A27575- Unrestricted

Report

The FLUIDE Framework for Specifying Emergency Response User Interfaces Employed to a Search and Rescue Case

Author(s)
Erik Gøsta Nilsson
Ketil Stølen

POLITI
POLITI
POLITI
Uløvlig passering medfører straffeansvar, jf. Politiloven § 30
AMBULANSE
113

SINTEF ICT
Networked Systems and Services
2016-12-29

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;
Search and rescue

Report

The FLUIDE Framework for Specifying Emergency Response User Interfaces Employed to a Search and Rescue Case

VERSION	DATE
Final	2016-12-29

AUTHOR(S)
Erik Gøsta Nilsson
Ketil Stølen

CLIENT(S)	CLIENT'S REF.
The EMERGENCY project supported by the Research Council of Norway, p.nr. 187799/S10	187799/S10
PROJECT NO.	**NUMBER OF PAGES/APPENDICES:**
90B261	74 incl. appendices

ABSTRACT

This report presents the FLUIDE Framework with particular emphasis on its specification languages. The FLUIDE Framework supports development of flexible emergency response user interfaces. We demonstrate the FLUIDE Framework by giving examples from the FLUIDE specification of the user interface of an application supporting management of unmanned vehicles in search and rescue operations. We also report the findings from an experiment investigating how easy FLUIDE specifications are to understand for systems developers not knowing FLUIDE. We discuss these findings, their implications on the design of the languages and how specifications should be presented to systems developers, as well as implications for other specification languages, concluding that to ensure comprehensibility, only concrete specifications need to be communicated. A more comprehensive presentation of the user interfaces of the search and rescue application, and the complete FLUIDE specifications of these user interfaces are given in an appendix. Two further appendices include the user tasks and introductory material used in the experiment.

PREPARED BY Erik G. Nilsson	SIGNATURE
CHECKED BY Jan Håvard Skjetne	SIGNATURE
APPROVED BY Bjørn Skjellaug	SIGNATURE

REPORT NO.	ISBN	CLASSIFICATION	CLASSIFICATION THIS PAGE
A27575	978-82-14-05931-1	Unrestricted	Unrestricted

Table of contents

1 Introduction

In this report, we present the FLUIDE Framework with particular emphasis on its specification languages and particularly the comprehensibility of these languages. The FLUIDE Framework supports development of flexible emergency response user interfaces.

The response by first responders to a simple everyday incident is very different from how a long-lasting serious catastrophe is handled. Increasing complexity tends to cause increasing change rate and decreasing predictability. Developing ICT solutions supporting such a range of responses is challenging. Designing user interfaces that are able to adapt to and reflect these variations is particularly challenging.

Local leaders at the incident site should be able to use the same applications on different equipment types with different screen sizes (Nilsson and Stølen, 2010). Field workers need non-intrusive ICT support, using non-visual modalities in addition to visual ones. ICT support for emergency responders, if at all available, tends to handle the needs for flexibility by being generic and data oriented. This forces responders to adapt to the solutions, and not the other way around. There are on the other hand many similarities and reoccurring patterns across emergency response operations. This includes tasks and information needs of individual operations and the involved actors. Previous research (Nilsson and Stølen, 2011) has shown that it is possible to characterize such similarities and patterns using a limited number of categories of functionality.

We suggest a development approach providing mechanisms for composing end-user solutions from flexible and tailor-friendly components supporting such categories of functionality. Such components combine being ready-to-use with being highly configurable and support composition and configuration both at design- and run-time. Using traditional programming languages to develop user interfaces for such solutions is very challenging if at all possible. Because all imaginable combinations of functionality, compositions and configurations must be supported, such user interface developments are very resource demanding. Existing model-based user interface development approaches are also seriously challenged by such solutions. Summarized, there is a need for building blocks that meet these four requirements:

R1. Are at a sufficiently high level of abstraction and provide compound structures of simple elements and containers/dialogs to support common specifications between platforms and modalities

R2. Have reflection mechanisms giving an awareness of model structures (including domain models) to support adaptation both at design- and run-time

R3. Support development of user interfaces where the layout depends on the instances at run-time, typically using icons, maps, graphical elements, and alternative modalities like speech

R4. Provide specific support for user interface patterns that are particularly useful for emergency response

Neither MARIA (Paternò et al., 2009), UsiXML (Limbourg et al., 2004), nor OMG's Interaction Flow Modeling Language (IFML)[1] (Brambilla and Fraternali, 2014) meet R1 fully. They provide abstract building blocks, but none of the approaches offer compound building blocks. The building blocks in these approaches are abstractions of simple user interface elements and containers for structuring these, but there are very few composed building blocks. With simple building blocks, the composition structure of the user interfaces, which is different when the platforms have large differences, is reflected in the specifications even at the abstract level. MARIA and UsiXML meet R2 to some extent,

[1] www.ifml.org

and MARIA partly meets R3 by composing external user interface services, while there exists extensions to UsiXML supporting maps and 3D user interfaces. ICOs (Navarre et al., 2009) on the other hand, meets R3 well by supporting development of post-WIMP user interfaces, but does not meet R1 and meets R2 only partly. R4 is best met by ICOs, which has been proven useful in domains like command and control, air traffic control and cockpit systems.

The FLUIDE Framework provides building blocks fulfilling R1-R4. This report, which is an extended version of Nilsson and Stølen (2016), presents selected parts of the FLUIDE Framework in an example-driven manner by specifying a search and rescue application. It also presents results from an experiment involving 29 potential users exploiting the FLUIDE specification of the user interface part of a search and rescue application.

2 The FLUIDE Framework

The FLUIDE Framework is a development environment for professional systems developers containing

- A collection of ready-to-use and highly configurable FLUIDE components supporting flexible composition of end-user solutions for emergency responders
- Composition and configuration approaches
- The FLUIDE specification languages
- A generic mechanism to generate FLUIDE components from FLUIDE specifications
- The FLUIDE method supporting the use of the framework

The FLUIDE Framework offers two specification languages: FLUIDE-A for specifying user interfaces in an abstract, platform-independent manner, and FLUIDE-D for specifying concrete, platform-specific designs based on the FLUIDE-A specifications. FLUIDE-D contains a library of user interface patterns that are particularly useful in the emergency response domain, thus meeting R4. FLUIDE-D specifications may automatically be transformed to a running user interface. Having specification languages at two different abstraction levels enables combining platform-independent specifications with platform-specific ones in a structured way. Usability of the user interfaces specified using FLUIDE is ensured through a predictable generation process from compound building blocks and the flexibility due to the component-based approach.

The FLUIDE specification languages meet R1 by providing compound building blocks, which enables common platform-independent specifications across platforms and modalities. It also allows compact platform-specific specifications of advanced user interfaces, including user interfaces where the layout depends on instances at run-time, thus meeting R3. By support of model patterns, the compound building blocks are versatile. R2 is met through reflection enabled by embedding domain models in the specifications.

Figure 2.1 gives an overview of three of the main language constructs in FLUIDE and how they support a natural break-down of emergency response work.

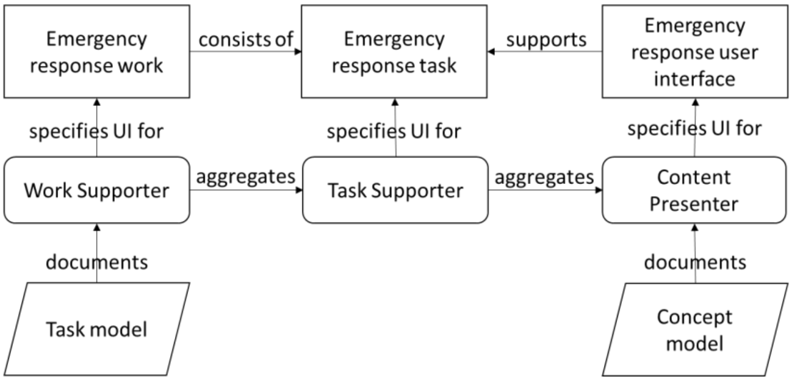
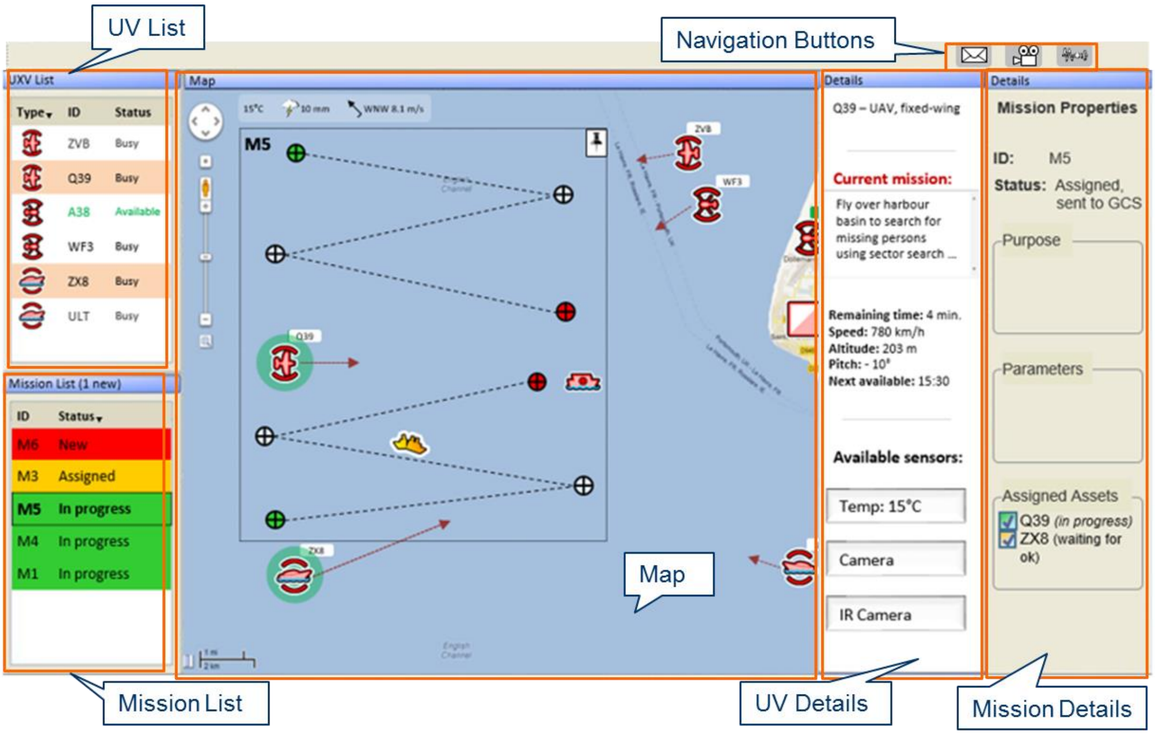


Figure 2.1 – Overview of the main constructs in FLUIDE-A

The work performed by emergency responders 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. In FLUIDE-A, such task structures are specified using the *Work Supporter* construct, which includes a task model to specify hierarchical and temporal structures. A user interface supporting one such task needs to manage certain information content that is 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 a (part of a) user interface is specified using the *Content Presenter* construct. The information to be presented by a Content Presenter is specified through 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 that is to be presented in a user interface at run-time, and how it will be presented.

To specify how a FLUIDE-A specification is transformed to a concrete user interface we provide FLUIDE-D, a separate language for specifying designs for the FLUIDE-A specifications. FLUIDE-D contains variants of the main constructs in FLUIDE-A, using the same names with the suffix *Design.* The constructs in FLUIDE-D are used to specify which parts of the concept and task models that are to be included in a user interface. FLUIDE-D's core is the library of user interface patterns, operationalized in the *View* constructs. 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.

3 The Search and Rescue Case



Type	ID	Status
	ZVB	Busy
	Q39	Busy
	A38	Available
	WF3	Busy
	ZX8	Busy
	ULT	Busy

Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

Available sensors:

Temp: 15°C

Camera

IR Camera

Mission Properties

ID: M5
Status: Assigned, sent to GCS

Purpose

Parameters

Assigned Assets
☑ Q39 *(in progress)*
☑ ZX8 (waiting for ok)

Figure 3.1 – The user interface of the GGS application used in the case

In the case study, we retrospectively specified the user interface of an existing application without any connections to FLUIDE. This application, the Generic Ground Station (GGS) application, was developed as part of the research project DARIUS[2]. We denote this the *target user interface*. By specifying the user interface of an existing application we ensure realism in the case, and do not need to design the user interface from scratch. The case plays two roles. First, the case is used to assess the suitability of the FLUIDE specification languages for their purpose. Second, as we have a particular focus on comprehensibility, the user interface of the GGS application and the corresponding FLUIDE specifications are used in the experiment described in Sections 4-6. In the following, we present examples from the FLUIDE specification. The full description of the search and rescue case, including the complete FLUIDE specification, is available in Appendix A.

The GGS application supports emergency personnel managing unmanned vehicles (UVs) as part of an emergency response. The user interface of the GGS application (Figure 3.1) consists of a map display showing among other the locations of the incident, the GGS, the UVs, and the search areas. UV search paths and trails are visualized using icons and graphics. Lists of and details for the UVs and missions are shown in separate panes.

Mission Presenter in Figure 3.2 specifies the Mission List and Mission Details panes (both shown in Figure 3.1) in FLUIDE-A. FLUIDE-A uses a subset of the UML class diagram notation (extended with the anchor symbol identifying the root entity of the model) to express concept models. Annotations specify additional platform-independent visual properties.

[2] http://www.darius-fp7.eu/

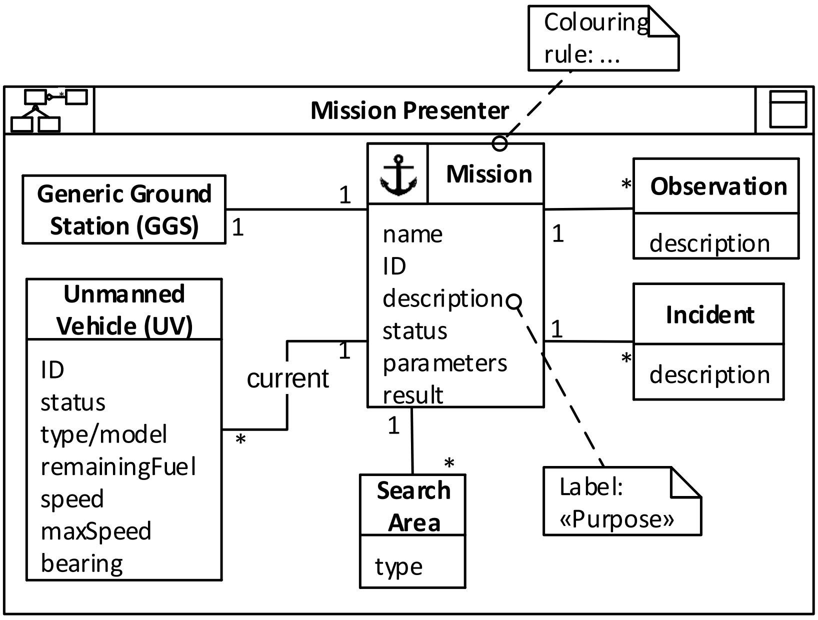


Figure 3.2 – FLUIDE-A specification of Mission Presenter

Figure 3.3 presents a FLUIDE-D design for the Mission List, Figure 3.4 shows the design for the Mission Details pane, while Figure 3.5 specifies how UVs are to be displayed as icons on the map (all part of Figure 3.1). The outer border of a FLUIDE-D specification specifies the user interface style(s) and modalities/platform(s) the design targets (PC for GGS). Designs contain one or more *views* in a hierarchical structure. Figure 3.3, Figure 3.4, and Figure 3.5 all use Content Views, i.e. views that present instances of one or more entities. Together with Content Integration Views, the available Content Views make up the library of emergency response user interface patterns. Such views represent one of the compound building blocks in FLUIDE-D and provide means for specifying advanced designs in a compact way also ensuring usability in the target user interface. They provide versatility by being based on model patterns, enabling specification of advanced user interfaces managing a wide variety of information as long as this information has a structure matching the model patterns of the view.

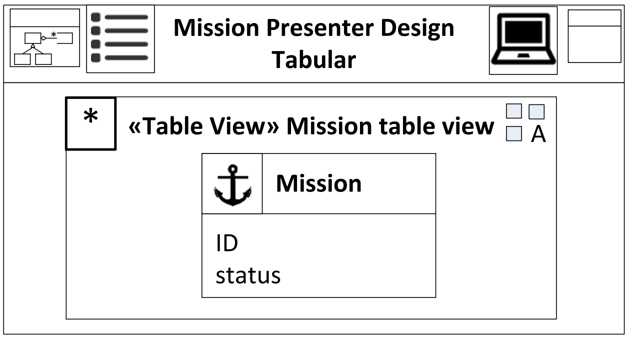

Mission Presenter Design Tabular
*
«Table View» Mission table view
A
Mission
ID
status

Figure 3.3 – FLUIDE-D specification of the tabular version of Mission Presenter Design

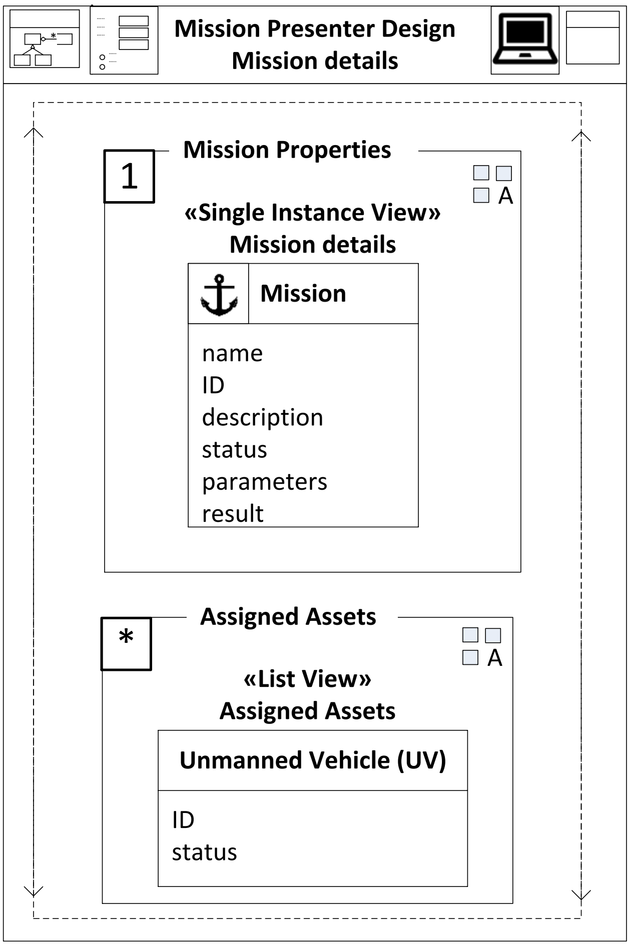


Figure 3.4 – FLUIDE-D specification of the forms-based version of Mission Presenter Design

For example the *Map Icons View (Figure 3.5)* provides means for presenting icon-based information in a map user interface as long as the model follows a given structure. Thus, this view may just as well be used for presenting incidents, UVs or victims in the map. Such views combine being specialized and powerful with respect to emergency response needs with being versatile with respect to the actual information they present.

In the case, we were able to fully describe the different parts of the user interface of the GGS application without meeting any major obstacles. We faced some challenges when specifying the Work Supporters, as the structure of the user tasks only partly matches the aggregation structure in the user interface. We have no indications that the specifications do not contain sufficient information for the target user interfaces to be schematically deducible.

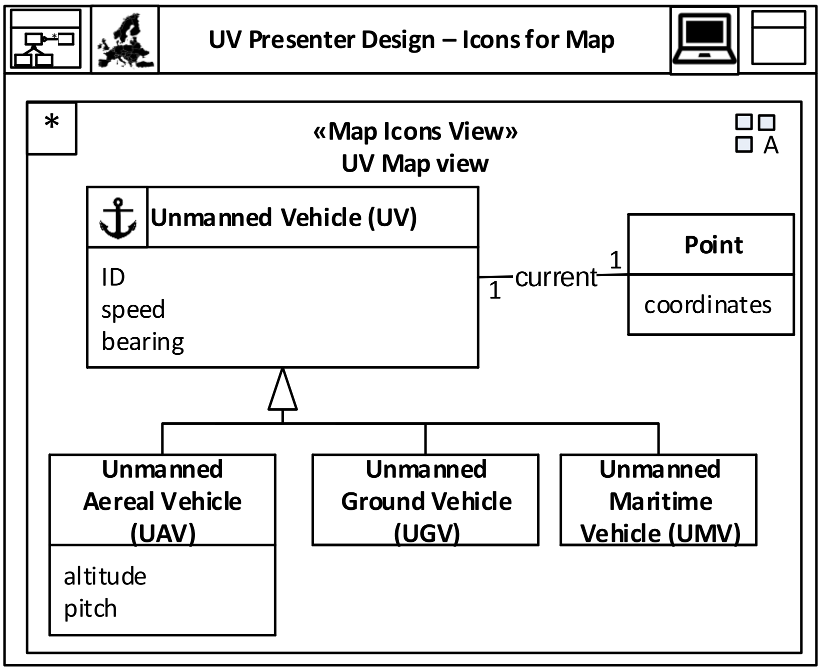

UV Presenter Design – Icons for Map
*
«Map Icons View»
UV Map view
A
Unmanned Vehicle (UV)
ID
speed
bearing
current
1
1
Point
coordinates
Unmanned Aereal Vehicle (UAV)
altitude
pitch
Unmanned Ground Vehicle (UGV)
Unmanned Maritime Vehicle (UMV)

Figure 3.5 – FLUIDE-D specification of map presentation of UVs

4 Experiment Design

The overarching research question we addressed in the experiment (Shadish et al., 2002) was: *To what extent are the FLUIDE specifications comprehensible to systems developers?* To address this research question, we conducted an experiment involving systems developers. In the experiment, the main way of gaining insight on the research question was by measuring to which degree the participants were able to match FLUIDE specifications and corresponding user interfaces from the search and rescue case. It was not our goal to obtain an absolute answer to the research question. Rather, we wanted to find out how easy the specifications are to understand for systems developers with different background and experience, as well as the comprehensibility of different parts of the languages.

To address the former, the subjects in the experiment had different education level, background and experience. To address the latter, we had three conditions in the experiment. The experiment involved both students and researchers. It was conducted in seminar rooms, lasted for one hour, and was divided into three parts. In the first part (approximately 15 minutes), the participants were given a short introduction to FLUIDE, the search and rescue case (with focus on the users and their tasks) and the goal and execution of the experiment. The presentation used for this purpose is included in Appendix C. In the second part (approximately 40 minutes), the participants solved the given tasks using pen and paper. In the third part (the last five minutes), the participants rated their experience from solving the tasks. In addition, the participants signed an informed consent and filled in a page with the background information. The user tasks (including the forms used for background information and for rating the experience) is included in Appendix B. All questions about knowledge used a Likert scale with 5 values.

There were in total 29 participants, 20 males and 9 females. The age varied from 20 to 58, with 34 as average (sd=9.4). Table 4.1 shows the knowledge profile of the participants. The average UML experience for the participants were 3.9 years (sd=0.78). Average task modelling experience were 0.5

years (sd=1.4). 13 participants had formal background or experience within model-driven development (MDD), 13 did not have this (yes/no).

	No knowledge	Minor knowledge	Some knowledge	Good knowledge	Expert
Knowledge of UML class models	1	11	4	11	2
Knowledge of task modelling	9	16	3	1	0
Knowledge of user interface design and usability	3	8	9	6	3
Knowledge of emergency response work	14	7	2	4	1

Table 4.1 - Knowledge profiles

The following three conditions were used in the experiment:

(i) both FLUIDE-A and FLUIDE-D specifications were shown

(ii) only FLUIDE-D specifications were shown

(iii) only FLUIDE-A specifications were shown

There were three task types in the experiment. Task type 1 consisted of 18 tasks (six for each condition – Task 1.* in Appendix B) where the subjects were asked to rate whether a FLUIDE specification matched a target user interface fully, partly or not at all. Task type 2 consisted of nine tasks (three for each condition – Task 2.*.* in Appendix B) where the subjects were asked to identify which of four specifications that best matched a given target user interface. Task type 3 consisted of nine tasks (three for each condition – Task 3.*.* in Appendix B) where the subjects were asked to identify which of four target user interfaces that best matches a given specification. In addition to the tasks, the participants were given an extra sheet of paper with an explanation of the most important parts of the FLUIDE specification languages (Slide 6 in Appendix C).

Questions about difficulty were answered using a Likert scale with six values. The question about the participants' reaction to integrating UML class models in the specifications was answered using a Likert scale with five values. The student participants received a small gift as gratitude for participation, and some larger prices were drawn among the participants.

