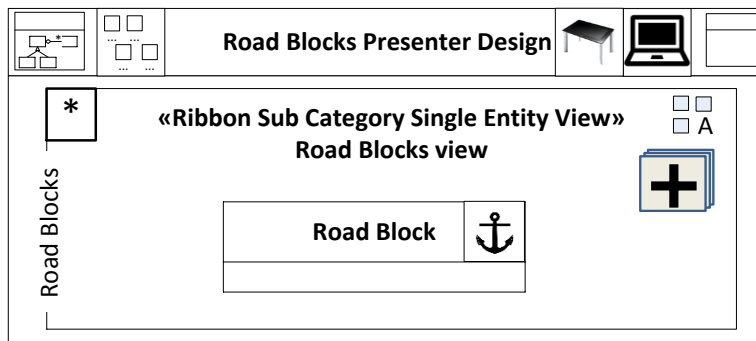


Figure A.31 - FLUIDE-D specification of the Road Block sub category (corresponding FLUIDE-A specification is shown in Figure A.26)

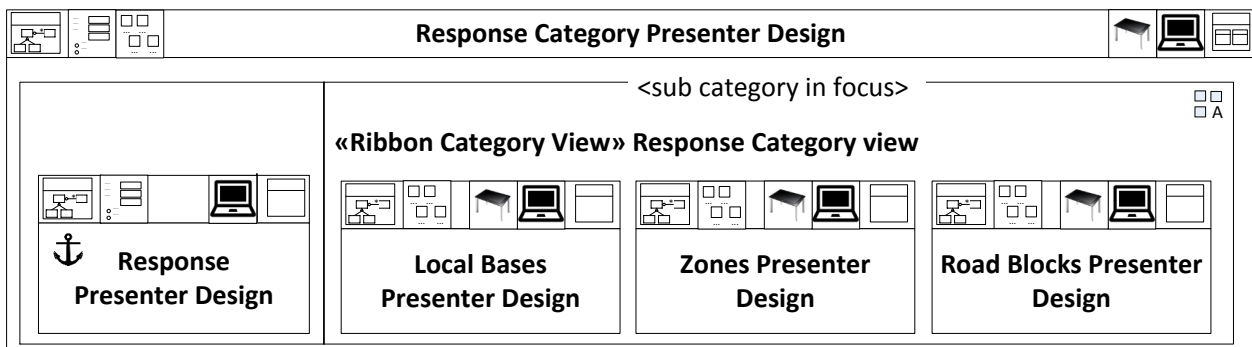


Figure A.32 - FLUIDE-D specification of the Response Category (i.e. the whole user interface in Figure A.22) (corresponding FLUIDE-A specification is shown in Figure A.27)

A.1.2.3 FLUIDE-D specifications of the Response category – Map overlays

Figure A.33 to Figure A.36 show how the user interface for map overlays (Figure A.2) for the response category may be specified in FLUIDE-D.

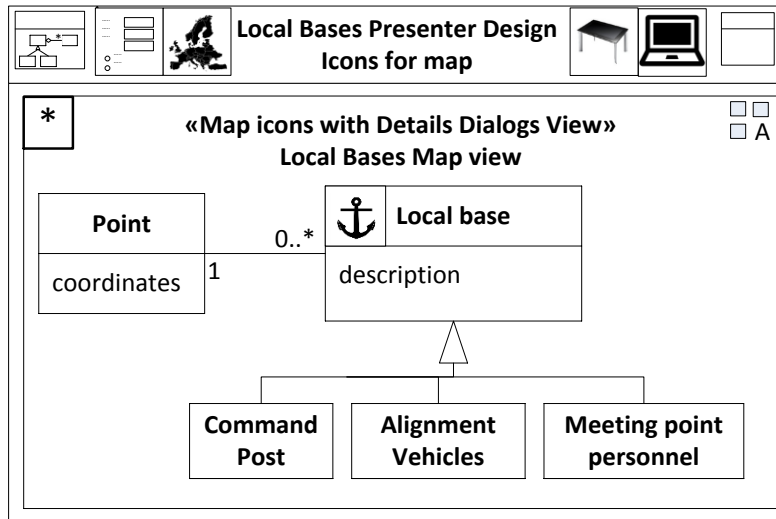


Figure A.33 - FLUIDE-D specification of map overlays for the Local Bases sub category (corresponding FLUIDE-A specification is shown in Figure A.24)

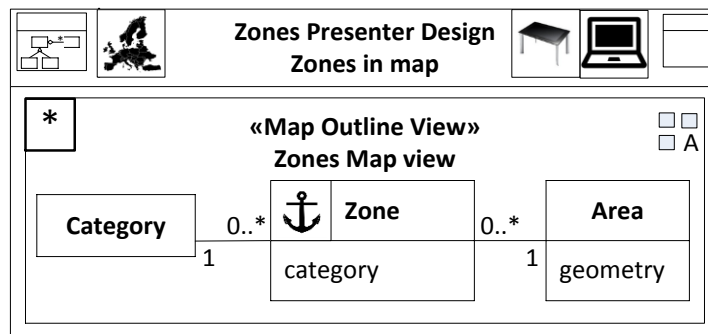


Figure A.34 - FLUIDE-D specification of map overlays for the Operational Area (zones) sub category (corresponding FLUIDE-A specification is shown in Figure A.25)

The presenter design in Figure A.34 contains one Map Outline View. This view is similar to the Map Icons with Details Dialog View (and Map Icons View presented in Figure A.104), except that it presents the entities as polygons instead of icons. As Map Icons View, Map Outline View does not provide forms based presentations of the icons.

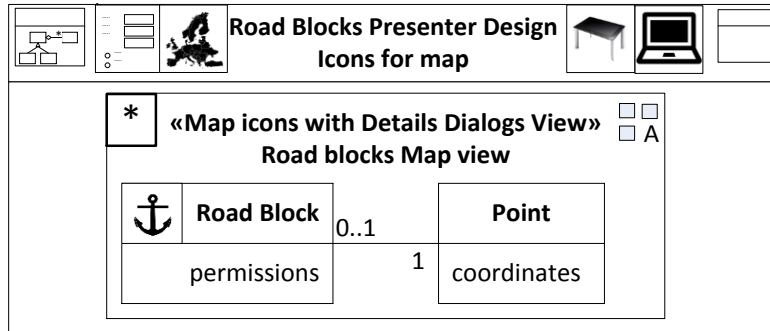


Figure A.35 - FLUIDE-D specification of map overlays for the Road Block sub category (corresponding FLUIDE-A specification is shown in Figure A.26)

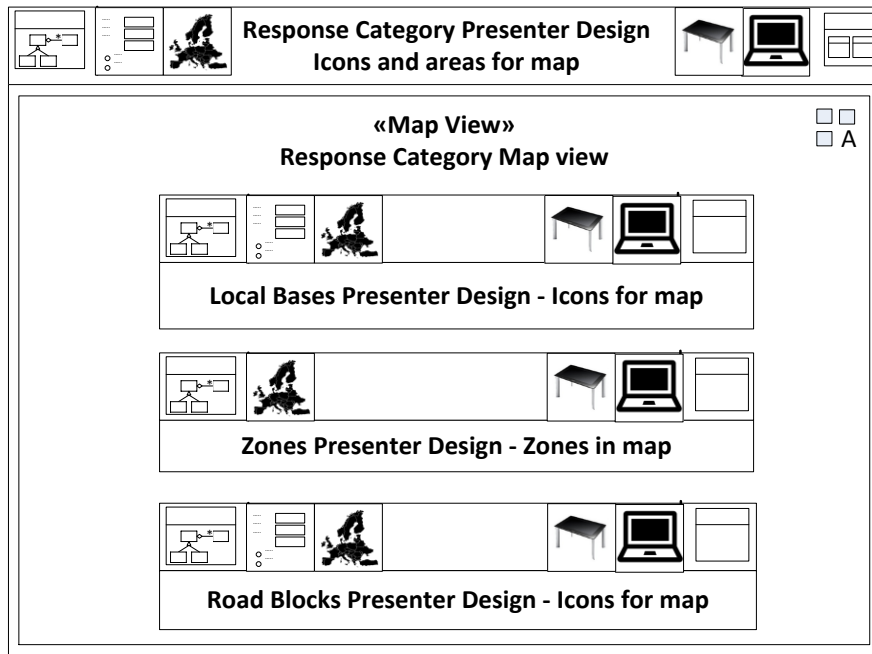


Figure A.36 - FLUIDE-D specification of map overlays for the Response Category (corresponding FLUIDE-A specification is shown in Figure A.27)

A.1.3 Resources category

The ribbon for the resources category is shown in Figure A.37.

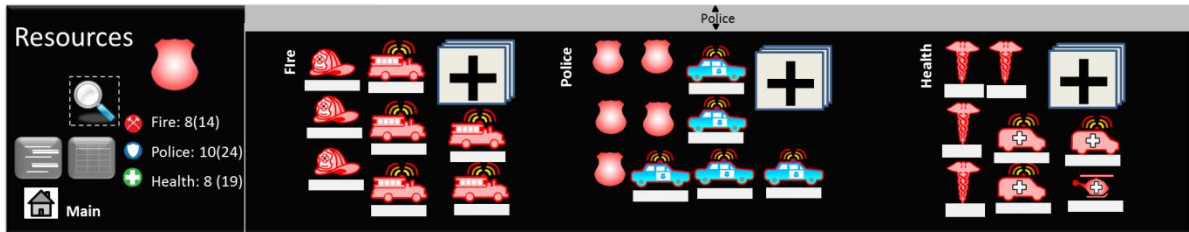


Figure A.37 – Ribbon content for the resources category

The information in the resources category may also be presented in a tabular form, as shown in Figure A.38.


Resources						
	Type	Agency	Status	Name	Radio	...
P1	Police unit, 2 officers	Oslo Politikammer	Patrolling	Hansen, Olsen	Ch. 5	...
P2	Police unit, 3 officers	Oslo Politikammer	Available	Johnsen, Person, Berg	Ch. 5	...
P4	Police unit, 2 officers	Follo Politikammer	Road block	Lium, Bjerke	Ch. 3	...
A1	Amb. unit, 2 param.	Oslo Amb.tjeneste	Available
A2	Amb. unit, 2 param., 1 doctor	Oslo Amb.tjeneste	One site
PO1	Police officer	Oslo Politikammer	Patrolling	Hansen	Ch. 5	...
PO2	Police officer	Follo Politikammer	Road block	Lium	Ch. 3	...
PA1	Paramedic	Oslo Amb.tjeneste	On site, triaging	Kvam

Figure A.38 – Tabular presentation of resource information

A.1.3.1 FLUIDE-A specifications of the Resources category

Figure A.39 to Figure A.41 show how these user interfaces may be specified in FLUIDE-A. The FLUIDE-A specification *Resource Presenter* (Figure A.40) also applies to the map overlay for this category.

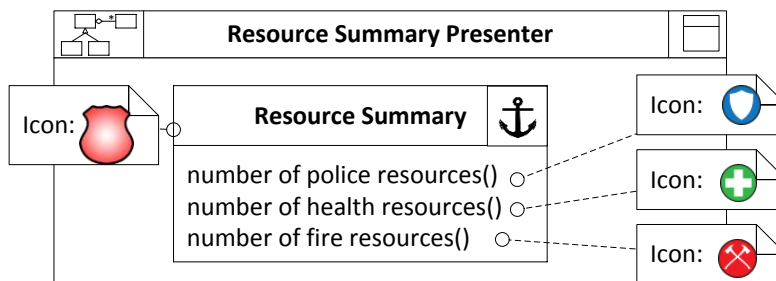


Figure A.39 - FLUIDE-A specification of the overview information shown at the left in Figure A.37

As the overview information for the resources category includes summary information for the across instances of the *Resource* entity, the presenter includes methods instead of attributes.

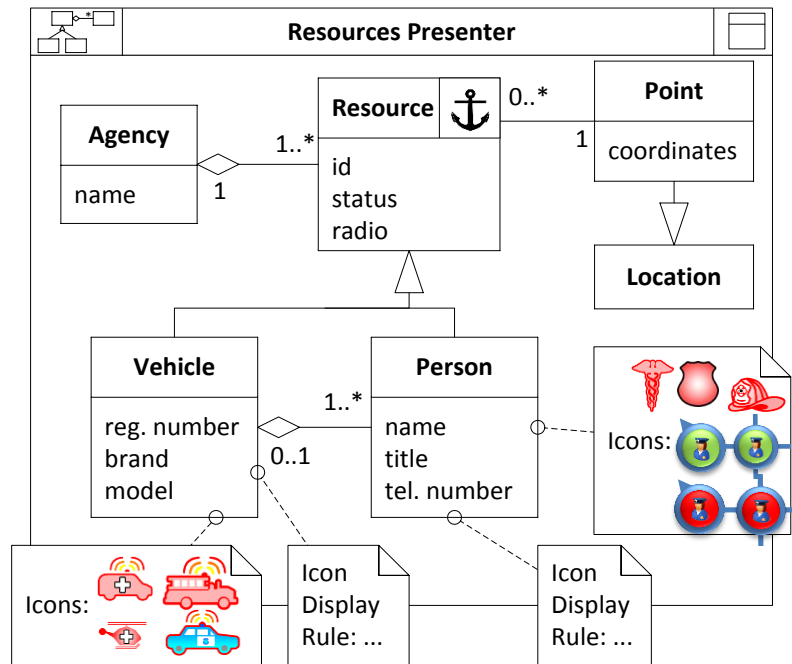


Figure A.40 - FLUIDE-A specification of all the Resources sub categories

In the incident and response categories, the sub categories were represented as different entities (with or without sub types). In the resources category, all sub categories are represented by the same entity (*Resource*). The sub categories are represented implicitly through the *Agency* entity in the connected concept model, which act as categorizer for the *Resource* entity. The sub types of *Resource* are used in a similar way to the sub category entities in the incident and response categories, and will cause different icons sets to be used within each sub category (determined by *Agency*). The FLUIDE-A specification in Figure A.40 also applies to the tabular presentation shown in Figure A.38.

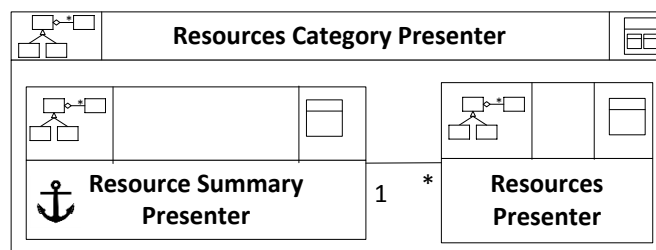


Figure A.41 - FLUIDE-A specification of the Resources Category (i.e. the whole user interface shown in Figure A.37)

A.1.3.2 FLUIDE-D specifications of the Resources category – Ribbon content

Figure A.42 to Figure A.44 show how the user interface for the ribbon content for the resources category (Figure A.37) may be specified in FLUIDE-D. These specifications only apply to the ribbon part of the user interface.

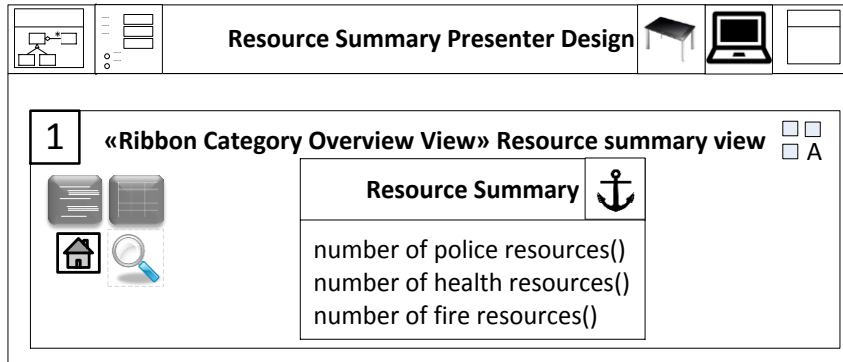


Figure A.42 - FLUIDE-D specification of the overview information shown at the left in Figure A.37 (corresponding FLUIDE-A specification is shown in Figure A.39)

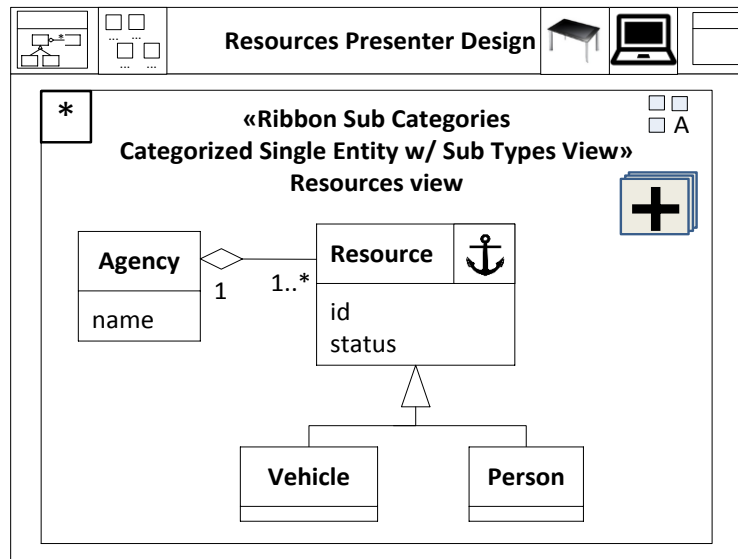


Figure A.43 - FLUIDE-D specification of all the Resources sub categories (corresponding FLUIDE-A specification is shown in Figure A.40)

The presenter design in Figure A.43 contains one Ribbon Sub Categories Categorized Single Entity with Sub Types View. This is the fourth content view type that may be member of a Ribbon Category View. It is a domain-specific content view type for presenting a number of sub category of a ribbon category based on a categorization entity. The model pattern must contain the entity (with subtypes) to be presented (*Resource*), and a categorizing entity (*Agency*). It presents a number of instances (documented using the “*” in the top left of the view) of the presented entity as a set of grids or tables of icons. The number of grids is determined by the number of instances of the categorizing entity. These instances also determine the labels for each sub type.

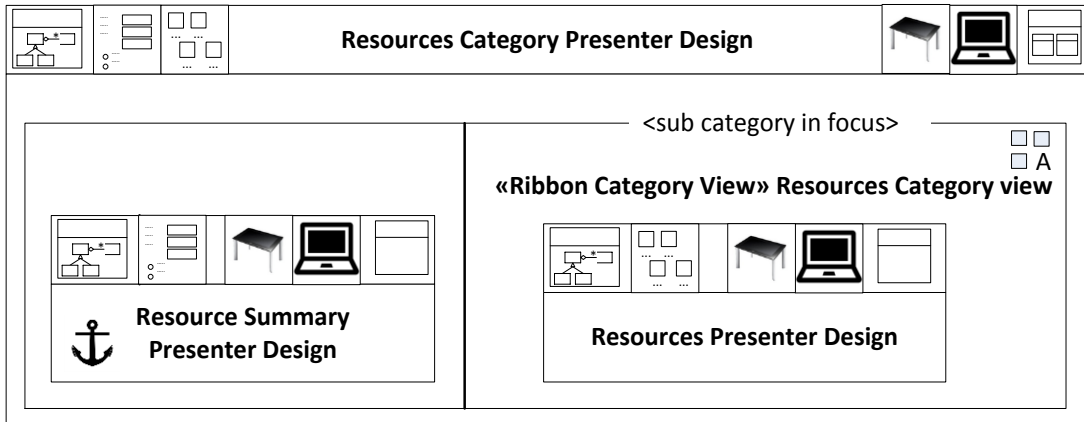


Figure A.44 - FLUIDE-D specification of the Resources Category (i.e. the whole user interface in Figure A.37) (corresponding FLUIDE-A specification is shown in Figure A.41)

The content integration view used in the presenter design in Figure A.44 (Ribbon Category View), is the same as was used for similar presenter designs in the incident and response categories. In the resources category, the presenter design in the right hand slot uses the Ribbon Sub Categories Categorized Single Entity with Sub Types View. There may only be one member presenter in this slot if it uses this view.

A.1.3.3 FLUIDE-D specifications of the Resources category – Tabular presentation

Figure A.45 shows how the tabular presentation of resource information (Figure A.38) may be specified in FLUIDE-D.

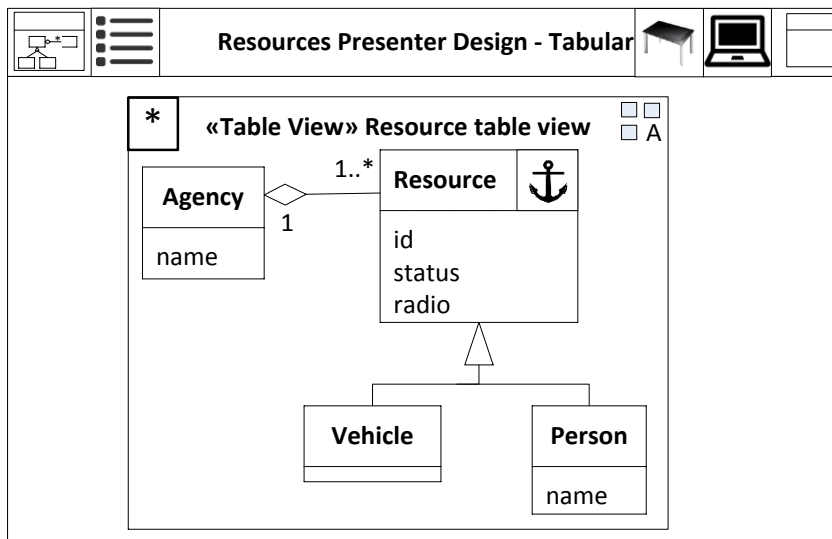


Figure A.45 - FLUIDE-D specification of the Tabular presentation of Resource information shown in Figure A.38 (corresponding FLUIDE-A specification is shown in Figure A.40)

The presenter design in Figure A.45 uses a list based user interface style, shown by the icon on the left side in header. It contains one Table View. This is generic content view type. The model pattern must contain one entity (possibly with subtypes) that determines the rows in the table (*Resource*). It may also include related entities, as long as the cardinality on the side of the related entity is one (as for *Agency* in the specification). It presents a number of instances (specified by the “*” in the top left of the view) of the presented entities in a table.