To operationalize the research question we formulated these hypotheses about the outcome of the experiment:

- H1: Competence in using modelling languages like UML class models and task modelling will improve the score in the test
- H2: The participants' assessment of difficulty of the tasks will correlate with the score in the test
- H3: Scores for the different conditions will vary, more precise:
  - H3.1: Score for condition (i) will be higher than for condition (ii)
  - H3.2: Score for condition (ii) will be higher than for condition (iii)
  - H3.3: Score for condition (i) will be higher than for condition (iii)

5 Findings from the Experiment

5.1 Task Scores

For each task, the participants' answers were given a score of 0 if wrong and 1 if correct. All scores reported in the following are averages. The average of all participants' (average) score was 0.57 (sd=0.13). The highest score obtained was 0.81, the lowest was 0.21. The correct answer was chosen by most participants for 29 of the 36 tasks. For 22 of these tasks, a majority of the participants answered correctly. The most difficult task had an average score of 0.19 (sd=0.40), while the easiest task was answered correctly by all.

5.2 Assessment Questions

The answers to the questions about difficulty are summarized in Table 5.1.

	Impossible	Very difficult	Somewhat difficult	Quite easy	Very easy	Trivial
Were the specifications easy to understand?	0	7	15	5	1	0
Was it easy to grasp the essential part of the specifications?	0	2	10	14	2	0
Was it easy to understand the connections between the specifications and the user interfaces?	0	4	13	10	1	0

Table 5.1 - Assessment of difficulty

Most participants did not react to the mixing of UML class diagrams and FLUIDE-specific notation, and among the participants who reacted, a majority reacted positive.

5.3 Findings Related to the Hypotheses

We assessed H1 by investigating the scores in relation to UML knowledge, as UML class diagrams are used in FLUIDE. The results from this shows a weak, but not significant trend that scores increase with UML knowledge. We found though that the participants with background or experience in MDD had a significantly higher score (0.63, sd=0.11) than the ones without this background (0.52, sd=0.14). An equal variance t test shows t=2.461 (24) $p<0.05$.

When investigating H2, we used the average assessment for the three ratings of difficulty (Table 5.1). The three ratings included in the measure had acceptable inter-item reliability (Cronbach's alpha 0.75) for doing this. When comparing the difficulty rating for the participants with background or experience in MDD (cf. H1) with the rest, we saw that the MDD group found the specifications significantly easier to understand. An equal variance t test shows t=2.215 (24) $p<0.05$.

To investigate H3.1, we compared the average scores for the tasks within condition (i) (0.58, sd=0.18) with the similar scores for the tasks within condition (ii) (0.63, sd=0.16), i.e. the score for condition (i) was lower than for condition (ii). The t test shows that the difference is not significant.

To investigate H3.2, we compared the average scores for the tasks within condition (ii) (0.63, sd=0.16) with the similar scores for the tasks within condition (iii) (0.51, sd=0.19). A paired t test shows that this difference is significant: t=3.452 (28) $p<0.01$.

To investigate H3.3, we compared the average scores for the tasks within condition (i) (0.58, sd=0.18) with the similar scores for the tasks within condition (iii) (0.51, sd=0.19). A paired t test shows that this difference is significant: t=1.1905 (28) p<0.05.

6 Discussion

6.1 The Search and Rescue Case

The specifications contain occurrences of all the main FLUIDE language constructs except the *Category Manager* construct. This construct couples a set of Work Supporters that support a category of functionality. As the GGS user interface supports one user doing very specialized work, all user tasks may be aggregated into a single Aggregated Work Supporter. Thus, there is not a need for any Category Managers. Both domain-specific and generic view types were exploited in the case, so we experienced that having a library combining domain specific and generic view types was useful when specifying the GGS user interface.

FLUIDE-A specifications were used as basis for a several FLUIDE-D specifications in the case. This supports the rationale behind having two languages on different levels of abstraction. The reuse mechanisms in FLUIDE, which are of prime importance for keeping the specifications simple, were applied whenever applicable. In addition, the complexity of the target user interfaces is well reflected in the complexity of the FLUIDE specifications. The experience from using the FLUIDE languages for specifying the search and rescue case indicates that they are well suited for specifying user interfaces in applications supporting a GGS user in a search and rescue operation.

6.2 Findings from the experiment

As the introduction to the FLUIDE languages was in total approximately 5 minutes, and solving the tasks was performed under time pressure, we argue that the experiment was conducted under conditions where we could expect rather low scores. Under this assumption, we rate the score level and the share of tasks with most correct or a majority of correct answers as an indication that understanding FLUIDE specifications is highly achievable for systems developers.

The difficulty scores show that it is possible to match FLUIDE specifications with user interfaces without fully understanding the specifications. This indicates that appropriate training will ensure good understanding. Our approach of using UML and task modelling notation as part of the FLUIDE notation is not common among comparable approaches, so it is thus reassuring that a majority of the participants were neutral or reacted positively to this. The fact that a majority of the participants who reacted to the mix reported that it enhanced the understanding provides support for an important design choices for the languages.

One explanation of the finding that UML knowledge is not sufficient to explain differences in understanding is that it is relatively simple for participants without much UML competence to assess the connection between the content of a UML class model and a target user interface just based on matching names of elements in the UML models and content of user interface annotations with labels and headings in the user interfaces. Being able to successfully do this type of *lexical matching* is an indication that the UML class models, as well as the FLUIDE constructs exploiting such models, are intuitively understandable. But regarding the correlation between UML knowledge and task performance, this raises the scores for the participants with low UML knowledge, and thus weakens the trend.

We found, though, that competence in MDD increases understanding significantly. That this competence is more important than UML competence may be explained in different ways. First, the

MDD group is usually familiar with seeing connections between abstract models and a concrete computer system. This may explain that this group scores higher than systems developers just using UML for understanding a domain and in early analysis phases. Second, the MDD characteristic is probably better than UML knowledge for identifying the participants that are able to match specifications and user interfaces in tasks where lexical matching is not effective.

As perceived difficulty is significantly lower in the MDD group than for the participants without this competence, it is likely that the latter group will have a lower motivation when working with FLUIDE specifications, and thus that training should be tailored for the groups.

The score differences between the conditions where the FLUIDE-D specifications are included and the conditions where only FLUIDE-A specifications are shown may be explained by the different abstraction levels, meaning that the "distance" between the specifications and the target user interfaces is larger in the FLUIDE-A than in the FLUIDE-D specifications. Furthermore, the FLUIDE-A specifications often contain model elements that are not shown in the target user interface. Lexical matching will be less effective, and may even be misleading in such specifications. Lexical matching is more effective in condition (ii) as the FLUIDE-D specifications never have such additional information.

The choice of having two languages is grounded in the traditional split in model-driven user interface development between abstract and concrete user interface specifications (Calvary et al., 2003). Comprehensibility is neither a rationale for having the same split, nor for connecting the user interface annotations to the FLUIDE-A specifications. Our main reasons for formulating H3.1, i.e. that the presence of FLUIDE-A specifications would increase the understanding of the FLUIDE-D specifications, is that the annotations are not shown in the FLUIDE-D specifications. The results from the experiment do not support this, and we conclude that to obtain an understanding of the match between a FLUIDE specification and the corresponding user interface, it is sufficient to use FLUIDE-D specifications. FLUIDE-A specifications should primarily be a means for the systems developers making the FLUIDE specifications.

To increase our insight into which aspects of the specifications that contribute to understanding, we inspected 21 tasks (equally distributed between the conditions) having particularly low or high scores. We inspected both characteristics of the tasks and how the answers were distributed.

Several of the most difficult tasks involving Content Presenters in conditions including FLUIDE-A specifications are categorized by having complex FLUIDE-A specifications, usually with information that may mislead when doing lexical matching, and thus reducing the success of this approach. This observation is also supported by two of the easy tasks in these conditions, both having simple FLUIDE-A specifications without any major information not present in the target user interface.

None of the Content Presenter Designs for the most difficult tasks in the condition using only FLUIDE-D revealed any indications of problems with matching the UML model with the user interface elements. Low scores are probably caused by lack of understanding of FLUIDE-D properties. A FLUIDE-A specification would not have contributed to understanding for any of these tasks.

6.3 Implications of the Findings

Our interpretation of the lack of effect on comprehensibility of including the FLUIDE-A specifications is that the larger and more complex models confuse systems developers. We have no indications that the presence of user interface annotation has a negative effect. Thus, a natural enhancement of the FLUIDE-D specifications is to add user interface annotations. It is important that these annotations (as the concept models) are specified as part of the FLUIDE-A specifications, and a subset of them are propagated to the FLUIDE-D specification. The subset should be determined by which of the annotations that are indeed needed for specifying the user interface in FLUIDE-D.

FLUIDE follows a widely used principle in MDD of distinguishing between platform-independent and platform-specific specifications, which is also applied in most model-driven user interface development approaches. The conclusions in this report regarding comprehensibility may thus be generalized to other languages following this principle. This means that the platform-specific specifications are easier to understand than platform-independent ones, and to further enhance the understanding, platform-specific specifications should be self-contained, meaning that it should not be necessary to consider the platform-independent specifications to understand the platform-specific ones.

7 Related Work

As argued in the introduction, the most influential languages and approaches supporting model-based user interface development do not fully meet all the requirement we have identified. The same holds for OMG's IFML[3] standard (Brambilla and Fraternali, 2014). In particular, none of the assessed approaches meet the requirement of having compound building blocks, and they do not provide explicit support for the emergency response domain (related domains are though supported in ICOs (Navarre et al, 2009)).

Research on ICT solutions for emergency response focuses mainly on developing concrete systems, usually also addressing their user interface. There is also focus on modelling different aspects of emergency response, like tasks, procedures, organization and standards – often operationalized through ontologies. Furthermore, there is also some focus on development methods, but surprisingly little on frameworks and tools for developing emergency response applications. A noteworthy exception is the framework presented in (Fitrianie and Rothkrantz, 2009). This framework supports development of multimodal user interfaces. Even though UML class diagrams play a role in the framework, specifications are not expressed at an abstract level. The ISyCri project (Truptil et al., 2009) makes use of the distinction between platform-independent and platform-specific models in their ontology-based approach. Their focus is not on development, though, but rather on facilitating collaboration. Balasubramanian et al.'s (2005) domain-specific language for specifying emergency response systems does not support user interface development.

One of the original features of FLUIDE is the combination and coupling of user interface patterns and model patterns. Ahmed and Ashraf (2007) use patterns extensively, but focus on task and user interface patterns. Lin and Landay (2008) also use user interface patterns in their cross-device development tool, but they rely on correspondence between user interface elements on different platforms rather than abstractions. In MyUI, Peissner et al. (2012) make extensive use of patterns combined with state charts for their abstract specifications. 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. Trætteberg (2002) uses model patterns as part of his languages, but does not apply it to a view mechanism in the platform-specific specifications.

The graphical syntax in FLUIDE differs from most approaches for model-driven user interface development by integrating UML models into the specifications. Most such approaches build on or relates to the CAMELEON Framework (Calvary et al., 2003), which depicts a close connection between platform-independent specifications and concept models. Despite this, the role of the concept models differ, e.g. in UsiXML (Limbourg et al., 2004), the language supports specification of such

[3] www.ifml.org

models, but the connection to the specifications is through graph transformations, while MARIA (Paternò et al., 2009) only refers to elements from concept models in the user interface specifications.

8 Conclusions and Future Research

In this report we have presented the FLUIDE Framework, focusing on the FLUIDE specification languages. These languages offer a unique combination of features making them suited for specifying traditional emergency response applications as well as ready-to-use and highly configurable components. Such components support flexible composition of user interfaces for emergency responders.

We have used FLUIDE to specify the user interface of an application supporting management of unmanned vehicles in search and rescue operations. We have also conducted an experiment involving 29 systems developers with varying background to assess the comprehensibility of these specifications for systems developers not knowing FLUIDE. The main conclusions are:

- FLUIDE is well suited for specifying user interfaces in the search and rescue case
- Understanding FLUIDE specifications is highly achievable for systems developers
- It is possible to match FLUIDE specifications with user interfaces without fully understanding the specifications
- Our approach of using UML and task modelling notation as part of the FLUIDE notation enhances the understanding
- Competence in model-driven systems development increases understanding significantly
- To obtain an understanding of the match between a FLUIDE specification and the corresponding user interface, it is sufficient to use FLUIDE-D specifications
- User interface annotations from the FLUIDE-A specifications should be included in the FLUIDE-D specifications
- To ensure understanding of specifications from languages distinguishing between platform-independent and platform-specific specifications, self-contained platform-specific specifications should be used

Because the score level and share of correctly solved tasks in the experiment are at a satisfactory level given the conditions under which the experiment was conducted, we conclude that understanding FLUIDE specifications is highly achievable for systems developers. As a majority of the participants found it quite or very easy to grasp the essential part of the specifications, we claim that it is possible to match FLUIDE specifications with user interfaces without fully understanding the specifications.

Furthermore, our approach of using UML and task modelling notation as part of the FLUIDE notation enhances the understanding because a majority of the participants who reacted to the mix assessed it to enhance their understanding. As there are significant difference both in the task scores and the participants' perceived difficulty, we conclude that competence in model-driven systems development increases understanding.

To obtain an understanding of the match between a FLUIDE specification and the corresponding user interface, it is sufficient to use FLUIDE-D specifications because the scores for the tasks involving only FLUIDE-D specifications and the tasks involving both FLUIDE-A and FLUIDE-D specifications are on the same level, and because the FLUIDE-A specifications in certain tasks with particularly low scores may be perceived as confusing.

User interface annotations from the FLUIDE-A specifications should be included in the FLUIDE-D specifications because this will make the FLUIDE-D specifications more complete compared to the user interface it specifies, and because our analyses show that any possible negative influence from FLUIDE-A specifications are not caused by the annotations.

The conclusions in this report regarding understandability may be generalized to specifications from other languages using the principle of distinguishing between platform-independent and platform-specific specifications, i.e. that to ensure understandability self-contained concrete specifications should be used.

Among our planned future research is to complement the framework, including tool support and adaptation mechanisms, and conducting experiments where systems developers use the FLUIDE Framework to specify user interfaces.

9 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.

10 References

1. Ahmed, S. and Ashraf,,G. (2007) Model-based user interface engineering with design patterns. *Journal of Systems and Software 80(8)*, 1408-1422.
2. Balasubramanian, K. et al. (2005). A platform-independent component modeling language for distributed real-time and embedded systems. *Proc. 11th Real Time and Embedded Technology and Applications Symposium.* IEEE.
3. Brambilla, M., and Fraternali, P. (2014). Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML. *Morgan Kaufmann*.
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. (2003). A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computer 15 (3)*, 289–308.
5. Fitrianie, S. and Rothkrantz, L. (2009). Computed Ontology-based Situation Awareness of Multi-User Observations. *Proc. ISCRAM'09.* ISBN 978-91-633-4604-0.
6. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L. and López-Jaquero, V. (2004). USIXML: a language supporting multi-path development of user interfaces. *Proc. of EHCI-DSVIS'04.* Springer.
7. Lin, J. and Landay, J.A. (2008) Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. *Proc. of CHI'08.*
8. Navarre, D. et al. (2009). ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Trans. Computer-Human Interaction 16(4).*
9. Nilsson, E.G. and Stølen, K. (2010). Ad Hoc Networks and Mobile Devices in Emergency Response – a Perfect Match? *Proc. 2nd International Conference on Ad Hoc Networks*. Springer.
10. Nilsson, E.G. and Stølen, K. (2011). Generic functionality in user interfaces for emergency response. *Proc. OZCHI'11*. ACM.
11. Nilsson, E.G. and Stølen, K. (2016). The FLUIDE Framework for Specifying Emergency Response User Interfaces Employed to a Search and Rescue Case. *Proc. ISCRAM'16*. ISBN 978-84-608-7984-8.
12. Paternò, F., Santoro, C. and Spano, L.D. (2009). MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Computer-Human Interaction 16(4).*
13. Peissner, M., Häbe, D., Janssen, D. and Sellner, T. (2012). MyUI: generating accessible user interfaces from multimodal design patterns. *Proc. of EICS'12.*
14. Shadish, W.R., Cook, T.D. and Campbell, D.T. (2002). Experimental and Quasi-experimental Designs for Generalized Causal Inference. *Houghton Mifflin.*

15. Truptil, S., Benaben, F. and Pingaud, H. (2009). Collaborative process design for Mediation Information System Engineering. *Proc. ISCRAM'09.* ISBN 978-91-633-4604-0.

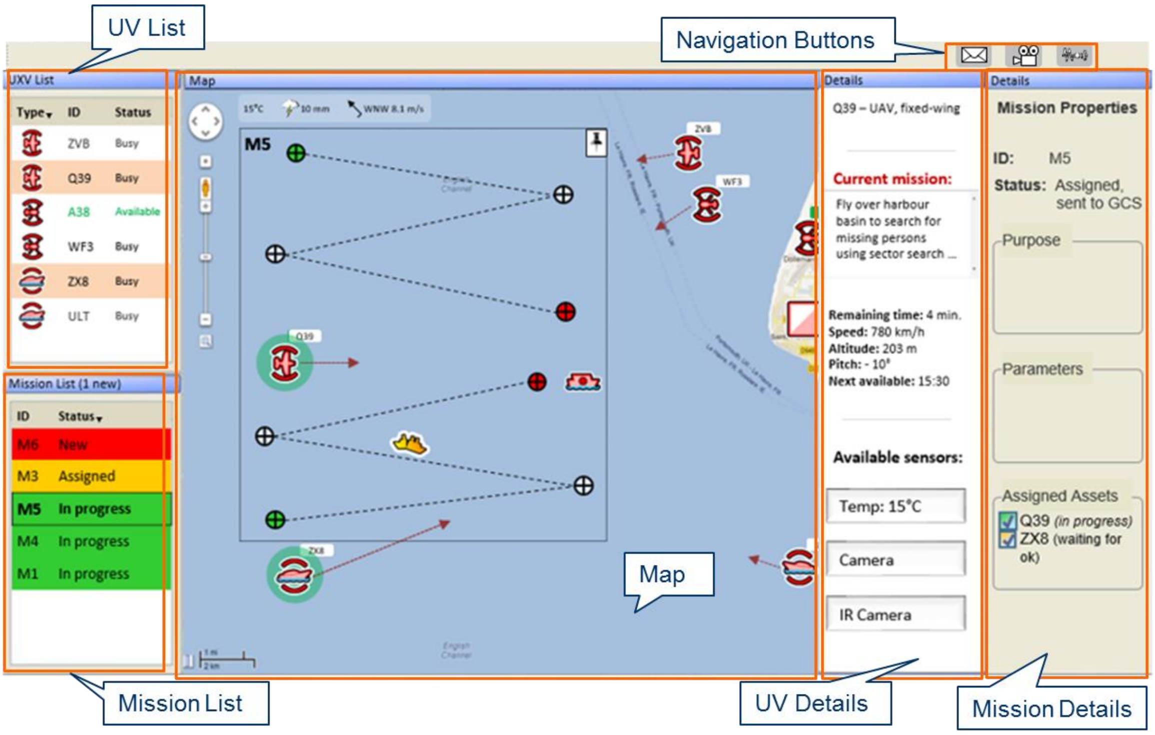
16. Trætteberg, H. (2002). *Model-based User Interface Design.* PhD thesis, NTNU.

Appendix A – Complete Specification of the Search and Rescue Case

In the case study, we retrospectively specified the user interface of an existing emergency application, i.e. the Generic Ground Station (GGS) application – which was developed as part of the research project DARIUS[4]. We denote this the *target user interface*. For the case study, specifying an already existing application has the advantages of realism and that we do not need to do user interface design from scratch. For the experiment, the case ensures realistic tasks. In this appendix we present the whole target user interface and the complete FLUIDE specification of this user interface.

The GGS application is designed for supporting emergency personnel managing unmanned vehicles (UVs) as part of an emergency response (search and rescue). The user of the application is located at a Generic Ground Station, and receives messages from command and control about incidents, missions and search areas. Based on this, the user assigns UVs to the mission and plans their search path. When the vehicles are operating, the GGS user monitors the UVs, including their positions, video feeds, images, sensor feeds, and observations. The UVs are either autonomous or operated by a dedicated operator, so the GGS user is not directly operating any UVs.

The user interface of the GGS application consists of a main map-based user interface and three pop-up windows presenting specialized information. The main user interface is a map display showing among other the locations of the incident, the GGS, the UVs, and the search areas. When a UV with a planned search path is selected, its search path as well as its trail is shown. This information is visualized using icons and graphics. In addition to the map display, the main user interface may show lists of and details for the UVs and missions in separate dockable panes. In Figure A.1, the map together with all lists and detail panes are shown.



UXV List

Type	ID	Status
	ZVB	Busy
	Q39	Busy
	A38	Available
	WF3	Busy
	ZX8	Busy
	ULT	Busy

Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress

Details

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

Available sensors:

Temp: 15°C

Camera

IR Camera

Details

Mission Properties

ID: M5
Status: Assigned, sent to GCS

Purpose

Parameters

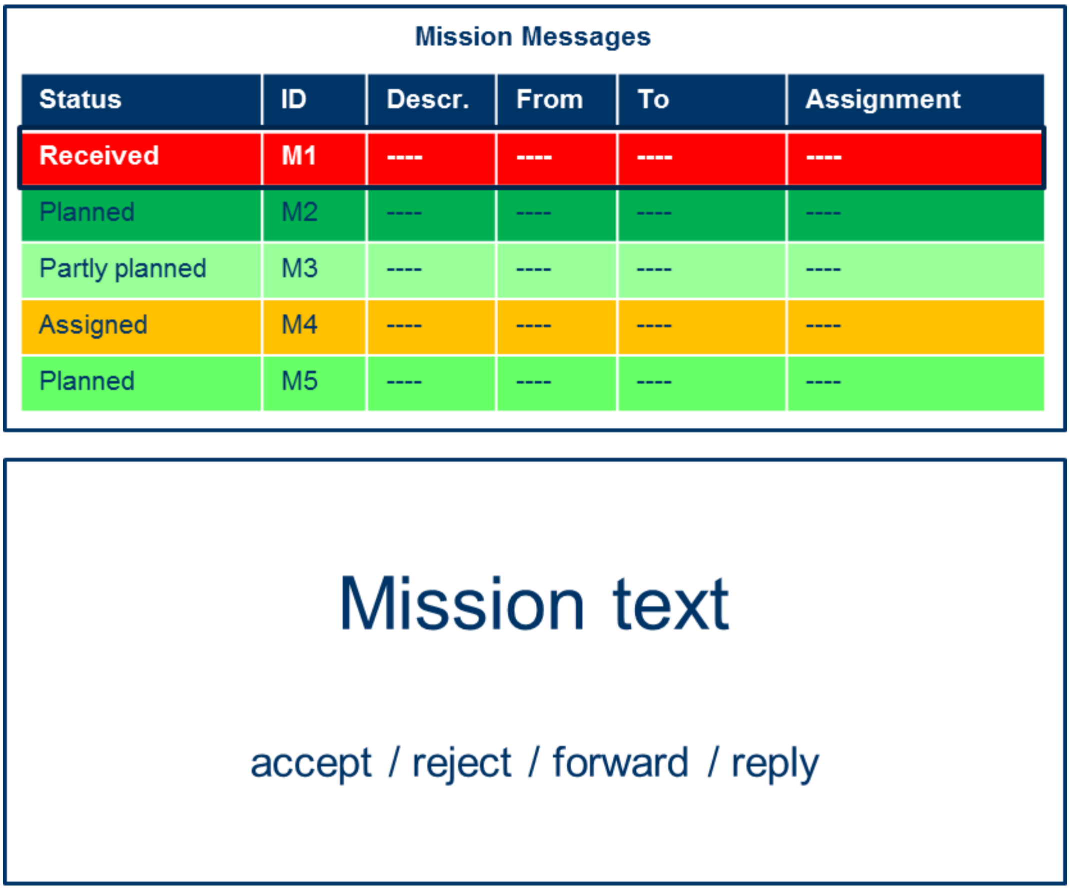
Assigned Assets

☑ Q39 *(in progress)*
☑ ZX8 (waiting for ok)

Figure A.1 – The main GGS user interface

[4] http://www.darius-fp7.eu/

In addition to the main user interface, the GGS application may also show three different pop-up windows. The first presents mission messages, as indicated in Figure A.2.



Mission Messages

Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

Mission text

accept / reject / forward / reply

Figure A.2 – The user interface for viewing mission messages

The second pop-up window presents video feeds from selected UVs, as indicated in Figure A.3.

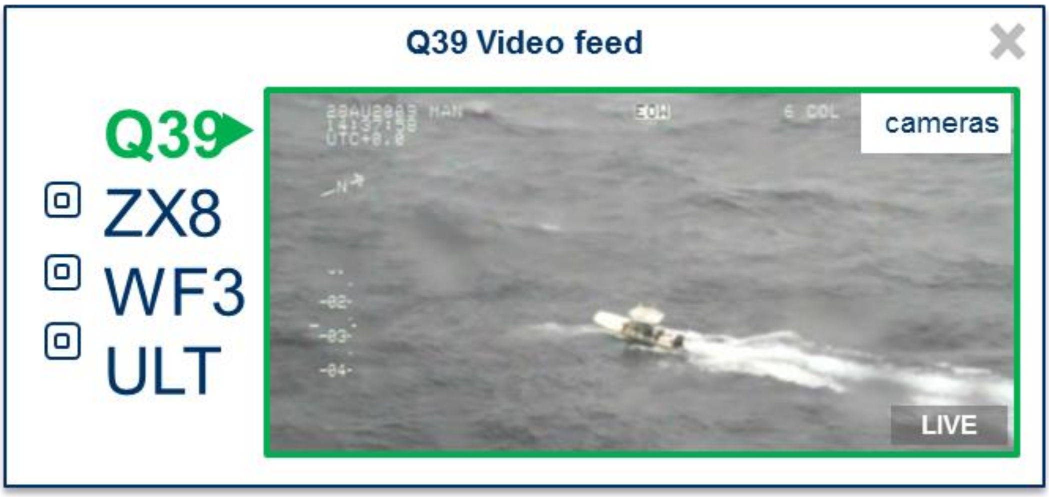


Figure A.3 – The user interface for viewing video feeds from UVs

Similarly, the third pop-up window presents sensor feeds from selected UVs, as indicated in Figure A.4.

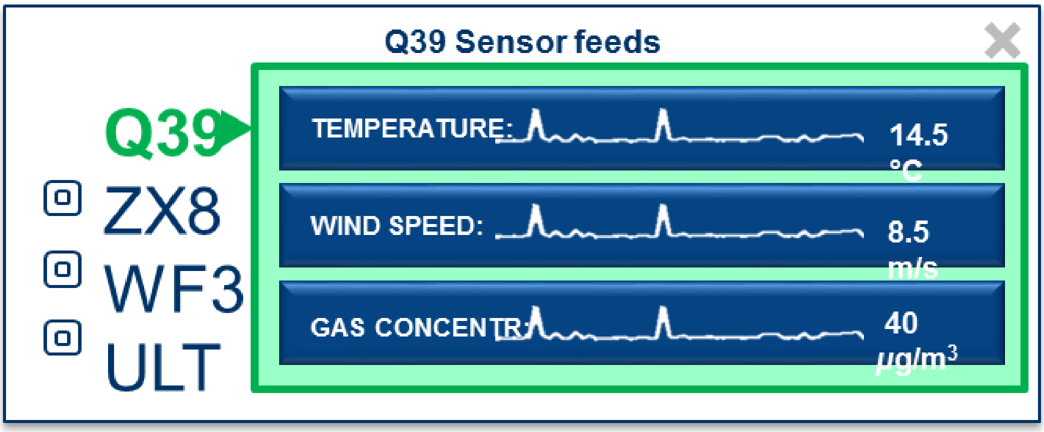


Figure A.4 – The user interface for viewing sensor feeds from UVs

In the following, we first present FLUIDE specifications of the different parts of the main GGS user interface and the three pop-up windows. Then we present how these specifications are coupled to specify the whole main GGS user interface, as well as how the main GGS user interface interplay with the three pop-up windows. FLUIDE-A and FLUIDE-D specifications are presented in separate sections.

A.1 FLUIDE-A specifications of individual parts of the main GGS user interface

In the FLUIDE-A specification of the individual parts of the main GGS user interface, there is one specification for each main concept in the user interface, i.e. Mission, UV, GGS, Incident, Observation, Weather, and Search Area.

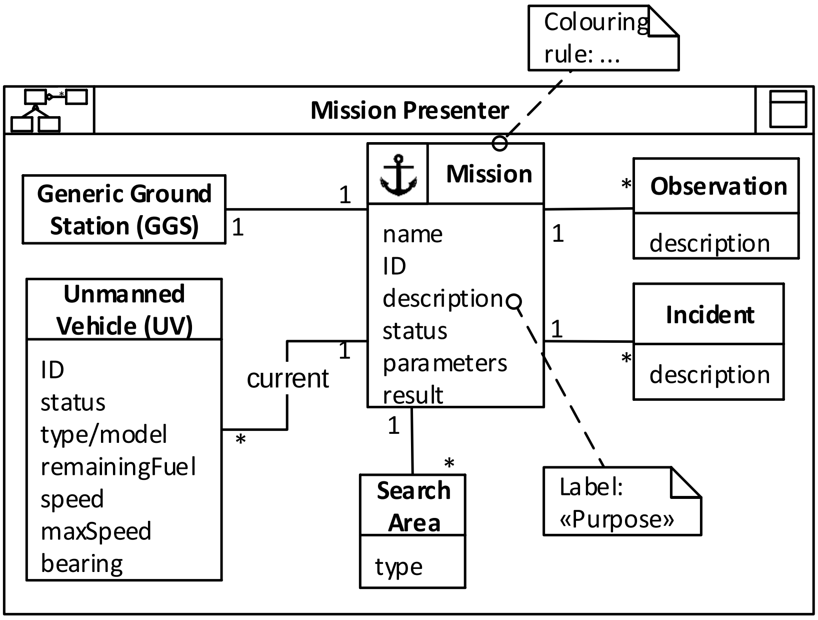


Figure A.5 – FLUIDE-A specification of Mission Presenter

In Figure A.5 we show *Mission Presenter*, which specifies Mission List and Mission Details pane (both shown in Figure A.1) in FLUIDE-A using the Content Presenter construct. 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 specification in Figure A.5 is a Basic Content Presenter). The graphical syntax of FLUIDE-A uses a subset of the UML class model notation (extended with the anchor symbol identifying the root entity of the model) to express the concept models. Additional platform-independent visual properties are expressed using annotations. The user interface annotations in Figure A.5 express rules for assigning colours to Missions (based on status) and a label.

In Figure A.6 we show *UV Presenter*, which specifies UV List and UV Details pane, as well as the visualization of the UVs on the map (all shown in Figure A.1) in FLUIDE-A.

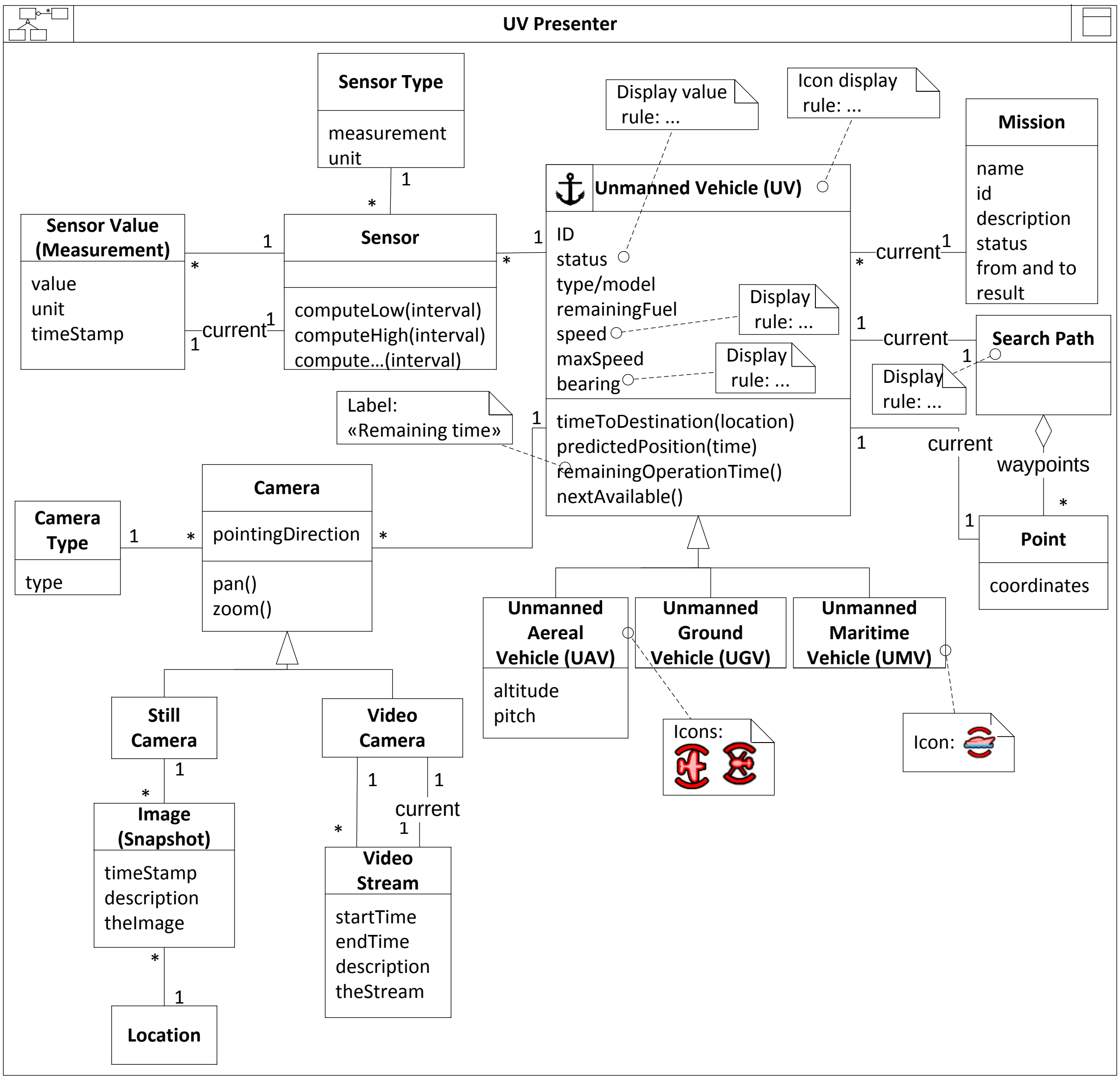


Figure A.6 – FLUIDE-A specification of UV Presenter

The user interface annotations in Figure A.6 include specifications of which icons to use for different types of UVs (given by annotations on the sub types of the entity *Unmanned Vehicle (UV))*.

The rest of the FLUIDE-A specifications for the main GGS user interface (Figure A.1) identify other types of information presented in the map. Figure A.7 shows the *GGS presenter*, specifying among other the icon for the Generic Ground Station. Figure A.8 and Figure A.9 give similar specifications for information about the incident and observations done during an operation.

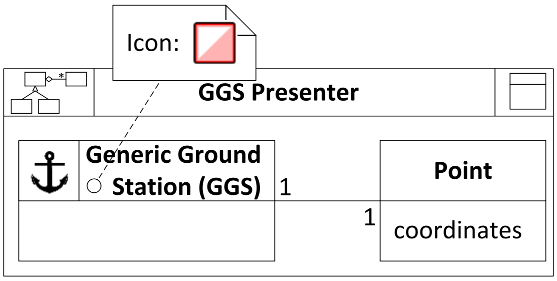


Figure A.7 – FLUIDE-A specification of GGS Presenter

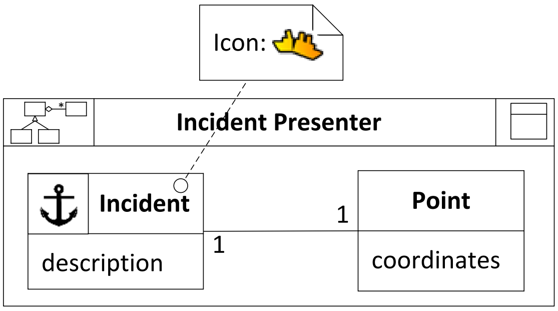


Figure A.8 – FLUIDE-A specification of Incident Presenter

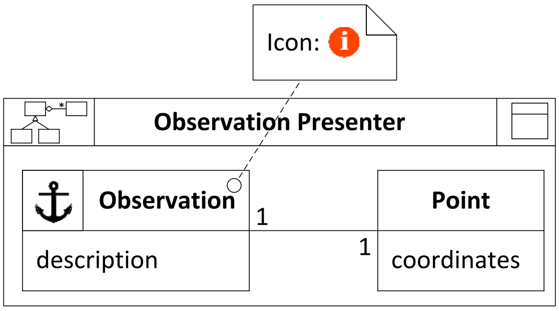


Figure A.9 – FLUIDE-A specification of Observation Presenter

A FLUIDE-A specification of the information that is needed for showing the weather icons on the top left of the map in the main GGS user interface (Figure A.1) is shown in Figure A.10.

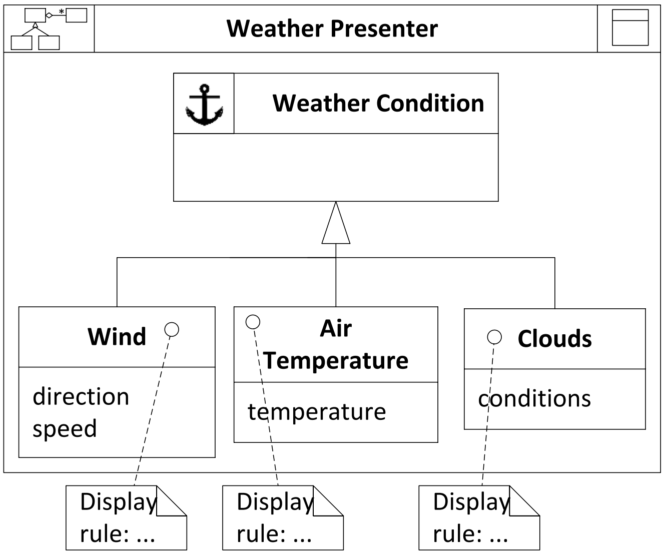


Figure A.10 – FLUIDE-A specification of Weather Presenter

A FLUIDE-A specification of the search area (the solid rectangle with the pin icon at the top right in the map part of Figure A.1) is shown in Figure A.11.

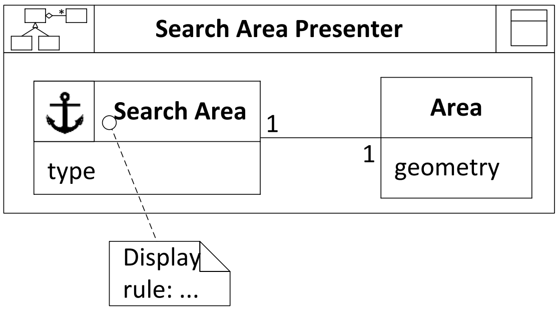


Figure A.11 – FLUIDE-A specification of Search Area Presenter

A.2 FLUIDE-D specifications of individual parts of the main GGS user interface

In the FLUIDE-D specification of the individual parts of the main GGS user interface, there is a varying number of specifications for the different FLUIDE-A specifications just presented.

There are two FLUIDE-D specifications refining the *Mission Presenter* (Figure A.5). Figure A.12 shows a FLUIDE-D Content Presenter Design specifying the Mission List, while Figure A.13 shows the design for Mission Details pane (both shown in Figure A.1). These two FLUIDE-D specifications use different subsets of the domain model in the corresponding FLUIDE-A specification in Figure A.5, both with regards to entities and attributes that are included.

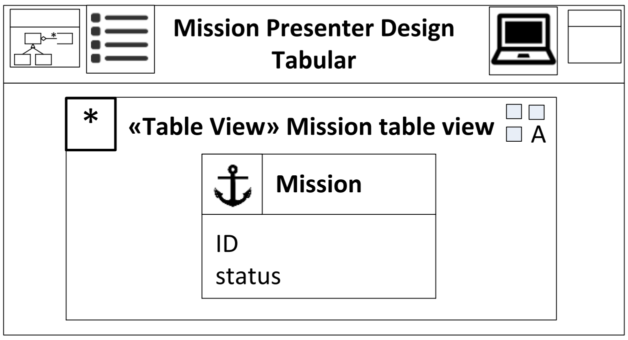


Figure A.12 – FLUIDE-D specification of the tabular version of Mission Presenter Design

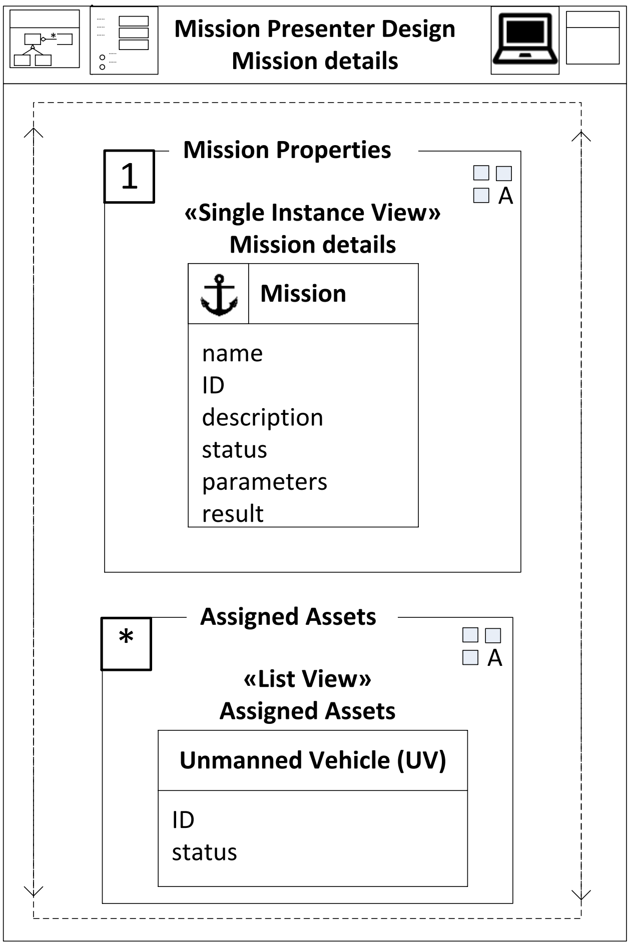


Figure A.13 – FLUIDE-D specification of the forms-based version of Mission Presenter Design

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 (PC for GGS). The content part contains the views that constitute the design. There are four main types of views, i.e. Layout Manager View, Decorational View, Content View and Content Integration View. The view shown in Figure A.12 is a Content View, i.e. a view that presents instances of one or more entities. Figure A.13 contains two Content Views. 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 many 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 it 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 generic Content View used in Figure A.12 (Table View) must contain one entity (possibly with subtypes) that determines the rows in the table *(Mission)*. It may also include related entities, as long as the cardinality on the side of the related entity is one. The Content Views used in Figure A.13 (Single Instance View and List View) use the same model pattern as Table View, even though Single Instance View presents one instance at the time. This design also uses a Layout Manager View. 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 (vertically in the example).

There are four FLUIDE-D specifications refining the *UV Presenter* (Figure A.6). Figure A.14 shows a FLUIDE-D Content Presenter Design specifying the UV List, Figure A.15 shows the design for UV Details pane (both shown in Figure A.1), Figure A.16 shows the specification of the map visualization of the UVs, while Figure A.17 shows the specification of the visualization of the search paths for the UVs.

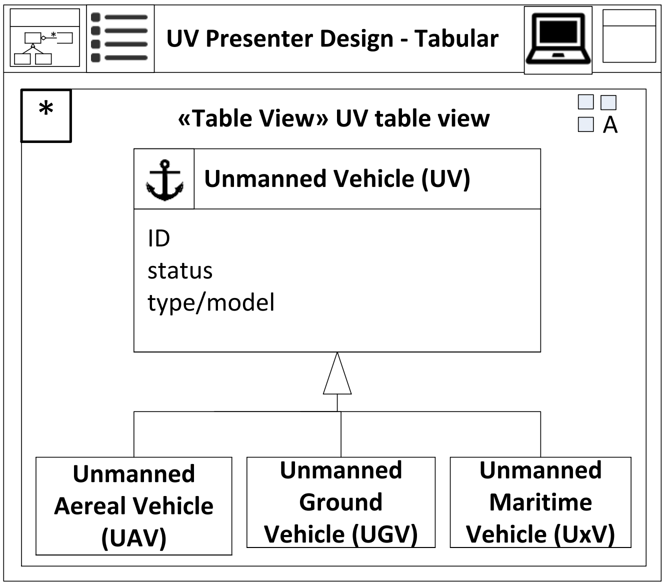

UV Presenter Design - Tabular
*
«Table View» UV table view
A
Unmanned Vehicle (UV)
ID
status
type/model
Unmanned Aereal Vehicle (UAV)
Unmanned Ground Vehicle (UGV)
Unmanned Maritime Vehicle (UxV)

Figure A.14 – FLUIDE-D specification of the tabular version of UV Presenter Design

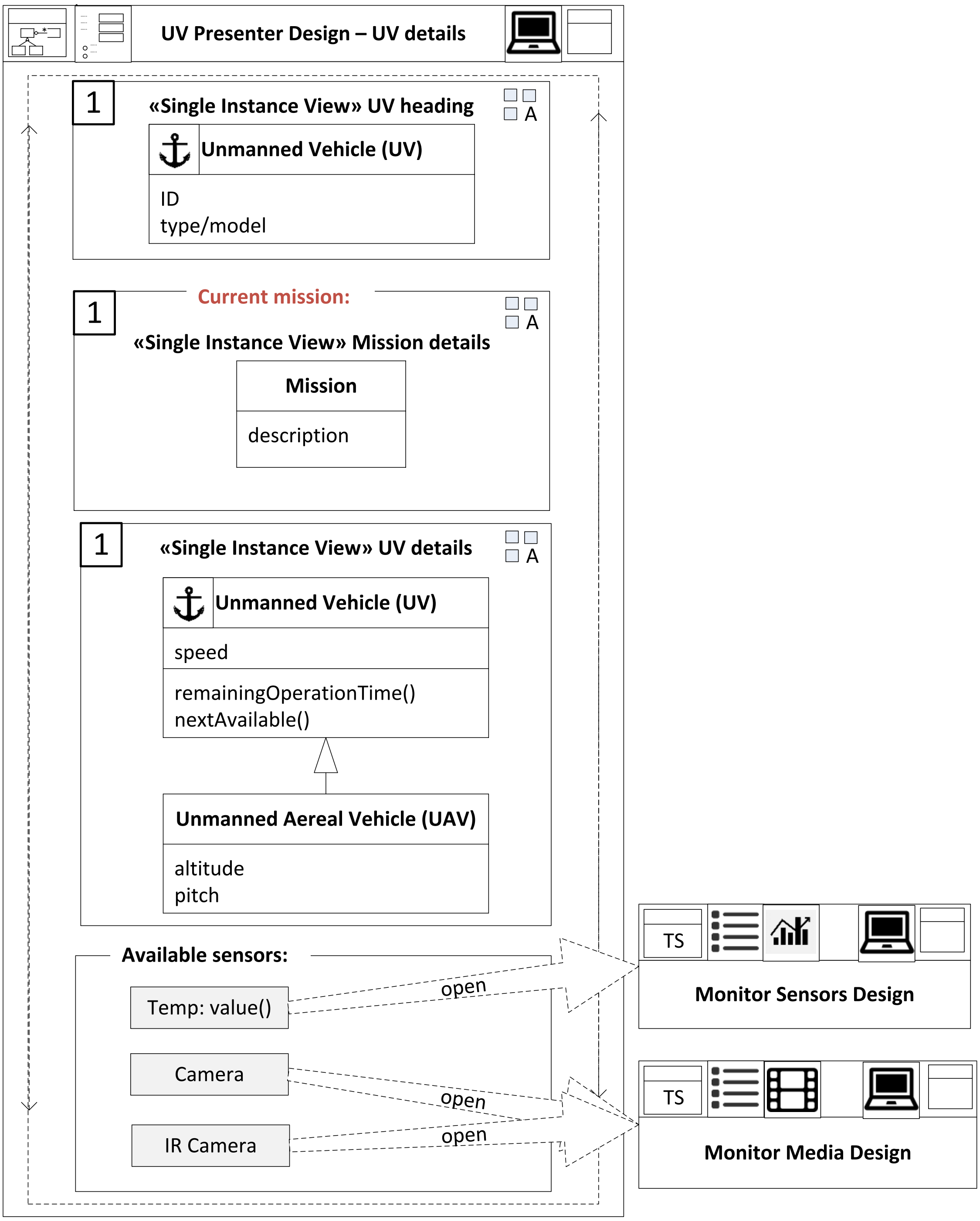


Figure A.15 – FLUIDE-D specification of the forms-based version of UV Presenter Design

The Basic Content Presenter in Figure A.15 specifies the intended dialog navigation to take place when the buttons are clicked. 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.