A.1.3.4 FLUIDE-D specifications of the Resources category – Map overlays

Figure A.46 and Figure A.47 show how the user interface for map overlays (Figure A.2) for the resources category may be specified in FLUIDE-D.

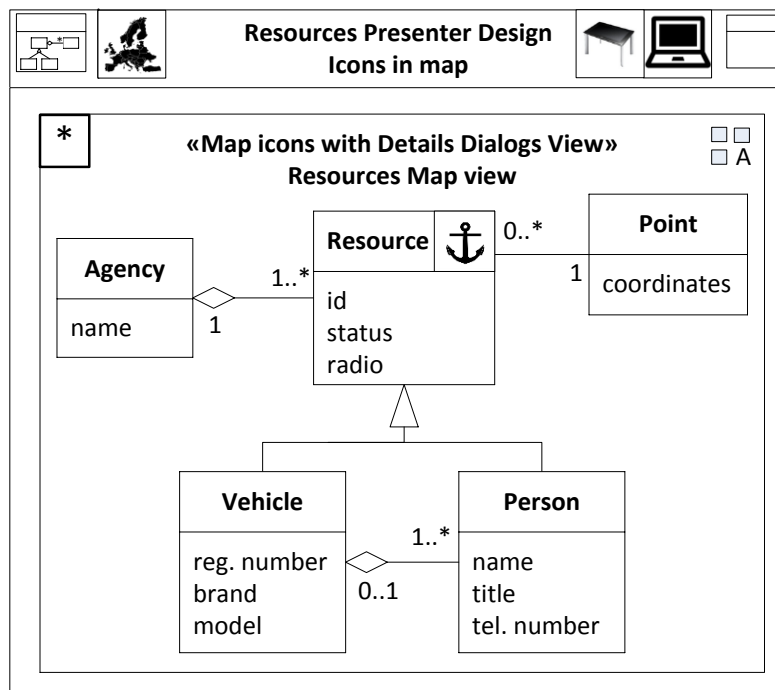


Figure A.46 - FLUIDE-D specification of map overlays for all the Resources sub categories (corresponding FLUIDE-A specification is shown in Figure A.40)

In the presenter design, the pop-up windows showing details will include an attribute from the related *Agency* entity. This entity will also be used to determine which icon to be used on the map (in the same way as the ribbon category icons, they are determined by combination of *Agency* and *Resource* sub type).

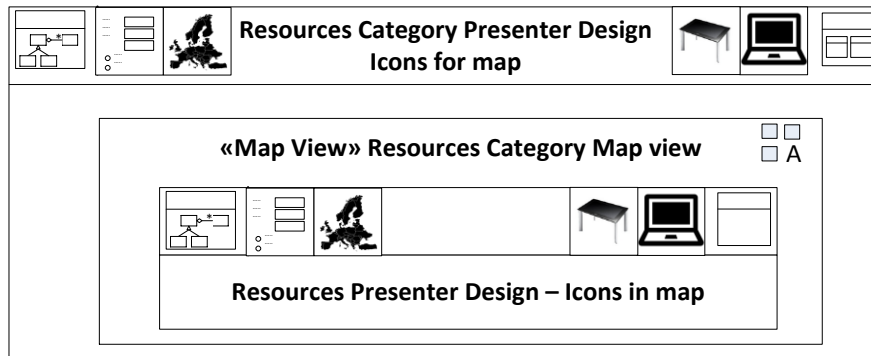


Figure A.47 - FLUIDE-D specification of map overlays for the Resources Category (corresponding FLUIDE-A specification is shown in Figure A.41)

A.1.4 Victims category

The ribbon for the victims category is shown in Figure A.48.

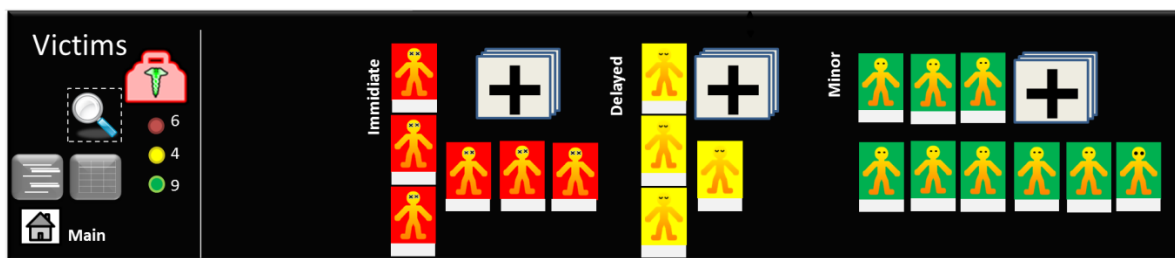


Figure A.48 – Ribbon content for the victims category

The information in the victims category may also be presenter in a tabular form, as shown in Figure A.49.


Patients						
	Status	Description	BP	Temp.	Name	...
1	Red	Unconscious	139/90	38	Unknown	...
8	Red	Seriously bleeding	80/40	35	Unknown	...
3	Red	Deliriously	200/140	40	Per Olsen	<input type="checkbox"/> ...
5	Yellow	Broken leg	Jon Tronstad	<input type="checkbox"/> ...
2	Yellow
4	Yellow
6	Yellow
11	Green	Scratches
15	Black	Dead

Figure A.49 – Tabular presentation of victim information

Logistic summaries of the information in the victims category may also be presented in a tabular form, as shown in Figure A.50.

Location	Number of red patients	Number of yellow patients	Number of green patients	SUM
Scene of Incident	2	6	5	13
Transportation	2	1	0	3
Hospital	1	1	0	2
SUM	5	8	5	18

Figure A.50 – Logistics summary of victim information

As both the FLUIDE-A and FLUIDE-D specifications are similar to the corresponding specifications for the Resources category, there are few or no comments and explanations to the specifications.

A.1.4.1 FLUIDE-A specifications of the Victims category

Figure A.51 to Figure A.53 show how these user interfaces may be specified in FLUIDE-A.

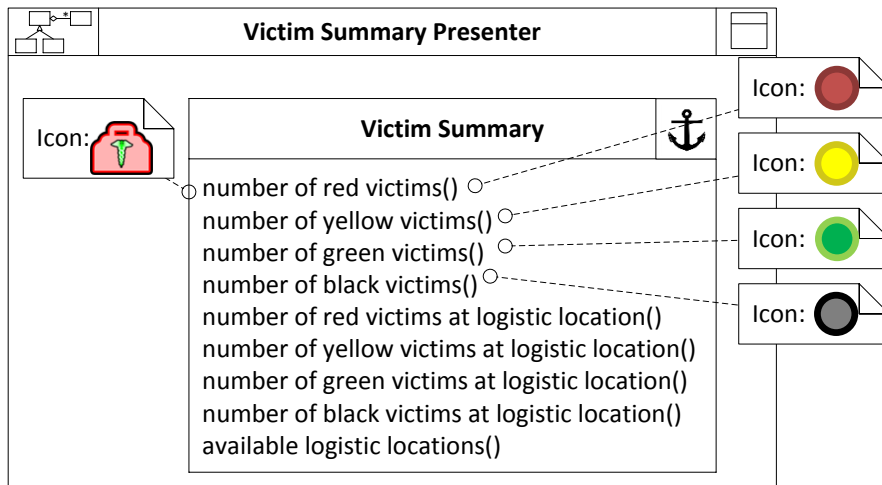


Figure A.51 - FLUIDE-A specification of the overview information shown at the left in Figure A.48

The FLUIDE-A specification in Figure A.51 also applies to the logistics summary presentation shown in Figure A.50.

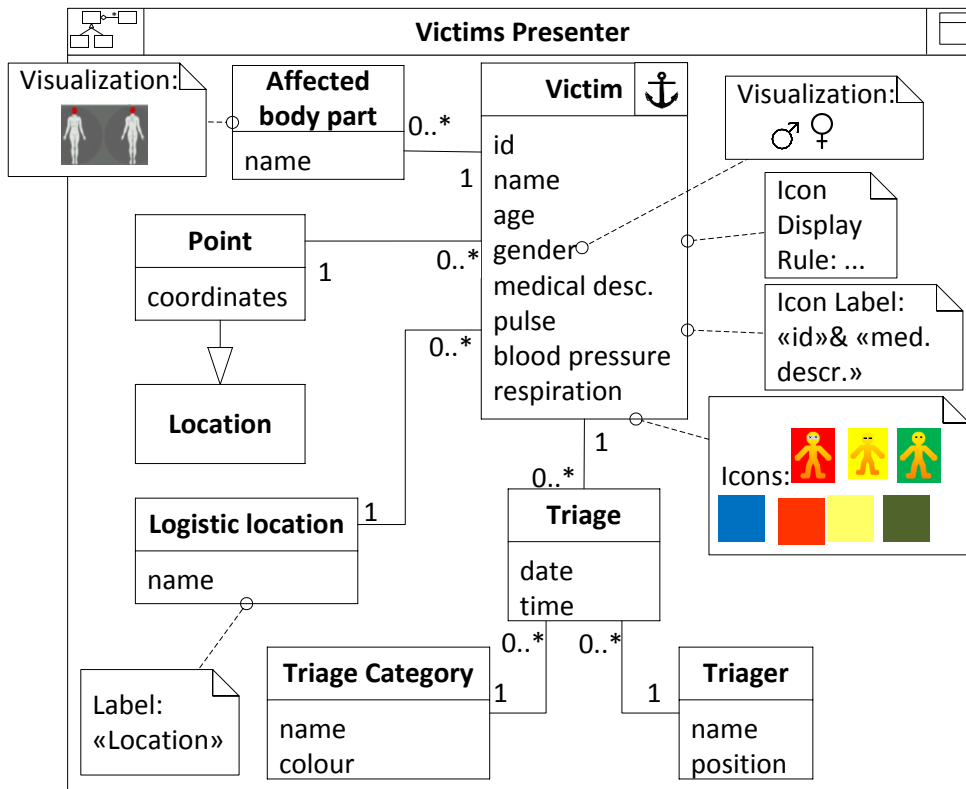


Figure A.52 - FLUIDE-A specification of all the Victims sub categories

The FLUIDE-A specification in Figure A.52 also applies to the tabular presentation shown in Figure A.49 and the map overlay for this category. The sub categories are represented implicitly through the *Triage*

Category entity in the connected concept model, which act as categorizer for the *Victim* entity. As opposed to the similar presenter in the resources category, the *Victim* entity does not have any sub types.

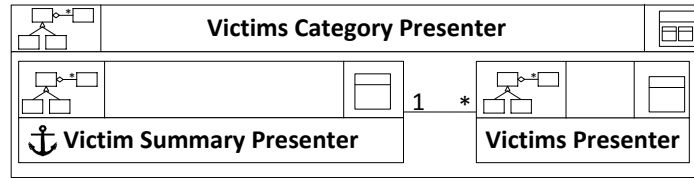


Figure A.53 - FLUIDE-A specification of the Victims Category (i.e. the whole user interface in Figure A.48)

A.1.4.2 FLUIDE-D specifications of the Victims category – Ribbon content

Figure A.54 to Figure A.56 show how the user interface for the ribbon content for the victims category (Figure A.48) may be specified in FLUIDE-D. These specifications only apply to the ribbon part of the user interface.

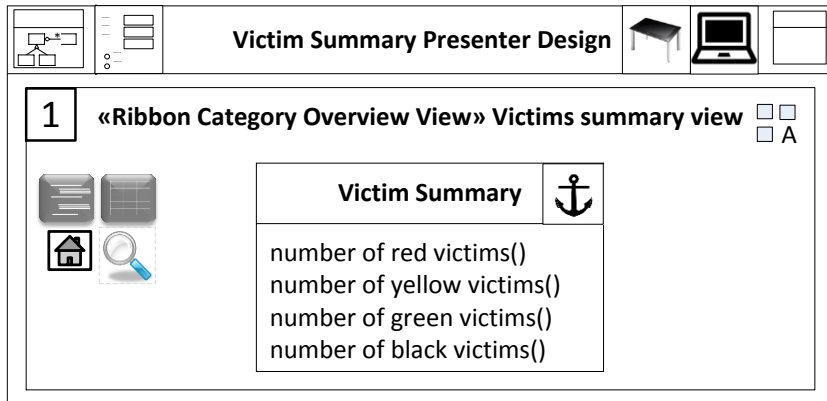


Figure A.54 - FLUIDE-D specification of the overview information shown at the left in Figure A.48 (corresponding FLUIDE-A specification is shown in Figure A.51)

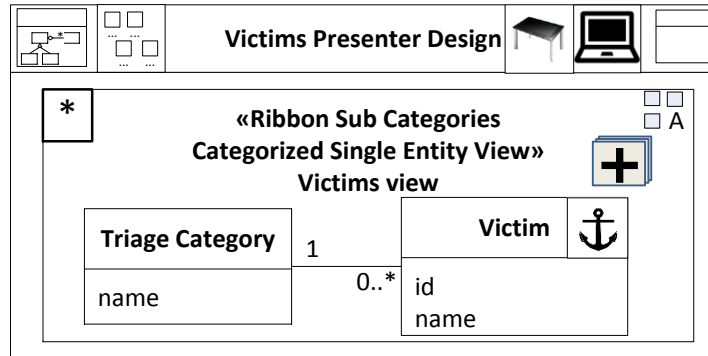


Figure A.55 - FLUIDE-D specification of all the Victims sub categories (corresponding FLUIDE-A specification is shown in Figure A.52)

The presenter design in Figure A.55 contains one Ribbon Sub Categories Categorized Single Entity View. This is the third content view type that may be member of a Ribbon Category View. It is a domain-specific content view type for presenting a number of sub category of a ribbon category based on a categorization entity. The model pattern must contain the entity to be presented (*Victim*), and a categorizing entity (*Triage Category*). It presents a number of instances (documented using the “*” in the top left of the view) of the presented entity as a set of grids or tables of icons. The number of grids is determined by the number of instances of the categorizing entity. These instances also determine the labels for each sub type.

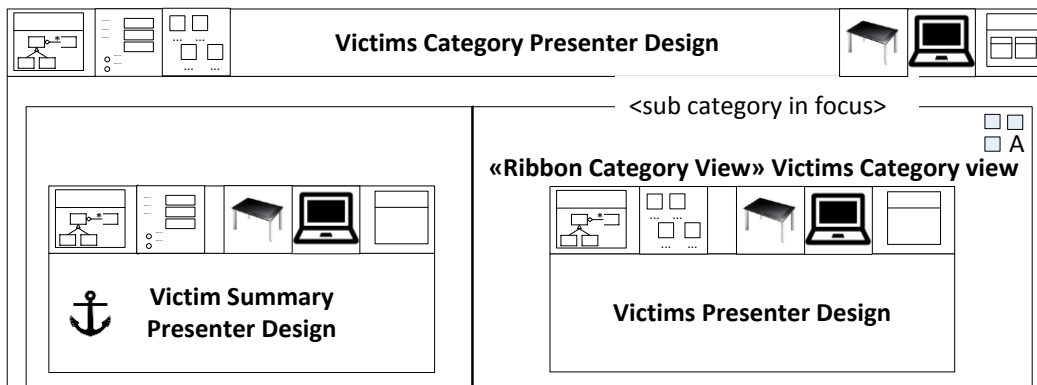


Figure A.56 - FLUIDE-D specification of the Victims Category (i.e. the whole user interface in Figure A.48) (corresponding FLUIDE-A specification is shown in Figure A.53)

The content integration view used in the presenter design in Figure A.56 (Ribbon Category View), is the same as was used for similar presenter designs in the incident, response and resources categories. In the victims category, the presenter design in the right hand slot uses the Ribbon Sub Categories Categorized Single Entity View. There may only be one member presenter in this slot if it uses this view.

A.1.4.3 FLUIDE-D specifications of the Victims category – Tabular presentations

Figure A.57 and Figure A.58 show how the tabular presentations of victims information (Figure A.49 and Figure A.50) may be specified in FLUIDE-D.

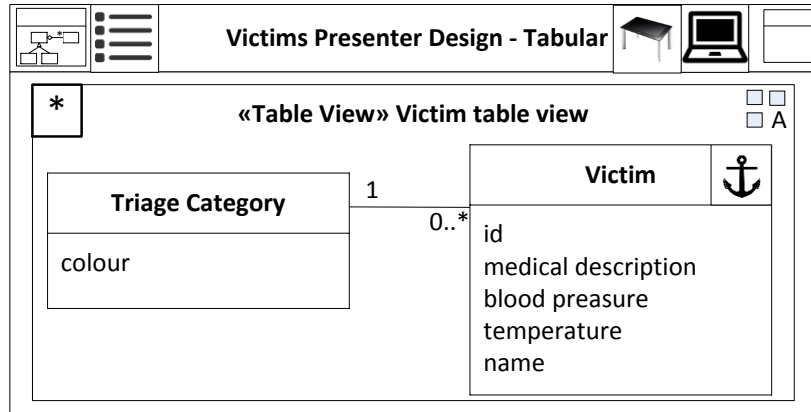


Figure A.57 - FLUIDE-D specification of the Tabular presentation of Victims information shown in Figure A.49 (corresponding FLUIDE-A specification is shown in Figure A.52)

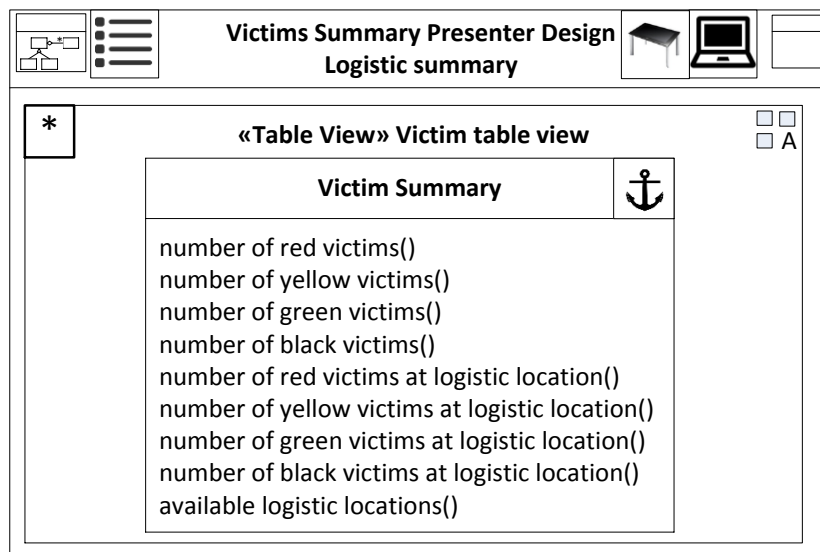


Figure A.58 - FLUIDE-D specification of the Logistics summary of Victims information shown in Figure A.50 (corresponding FLUIDE-A specification is shown in Figure A.51)

A.1.4.4 FLUIDE-D specifications of the Victims category – Map overlays

Figure A.59 and Figure A.60 show how the user interface for map overlays (Figure A.2) for the victims category may be specified in FLUIDE-D.

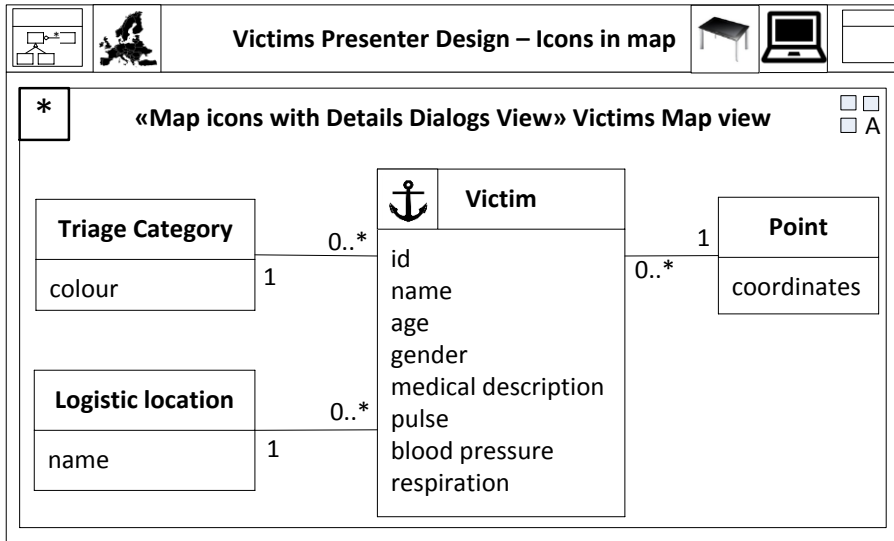


Figure A.59 - FLUIDE-D specification of map overlays for all the Victims sub categories (corresponding FLUIDE-A specification is shown in Figure A.52)

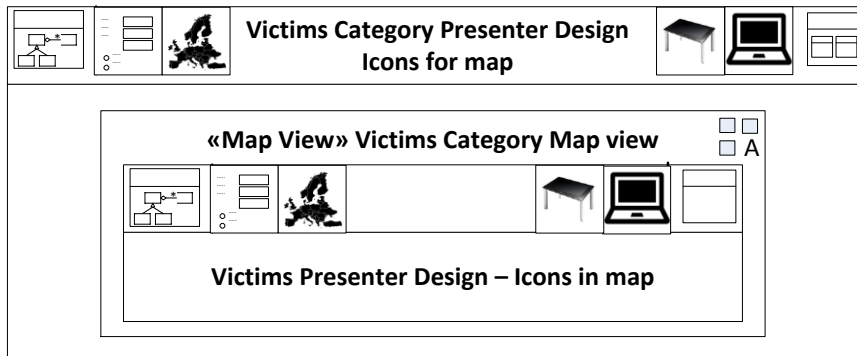


Figure A.60 - FLUIDE-D specification of map overlays for the Victims Category (corresponding FLUIDE-A specification is shown in Figure A.53)

A.1.5 Risks category

The ribbon for the risks category is shown in Figure A.61.