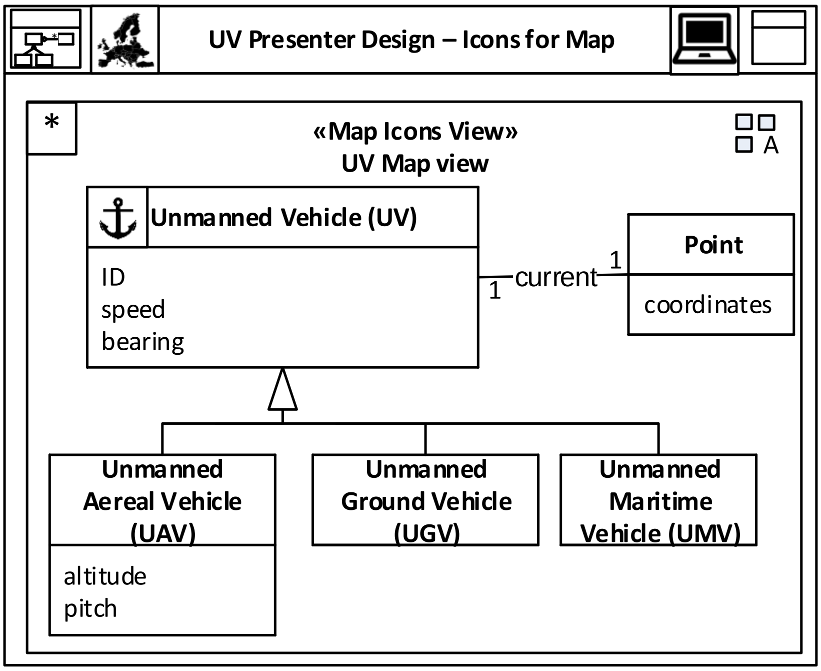

UV Presenter Design – Icons for Map
*
«Map Icons View»
UV Map view
A
Unmanned Vehicle (UV)
ID
speed
bearing
1
current
1
Point
coordinates
Unmanned Aereal Vehicle (UAV)
altitude
pitch
Unmanned Ground Vehicle (UGV)
Unmanned Maritime Vehicle (UMV)

Figure A.16 – FLUIDE-D specification of map presentation of UVs

The specification in Figure A.16 uses a map-based user interface style, shown by the icon on the left side in header. The design uses the domain-specific view Map Icons View. The model pattern for this view type 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 that is presented is one. Map Icons View visualizes one or more instances of the presented entities as icons on a map. If the presenter design using this view type (or other map-based Content Views) 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 (or another map-based Content View) does not have a parent providing a Map View, it will provide its own map.

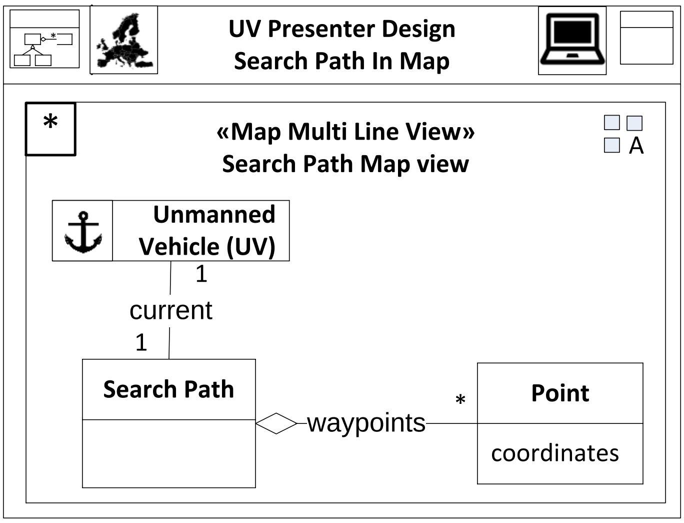


Figure A.17 – FLUIDE-D specification of the presentation of the UVs' search path

The specification in Figure A.17 is also map-based, but instead of showing icons, the Map Multi Line View presents a set of line segments determined by a set of points.

The other five FLUIDE-A specifications in Section A.1 have just one design each. All these designs specify map-based presentations, and are shown in Figure A.18-Figure A.22.

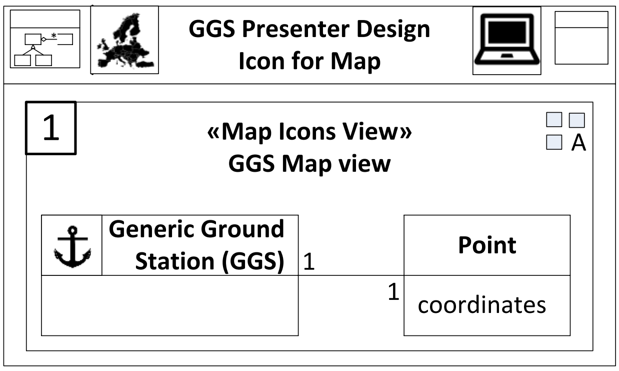


Figure A.18 – FLUIDE-D specification of map presentation of the GGS

As there is always one GGS, the cardinality of the view in Figure A.18 is one.

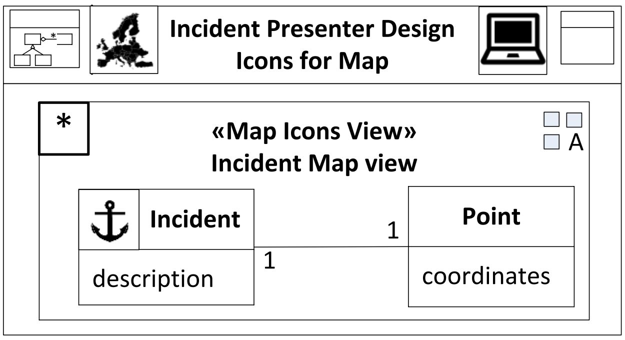


Figure A.19 – FLUIDE-D specification of map presentation of one or more incidents

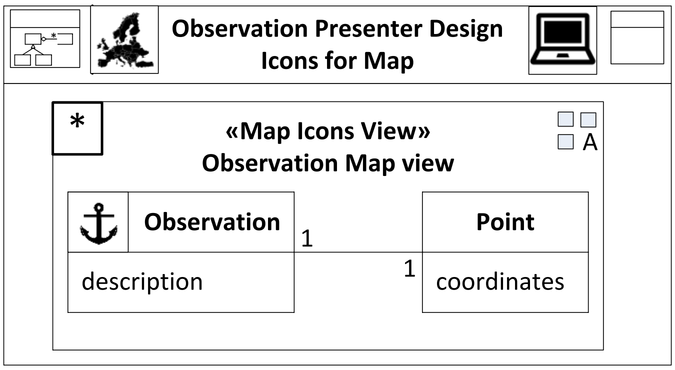


Figure A.20 – FLUIDE-D specification of map presentation of observations

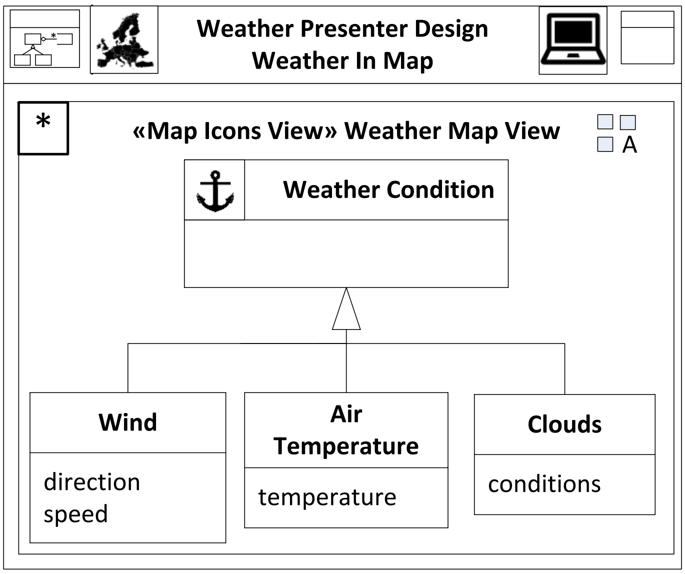


Figure A.21 – FLUIDE-D specification of map presentation of weather information

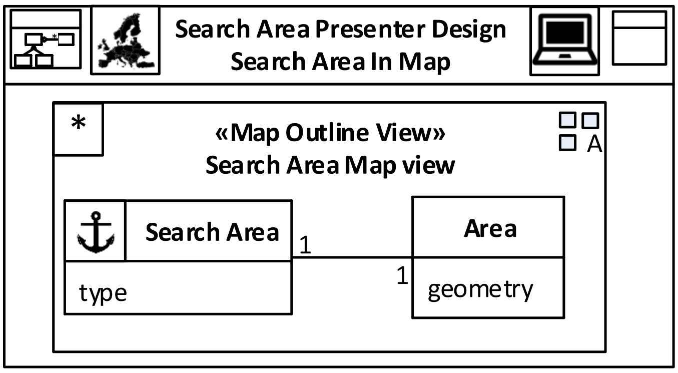


Figure A.22 – FLUIDE-D specification of map presentation of search area

While all the specifications in Figure A.18-Figure A.21 employ Map Icons Views, the *Search Area Presenter Design – Search Area In Map* in Figure A.22 uses the domain-specific view Map Outline View. This is a different map-based Content View using a similar model pattern as Map Icons View. The model pattern for Map Outline View must contain one entity (possibly with subtypes) that has a relation to a location entity (providing an area). It may also include related entities, as long as the cardinality on the side of the related entity that is presented is one. Map Outline View visualizes one or more instances using rectangles on a map.

A.3 FLUIDE-A specifications of the three pop-up windows

The FLUIDE-A specification in Figure A.6 *(UV Presenter)* includes all the information in the user interfaces in Figure A.3 (video feeds from UVs) and Figure A.4 (sensor feeds from UVs). This means that the only additional FLUIDE-A specification that is needed to specify the three pop-up windows is the *Message Reader* shown in Figure A.23.

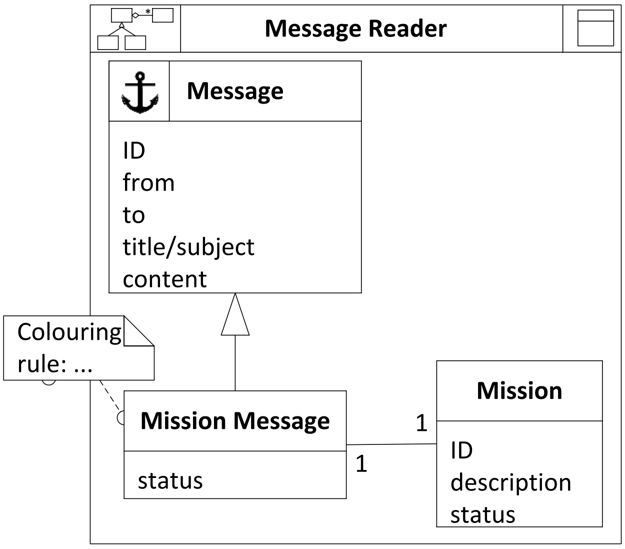


Figure A.23 – FLUIDE-A specification of Message Reader

This Basic Content Presenter contains all the information in the user interface for viewing mission messages (Figure A.2).

A.4 FLUIDE-D specifications of the three pop-up windows

There is one FLUIDE-D specification for each of the pop-up windows (Figure A.2-Figure A.4).

Figure A.24 shows a FLUIDE-D specification of the user interface for viewing mission messages (Figure A.2 – corresponding FLUIDE-A specification is shown in Figure A.23).

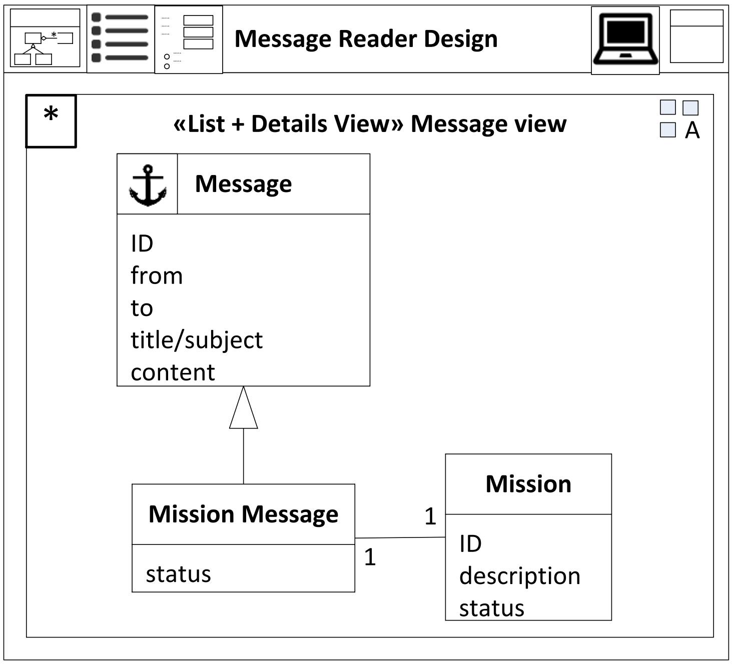


Figure A.24 – FLUIDE-D specification of the user interface for viewing mission messages

The Basic Content Presenter Design in Figure A.24 uses the generic view List + Details View. It presents many instances of one entity type in a list, and shows details for the instance selected in the list in a separate view. The model pattern for List + Details View must contain one entity (possibly with subtypes). It may also include related entities, as long as the cardinality on the side of the related entity that is presented is one, as the *Mission* entity in Figure A.24.

Figure A.25 shows a FLUIDE-D specification of the user interface for viewing video feeds from UVs (Figure A.3 – corresponding FLUIDE-A specification is shown in Figure A.6).

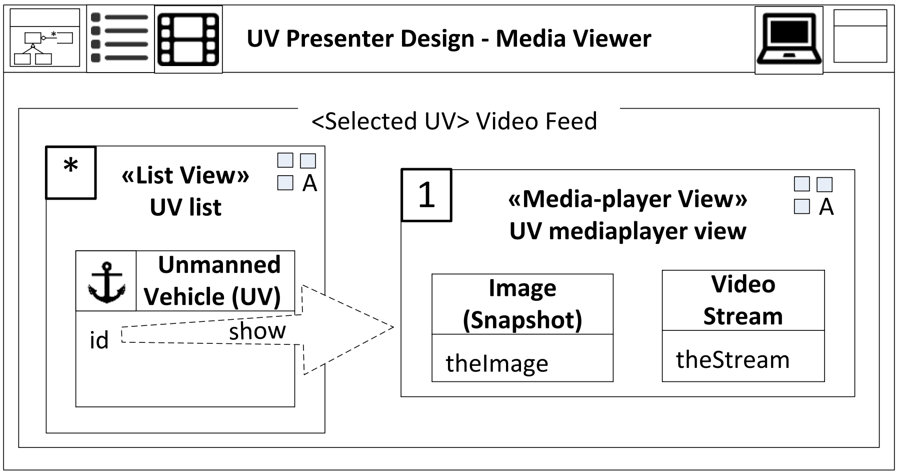


Figure A.25 – FLUIDE-D specification of the user interface for viewing video feeds from UVs

The Basic Content Presenter Design in Figure A.25 applies list-based and multimedia user interface styles. It uses two generic views (List View and Media-player View). The design also specifies the intended dialog navigation to take place when a UV identifier is selected in the list.

Figure A.26 shows a FLUIDE-D specification of the user interface for viewing sensor feeds from UVs (Figure A.4– corresponding FLUIDE-A specification is shown in Figure A.6). The Basic Content Presenter Design in Figure A.26 applies list- and graph-based user interface styles. In addition to the generic view List View, it also uses the domain-specific view Sensor Feeds View. The dialog navigation is similar to the one used in the design in Figure A.25.

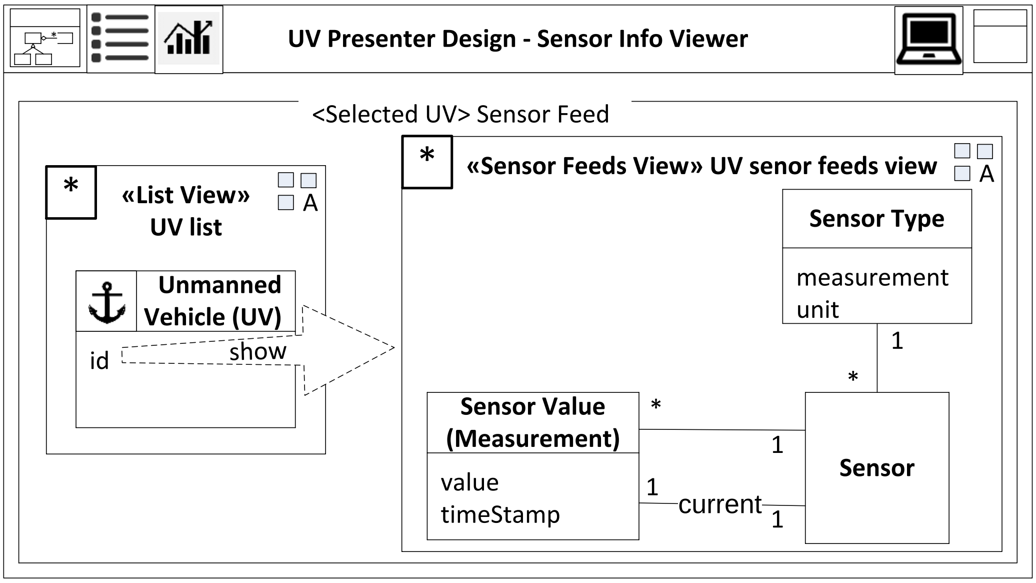

UV Presenter Design - Sensor Info Viewer
<Selected UV> Sensor Feed
*
«List View»
UV list
□ □
□ A
Unmanned Vehicle (UV)
id
show
*
«Sensor Feeds View» UV senor feeds view
□ □
□ A
Sensor Type
measurement
unit
1
*
Sensor Value (Measurement)
value
timeStamp
*
1
1
current
1
Sensor

Figure A.26 – FLUIDE-D specification of the user interface for viewing sensor feeds from UVs

A.5 FLUIDE-A specifications of the coupling of the different parts of the GGS user interface

To specify how the different types of information presented in the Map part of Figure A.1 is integrated, several of FLUIDE-A's aggregation mechanisms are used. As the same information may be useful when solving different tasks, and because different tasks may require different subsets of the same information, Content Presenters may be specified hierarchically.

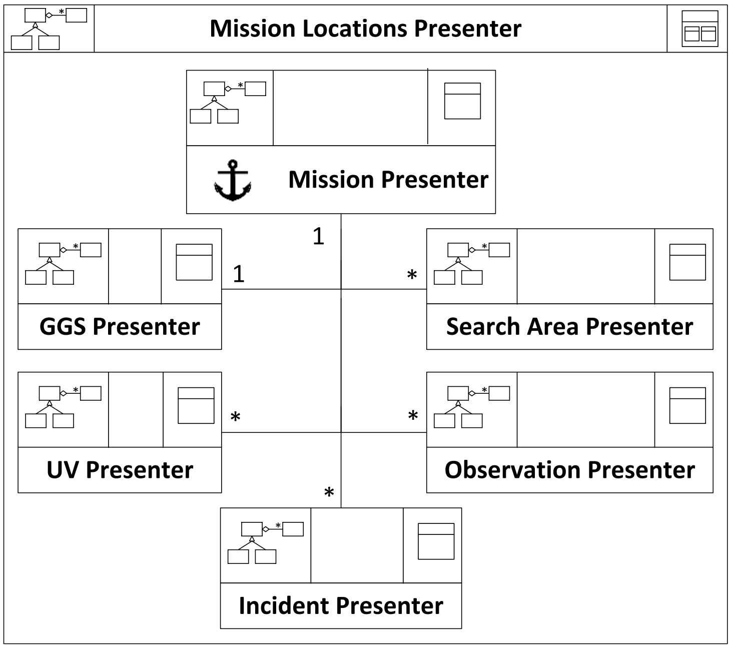


Figure A.27 – FLUIDE-A specification of Mission Locations Presenter

The hierarchical structure is specified using the Aggregated Content Presenter construct that aggregates other Content Presenters recursively. All mission related information that is presented in the map is specified using the Aggregated Content Presenter *Mission Locations Presenter* shown in Figure A.27.

The content part of aggregated presenters referring to other content presenters, includes only the border parts of their members, and shows the members' names inside the presenter instead of in the heading. The anchor symbol indicates which of the member presenter the aggregated one inherits its anchor from.

To include all the content in the map, an additional specification is needed, i.e. the Task Supporter *Use Map* shown in Figure A.28. It aggregates the *Mission Location Presenter* just discussed and the Content Presenter *Weather Presenter* (Figure A.10).

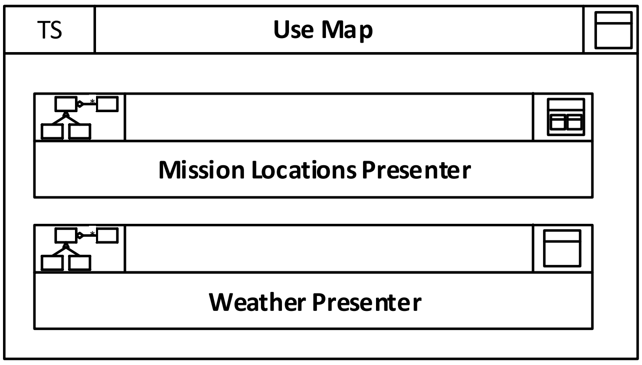


Figure A.28 – FLUIDE-A specification of the Use Map task support

The Task Supporter in Figure A.28 integrates one Basic and one Aggregated Content Presenter.

There are also several other Task Supporters, wrapping one or more Content Presenter(s), shown in Figure A.29-Figure A.33. These are among other needed by the Work Supporters specified below.

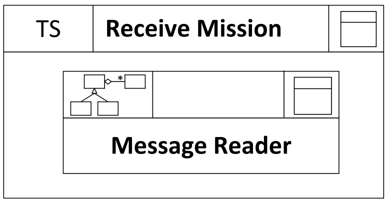


Figure A.29 – FLUIDE-A specification of the Receive Mission task support

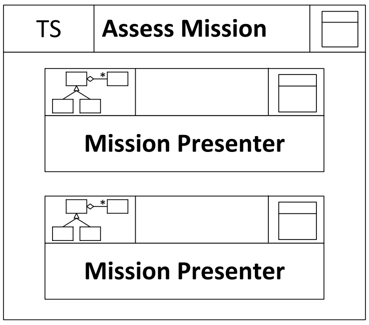


Figure A.30 – FLUIDE-A specification of the Assess Mission task support

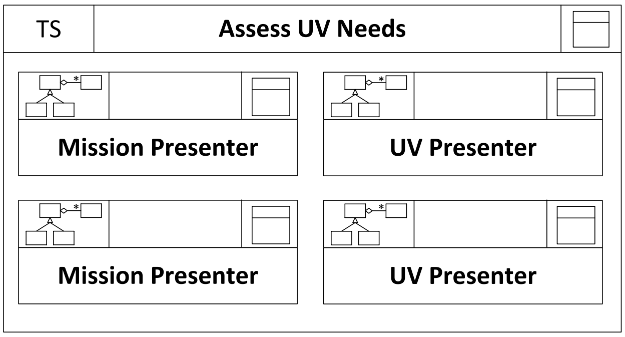


Figure A.31 – FLUIDE-A specification of the Assess UV Needs task support

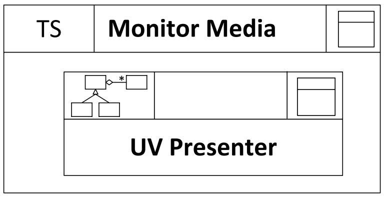


Figure A.32 – FLUIDE-A specification of the Monitor Media task support

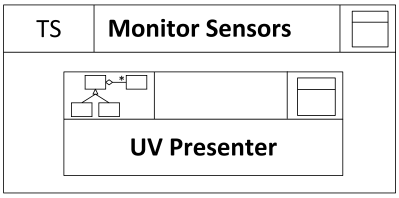


Figure A.33 – FLUIDE-A specification of the Monitor Sensors task support

To specify how these supported tasks contribute to different parts of the work of the user, five Work Supporters are specified. The most complex of these, the *Plan Mission Supporter* is shown in Figure A.34.

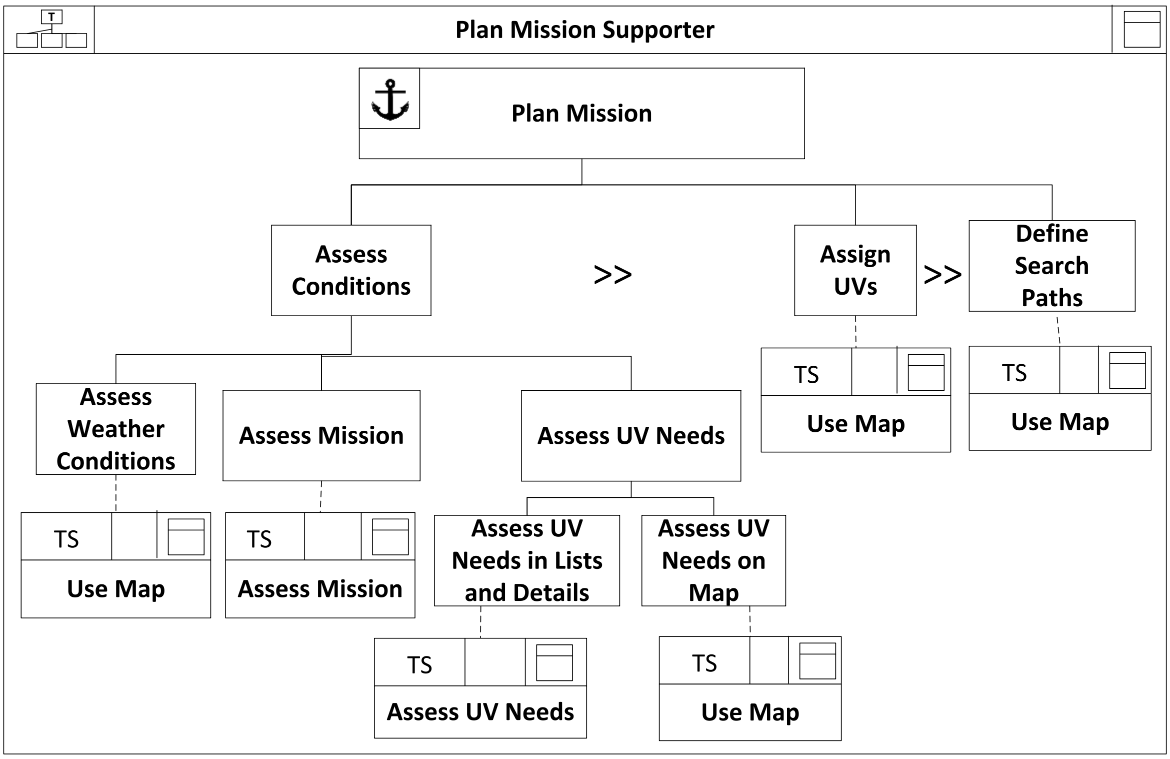


Figure A.34 – FLUIDE-A specification of the Plan Mission Supporter work supporter

Each task in a Work Supporter may have a connected Task Supporter. There are nine tasks organized in four levels in the task model in the *Plan Mission Supporter*, of which all the leaf tasks have Task Supporters. The operator “>>” indicates sequence in task performance.

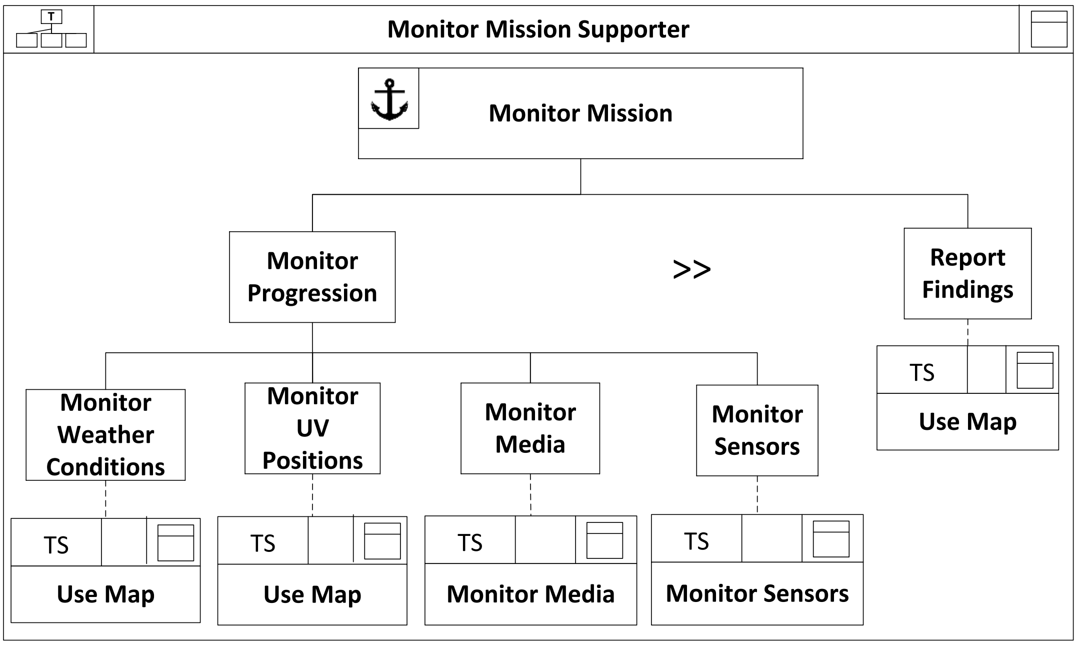


Figure A.35 – FLUIDE-A specification of the Monitor Mission Supporter

There is a similar Work Supporter for the tasks involved in monitoring a mission, i.e. the *Monitor Mission Supporter* shown in Figure A.35.

The Work Supporters in Figure A.36 and Figure A.37 are wrappers for Task Supporters to allow them to be members of the Aggregated Work Supporter *Manage Missions Supporter* (Figure A.38).

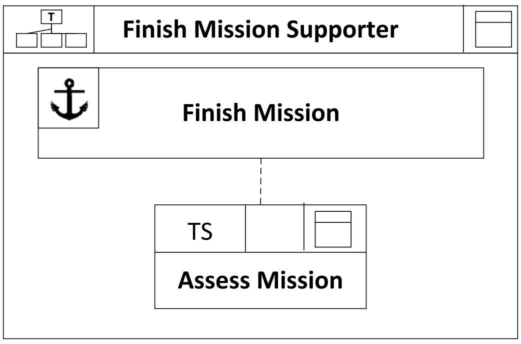


Figure A.36 – FLUIDE-A specification of the Finish Mission Supporter

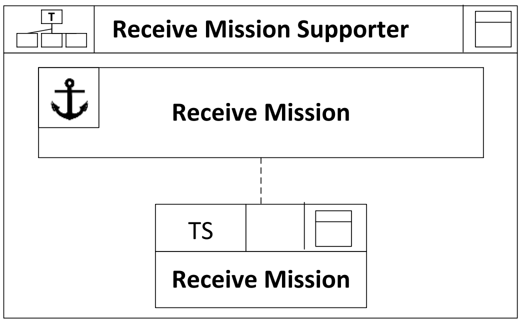


Figure A.37 – FLUIDE-A specification of the Receive Mission Supporter

The structure of the work supported by the four Basic Work Supporters just presented is specified in the Aggregated Work Supporter *Manage Missions Supporter* (Figure A.38). This work supporter specifies the whole GGS application.

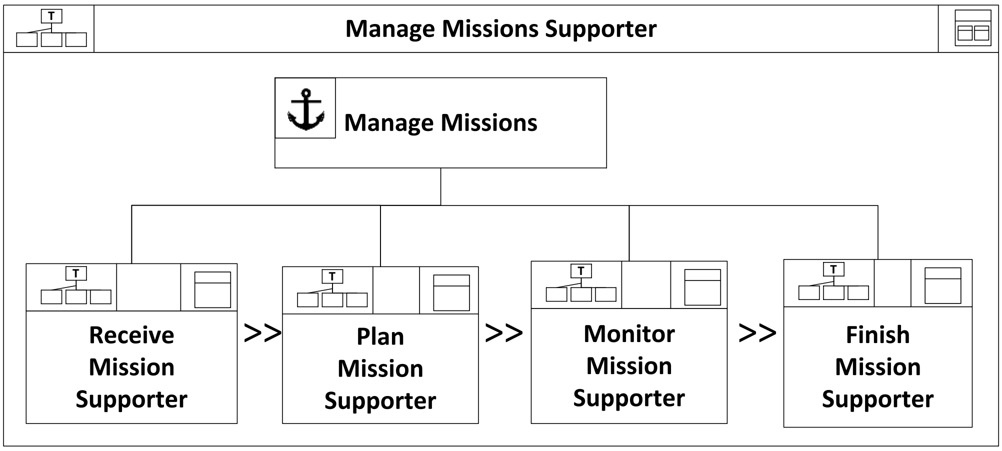

T
Manage Missions Supporter
Manage Missions
T
Receive Mission Supporter
>>
T
Plan Mission Supporter
>>
T
Monitor Mission Supporter
>>
T
Finish Mission Supporter

Figure A.38 – FLUIDE-A specification of the Manage Missions Supporter

An Aggregated Work Supporter must add exactly one task (possibly with a Task Supporter) on the level above the member supporters.

A.6 FLUIDE-D specifications of the coupling of the different parts of the GGS user interface

A design for *Mission Locations Presenter* (Figure A.27) is shown in Figure A.39.

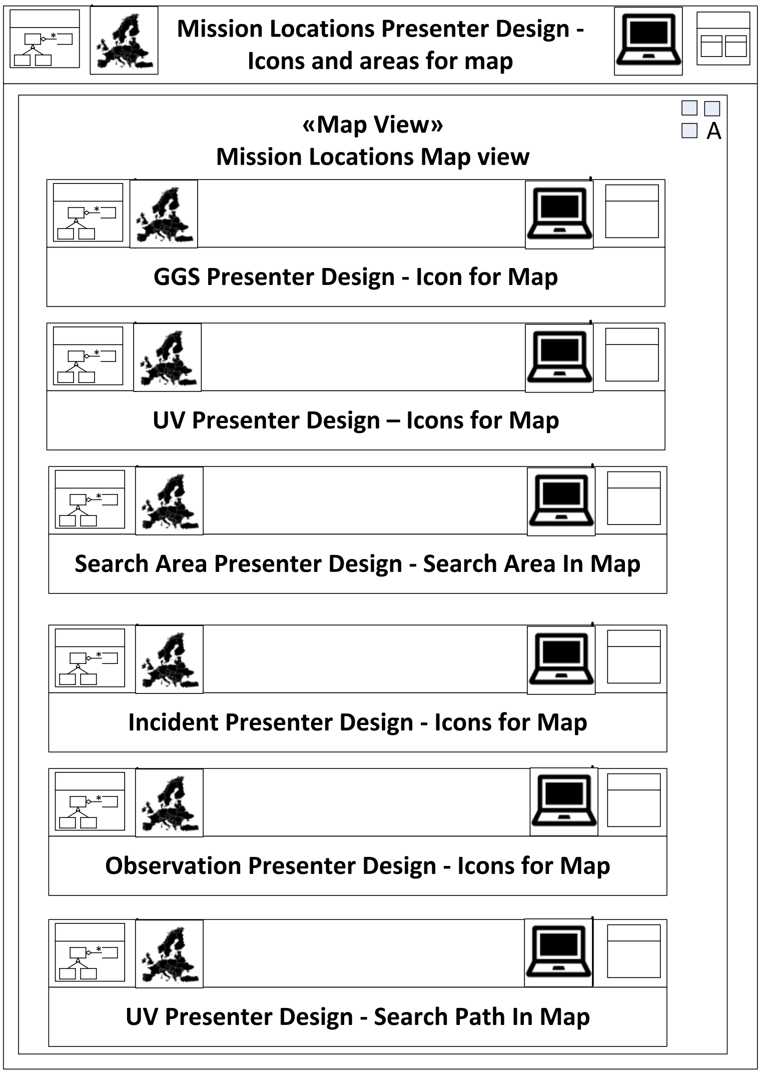

Mission Locations Presenter Design - Icons and areas for map
«Map View»
Mission Locations Map view
A
GGS Presenter Design - Icon for Map
UV Presenter Design – Icons for Map
Search Area Presenter Design - Search Area In Map
Incident Presenter Design - Icons for Map
Observation Presenter Design - Icons for Map
UV Presenter Design - Search Path In Map

Figure A.39 – FLUIDE-D specification of a map-based design for the Mission Locations Presenter

The Aggregated Content Presenter Design in Figure A.39 contain a Content Integration View of the type Map View. Content Integration Views integrate related content from different interactor designs. Content Integration Views require that their member presenter designs use specific Content Views or Content Integration Views. Among the views that may be aggregated into Map Views are Map Icons View and Map Outline View used in the Content Presenter Designs in Figure A.16 and Figure A.22, as well as other Map Views.

The design for the Task Supporter *Use Map* (Figure A.28) is shown in Figure A.40.

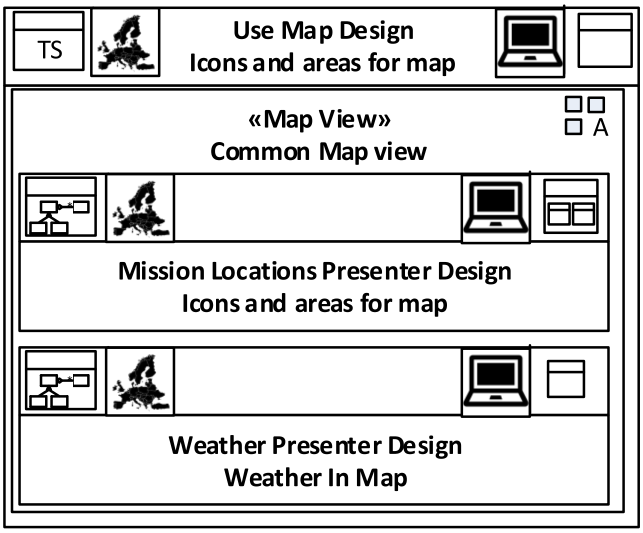


Figure A.40 – FLUIDE-D specification of a map-based design for the Use Map task supporter

The Aggregated Content Presenter Design in Figure A.40 also uses a Map View.

The design for the Task Supporter *Receive Mission* (Figure A.29) is shown in Figure A.41.

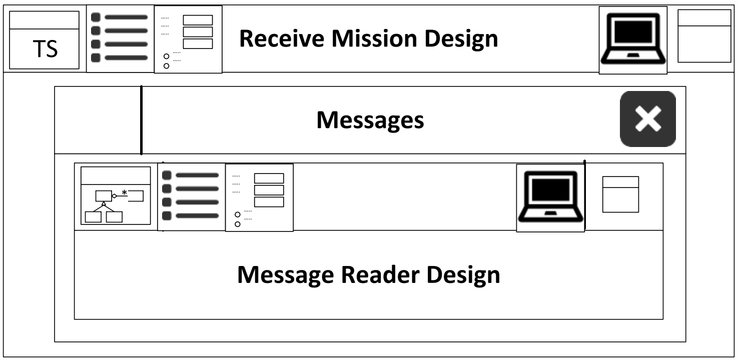


Figure A.41 – FLUIDE-D specification of a design for the Receive Mission task supporter

The Task Supporter Design in Figure A.41 uses a decorational 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.

There is no need for designs for the Task Supporters in Figure A.30 and Figure A.31, as the designs of their member Content Presenters will be used directly in the design for the Aggregated Work Supporter *Manage Missions Supporter* (Figure A.38). This design is shown below in Figure A.44.

The design for the Task Supporter *Monitor Media* (Figure A.32) is shown in Figure A.42.

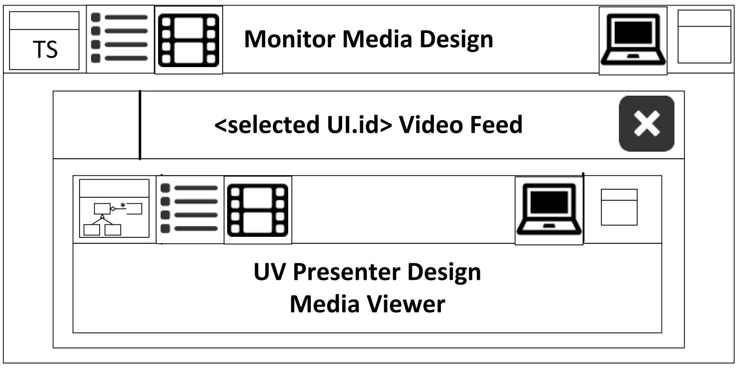


Figure A.42 – FLUIDE-D specification of a design for the Monitor Media task supporter

The heading of the window type decorational view in the Task Supporter Design in Figure A.42 uses a dynamic value collected from a member Content Presenter.

The design for the Task Supporter *Monitor Sensors* (Figure A.33) is shown in Figure A.43.

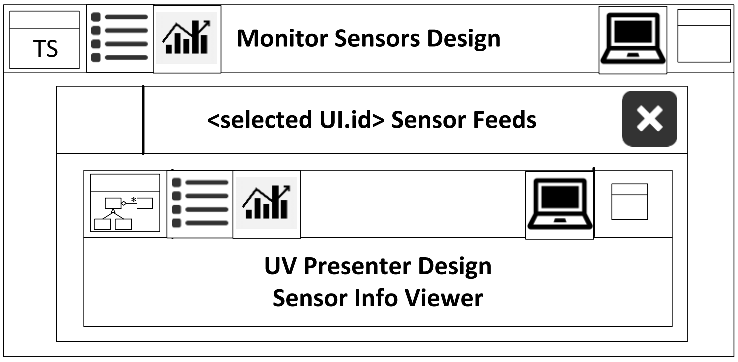


Figure A.43 – FLUIDE-D specification of a design for the Monitor Sensors task supporter

There is no need for designs for any of the Basic Work Supporters (Figure A.34-Figure A.37), as the designs of their member Task Supporters will be used directly in the design for the top level Aggregated Work Supporter *Manage Missions Supporter* (Figure A.38). This design is shown in Figure A.44.

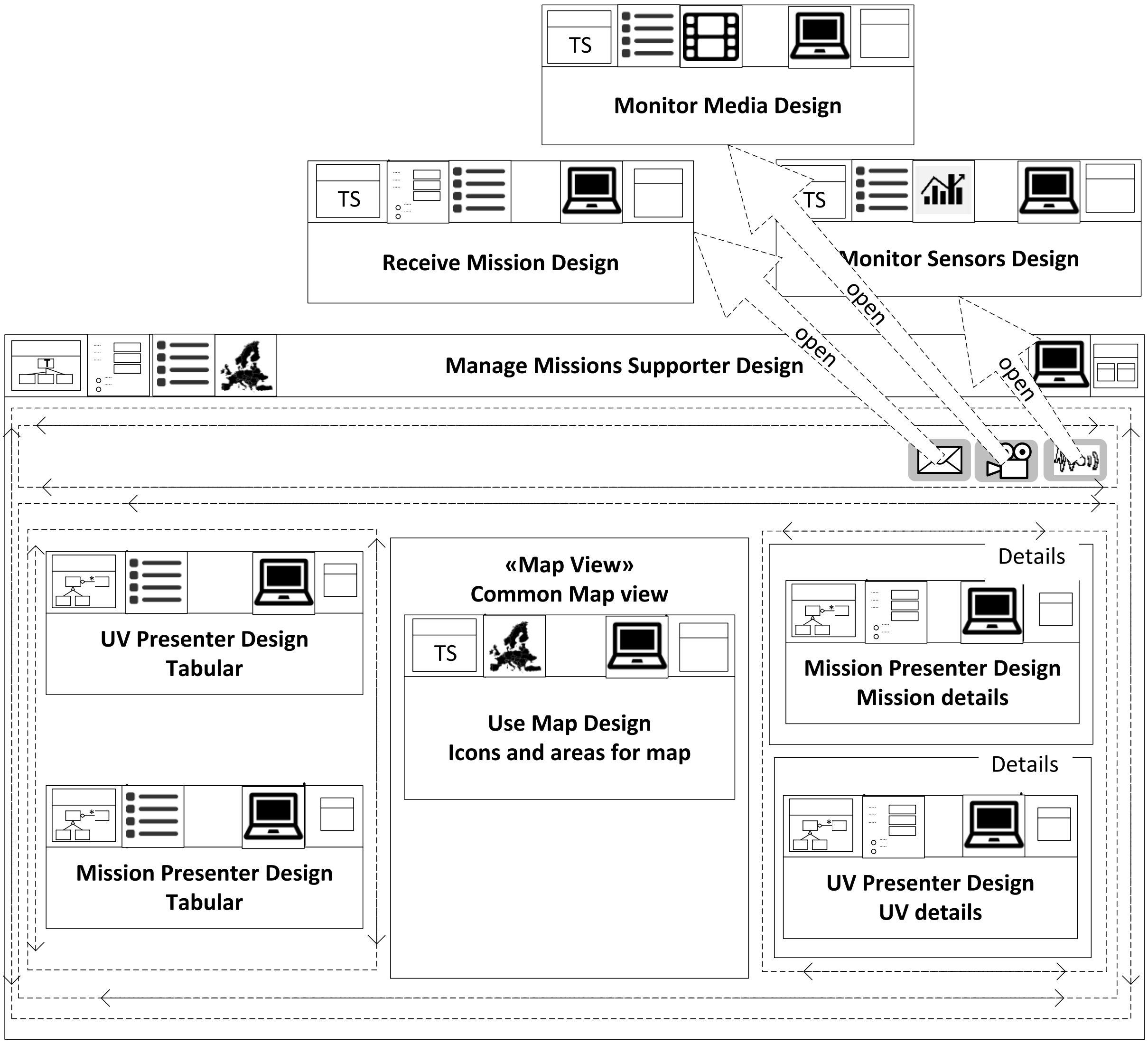


Figure A.44 – FLUIDE-D specification of the design for the Manage Mission Supporter

In addition to five Layout Manager Views, the Aggregated Work Supporter Design in Figure A.44 uses two standard Decorational Views to specify the border and heading ("Details") for the details panes on the right-hand side of the user interface. All designs having children exploit the sum of styles and modality/platforms of their children (and their children recursively). The design in Figure A.44 also specifies the navigation structure between the main user interface and the pop-up windows. It specifies the design for the user interface in Figure A.1, collects the designs for the user interfaces in Figure A.2-Figure A.4, and specifies the navigation structure between these user interfaces.

Appendix B – The tasks solved by the participants in the experiment

This appendix contains the complete set of user tasks used in the experiment, including the forms used for background information and for rating the experience. Note that we have made some small adjustments to the user interfaces and specifications after the experiment was conducted (mainly layout adjustments). This means that the user interfaces and specifications in Appendix A and Appendix B might be slightly different.

Background information

Gender: Male Female

Age: ....

Occupation: Student Research scientist Other: ....

Education level: Bachelor Master PhD

Educational background: Informatics Psychology Other: ....

Programme/direction: User interface design Software Engineering Other..................

Knowledge of UML class models:

No knowledge	Minor knowledge	Some knowledge	Good knowledge	Expert

Experience in using UML class models (number of years):.....................

Knowledge of task modelling:

No knowledge	Minor knowledge	Some knowledge	Good knowledge	Expert

Experience in using tasks modelling (number of years):.....................

Knowledge of user interface design and usability:

No knowledge	Minor knowledge	Some knowledge	Good knowledge	Expert

Knowledge of emergency response work:

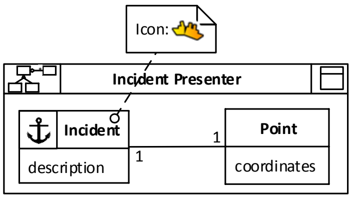
No knowledge	Minor knowledge	Some knowledge	Good knowledge	Expert

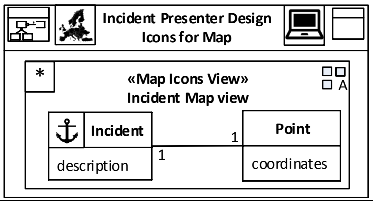
Formal background or experience within model-driven development (Yes/No):....................

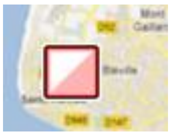
Email address (will only be used for contacting the winner):.......................

Task 1.1

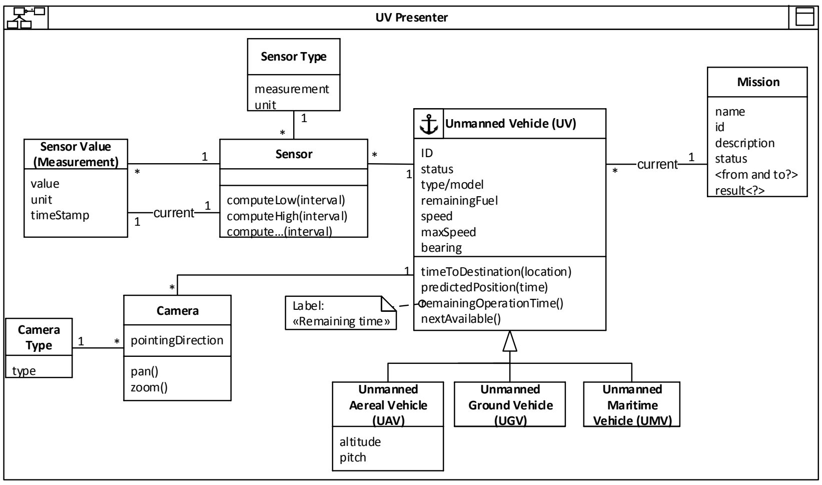
Your task is to judge whether a pair of FLUIDE-A and FLUIDE-D specifications (on the left) and a user interface (on the right) match. The alternatives are: yes, partly, no.