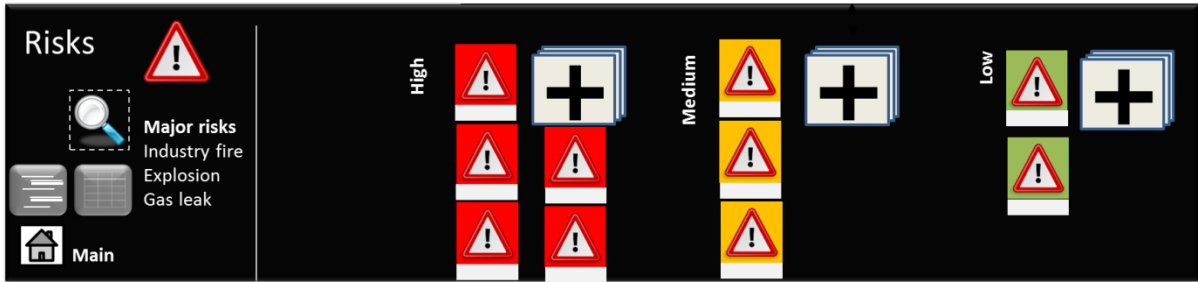


Figure A.61 – Ribbon content for the risks category

As both the FLUIDE-A and FLUIDE-D specifications are similar to the corresponding specifications for the Resources and Victims categories, there are few or no comments and explanations to the specifications.

A.1.5.1 FLUIDE-A specifications of the Risks category

Figure A.62 to Figure A.64 show how this user interface may be specified in FLUIDE-A. The FLUIDE-A specification *Risks Presenter* (Figure A.63) also applies to the map overlay for this category.

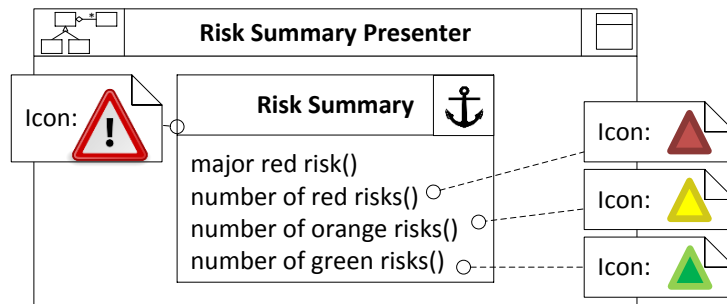


Figure A.62 - FLUIDE-A specification of the overview information shown at the left in Figure A.61

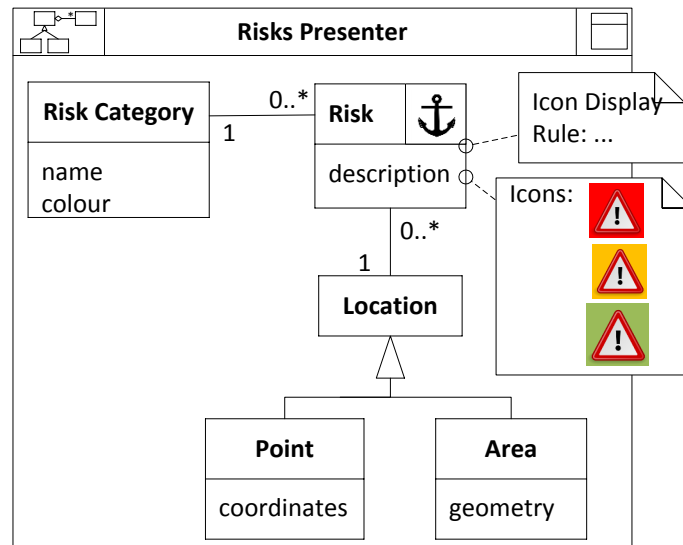


Figure A.63 - FLUIDE-A specification of all the Risks sub categories

The sub categories are represented implicitly through the *Risk Category* entity in the connected concept model, which act as categorizer for the *Risk* entity.

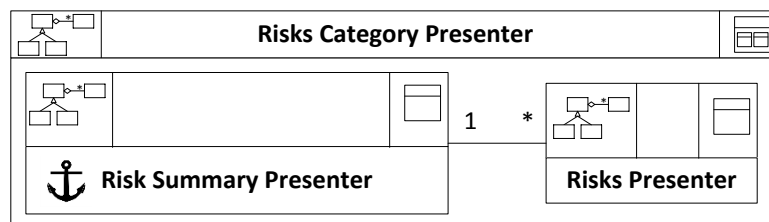


Figure A.64 - FLUIDE-A specification of the Risks Category (i.e. the whole user interface in Figure A.61)

A.1.5.2 FLUIDE-D specifications of the Risks category – Ribbon content

Figure A.65 to Figure A.67 show how the user interface for the ribbon content for the risks category (Figure A.61) may be specified in FLUIDE-D. These specifications only apply to the ribbon part of the user interface.

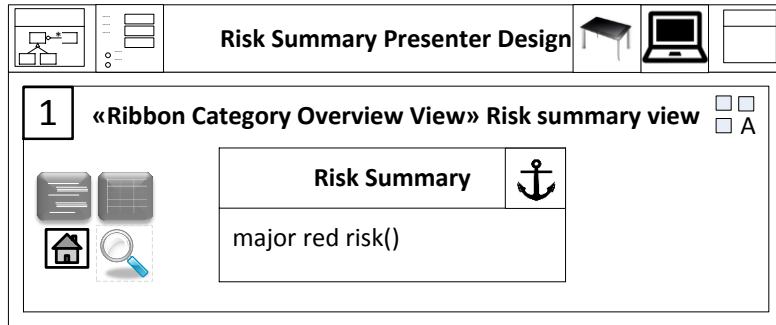


Figure A.65 - FLUIDE-D specification of the overview information shown at the left in Figure A.61 (corresponding FLUIDE-A specification is shown in Figure A.62)

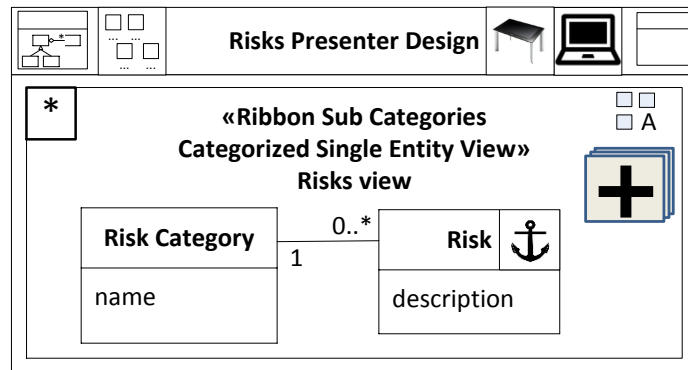


Figure A.66 - FLUIDE-D specification of all the Risks sub categories (corresponding FLUIDE-A specification is shown in Figure A.63)

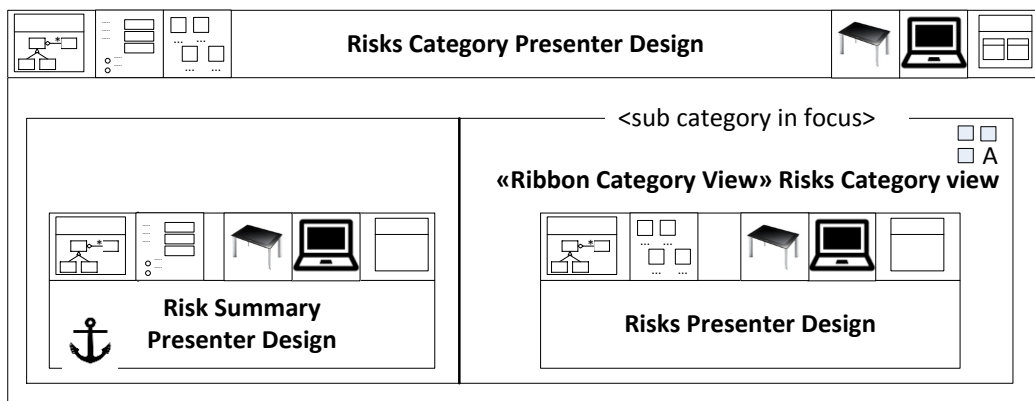


Figure A.67 - FLUIDE-D specification of the Risks Category (i.e. the whole user interface in Figure A.61) (corresponding FLUIDE-A specification is shown in Figure A.64)

A.1.5.3 FLUIDE-D specifications of the Risks category – Map overlays

Figure A.68 and Figure A.69 show how the user interface for map overlays (Figure A.2) for the risks category may be specified in FLUIDE-D.

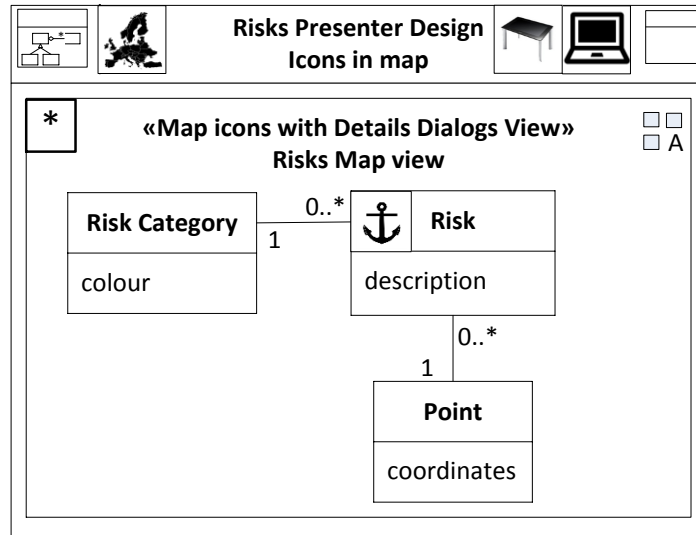


Figure A.68 - FLUIDE-D specification of map overlays for all the Risks sub categories (corresponding FLUIDE-A specification is shown in Figure A.63)

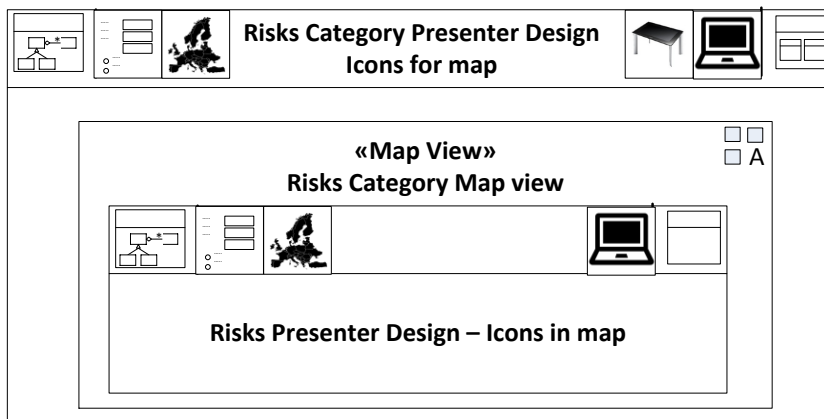


Figure A.69 - FLUIDE-D specification of map overlays for the Risks Category (corresponding FLUIDE-A specification is shown in Figure A.64)

A.1.6 Top level of ribbon (ribbon buttons and ticker)

Above, the user interfaces for each ribbon category have been specified. In this section, we look at the top level of the ribbon, i.e. the user interface in Figure A.3. This user interface consists of a set of buttons and a ticker. We present the specifications for these parts separately. As there are only one new FLUIDE-A specification for each of these parts, we present the FLUIDE-A and FLUIDE-D specifications together.

A.1.6.1 FLUIDE specifications the ribbon buttons

The FLUIDE-A specifications presented above may be used as basis for specifying almost all part of the ribbon buttons in FLUIDE-D. One additional FLUIDE-A specification is needed, i.e. a specification of which information that makes up the ribbon buttons. The Task Supporter presented in Figure A.70 collects the needed presenters for the ribbon buttons.

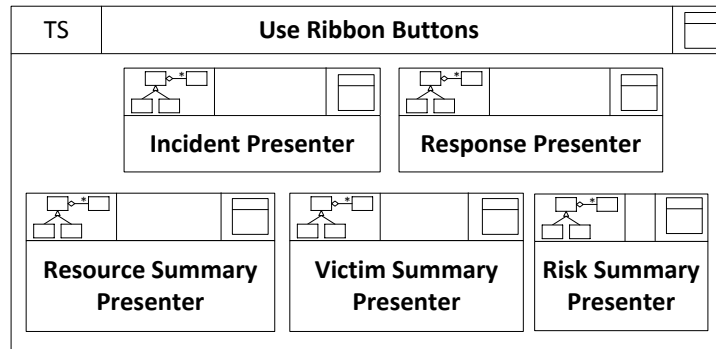


Figure A.70 - FLUIDE-A specification of the top level set of ribbon buttons

Figure A.71 to Figure A.76 show how the user interface for the button parts of the top level of the ribbon (Figure A.3) may be specified in FLUIDE-D.

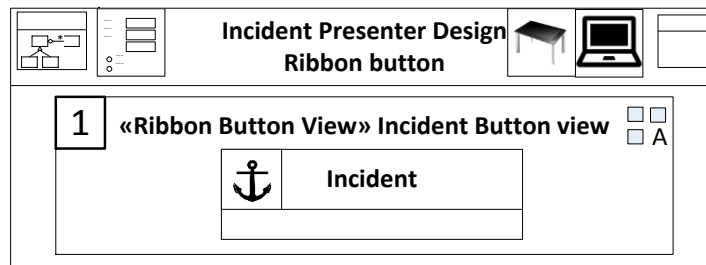


Figure A.71 - FLUIDE-D specification of the ribbon button for accessing the incident category (corresponding FLUIDE-A specification is shown in Figure A.5)

The presenter design in Figure A.71 uses the domain-specific content view Ribbon Button View, which presents one ribbon button and provides the functionality for coupling this button to the ribbon category it should open. The content view requires a single entity as its model pattern.

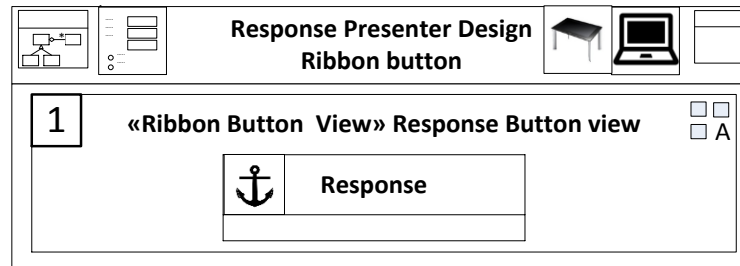


Figure A.72 - FLUIDE-D specification of the ribbon button for accessing the response category (corresponding FLUIDE-A specification is shown in Figure A.23)

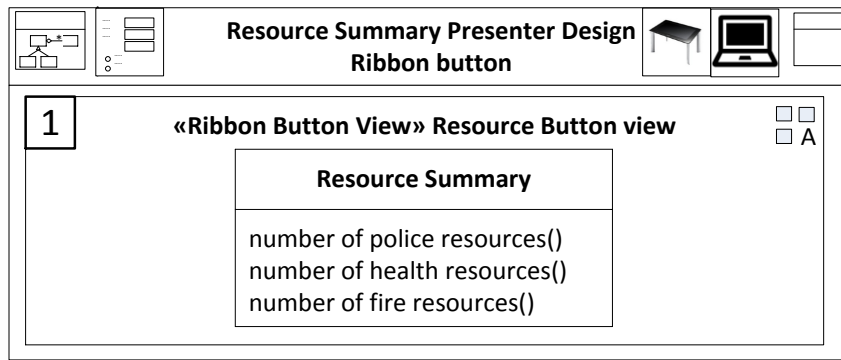


Figure A.73 - FLUIDE-D specification of the ribbon button for accessing the resources category (corresponding FLUIDE-A specification is shown in Figure A.39)

In the presenter design in Figure A.73, the methods from the model fragment are used to provide status information in the button.

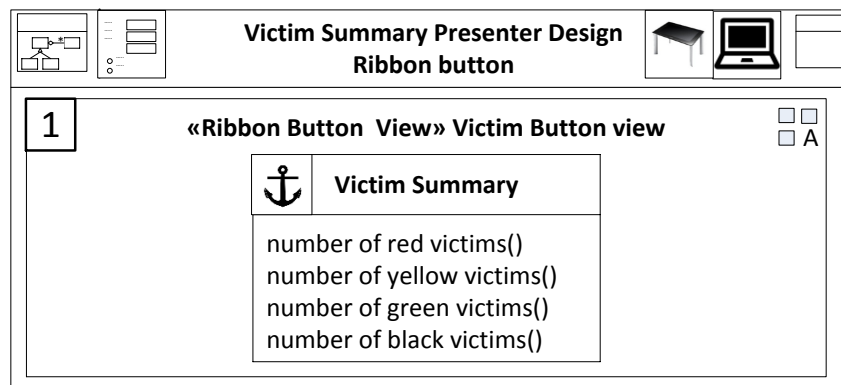


Figure A.74 - FLUIDE-D specification of the ribbon button for accessing the victims category (corresponding FLUIDE-A specification is shown in Figure A.51)

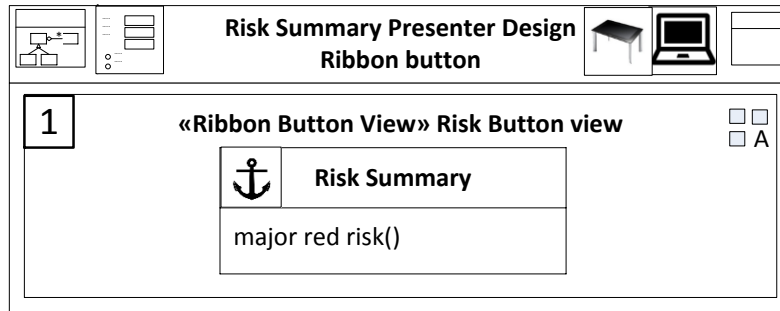


Figure A.75 - FLUIDE-D specification of the ribbon button for accessing the risks category (corresponding FLUIDE-A specification is shown in Figure A.62)

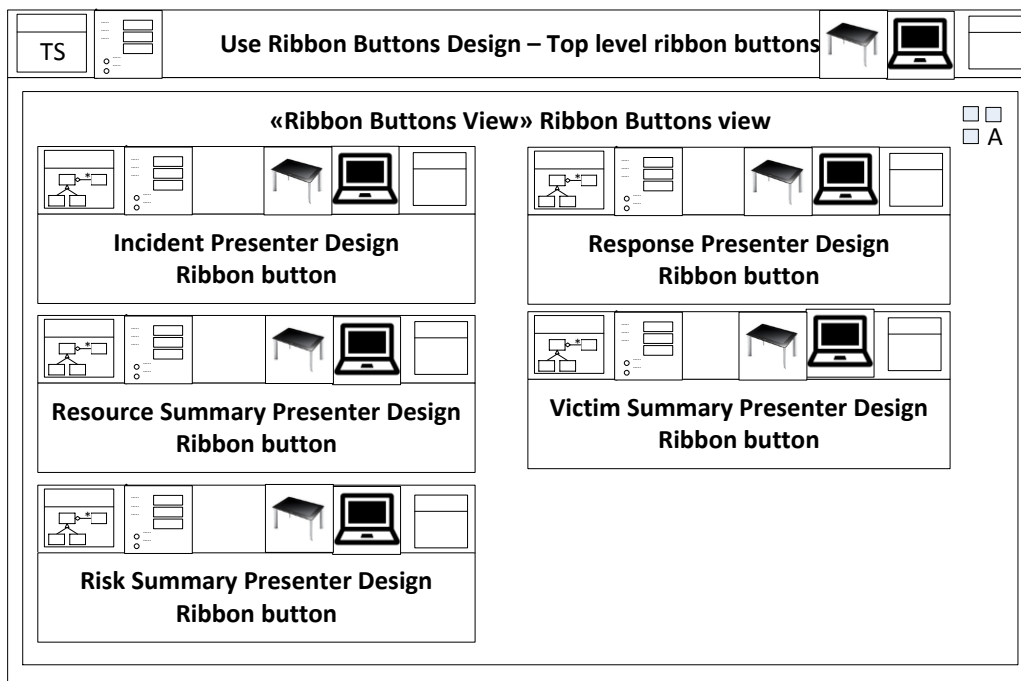


Figure A.76 - FLUIDE-D specification of the collections of ribbon buttons on the top level (corresponding FLUIDE-A specification is shown in Figure A.70)

The Task Supporter Design in Figure A.76 uses the domain-specific content integration view Ribbon Buttons View, which puts together a set of presenter designs containing Ribbon Button Views, making up the button part of the top level of the ribbon.