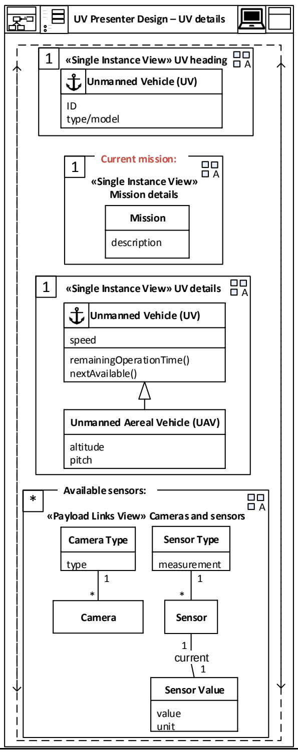


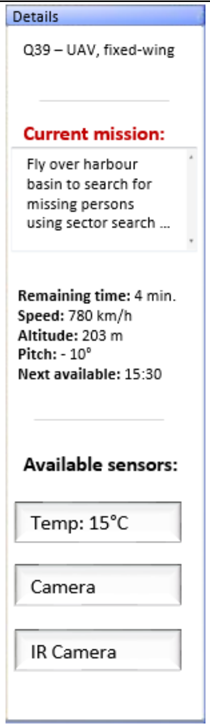




The specification and user interface match:	Yes	Partly	No







Details

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

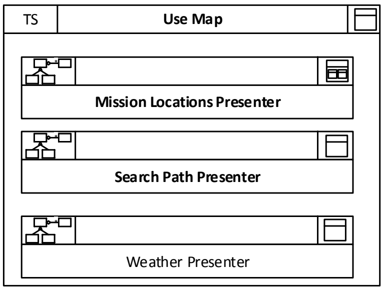
Available sensors:

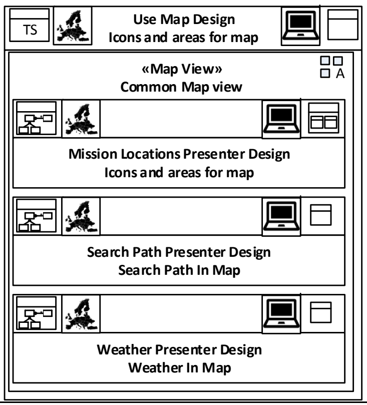
Temp: 15°C

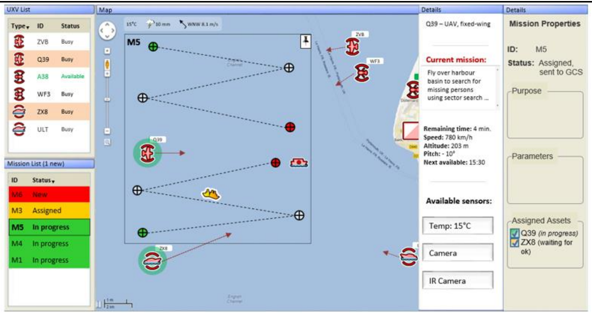
Camera

IR Camera

The specification and user interface match:	Yes	Partly	No







UXV List

Type	ID	Status
	ZVB	Busy
	Q39	Busy
	A38	Available
	WF3	Busy
	ZX8	Busy
	ULT	Busy

Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress

Map

15°C 10 mm WNW 8.1 m/s

M5

Details

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

Available sensors:

Temp: 15°C

Camera

IR Camera

Details

Mission Properties

ID: M5

Status: Assigned, sent to GCS

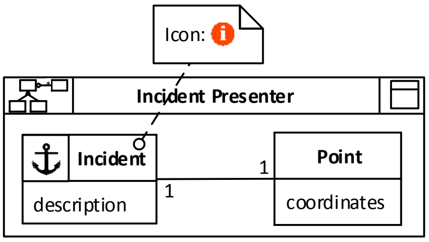
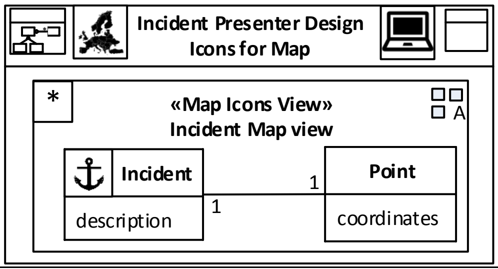
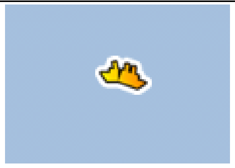
Purpose

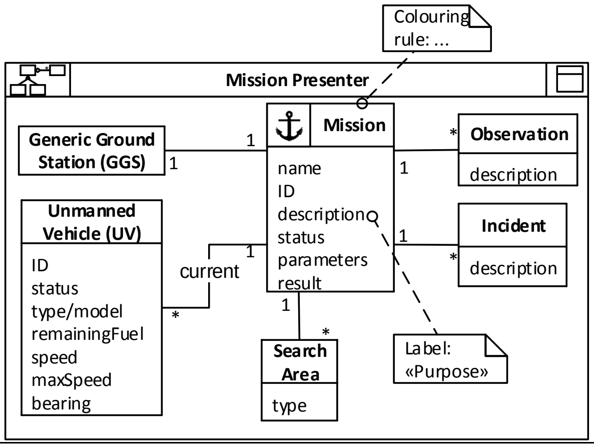
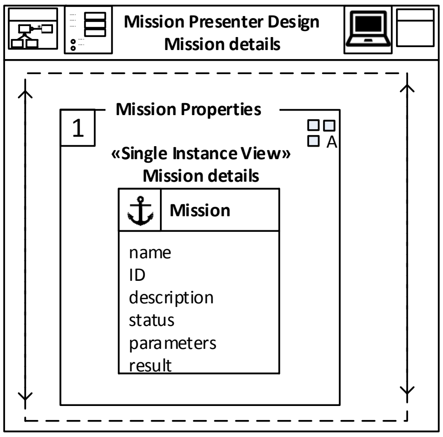
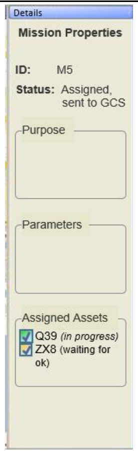
Parameters

Assigned Assets

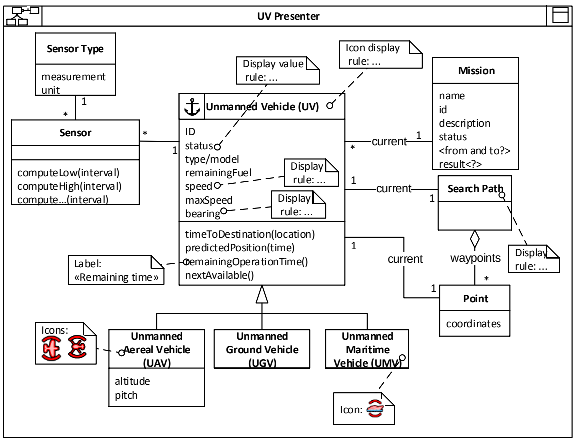
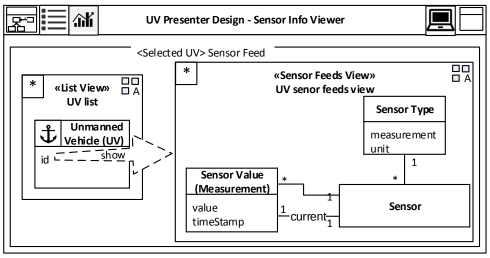
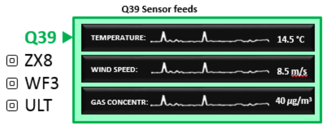
☑ Q39 *(in progress)*
☑ ZX8 (waiting for ok)

The specification and user interface match:	Yes	Partly	No

 			
The specification and user interface match:	Yes	Partly	No

 			
The specification and user interface match:	Yes	Partly	No

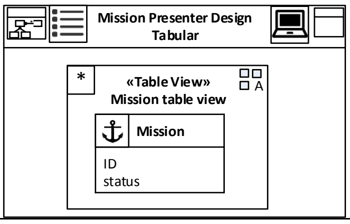
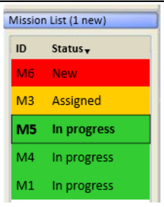
Details
Mission Properties
ID: M5
Status: Assigned, sent to GCS
Purpose
Parameters
Assigned Assets
☑ Q39 *(in progress)*
☑ ZX8 (waiting for ok)

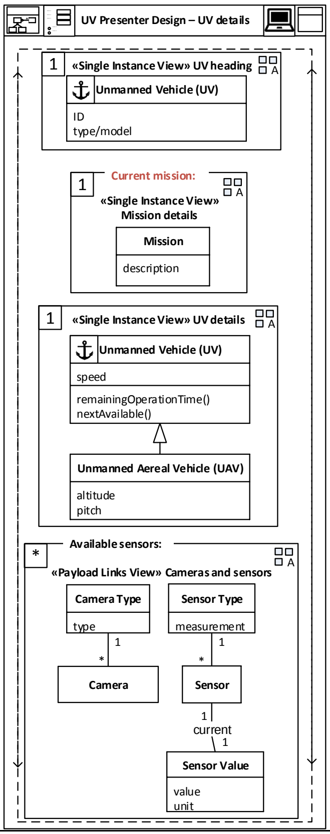
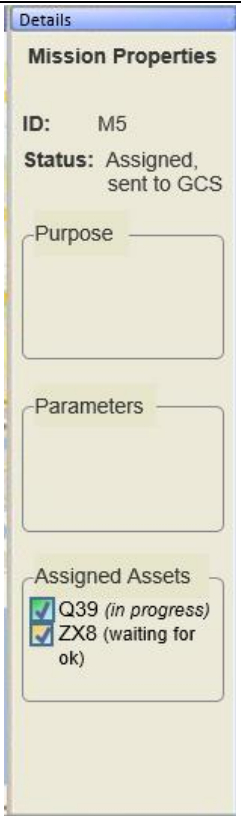
 			
The specification and user interface match:	Yes	Partly	No

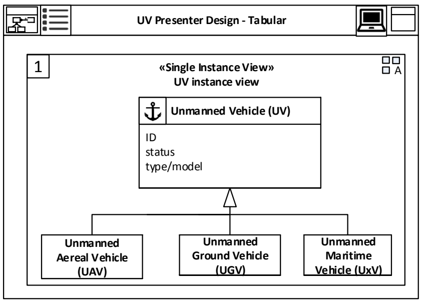
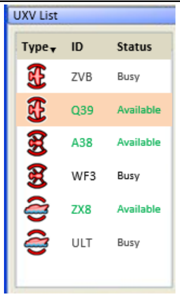
Q39 Sensor feeds
Q39 ▶
⊡ ZX8
⊡ WF3
⊡ ULT
TEMPERATURE: 14.5 °C
WIND SPEED: 8.5 m/s
GAS CONCENTR: 40 µg/m³

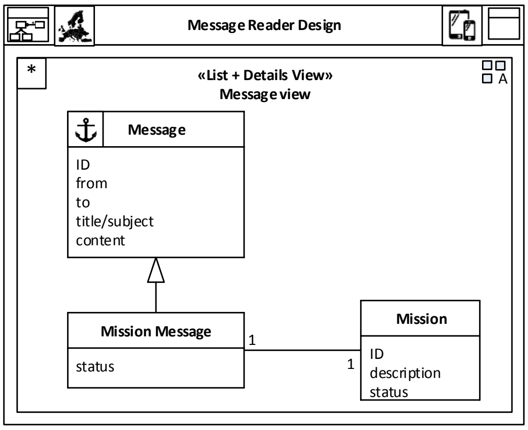
Task 1.2

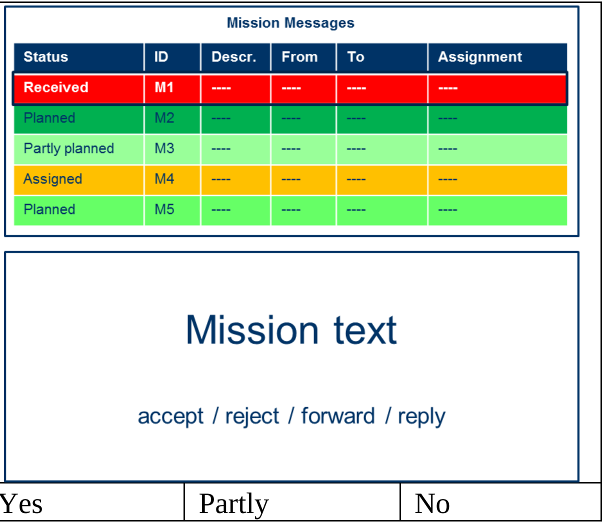
Your task is to judge whether a FLUIDE-D specification (on the left) and a user interface (on the right) match. The alternatives are: yes, partly, no.

 Mission Presenter Design Tabular * «Table View» Mission table view A Mission ID status	Mission List (1 new) ID Status M6 New M3 Assigned M5 In progress M4 In progress M1 In progress		
The specification and user interface match:	Yes	Partly	No

 UV Presenter Design – UV details 1 «Single Instance View» UV heading A Unmanned Vehicle (UV) ID type/model Current mission: 1 «Single Instance View» Mission details A Mission description 1 «Single Instance View» UV details A Unmanned Vehicle (UV) speed remainingOperationTime() nextAvailable() Unmanned Aereal Vehicle (UAV) altitude pitch Available sensors: * «Payload Links View» Cameras and sensors A Camera Type type Sensor Type measurement 1 * 1 * Camera Sensor 1 current 1 Sensor Value value unit	Details Mission Properties ID: M5 Status: Assigned, sent to GCS Purpose Parameters Assigned Assets ☑ Q39 *(in progress)* ☑ ZX8 (waiting for ok)		
The specification and user interface match:	Yes	Partly	No

 UV Presenter Design - Tabular 1 «Single Instance View» UV instance view A Unmanned Vehicle (UV) ID status type/model Unmanned Aereal Vehicle (UAV) Unmanned Ground Vehicle (UGV) Unmanned Maritime Vehicle (UxV)	UXV List Type ID Status ZVB Busy Q39 Available A38 Available WF3 Busy ZX8 Available ULT Busy		
The specification and user interface match:	Yes	Partly	No





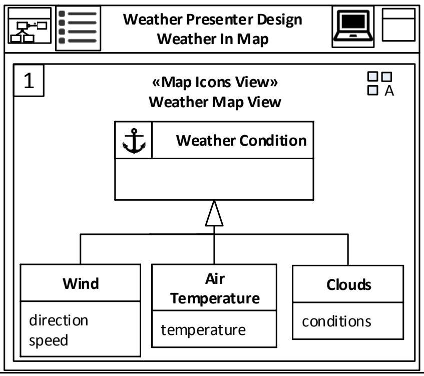
Mission Messages

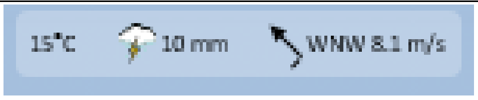
Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

Mission text

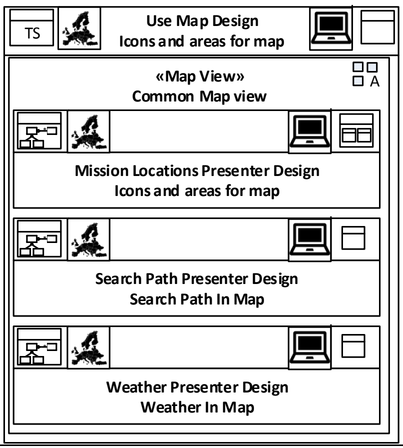
accept / reject / forward / reply

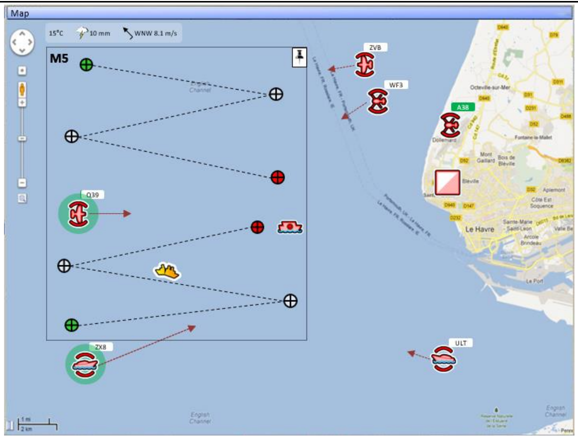
The specification and user interface match:	Yes	Partly	No





The specification and user interface match:	Yes	Partly	No

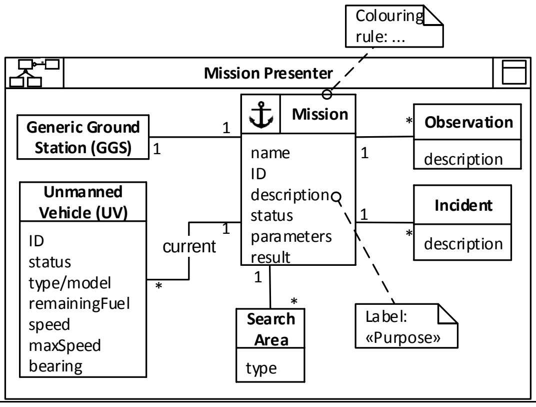


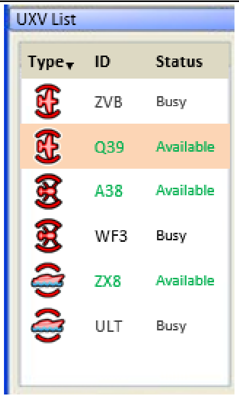


The specification and user interface match:	Yes	Partly	No

Task 1.3

Your task is to judge whether a FLUIDE-A specification (on the left) and a user interface (on the right) match. The alternatives are: yes, partly, no.

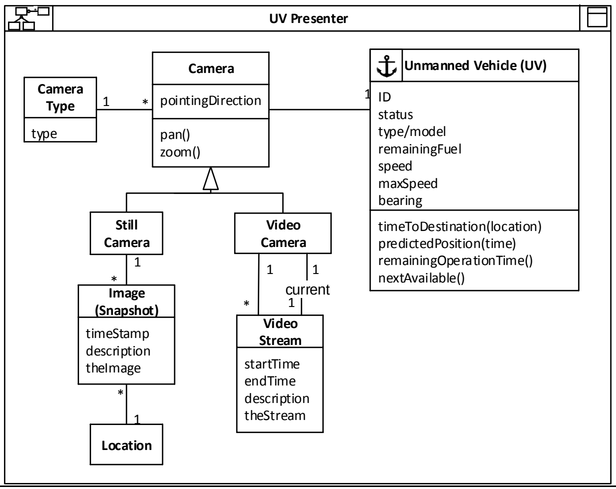


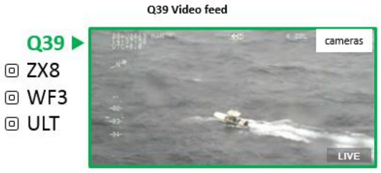


UXV List

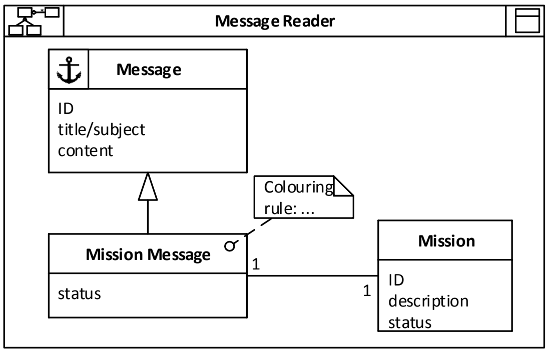
Type	ID	Status
	ZVB	Busy
	Q39	Available
	A38	Available
	WF3	Busy
	ZX8	Available
	ULT	Busy

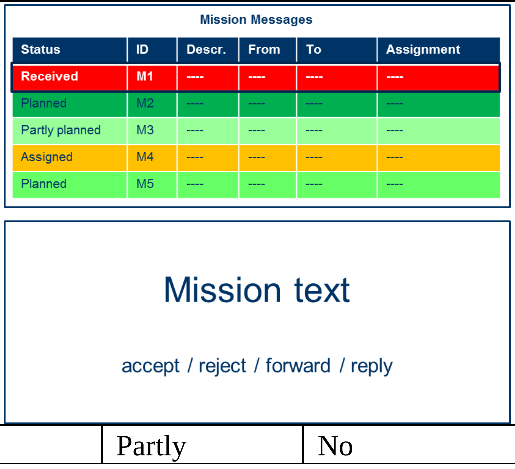
The specification and user interface match:	Yes	Partly	No





The specification and user interface match:	Yes	Partly	No





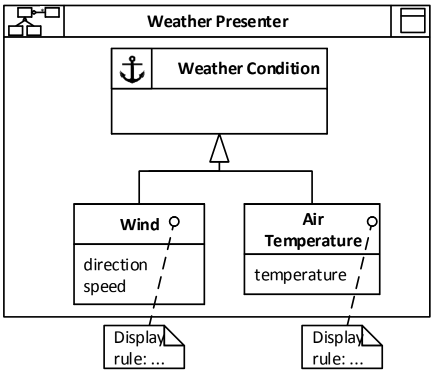
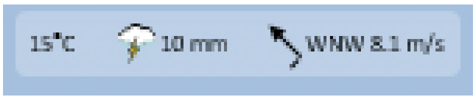
Mission Messages

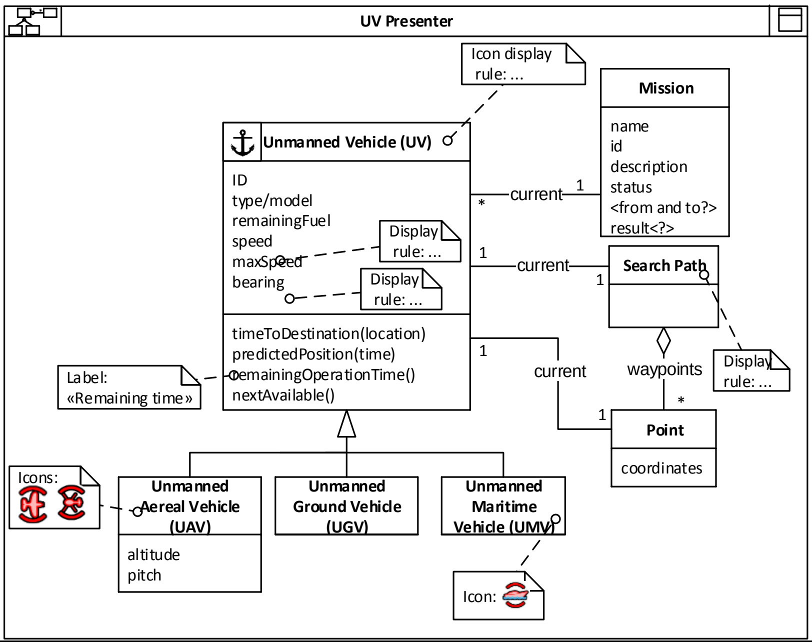
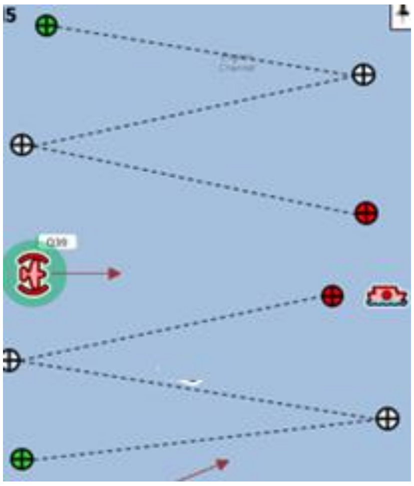
Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

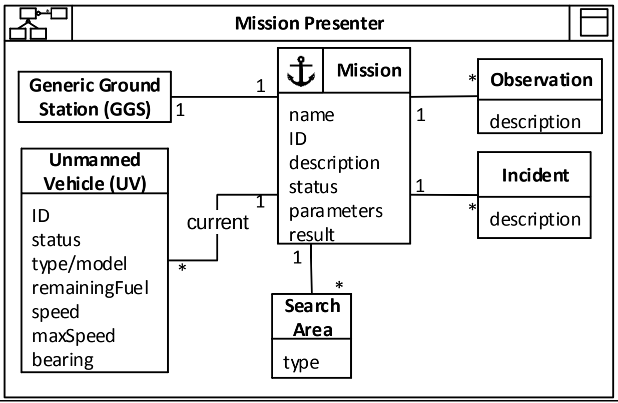
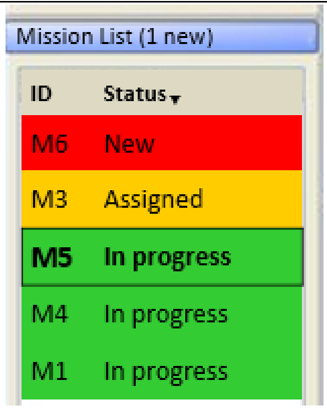
Mission text

accept / reject / forward / reply

The specification and user interface match:	Yes	Partly	No

<table>
<tr><td colspan="1"> </td><td colspan="3"> </td></tr>
<tr><td>The specification and user interface match:</td><td>Yes</td><td>Partly</td><td>No</td></tr>
</table>

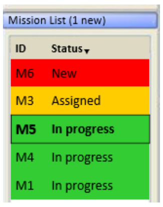
<table>
<tr><td colspan="1"> </td><td colspan="3"></td></tr>
<tr><td>The specification and user interface match:</td><td>Yes</td><td>Partly</td><td>No</td></tr>
</table>

<table>
<tr><td colspan="1"> </td><td colspan="3">Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress
</td></tr>
<tr><td>The specification and user interface match:</td><td>Yes</td><td>Partly</td><td>No</td></tr>
</table>

Task 2.1.1

In this task, one user interface (on the top) and four pairs of FLUIDE-A and FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the pairs of FLUIDE-A and FLUIDE-D specifications that best matches the user interface. Identify the specification pair by its letter.

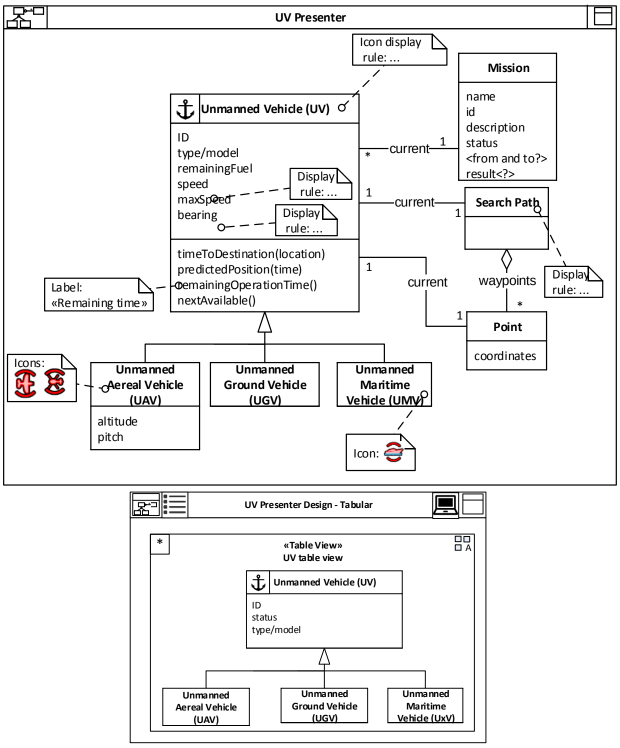


Mission List (1 new)

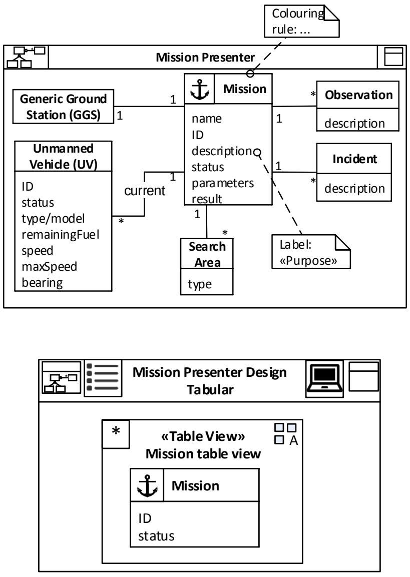
ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress

The specification pair that best matches the user interface is: A / B / C / D

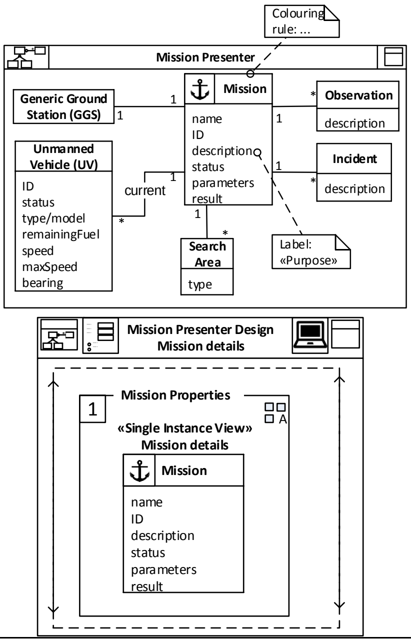
A



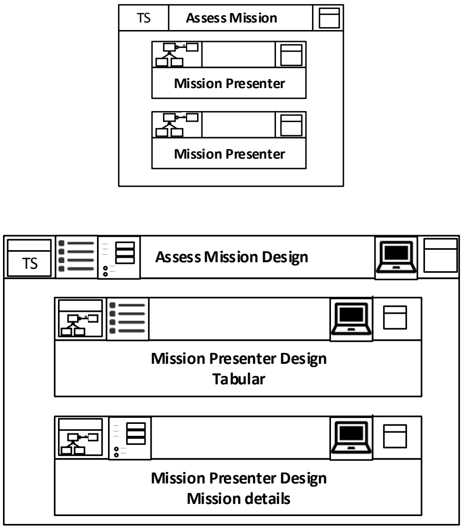
B



C



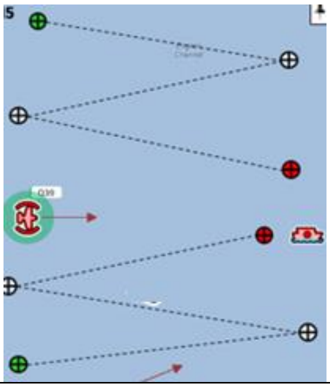
D



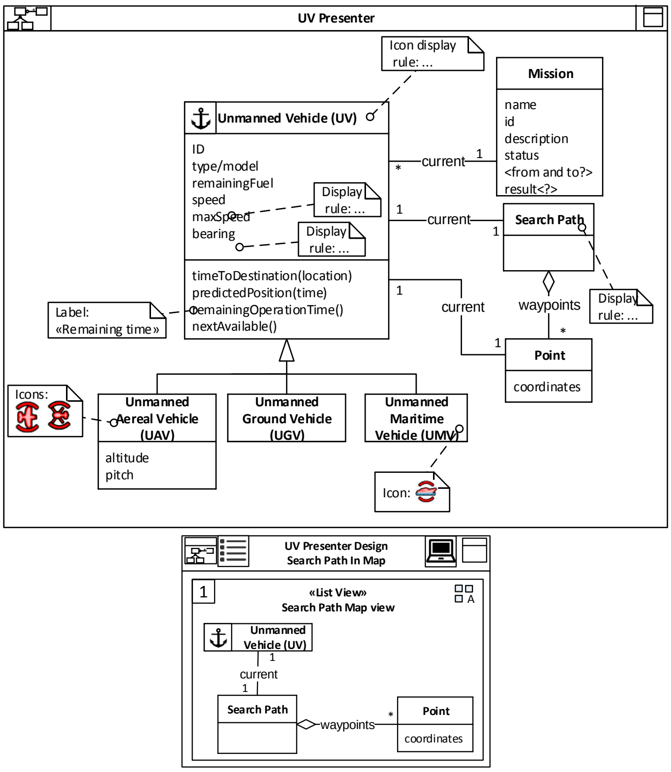
Task 2.1.2

In this task, one user interface (on the top) and four pairs of FLUIDE-A and FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the pairs of FLUIDE-A and FLUIDE-D specifications that best matches the user interface. Identify the specification pair by its letter.

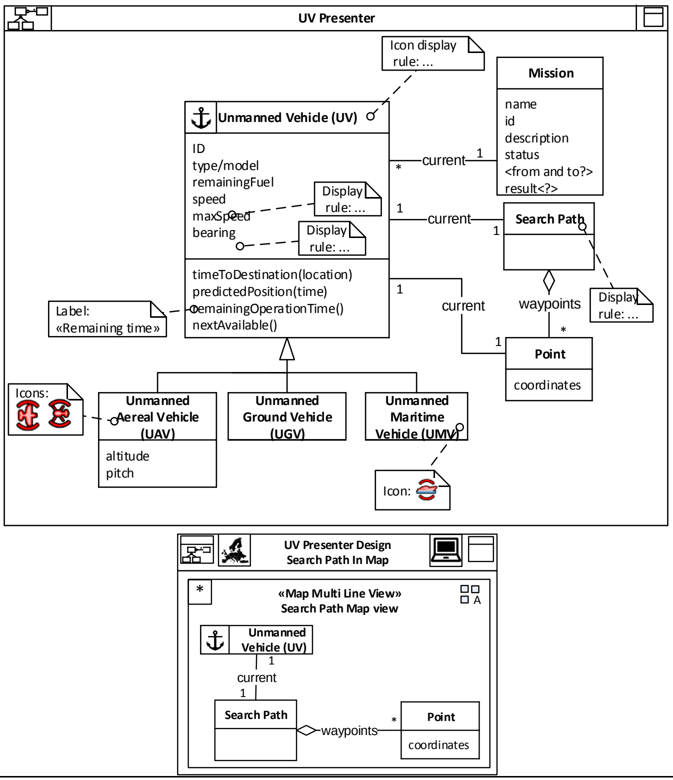
The specification pair that best matches the user interface is: A / B / C / D



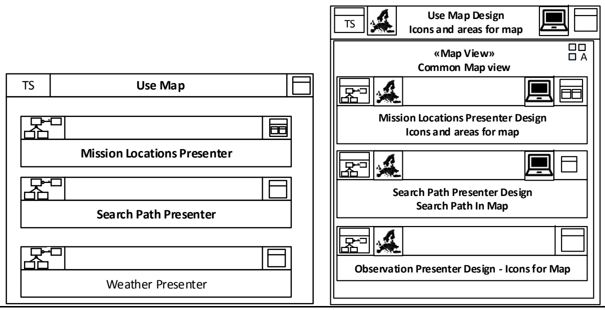
A



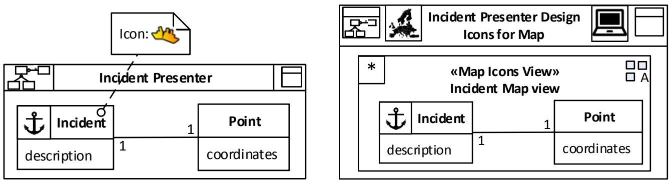
B



C



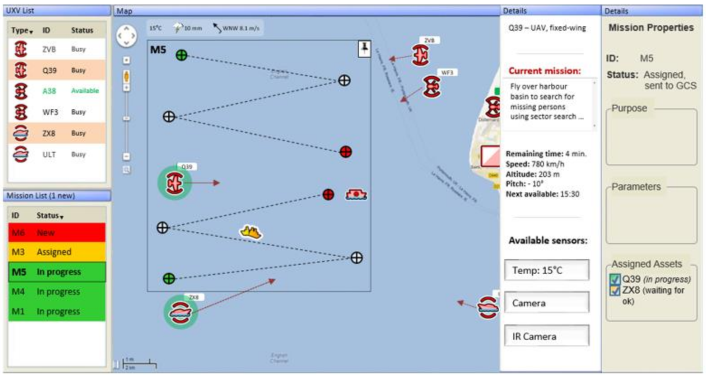
D



Task 2.1.3

In this task, one user interface (on the top) and four pairs of FLUIDE-A and FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the pairs of FLUIDE-A and FLUIDE-D specifications that best matches the user interface. Identify the specification pair by its letter.

The specification pair that best matches the user interface is: A / B / C / D



Type	ID	Status
	ZVB	Busy
	Q39	Busy
	A38	Available
	WF3	Busy
	ZX8	Busy
	ULT	Busy

Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	In progress
M4	In progress
M1	In progress

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

Available sensors:

Temp: 15°C

Camera

IR Camera

Mission Properties

ID: M5

Status: Assigned, sent to GCS

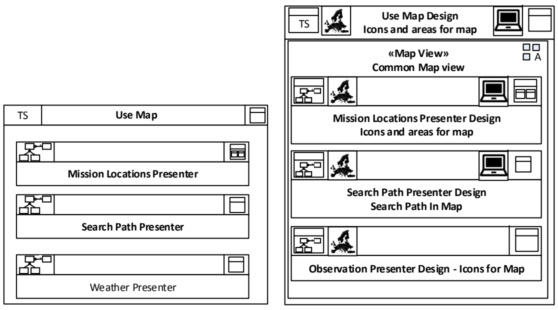
Purpose

Parameters

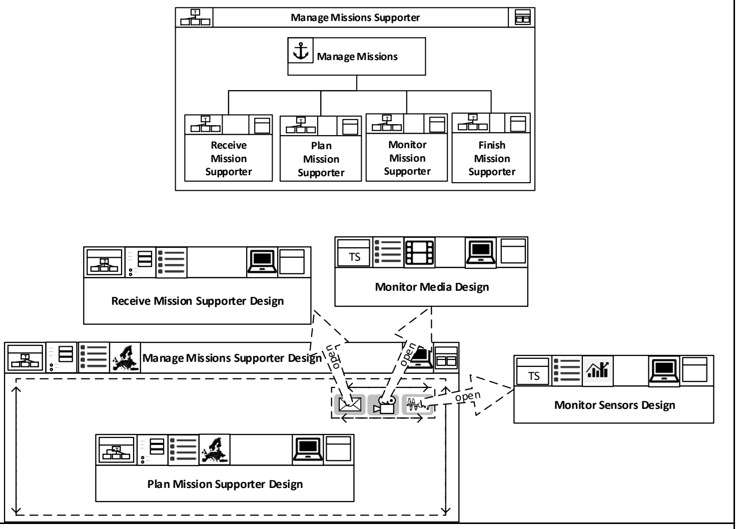
Assigned Assets

☑ Q39 *(in progress)*
☑ ZX8 (waiting for ok)

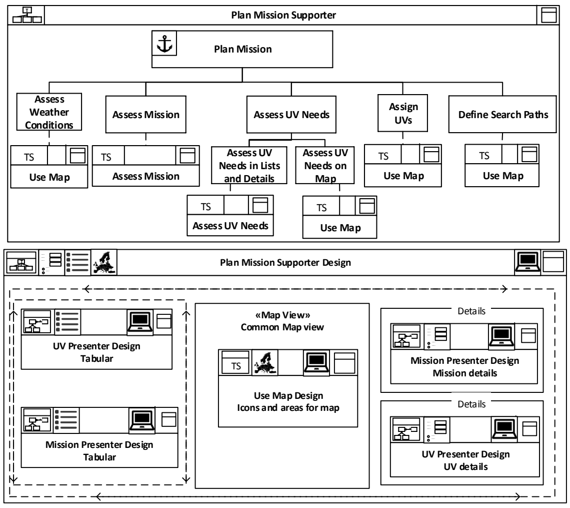
A



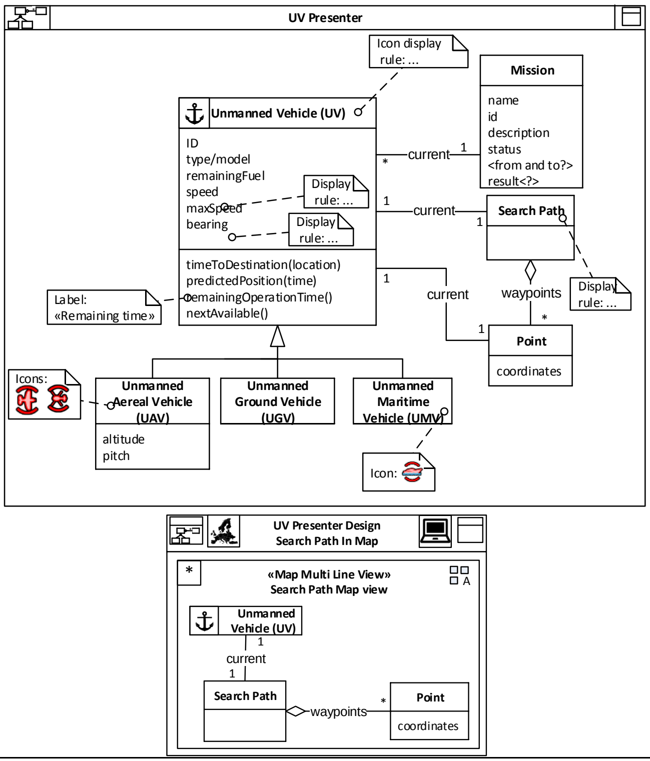
B


Manage Missions Supporter
Manage Missions
Receive Mission Supporter
Plan Mission Supporter
Monitor Mission Supporter
Finish Mission Supporter
Receive Mission Supporter Design
TS
Monitor Media Design
Manage Missions Supporter Design
open
open
open
TS
Monitor Sensors Design
Plan Mission Supporter Design

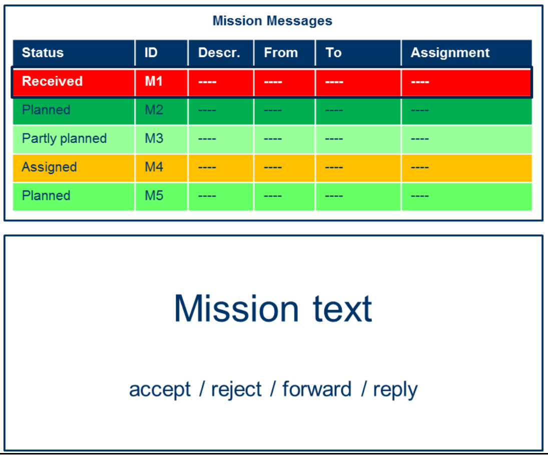
C



D



Task 2.2.1
In this task, one user interface (on the top) and four FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-D specifications that best matches the user interface. Identify the specification by its letter.
The specification that best matches the user interface is: A / B / C / D



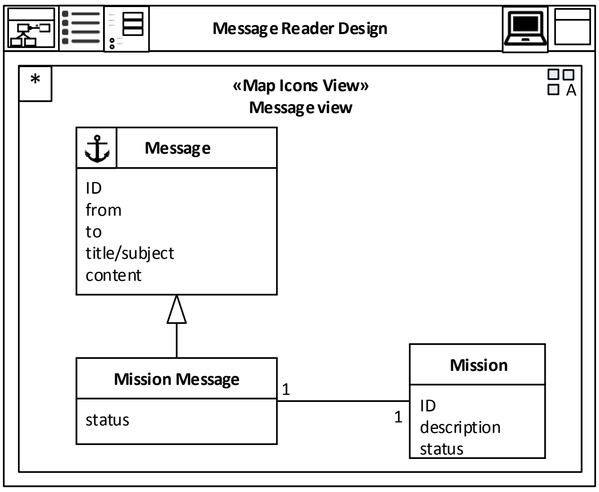
Mission Messages

Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

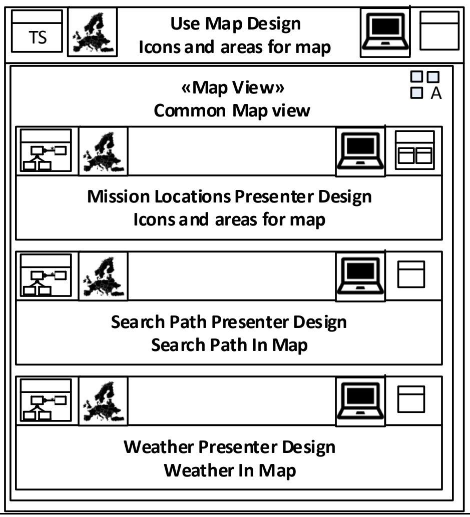
Mission text

accept / reject / forward / reply

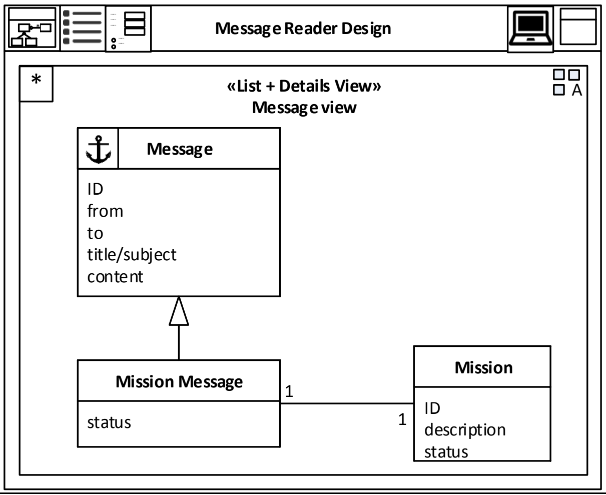
A



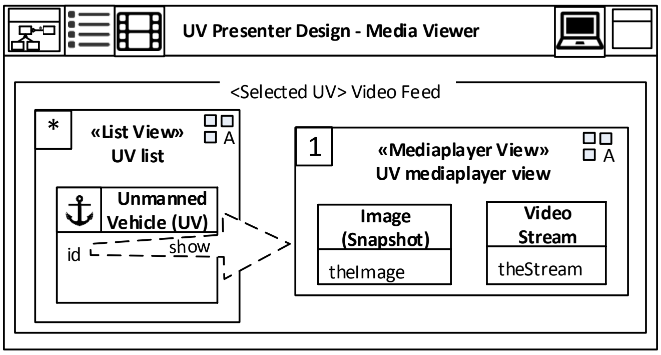
B



C

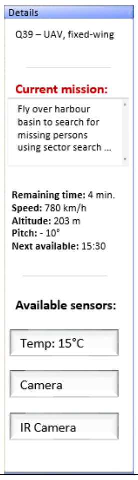


D



Task 2.2.2
In this task, one user interface (on the top) and four FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-D specifications that best matches the user interface. Identify the specification by its letter.

The specification that best matches the user interface is: A / B / C / D



Details

Q39 – UAV, fixed-wing

Current mission:

Fly over harbour basin to search for missing persons using sector search ...