A.1.6.2 FLUIDE specifications the ribbon ticker

The FLUIDE-A specifications presented above may be used as basis for specifying almost all part of the ribbon ticker. One additional FLUIDE-A specification is needed, i.e. a specification of which information that makes up the ticker. The Task Supporter presented in Figure A.77 collects the needed presenters for the ribbon ticker.

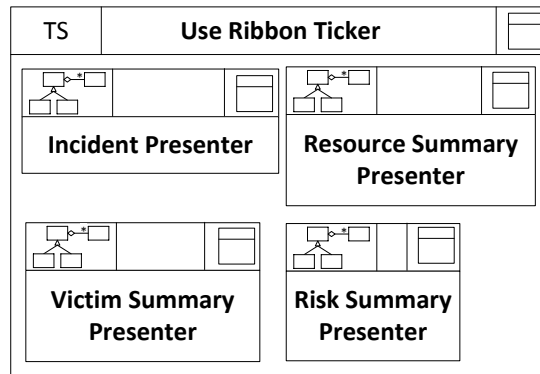


Figure A.77 - FLUIDE-A specification of the ribbon ticker (top part of the top level of the ribbon)

Note that there is no response information in the ticker, thus the presenter for this is not included. Except for this the Task Supporter is identical to the *Use Ribbon Buttons Task Supporter* in Figure A.70.

Figure A.78 to Figure A.82 show how the user interface for the ticker part of the top level of the ribbon (Figure A.3) may be specified in FLUIDE-D.

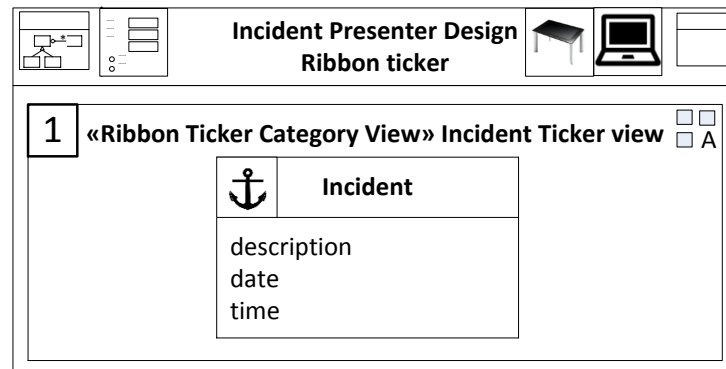


Figure A.78 - FLUIDE-D specification of the ribbon ticker for the incident category (corresponding FLUIDE-A specification is shown in Figure A.5)

The presenter design in Figure A.78 uses the domain-specific content view Ribbon Ticker Category View, which presents a part of a ticker. The information provided in the part of the ticker being specified is determined by the attributes or methods included in the presenter design's model fragment. The content view requires a single entity as its model pattern.

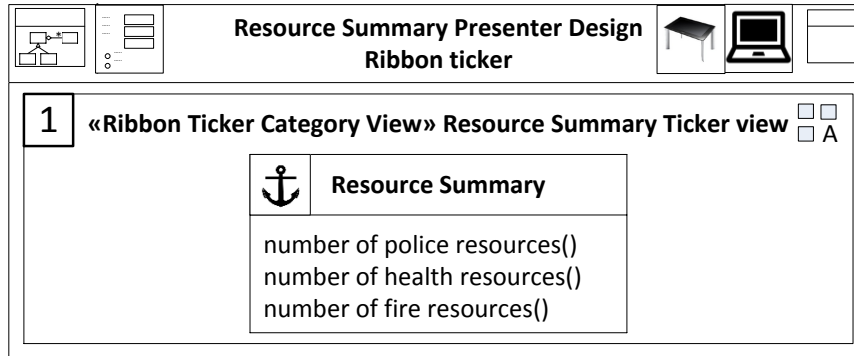


Figure A.79 - FLUIDE-D specification of the ribbon ticker for the resources category (corresponding FLUIDE-A specification is shown in Figure A.39)

In the presenter design in Figure A.79, the methods from the model fragment are used to provide the information to show in this part of the ticker.

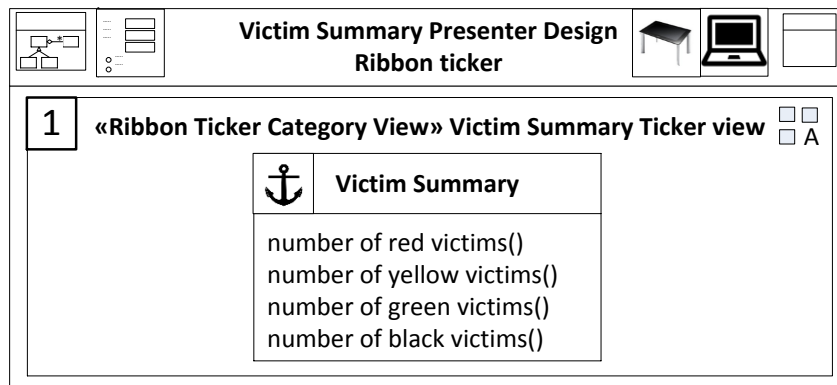


Figure A.80 - FLUIDE-D specification of the ribbon ticker for the victims category (corresponding FLUIDE-A specification is shown in Figure A.51)

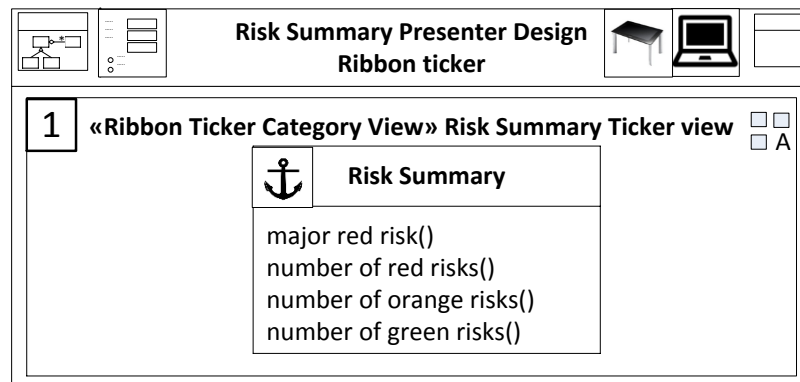


Figure A.81 - FLUIDE-D specification of the ribbon ticker for the risks category (corresponding FLUIDE-A specification is shown in Figure A.62)

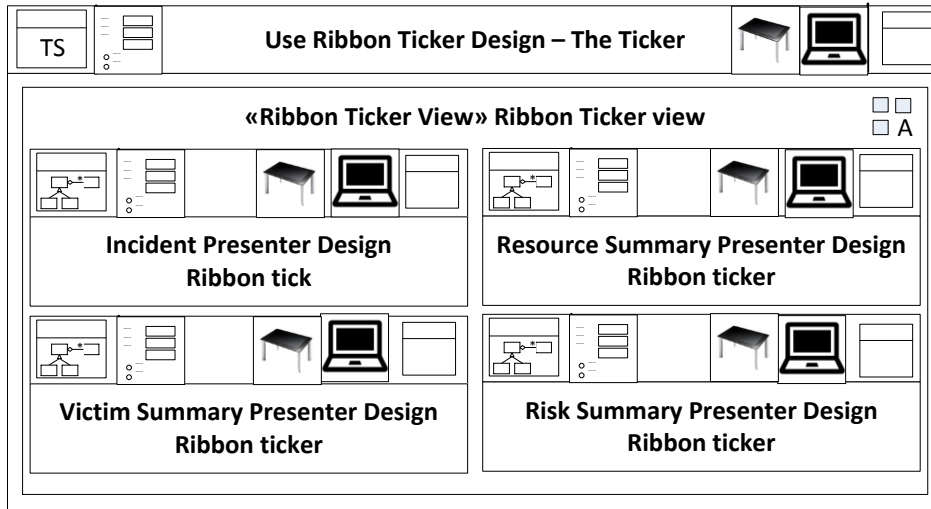


Figure A.82 - FLUIDE-D specification of the whole ribbon ticker (corresponding FLUIDE-A specification is shown in Figure A.77)

The Task Supporter Design in Figure A.82 uses the domain-specific content integration view Ribbon Ticker View, which puts together a set of presenter designs containing Ribbon Ticker Category Views, making up the ticker part of the top level of the ribbon.

A.1.7 Coupling the different parts of the ribbon

Above, the user interfaces for each ribbon category, as well as the top level of the ribbon have been specified. In this section, we look at the how all parts of the ribbon may be put together to provide the intended ribbon functionality.

A.1.7.1 FLUIDE-A specifications for coupling the different parts of the ribbon

To couple the different parts of the ribbon, four new FLUIDE-A specifications are needed, using the language constructs Task Supporter, Basic Work Supporter and Aggregated Work Supporter.

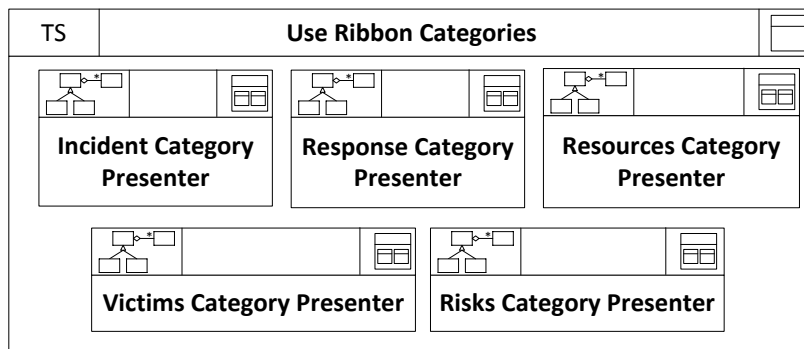


Figure A.83 - FLUIDE-A specification of a Task Supporter connecting the five ribbon categories

Earlier, five ribbon categories have been specified, both in FLUIDE-A and FLUIDE-D. To couple these to the top level part of the ribbon, they must be put together in the Task Supporter in Figure A.83.

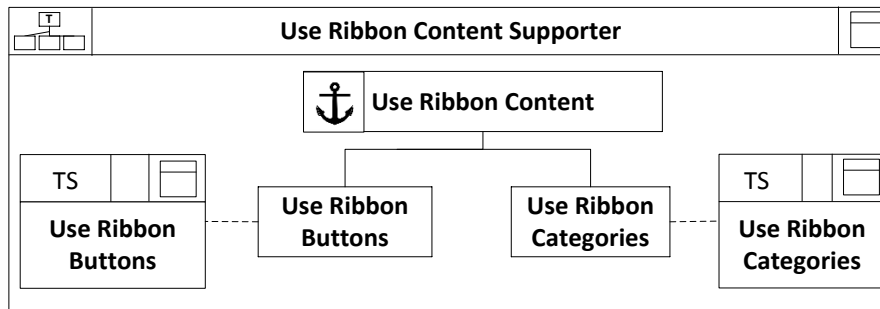


Figure A.84 - FLUIDE-A specification of a Basic Work Supporter coupling the ribbon categories with the top level set of ribbon buttons

The FLUIDE-A specification in Figure A.84 uses the language construct Work Supporter. A Work Supporter is similar to a Content Presenter, but has an associated task model instead of a concept model. This is indicated by the icon at the top left corner. The top right icon indicates that the supporter is basic. The task model is expressed in neutral hierarchical task modelling notation, with the addition that each task may have connected Task Supporter. In a Work Supporter, the anchor is always the root task. The supporter in Figure A.84 couples the ribbon categories (the Task Supporter just specified) with the button part of the top level part of the ribbon.

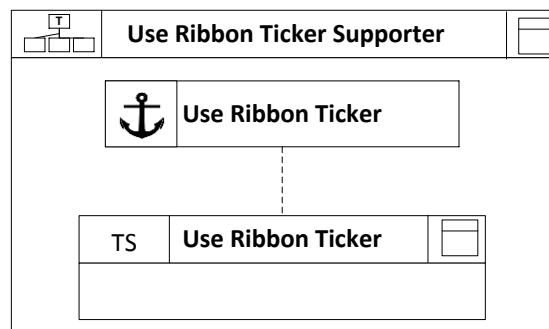


Figure A.85 - FLUIDE-A specification of a Basic Work Supporter for the Use Ribbon Ticker task

The supporter in Figure A.85 wraps the Task Supporter Use Ribbon Ticker in a Basic Work Supporter. This wrapping is needed in order to couple the ribbon contents (the top level buttons and the ribbon categories specified above) with ribbon ticker in the Aggregated Work Supporter presented in Figure A.86.

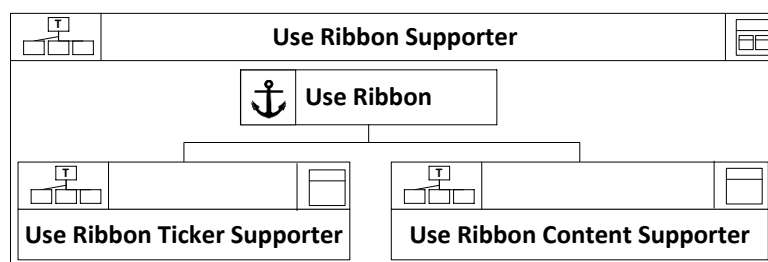


Figure A.86 - FLUIDE-A specification of an Aggregated Work Supporter for the Use Ribbon task

The FLUIDE-A specification in Figure A.86 uses the aggregated variant of the Work Supporter language construct (indicated by the icon in the top right corner). Such supporters may only aggregate other Work Supporters (basic or aggregated), and must add exactly one task anchor (with or without a Task Supporter) on the level above the aggregated ones. The supporter in Figure A.86 couples the ribbon contents (i.e. the top level buttons coupled with the five ribbon categories) with the ribbon ticker. In this way, this supporter specifies the whole ribbon in FLUIDE-A.

A.1.7.2 FLUIDE-D specifications for coupling the different parts of the ribbon

Figure A.87 to Figure A.90 show how the coupling of the different parts of the ribbons may be specified in FLUIDE-D, i.e. the designs for the four FLUIDE-A specifications presented in the previous section. This ends up with a specification of the complete ribbon.

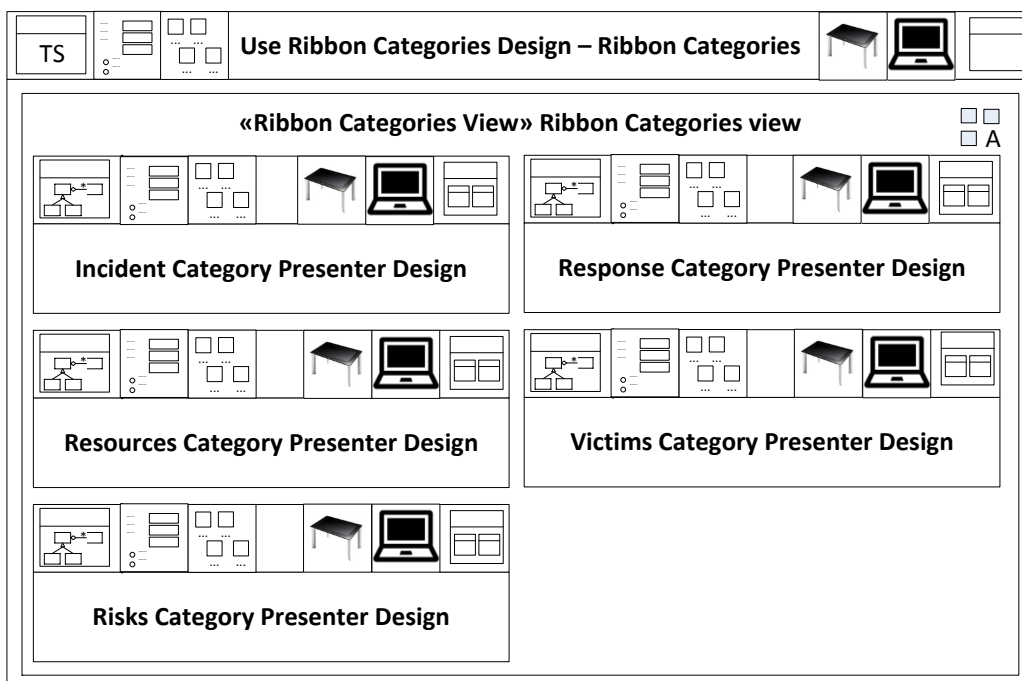


Figure A.87 - FLUIDE-D specification of connection of the five ribbon categories (corresponding FLUIDE-A specification is shown in Figure A.83)

The Task Supporter Design in Figure A.87 use the domain-specific content integration view Ribbon Categories View, which puts together a set of designs containing Ribbon Category Views. This coupling is functional rather than visual.

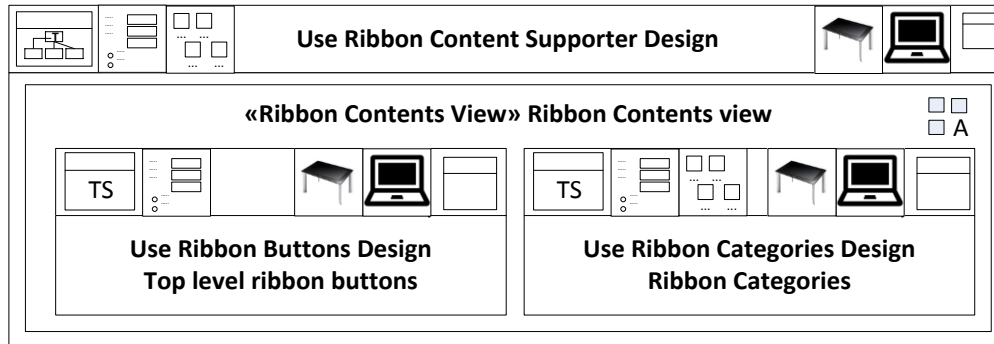


Figure A.88 - FLUIDE-D specification of coupling of the five ribbon categories with the ribbon buttons (corresponding FLUIDE-A specification is shown in Figure A.84)

The Work Supporter Design in Figure A.88 use the domain-specific content integration view Ribbon Contents View, which puts together one design containing a Ribbon Categories View with another design containing a Ribbon Buttons View. Also this coupling is functional rather than visual, and provides the functionality navigating between the single ribbon buttons the corresponding ribbon category and up again.

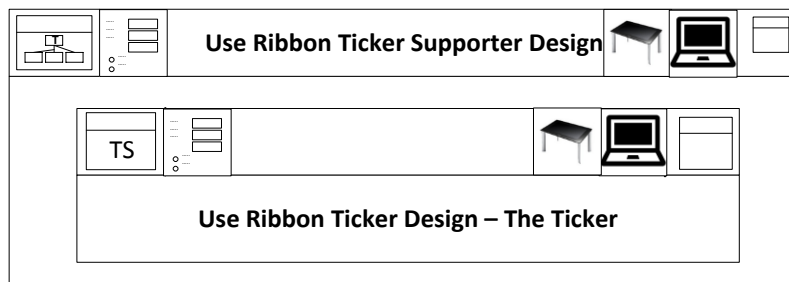


Figure A.89 - FLUIDE-D specification design for the Use Ribbon Ticker Supporter (corresponding FLUIDE-A specification is shown in Figure A.85)

In the Work Supporter Design in Figure A.89 there is no need for a content or content integration view, as it is only used to indicate which supporter design that should be used for the corresponding supporter.

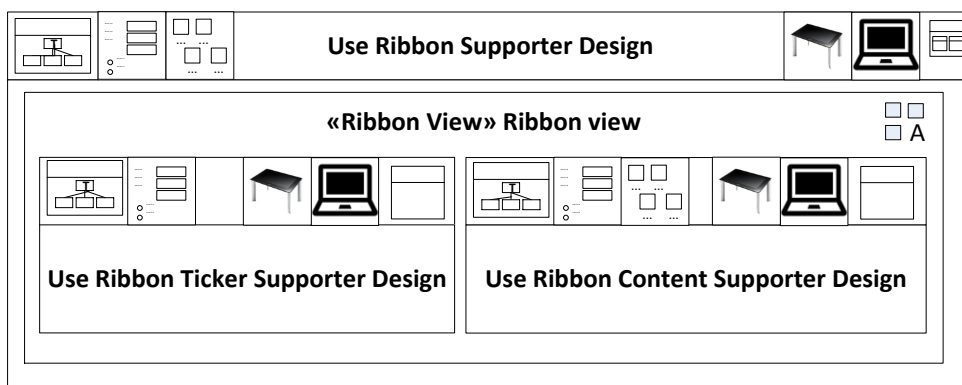


Figure A.90 - FLUIDE-D specification of the complete ribbon (corresponding FLUIDE-A specification is shown in Figure A.86)

The Aggregated Work Supporter Design in Figure A.90 use the domain-specific content integration view Ribbon View, which puts together one design containing a Ribbon Contents View with another design containing a Ribbon Ticker View, together making up a complete ribbon.

A.1.8 Coupling the map overlays

Above, the user interfaces for map overlays for the information in each ribbon category have been specified. In this section, we look at how these overlays are coupled to be presented together in a map (Figure A.2).

To utilize this, only one additional FLUIDE-A specification with a connected FLUIDE-D specification is needed.

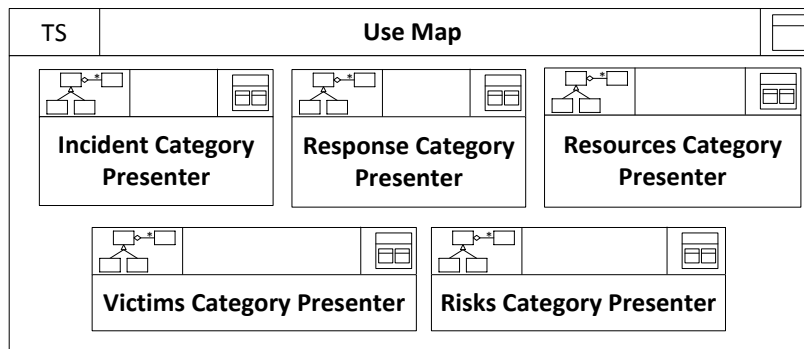


Figure A.91 - FLUIDE-A specification of the complete information to be presented in the common map (Figure A.2)

The Task Supporter presented in Figure A.91 collects the needed presenters for the sum of icons and polygons to be presented in the common map.

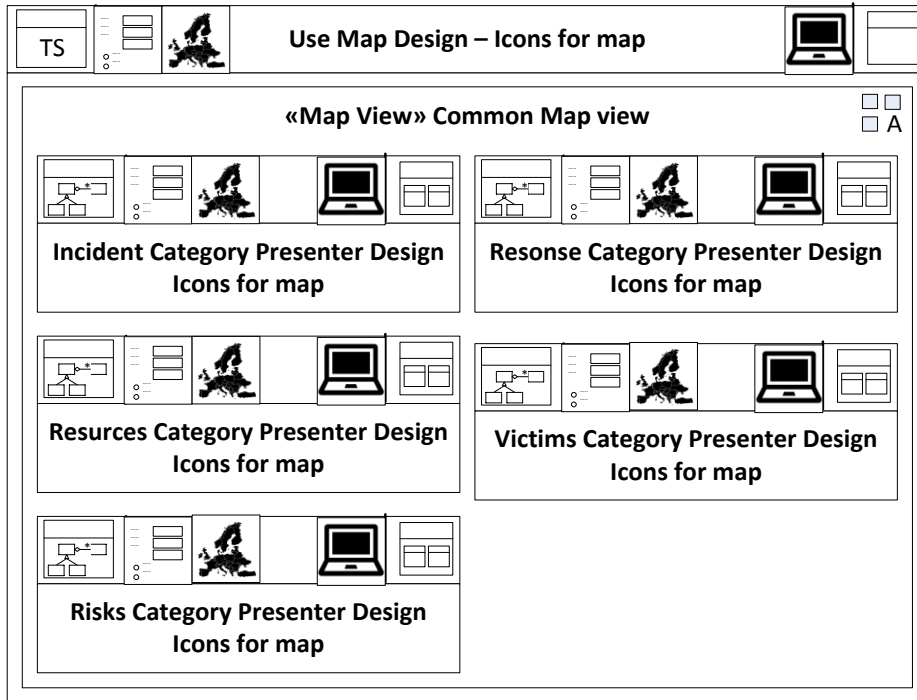


Figure A.92 - FLUIDE-D specification of the complete visualization in the common map (corresponding FLUIDE-A specification is shown in Figure A.91)

The Task Supporter Design in Figure A.92 uses the same content integration view as its member designs, i.e. Map View (which may be nested in any number of levels).

A.1.9 Putting all parts of the MASTER together

Above, the user interfaces for all parts of the MASTER application have been specified. In this section, the final integration of these parts is specified.

A.1.9.1 FLUIDE-A specifications for putting all parts of the MASTER together

To couple the map and ribbon parts of the master, two new FLUIDE-A specifications are needed.

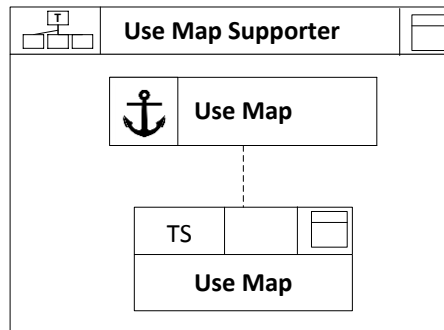


Figure A.93 - FLUIDE-A specification of a Basic Work Supporter for the *Use Map* task

The Work Supporter in Figure A.93 wraps the Task Supporter *Use Map* in a Basic Work Supporter. This wrapping is needed in order to couple the map with the ribbon in the Category Manager presented in Figure A.94.

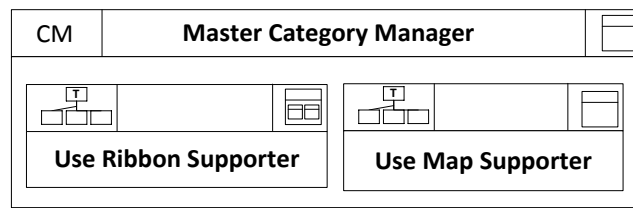


Figure A.94 - FLUIDE-A specification of a Category Manager for the Master

The FLUIDE-A specification in Figure A.94 is Category Manager (indicated by the “CM” icon at the top left) that couples the ribbon with the map (one being a Basic Work Supporter, the other being an Aggregated one). In this way, this Category Manager specifies the whole MASTER application in FLUIDE-A.

A.1.9.2 FLUIDE-D specifications for putting all parts of the MASTER together

Figure A.95 and Figure A.96 show how the coupling of the map and ribbon parts may be specified in FLUIDE-D, i.e. the designs for the two FLUIDE-A specifications presented in the previous section. This ends up with a specification of the complete MASTER user interface.

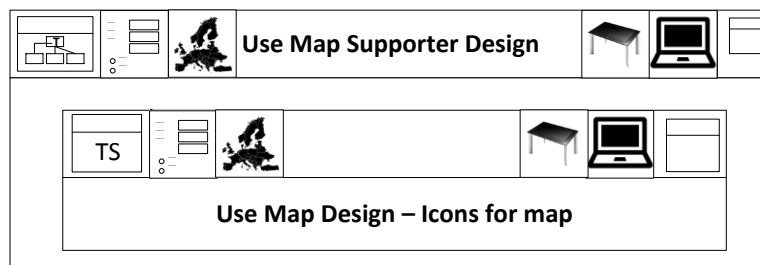


Figure A.95 - FLUIDE-D specification of the design for the *Use Map Supporter* (corresponding FLUIDE-A specification is shown in Figure A.93)

In the Work Supporter Design in Figure A.95, there is no need for a content or content integration view, as it is only used to indicate which supporter design that should be used for the corresponding supporter.

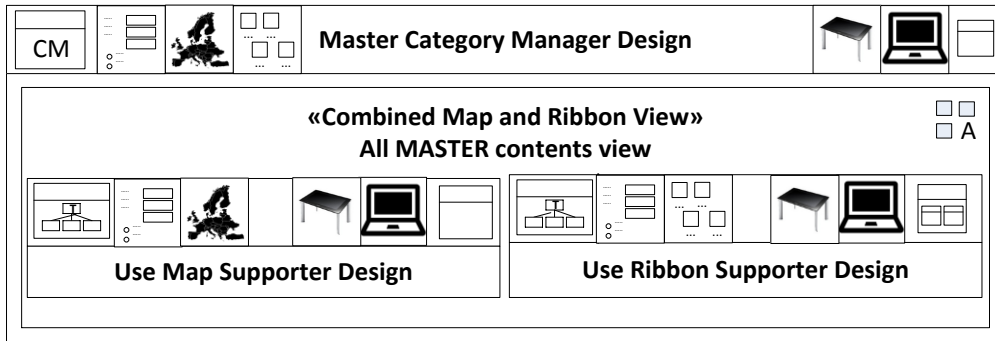


Figure A.96 - FLUIDE-D specification of the MASTER user interface (corresponding FLUIDE-A specification is shown in Figure A.94)

The Category Manager Design in Figure A.96 uses the domain-specific content integration view Combined Map and Ribbon View, which puts together one design containing a Map view with another design containing a Ribbon View, together making up a complete user interface for the MASTER application with the map and the ribbon working together.

A.2 The Resource Manager Application

The Resource Manager is a smartphone application supporting personnel in the field by managing locations as well as receiving and responding to task allocations. Below we present these two main parts of the application and the corresponding FLUIDE-A and FLUIDE-D specifications, as well as the coupling of the two.

A.2.1 Location tracking and overview of locations

The main user interface of the Resource Manager application (Figure A.97) is used for location tracking and overview of location.

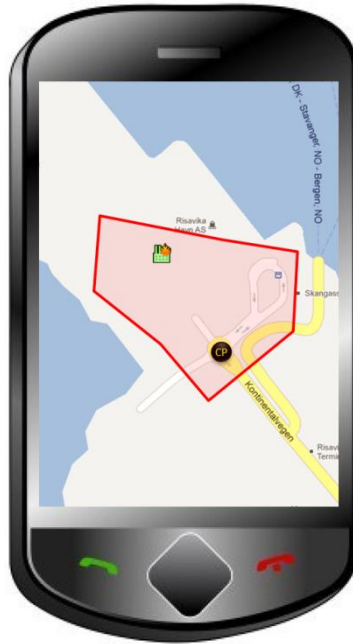


Figure A.97 – Main user interface of the Resource Manager

A.2.1.1 FLUIDE-A specifications of location tracking and overview of locations

Figure A.98 to Figure A.103 show how this user interface may be specified in FLUIDE-A. Some of these FLUIDE-A specifications are also used in the MASTER application, but are included here to enhance readability.

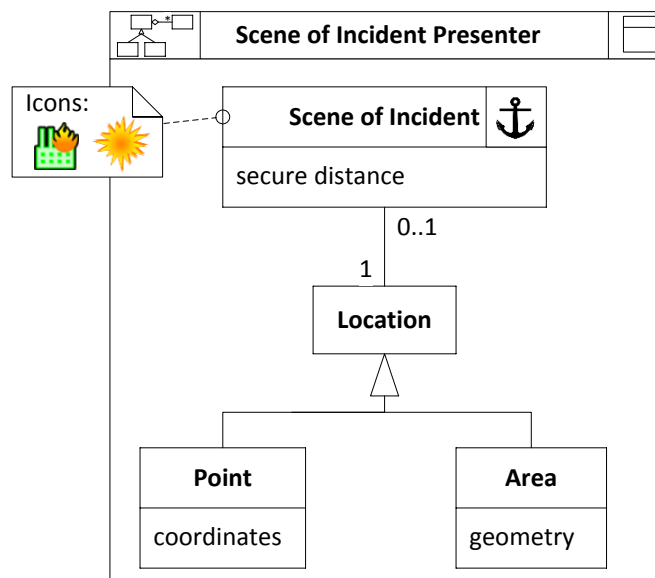


Figure A.98 - FLUIDE-A specification of the Scene of Incident

The Basic Content Presenter in Figure A.98 is needed for showing the incident icon on the map shown in Figure A.97. This specification is identical to the presenter with the same name presented for the Master application (Figure A.6).

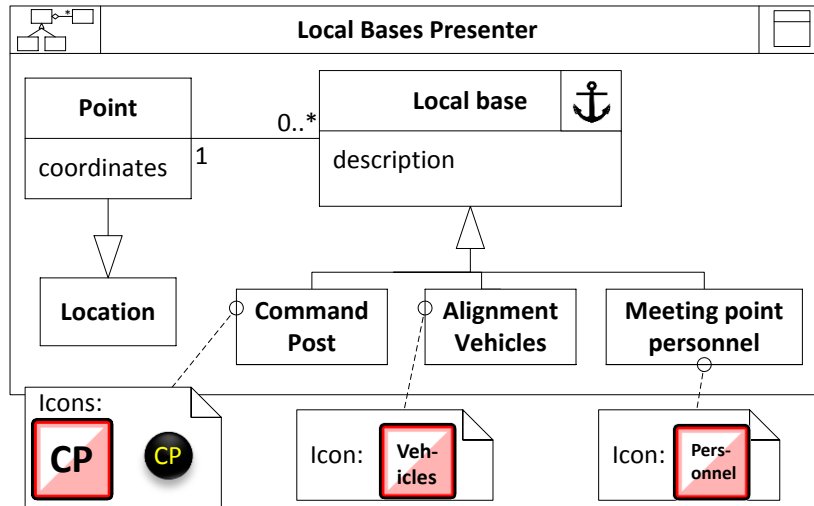


Figure A.99 - FLUIDE-A specification of the Local Bases

The Basic Content Presenter in Figure A.99 is needed for showing the command post icon on the map. The specification is identical to the presenter with the same name presented for the Master application (Figure A.24).

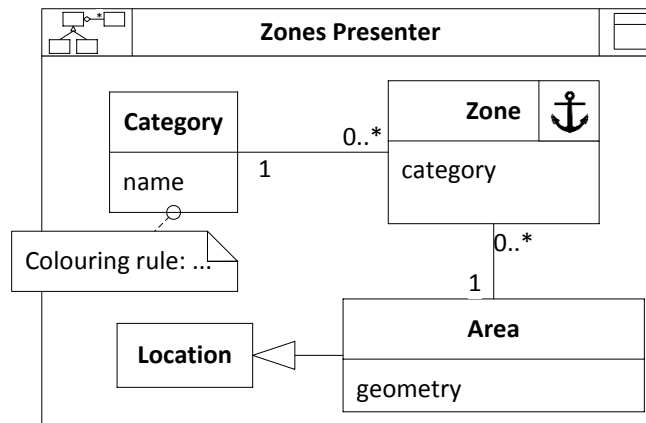


Figure A.100 - FLUIDE-A specification of the Operational Area (zones)

The Basic Content Presenter in Figure A.100 is needed for showing the zone polygon(s) on the map. The specification is identical to the presenter with the same name presented for the Master application (Figure A.25).

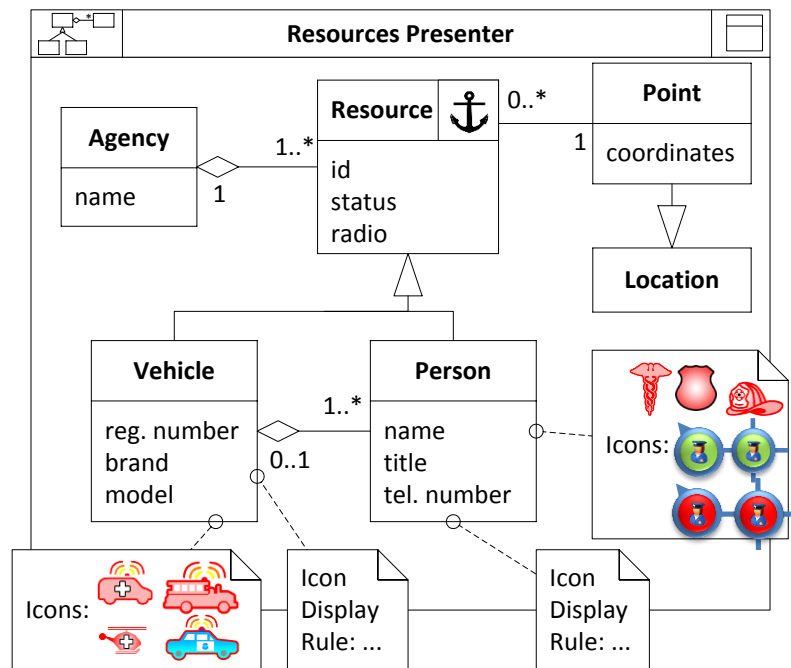


Figure A.101 - FLUIDE-A specification of Resources

The Basic Content Presenter in Figure A.101 is needed for showing resource icons on the map (both representing the user and other resources). The specification is identical to the presenter with the same name presented for the Master application (Figure A.40).

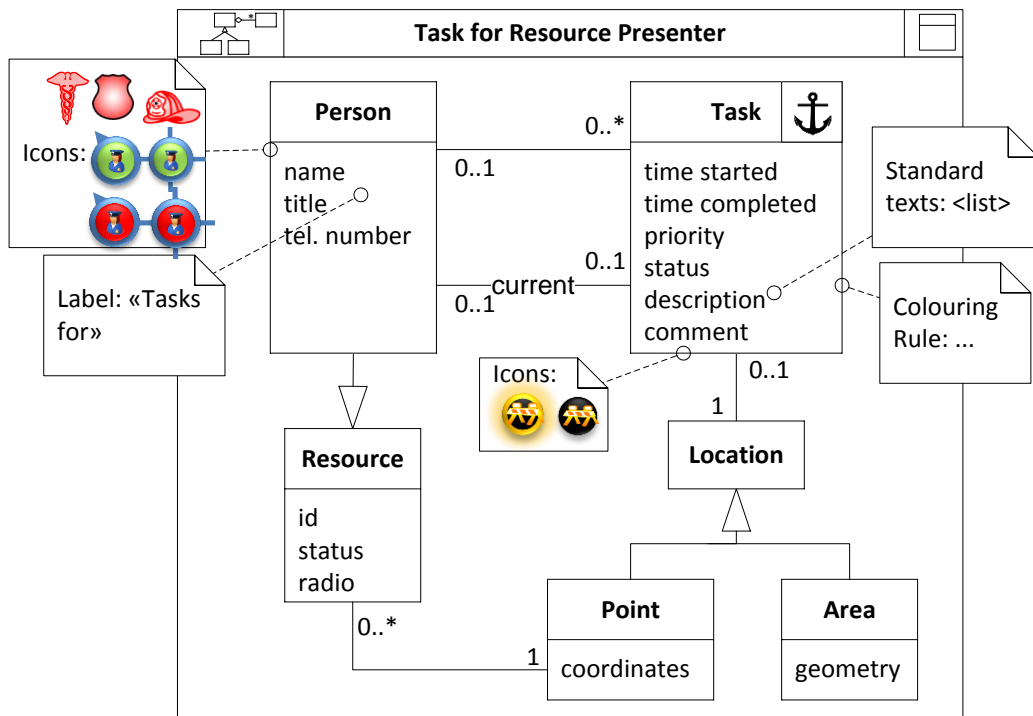


Figure A.102 - FLUIDE-A specification of Tasks for Resource

Although the Basic Content Presenter in Figure A.102 overlaps with the one in Figure A.101, it is needed for showing task icons on the map (the presenter in Figure A.101 is needed for ensuring that unallocated resources are shown). This specification is not used as part of the Master application.

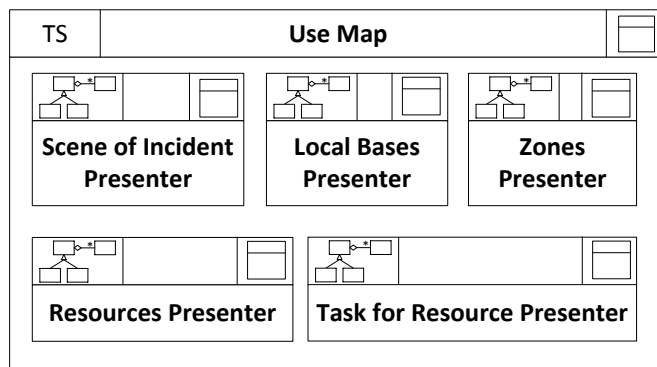


Figure A.103 - FLUIDE-A specification of Use Map Task Supporter

The Task Supporter in Figure A.103 is needed for specifying which information that is shown together on the map. This specification is not used as part of the Master application.

A.2.1.2 FLUIDE-D specifications of location tracking and overview of locations

Figure A.104 to Figure A.109 show how the user interface for location tracking and overview of locations may be specified in FLUIDE-D.

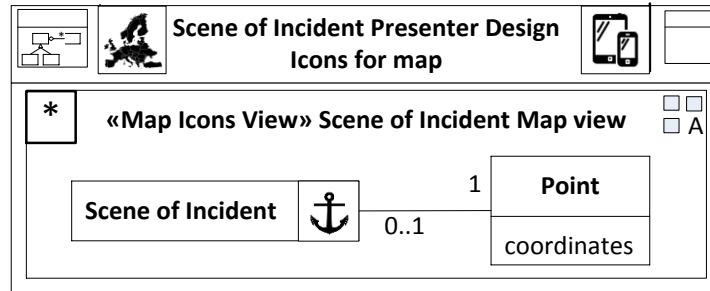


Figure A.104 - FLUIDE-D specification of map overlays for the Scene of Incident (corresponding FLUIDE-A specification is shown in Figure A.98)

Compared to the similar presenter design in the MASTER application (Figure A.17), the variant for Resource Manager (Figure A.104) uses touch based mobile device as modality and/or platform(s) (shown by the icon showing two screens on the top right). Furthermore, as this user interface does not provide pop-up windows connected to the icons, a simpler, domain-specific view is used, i.e. Map Icons View. The Map Icons View provides the same functionality and rules as Map Icons with Details Dialog View, except for forms based presentations of the icons. Thus, it only exploits the map based user interface style.

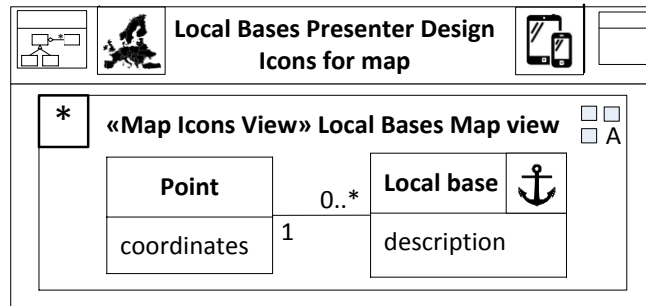


Figure A.105 - FLUIDE-D specification of map overlays for the Local Bases (corresponding FLUIDE-A specification is shown in Figure A.99)

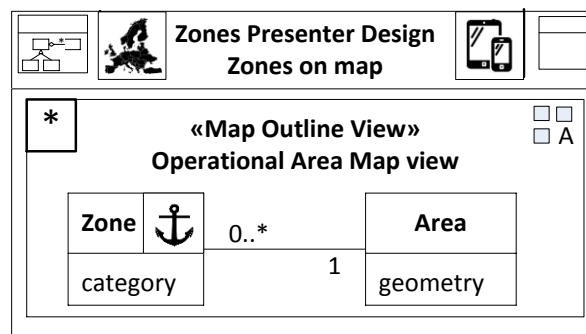


Figure A.106 - FLUIDE-D specification of map overlays for the Operational Area (zones) (corresponding FLUIDE-A specification is shown in Figure A.100)

The presenter design in Figure A.106 uses the same view as the corresponding one in the MASTER application.

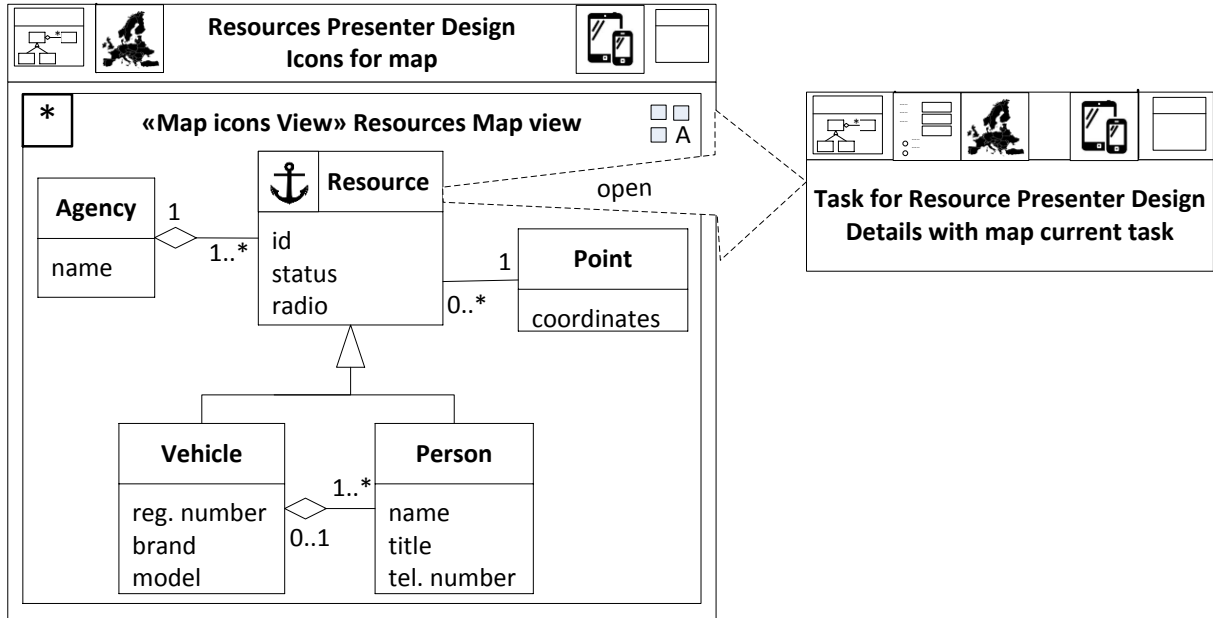


Figure A.107 - FLUIDE-D specification of map overlays for Resources (corresponding FLUIDE-A specification is shown in Figure A.101)

The presenter design in Figure A.107 includes specification of the intended dialog navigation to take place when an icon on the map is tapped. A dialog navigation specification is shown as a dashed-lined arrow with a growing size. The type of dialog navigation (in this case "open") is shown as text on the arrow. Such arrows point from a Content Presenter Design specification to a representation of another interactor design specification (using the same syntax as for members of aggregated interactor designs). The small end indicates which element of the user interface that triggers the dialog navigation, i.e. the user interface representation of the entity or attribute that the small end starts from. The point of the arrow indicates which design or view that is triggered by the dialog navigation (the target). The target of the dialog navigation is a representation of the presenter design in Figure A.119.

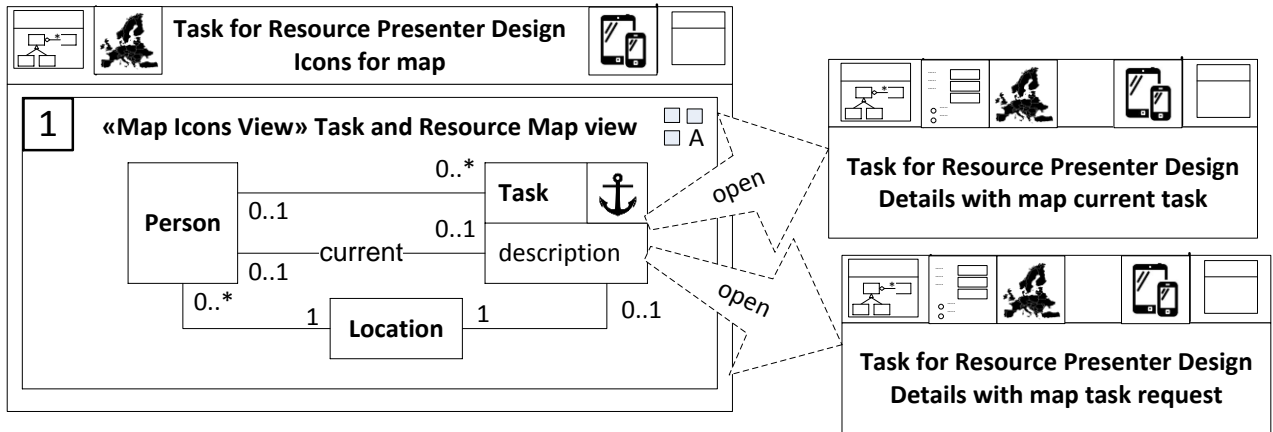


Figure A.108 - FLUIDE-D specification of map overlays for Tasks for Resource (corresponding FLUIDE-A specification is shown in Figure A.102)

Unlike the other map-based presenter designs just specified, the presenter design in Figure A.108 will only show one icon, as a resource may only have one task allocated at a time in the Resource Manager. It also includes specification of dialog navigation to the presenter designs in Figure A.119 and Figure A.120. In the specification in Figure A.108 there are two different targets for the dialog navigation, depending on whether the source is a task request or an allocated task.

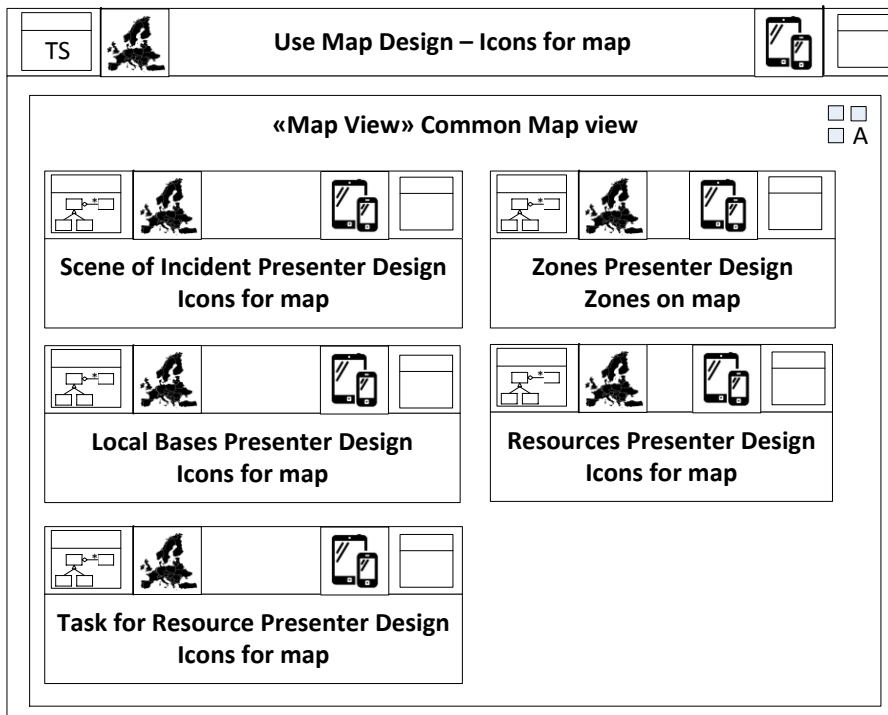


Figure A.109 - FLUIDE-D specification of the complete user interface in Figure A.97 (corresponding FLUIDE-A specification is shown in Figure A.103)

A.2.2 Task overview and allocation

The user interface to get an overview of the current task is shown in Figure A.110.

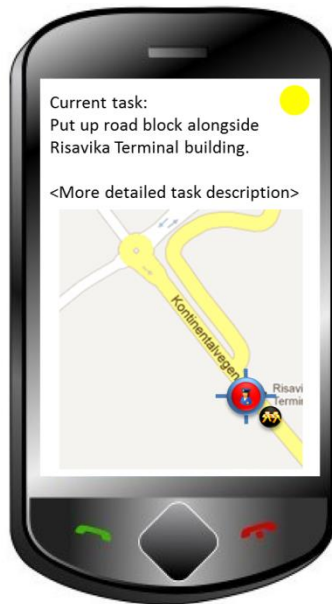


Figure A.110 – User interface for seeing details about the current task

When a task request arrives the user interface in Figure A.111 is displayed.

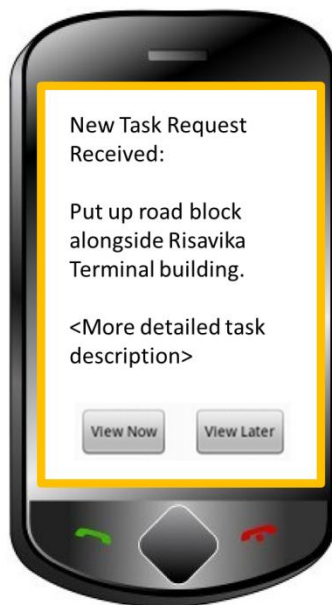


Figure A.111 – Task request user interface

If the *View Now* button is selected the user interface in Figure A.112 is displayed.

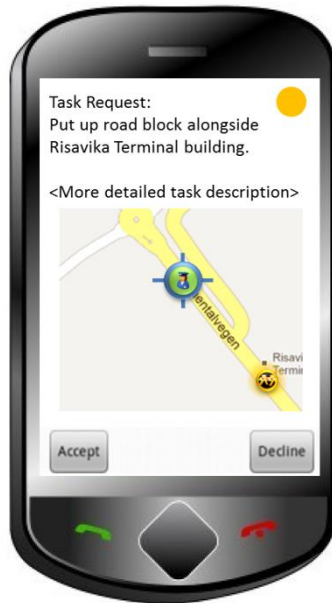


Figure A.112 – User interface for seeing details about the task request received

If the *View Later* button is selected in the task request user interface (Figure A.111) the request can be brought up later using the user interface in Figure A.113.

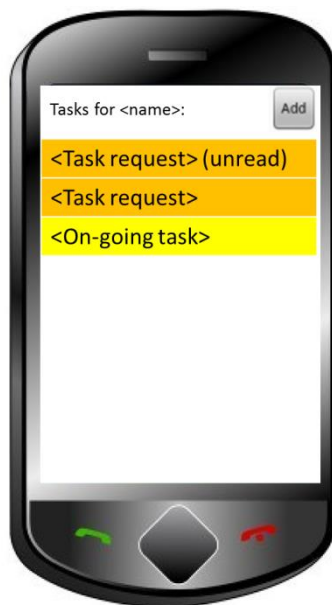


Figure A.113 – User interface for seeing a list of task requests (and current task if any)

If the *Decline* button is selected in the user interface for seeing details about the task request received (Figure A.112) the user interface in Figure A.114 is displayed.



Figure A.114 – User interface for declining a task request

If the *Choose standard decline text* button is selected the user interface in Figure A.115 is displayed.

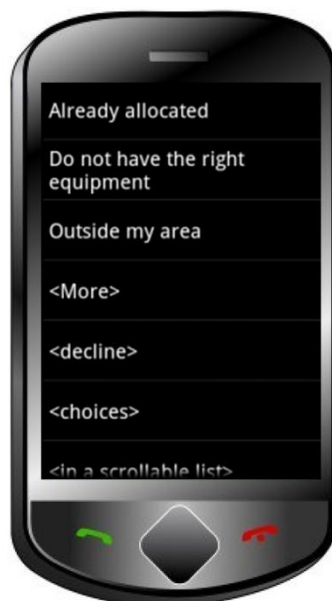


Figure A.115 – User interface for entering stand decline explanations

A.2.2.1 FLUIDE-A specifications of task overview and allocation

All the user interfaces in Figure A.110 to Figure A.115 may be specified using the FLUIDE-A specification in Figure A.102.

To describe how the different user interfaces supporting task overview and allocation support different user tasks, a number of Task Supporters are specified. How these supporters are used in a task model for the whole Resource Manager application is specified below in Figure A.116 - Figure A.118.

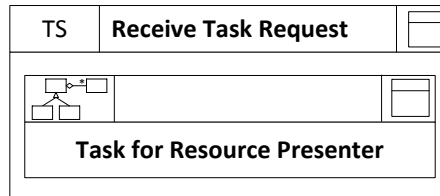


Figure A.116 - FLUIDE-A specification of the Task Supporter for the task *Receive Task Request*

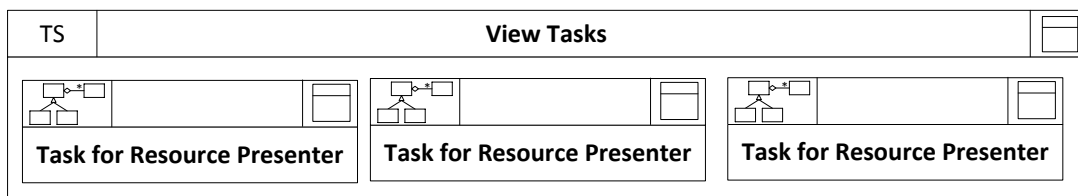


Figure A.117 - FLUIDE-A specification of the Task Supporter for the task *View Tasks*

As all the user interfaces supporting the task *View Tasks* are based on the same presenter in FLUIDE-A, there are three instances of this presenter in the Task Supporter. In the design specification of this Task Supporter (Figure A.125), three different Task Supporter Designs will be used.

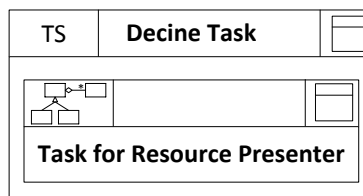


Figure A.118 - FLUIDE-A specification of the Task Supporter for the task *Decline Task*

A.2.2.2 FLUIDE-D specifications of task overview and allocation

Figure A.119 to Figure A.123 show how the user interfaces for task overview and allocation (Figure A.110 to Figure A.115) may be specified in FLUIDE-D. All these FLUIDE-D specifications correspond to the FLUIDE-A specification in Figure A.102.

A FLUIDE-D specification of the user interface in Figure A.110 is shown in Figure A.119.

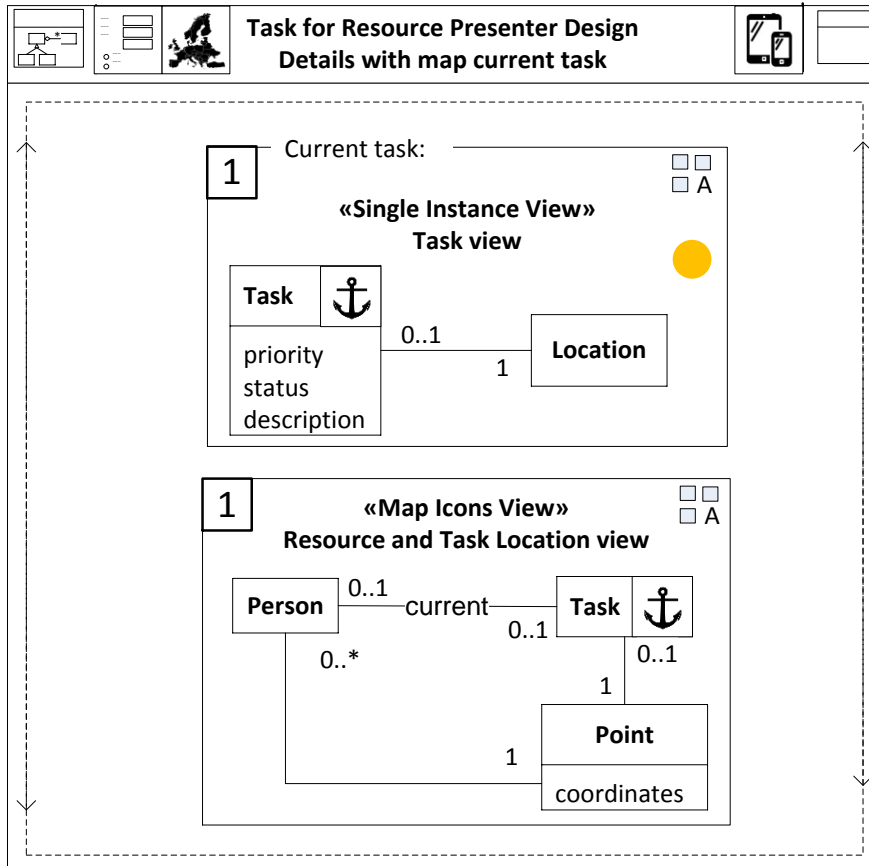


Figure A.119 - FLUIDE-D specification of user interface for seeing details about the current task

The presenter design in Figure A.119 puts together two content views using a layout manager view. Layout manager views are not given names, and are visualized by dashed lines (to indicate that it is usually not visible). The arrows on the dashed line specify whether the children are organized horizontally or vertically. The layout manager view in Figure A.119 organizes its children vertically. The first content view type used is a Single Instance View, which is a generic view type for presenting one instance at a time of a single entity. The second content view provides a map with the resource and task displayed.

A FLUIDE-D specification of the user interface in Figure A.112 is shown in Figure A.121.

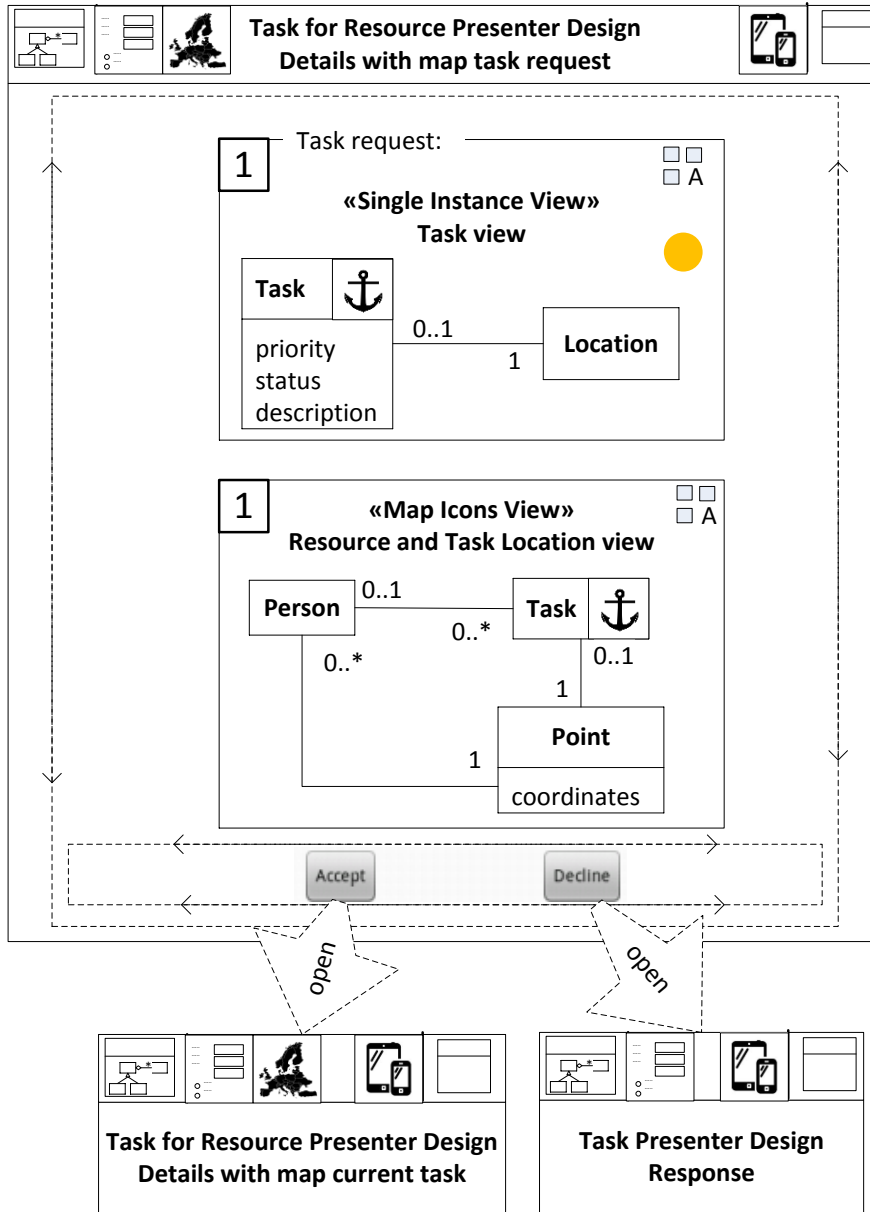


Figure A.120 - FLUIDE-D specification of user interface for seeing details about a received task request

As can be seen, the specification is similar to the one shown in Figure A.119, except for the label on the *Task view*, the relation between *Person* and *Task*, and the layout manger view including two buttons at the bottom of the user interface. Specification of the dialog navigation from these buttons is also included (the target presenter designs are shown in Figure A.119 and Figure A.123).

A FLUIDE-D specification of the task request user interface in Figure A.111 is shown in Figure A.121.

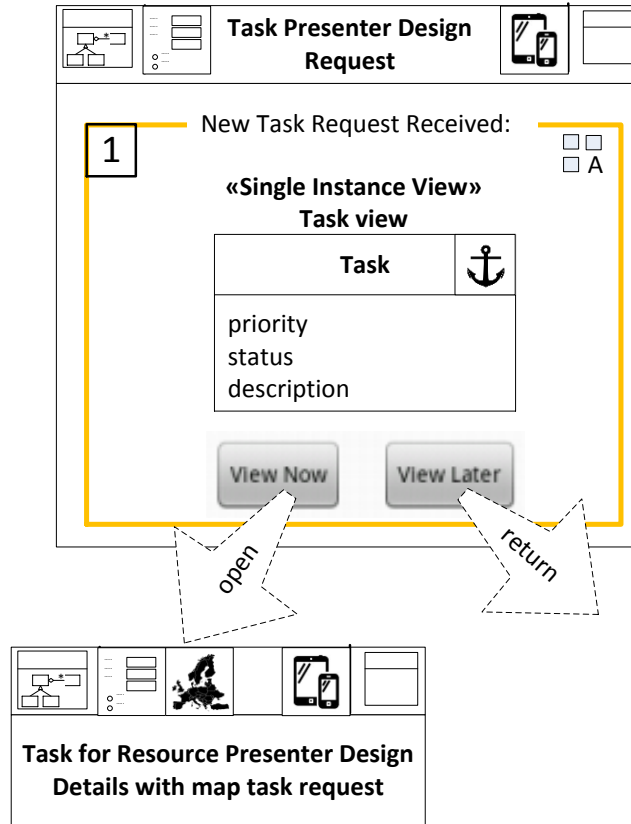


Figure A.121 - FLUIDE-D specification of the task request user interface

The orange border on the Single Instance View called Task view indicates that the view should have a border colour. Which colour to use is specified in a property for the view, where it is specified that the border colour should be determined by the colouring rule annotation in the corresponding FLUIDE-A specification. One of the dialog navigation specifications (using the "return" type of dialog navigation) has no specified target, as it specifies that that the last open dialog should be shown.

A FLUIDE-D specification of the task list user interface in Figure A.113 is shown in Figure A.122.

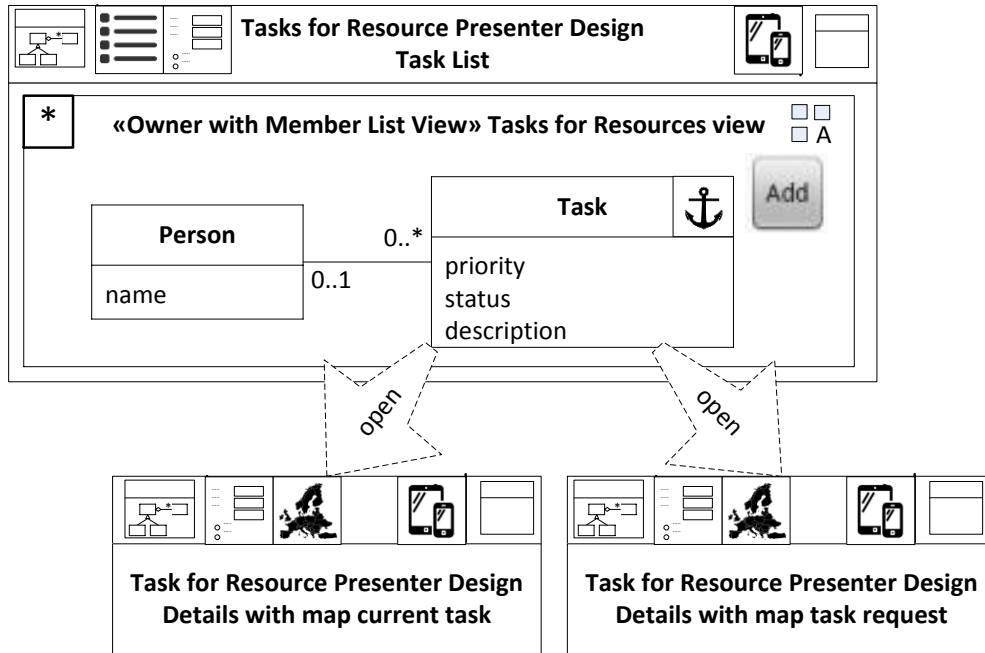


Figure A.122 - FLUIDE-D specification of the task list user interface

The presenter design in Figure A.122 uses a generic content view for presenting an entity with a list of an arbitrary number of related entities. It also includes specification of dialog navigation to the presenter designs in Figure A.119 and Figure A.120. In this specification there are two different targets for the dialog navigation, depending on whether the source is a task request or an allocated task. The dialog navigation specification for the *Add* button is omitted in this specification.

Figure A.123 shows a FLUIDE-D specification of the user interfaces for declining a task request shown in Figure A.114 and Figure A.115.

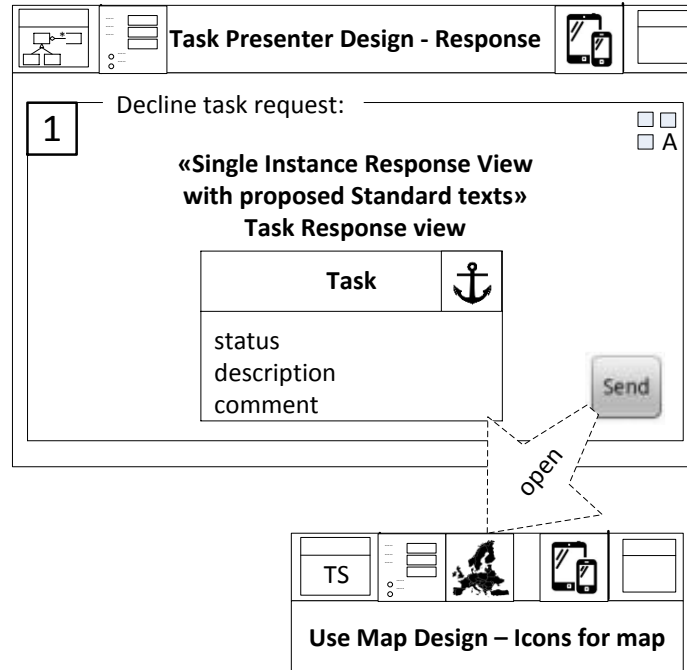


Figure A.123 – FLUIDE-D specification of user interfaces for declining a task request

The content view provides the list based user interface in Figure A.115 as part of its functionality. The navigation design specification use the Task Supporter Design in Figure A.109 as its target.

To describe how the different user interfaces supporting task overview and allocation support different user tasks, a number of Task Supporters were specified in Figure A.116 to Figure A.118. Below, designs for these Task Supporters are specified.

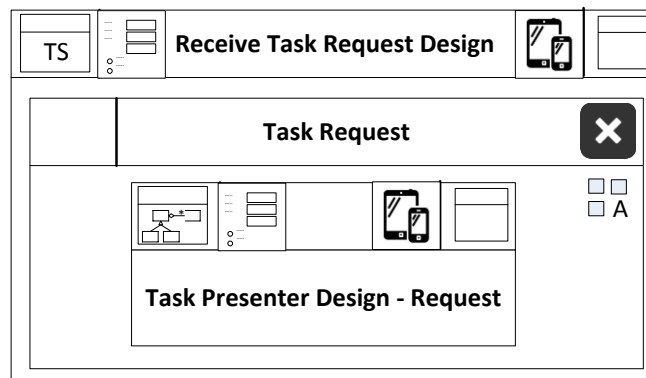


Figure A.124 – FLUIDE-D specification of Task Supporter Design for the task *Receive Task Request* (corresponding FLUIDE-A specification is shown in Figure A.116)

The Task Supporter Design in Figure A.124 utilizes a decorative view specifying that the dialog should be presented as a window (or as a full screen dialog on a mobile device). This is indicated by the close icon at the top right corner of the view. The text "*Task Request*" is the heading text for the view. The only child of this view is a view for the presenter design that is to be shown inside the window.

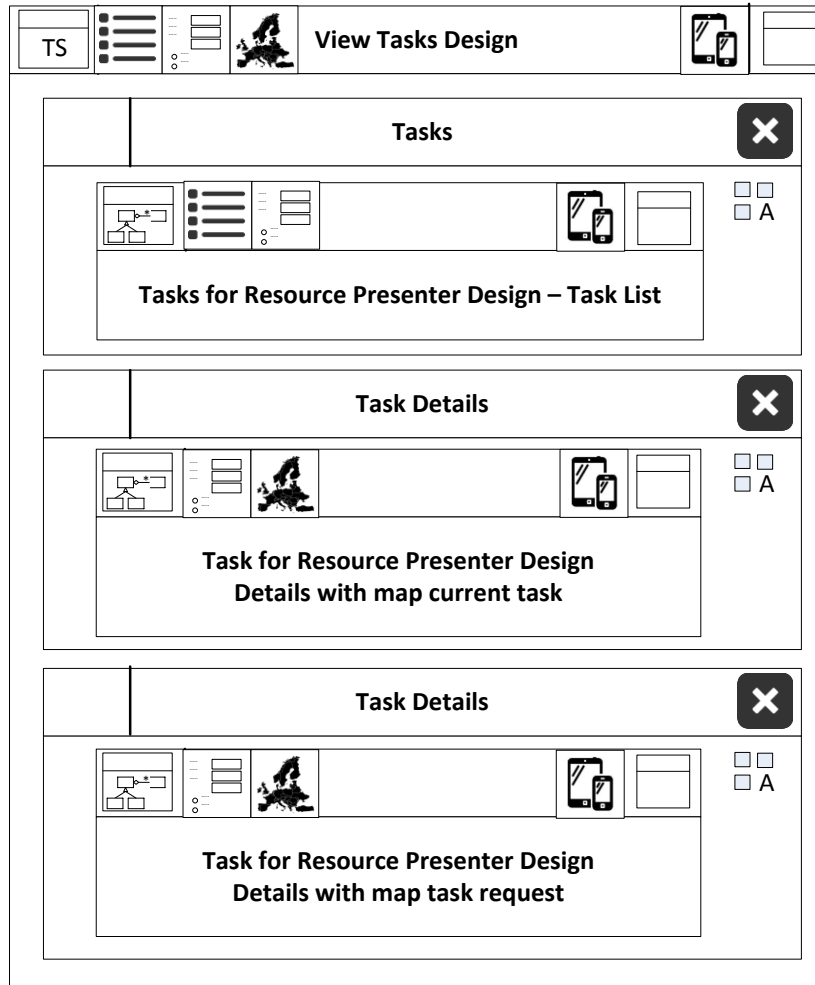


Figure A.125 – FLUIDE-D specification of Task Supporter Design for the task *View Tasks* (corresponding FLUIDE-A specification is shown in Figure A.117)

The Task Supporter Design in Figure A.125 includes three decorative views that are to be presented as windows. Comparing the Task Supporter Design in Figure A.117, one will see that the three presenter designs are different designs for the same presenter (of which the corresponding Task Supporter in FLUIDE-A contains three instances). Two of the views that are to be presented as windows have the same heading text.

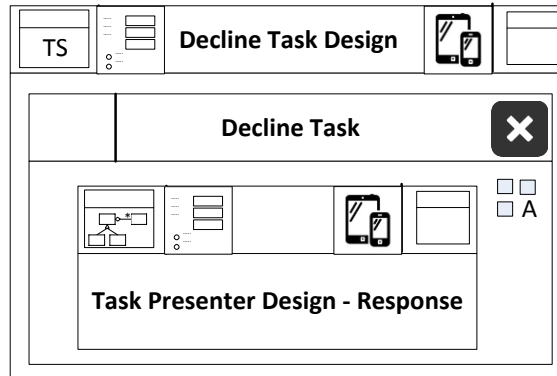


Figure A.126 – FLUIDE-D specification of Task Supporter Design for the task *Decline Task* (corresponding FLUIDE-A specification is shown in Figure A.118)

A.2.3 Putting all parts of the Resource Manager together

Above, the user interfaces for all parts of the Resource Manager application have been specified. In this section, the integration of these parts is specified.

A.2.3.1 FLUIDE-A specification for putting all parts of the Resource Manager together

To couple the location tracking and task overview and allocation parts, one new FLUIDE-A specification is needed. This specification is a Basic Work Supporter specifying which tasks the Resource Manager supports, the structure between these tasks, as well as how the Task Supporters specified above fit into this structure.

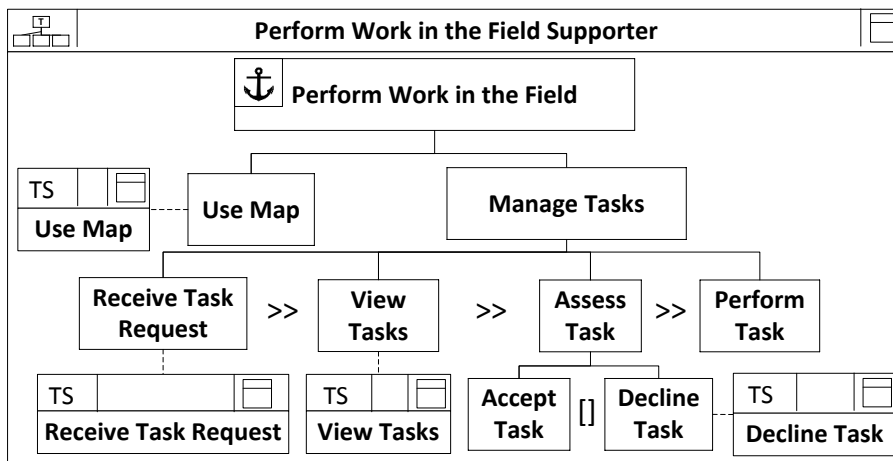


Figure A.127 - FLUIDE-A specification of a Basic Work Supporter for the *Perform Work in the Field* task

In the task model, the task *Use Map* corresponds to the location tracking part of the Resource Manager, while the task *Manage Tasks* corresponds to the task overview and allocation part. The task *Manage Tasks* is further decomposed into four tasks, of which three have a Task Supporter, while the fourth is further decomposed into two tasks without task supporters. The tasks without task supporters are not directly supported by the Resource Manager application. The task model also contains some operators (using the same symbols and precedence rules as CTT [21]): ">>" indicates sequence in task performance, while "[]" indicates a choice between tasks. No operator indicates that the tasks may be performed in an arbitrary sequence, including in parallel.

As there is only one Work Supporter in the specification of the Resource Manager, this supporter aggregates all the Task Supporters, and thus all the different part of the user interface of the Resource Manager. Despite this, the specification contains the Category Manager in Figure A.128, representing the complete Resource Manager application. The only child of this Category Manager is the Basic Work Supporter in Figure A.127.

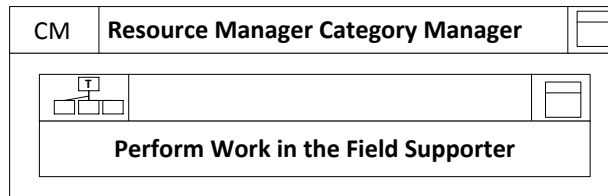
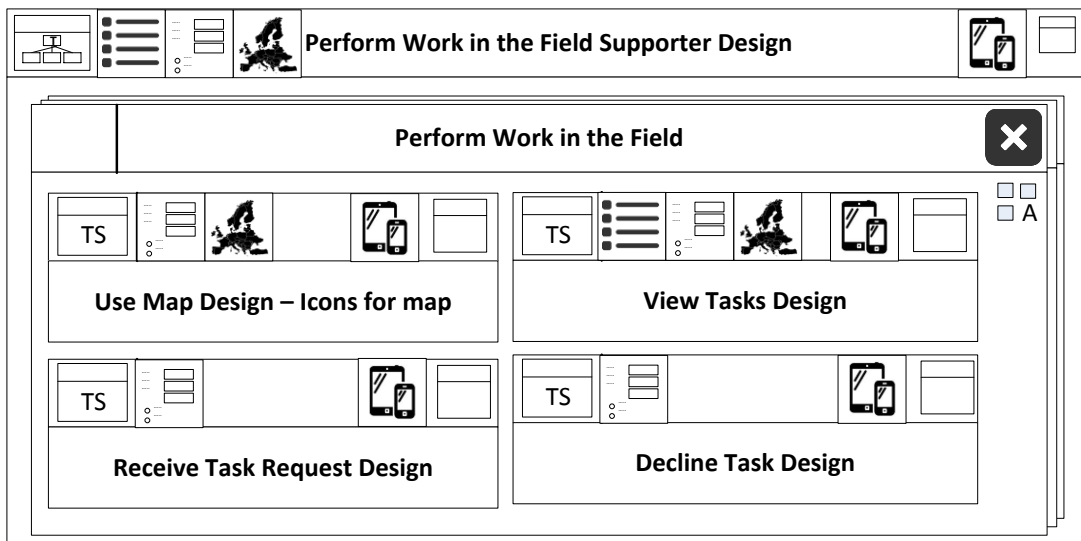


Figure A.128 - FLUIDE-A specification of Category Manager for the Resource Manager application

A.2.3.2 FLUIDE-D specifications for putting all parts of the Resource Manager together



**Figure A.129 - FLUIDE-D specification of a Basic Work Supporter Design for the *Perform Work in the Field* task
(corresponding FLUIDE-A specification is shown in Figure A.127)**

The Basic Work Supporter Design in Figure A.129 utilizes a decorative view specifying that the child designs are a set of loosely connected windows, indicated by using the same notation as for window type decorative views, augmented with a stacking look to indicate that the view contains a number of windows. How the user may navigate between the child designs is determined by the dialog navigation specified for the children (possibly at different levels). The child designs are designs for the Task Supporters that are children of the corresponding Work Supporter in Figure A.127.

The FLUIDE-A specification of Resource Manager also includes a Category Manager representing the whole Resource Manager application (Figure A.128). The design for this Category Manager is specified in Figure A.130.

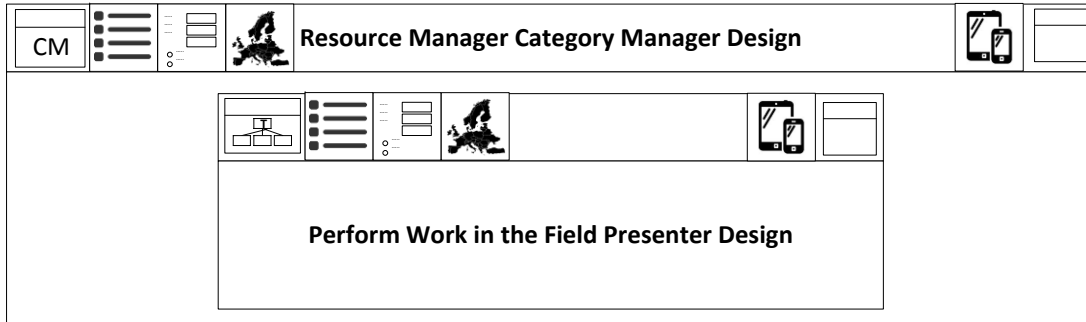


Figure A.130 - FLUIDE-D specification of a Category Manager Design for the *Resource Manager Category Manager* (corresponding FLUIDE-A specification is shown in Figure A.128)

As there the Category Manager Design in Figure A.130 only has one child, and this child contains all necessary views, there is no need for any additional views in the Category Manager Design.

A.3 The eTriage Application

The eTriage application supports personnel in the field triaging victims. The user interface for obtaining an overview of victims already triaged is shown in Figure A.131.

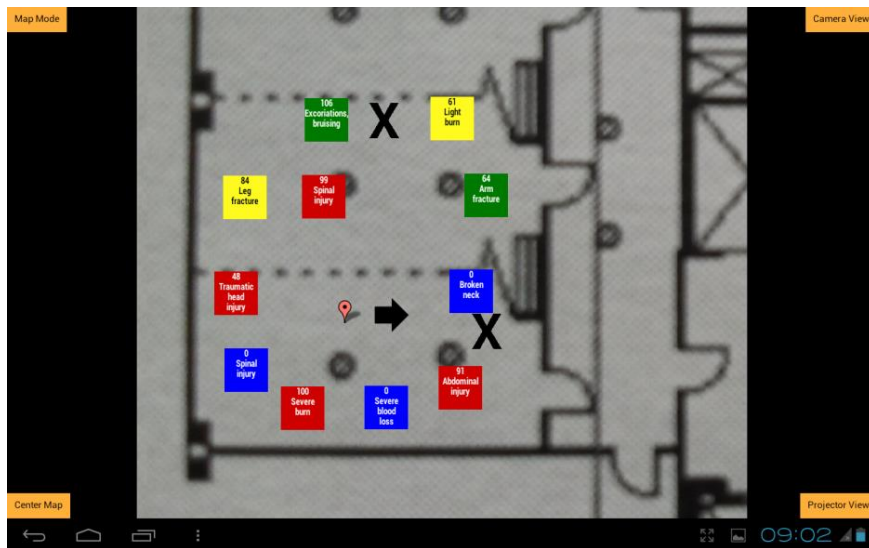


Figure A.131 – Map based user interface for getting an overview of already triaged victims

The user interface used when triaging a single victim or looking at details about a victim already triaged is shown in Figure A.132.

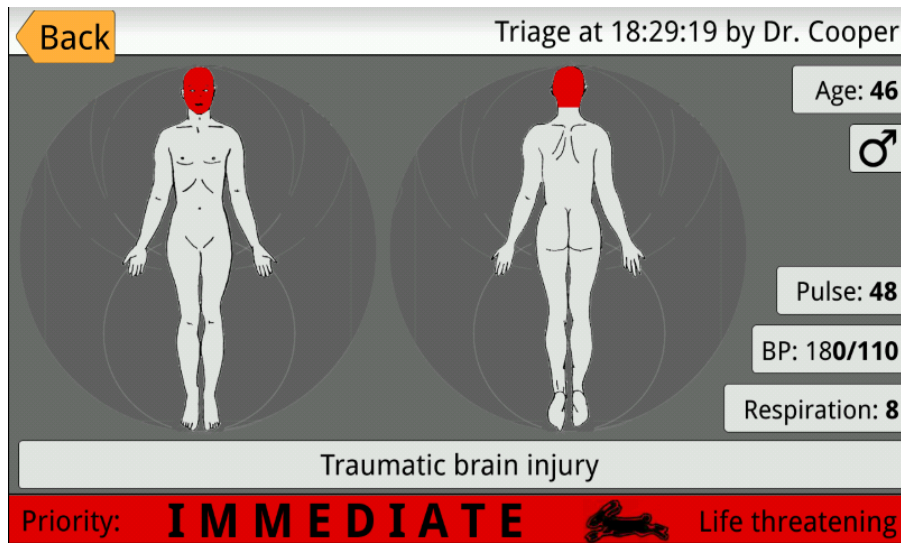


Figure A.132 – User interface for looking at or entering details about a triaged victim

Figure A.133 shows an augmented reality user interface aiding localizing patients.

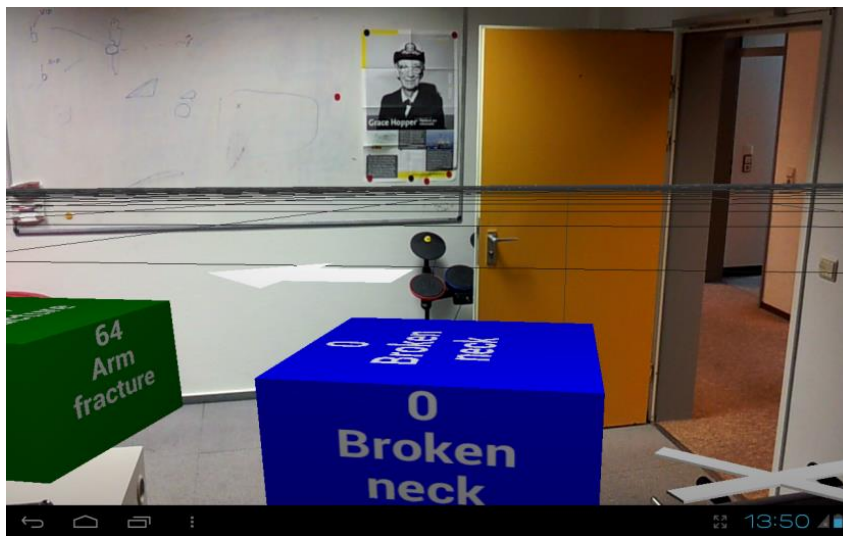


Figure A.133 – Augmented reality user interface for locating already triaged victims

A.3.1 FLUIDE-A specifications of the eTriage user interface

All the user interfaces in Figure A.131 to Figure A.133 may be specified using the FLUIDE-A specification in Figure A.134.

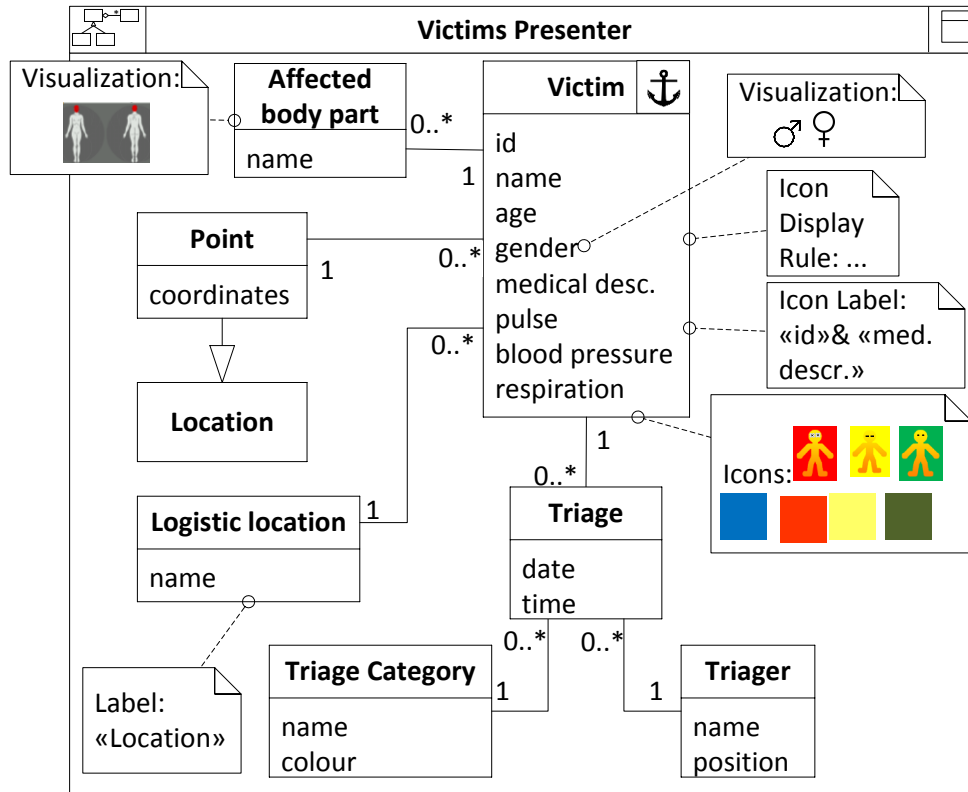


Figure A.134 - FLUIDE-A specification of the information used in the eTriage application

The presenter in Figure A.134 is also used for specifying the victims category in the MASTER application (Figure A.52), but is repeated here for readability.

As there are only three dialogs in the eTriage application, and no particular sequence in which these are used, there are no Task or Work Supporters in its specification. Instead, the three instances of the only Basic Content Presenter are aggregated into the Category Manager shown in Figure A.135 representing the user interface of the eTriage application.

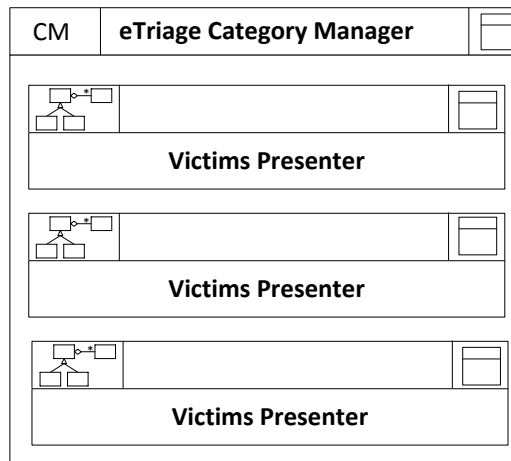


Figure A.135 - FLUIDE-A specification of the Category Manager for the eTriage application

As all the user interfaces in eTriage are based on the same presenter in FLUIDE-A, there are three instances of this presenter in the Category Manager. In the design specification of this Category Manager, three different Content Presenter Designs will be used.

A.3.2 FLUIDE-D specifications of the eTriage user interface

Figure A.136 to Figure A.138 show how the eTriage user interfaces (Figure A.131 to Figure A.133) may be specified in FLUIDE-D. All these FLUIDE-D specifications correspond to the FLUIDE-A specification in Figure A.134.

A FLUIDE-D specification of the user interface in Figure A.131 is shown in Figure A.136.

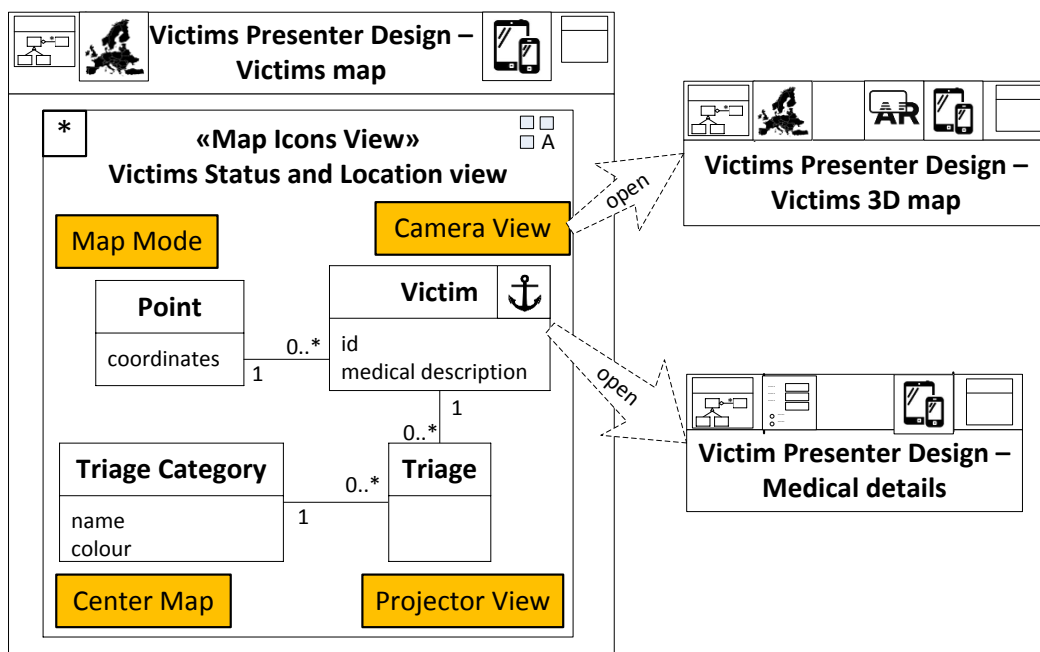


Figure A.136 - FLUIDE-D specification of the user interface for getting an overview of already triaged victims

Which dialog to open when a victim icon is tapped is specified explicitly in Figure A.136 (needed as Map Icons View is used). Three of the orange buttons inside the views are property values of the view, the fourth (the button labeled "Camera View") is an explicitly specified button providing navigation.

A FLUIDE-D specification of the user interface in Figure A.132 is shown in Figure A.137.

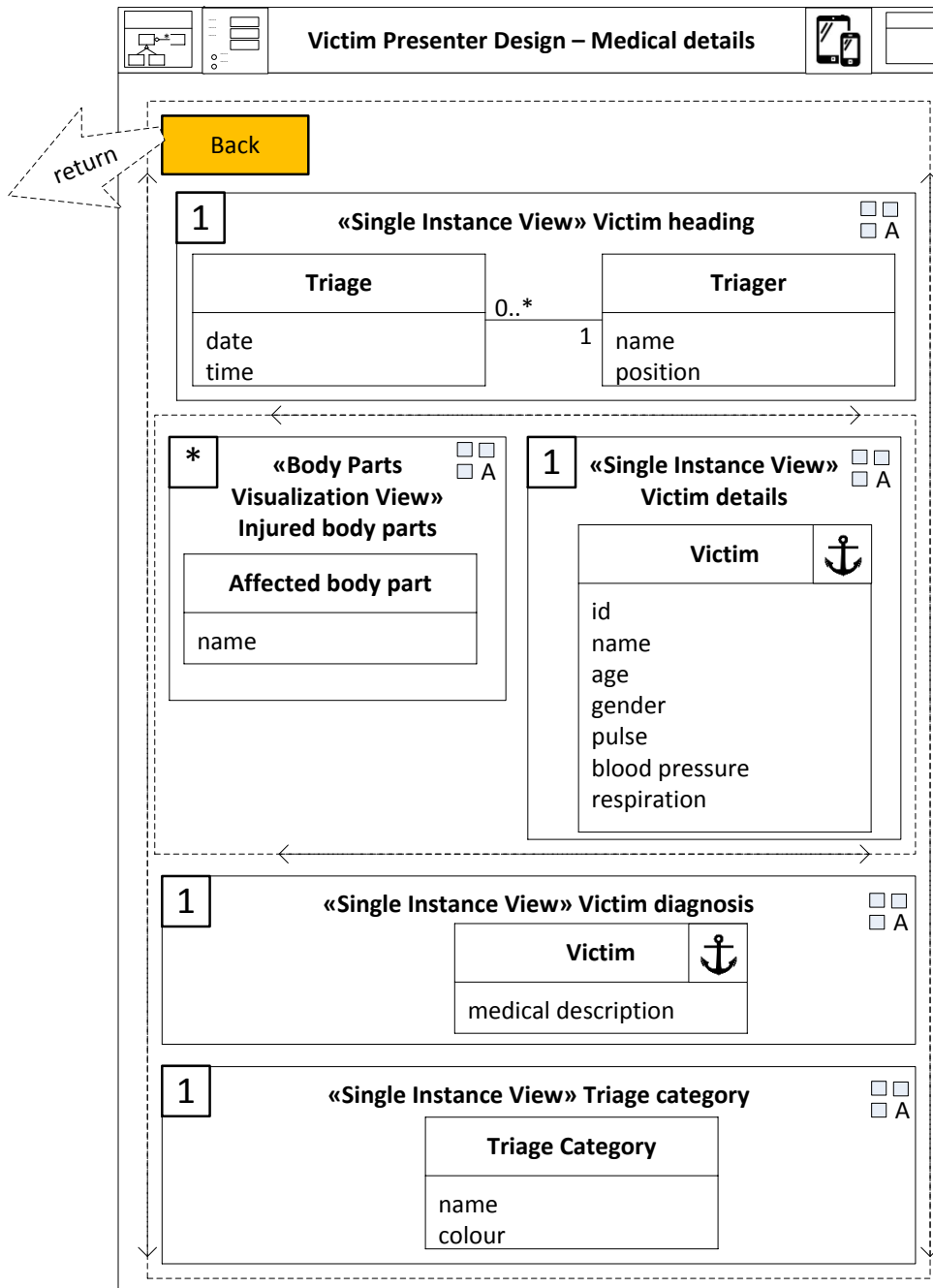


Figure A.137 - FLUIDE-D specification of user interface for getting an overview of already triaged victims

The presenter design in Figure A.137 uses a combination of generic content views and the domain-specific Body Parts Visualization View, wrapped in a number of layout manager views. It also contains a button for navigating back to the dialog from which the *Medical details* was opened.

A FLUIDE-D specification of the user interface in Figure A.133 is shown in Figure A.138.

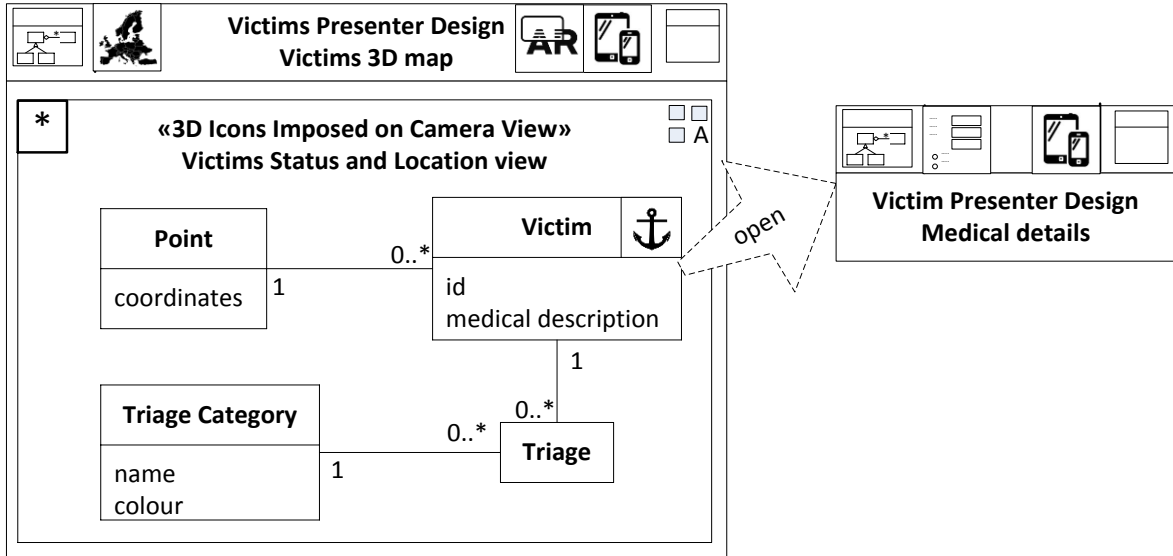


Figure A.138 - FLUIDE-D specification of augmented reality user interface for locating already triaged victims

Except for the addition of augmented reality user interface modality/platform (shown by the additional icon at the top right), the content view (3D Icons Imposed on Camera View), and the absence of orange buttons, the presenter design is identical to the one specifying the map based user interface (Figure A.136). The 3D Icons Imposed on Camera View requires the same model pattern as Map Icons View, and provides an augmented reality view on localized information. It has the same aggregation and display rules as the family of content and content integration views for displaying map overlays.

In addition to the Content Presenter, the FLUIDE-A specification of eTriage includes a Category Manager representing the whole eTriage application (Figure A.135). The design for this Category Manager is specified in Figure A.139.

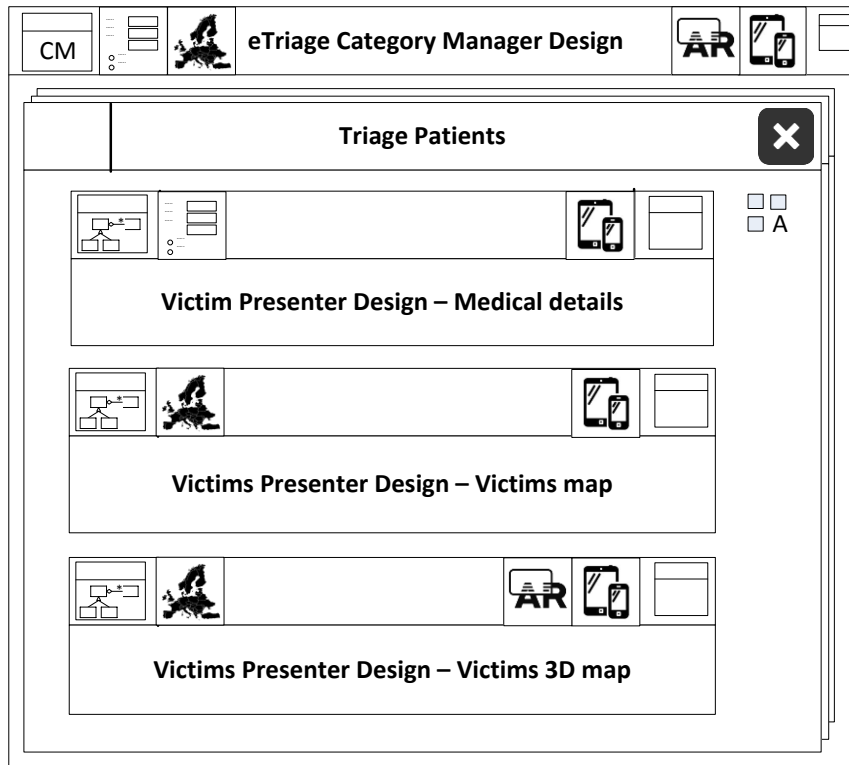


Figure A.139 – FLUIDE-D specification of Category Manager Design for the iTriage application (corresponding FLUIDE-A specification is shown in Figure A.135)

The Category Manager Design in In Figure A.139 utilizes the same type of decorative view that is used for the top level of Resource Manager (see Figure A.129). How the user may navigate between the child designs is determined by the dialog navigation specified for the children.



Technology for a better society

www.sintef.no