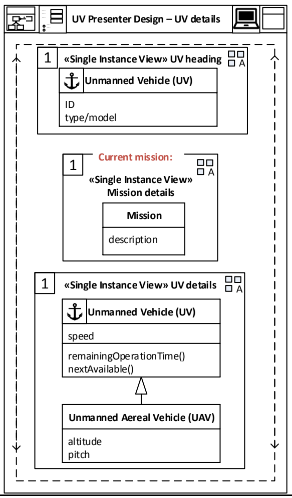
Remaining time: 4 min.
Speed: 780 km/h
Altitude: 203 m
Pitch: - 10°
Next available: 15:30

Available sensors:

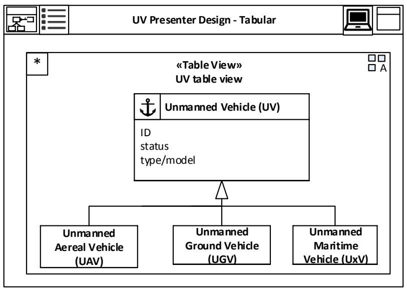
Temp: 15°C

Camera

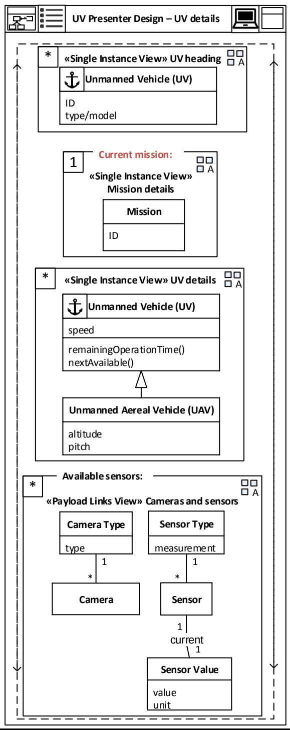
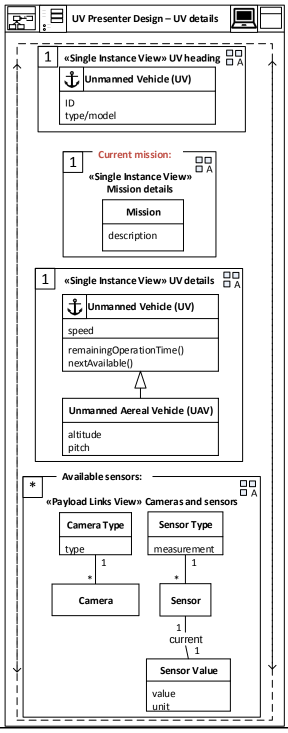
IR Camera

<table>
<tr><th>A</th><th>B</th></tr>
<tr><td>

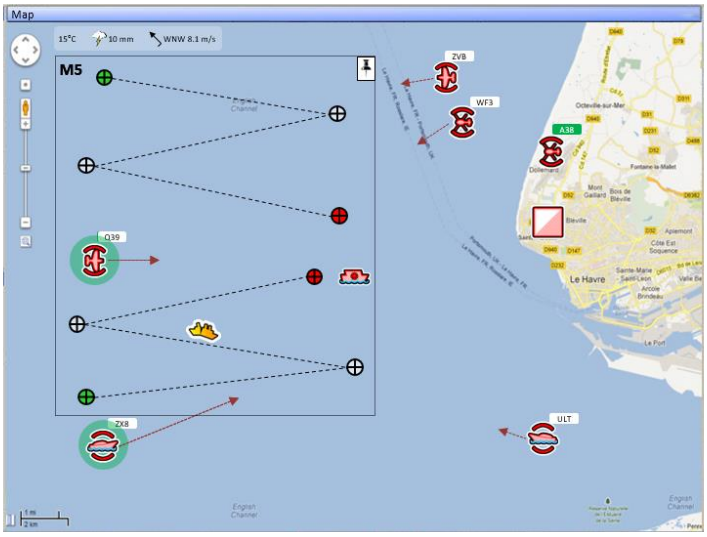
UV Presenter Design – UV details
1
«Single Instance View» UV heading
A
Unmanned Vehicle (UV)
ID
type/model
1
Current mission:
A
«Single Instance View» Mission details
Mission
description
1
«Single Instance View» UV details
A
Unmanned Vehicle (UV)
speed
remainingOperationTime()
nextAvailable()
Unmanned Aereal Vehicle (UAV)
altitude
pitch

</td><td>

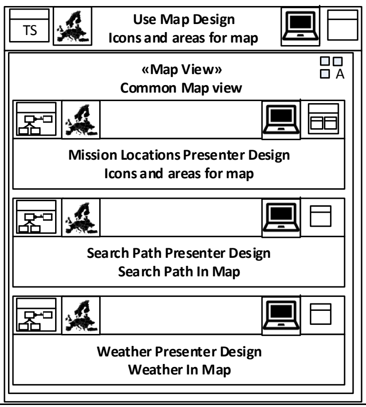
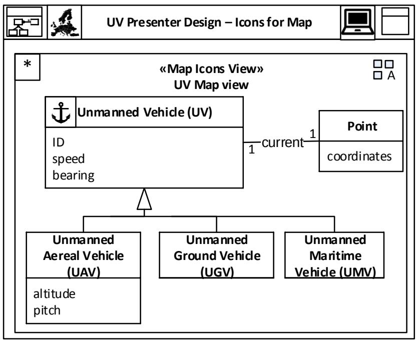
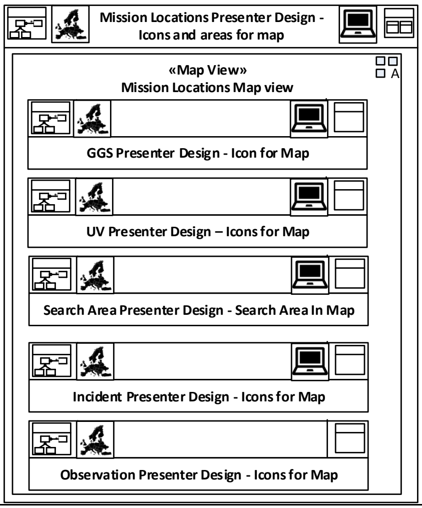
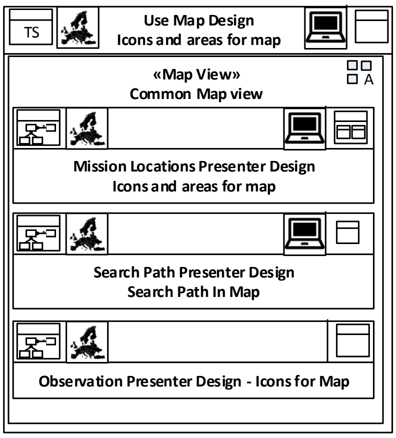
UV Presenter Design - Tabular
*
«Table View» UV table view
A
Unmanned Vehicle (UV)
ID
status
type/model
Unmanned Aereal Vehicle (UAV)
Unmanned Ground Vehicle (UGV)
Unmanned Maritime Vehicle (UxV)

</td></tr>
<tr><th>C</th><th>D</th></tr>
<tr><td> </td><td> </td></tr>
</table>

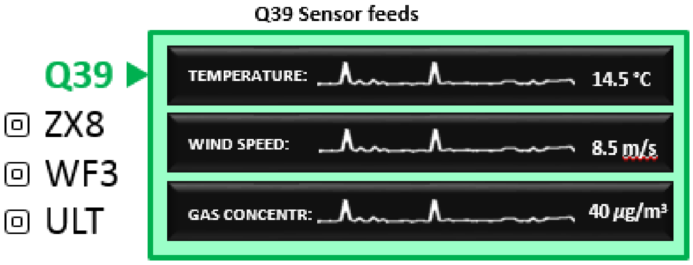
Task 2.2.3
In this task, one user interface (on the top) and four FLUIDE-D specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-D specifications that best matches the user interface. Identify the specification by its letter.



The specification that best matches the user interface is: A / B / C / D

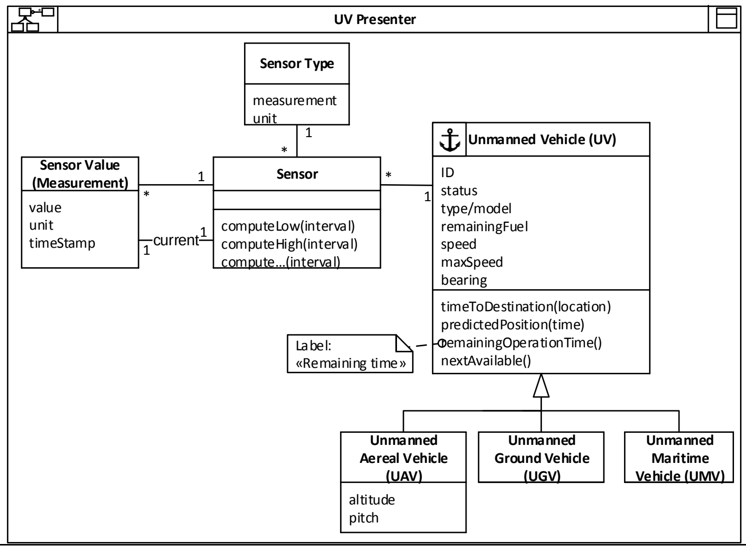
A	B
 TS Use Map Design Icons and areas for map «Map View» Common Map view A Mission Locations Presenter Design Icons and areas for map Search Path Presenter Design Search Path In Map Weather Presenter Design Weather In Map	 UV Presenter Design – Icons for Map * «Map Icons View» UV Map view A Unmanned Vehicle (UV) ID speed bearing current 1 1 Point coordinates Unmanned Aereal Vehicle (UAV) altitude pitch Unmanned Ground Vehicle (UGV) Unmanned Maritime Vehicle (UMV)
C	D
 Mission Locations Presenter Design - Icons and areas for map «Map View» Mission Locations Map view A GGS Presenter Design - Icon for Map UV Presenter Design – Icons for Map Search Area Presenter Design - Search Area In Map Incident Presenter Design - Icons for Map Observation Presenter Design - Icons for Map	 TS Use Map Design Icons and areas for map «Map View» Common Map view A Mission Locations Presenter Design Icons and areas for map Search Path Presenter Design Search Path In Map Observation Presenter Design - Icons for Map

Task 2.3.1
In this task, one user interface (on the top) and four FLUIDE-A specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-A specifications that best matches the user interface. Identify the specification by its letter.

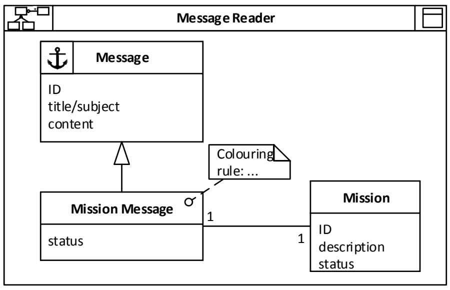


The specification that best matches the user interface is: A / B / C / D

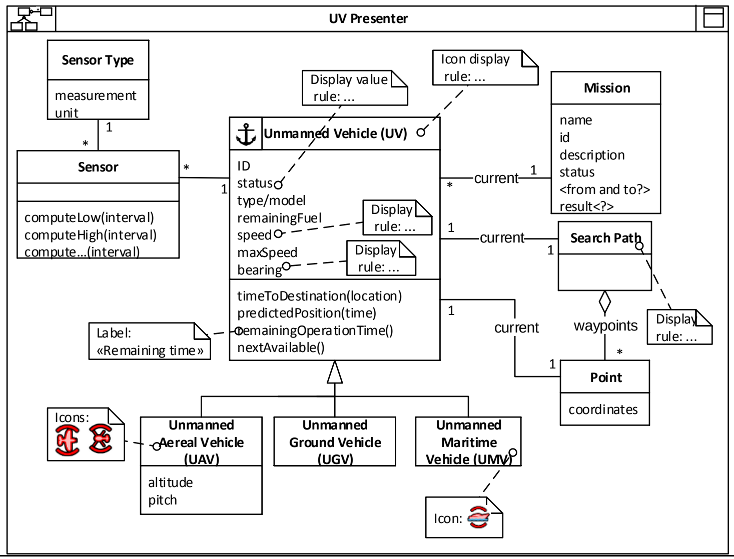
A



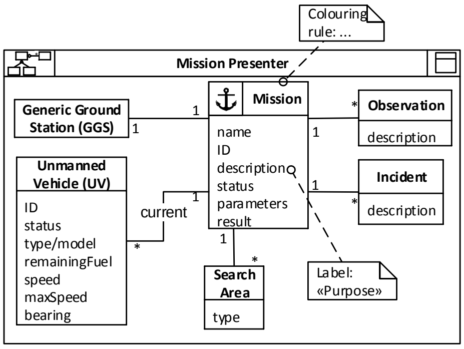
B



C

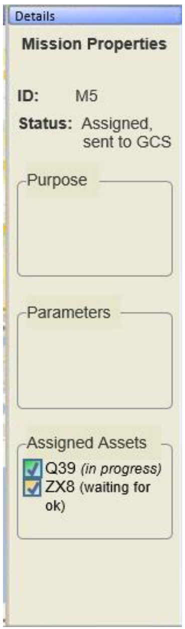


D



Task 2.3.2
In this task, one user interface (on the top) and four FLUIDE-A specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-A specifications that best matches the user interface. Identify the specification by its letter.

The specification that best matches the user interface is: A / B / C / D



Details

Mission Properties

ID: M5

Status: Assigned, sent to GCS

Purpose

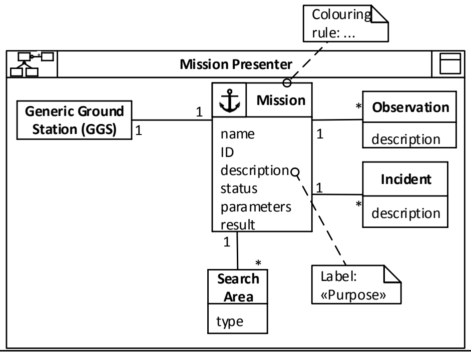
Parameters

Assigned Assets

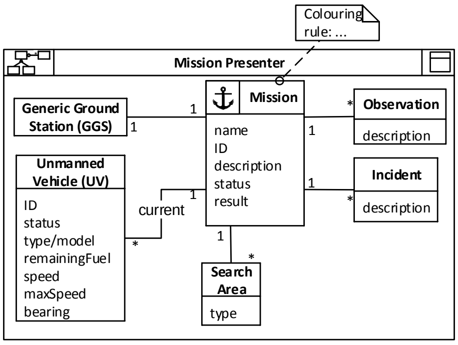
☑ Q39 *(in progress)*

☑ ZX8 (waiting for ok)

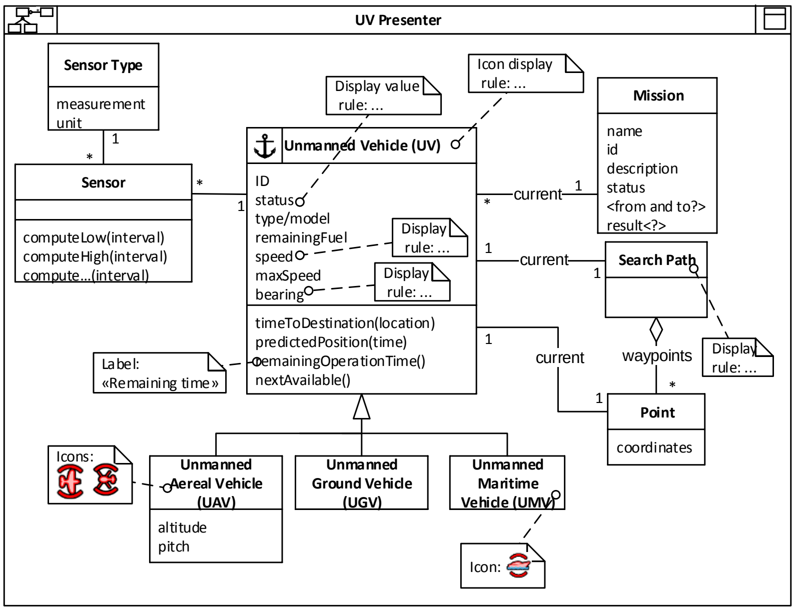
A



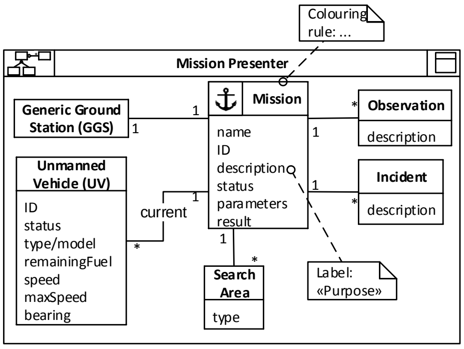
B



C



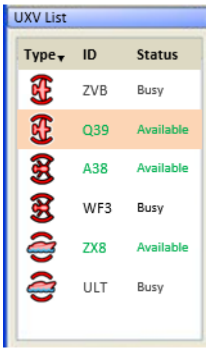
D



Task 2.3.3

In this task, one user interface (on the top) and four FLUIDE-A specifications (on the bottom) are shown. Your task is to identify which of the FLUIDE-A specifications that best matches the user interface. Identify the specification by its letter.

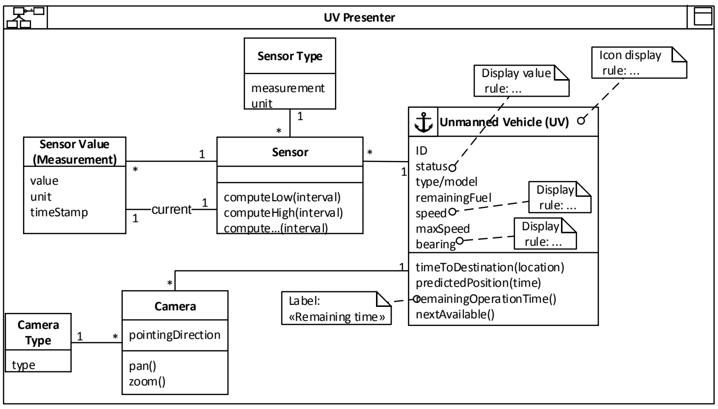
The specification that best matches the user interface is: A / B / C / D



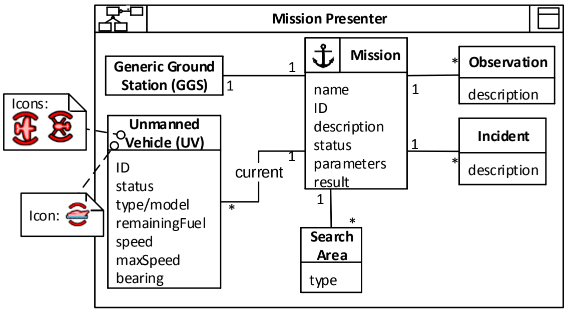
UXV List

Type	ID	Status
	ZVB	Busy
	Q39	Available
	A38	Available
	WF3	Busy
	ZX8	Available
	ULT	Busy

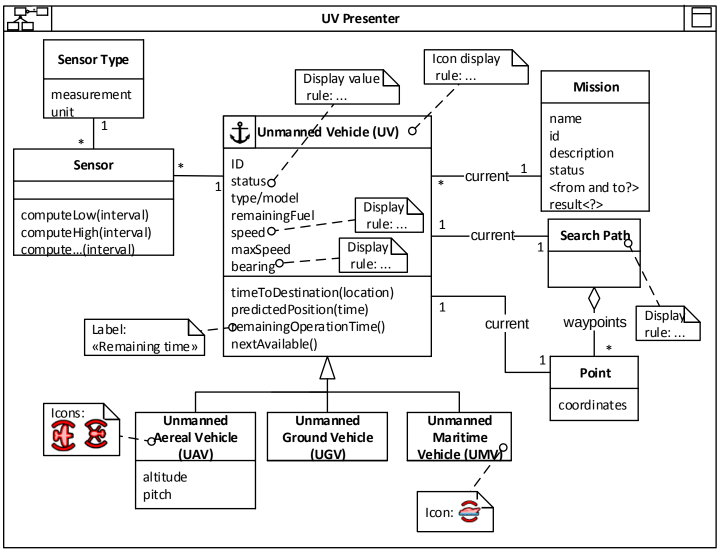
A



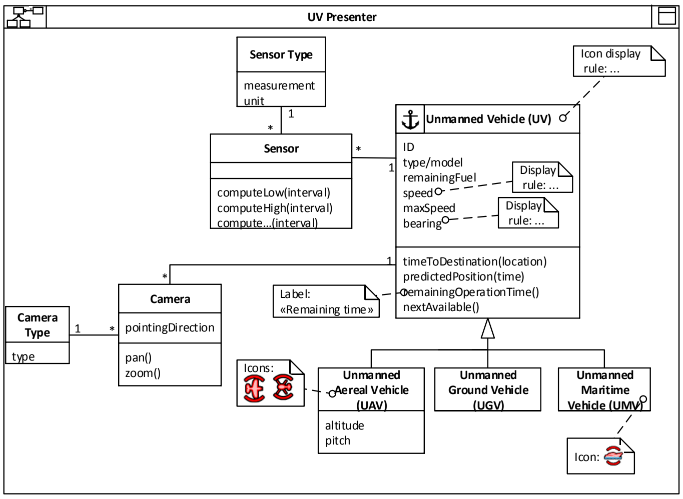
B



C

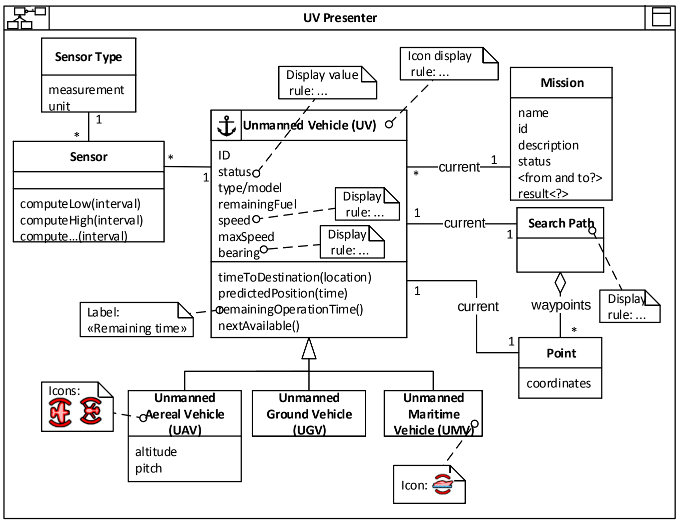


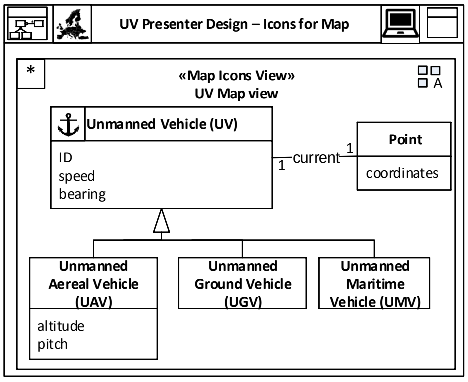
D



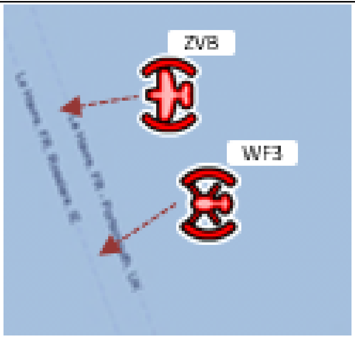
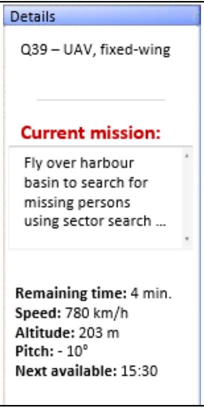
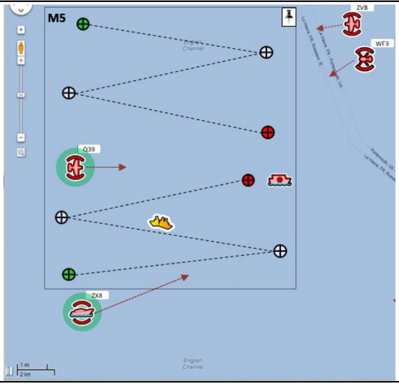
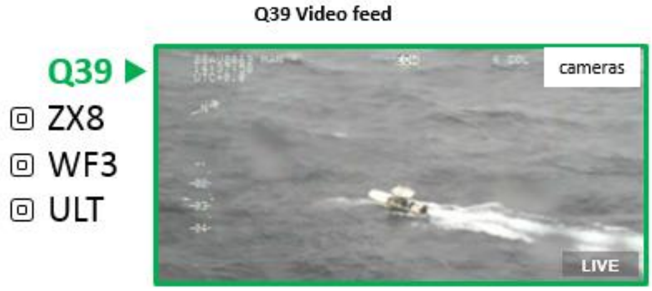
Task 3.1.1

In this task, one pair of FLUIDE-A and FLUIDE-D specifications (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A and FLUIDE-D specification. Identify the user interface by its letter.



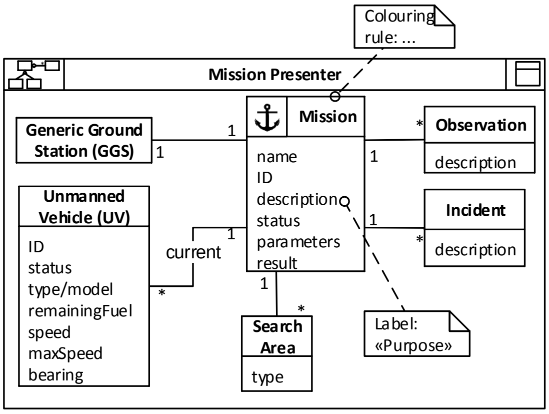

UV Presenter Design – Icons for Map
*
«Map Icons View»
UV Map view
A
Unmanned Vehicle (UV)
ID
speed
bearing
current
1
1
Point
coordinates
Unmanned Aereal Vehicle (UAV)
altitude
pitch
Unmanned Ground Vehicle (UGV)
Unmanned Maritime Vehicle (UMV)

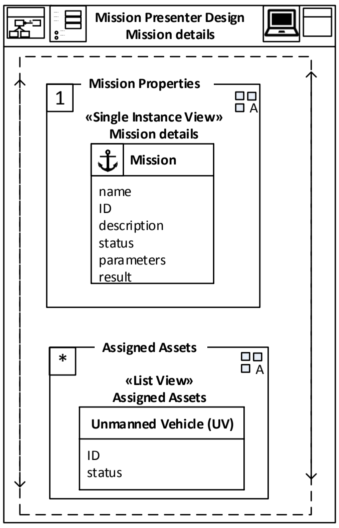
The user interface that best matches the specification is: A / B / C / D

A	B
	 Details Q39 – UAV, fixed-wing Current mission: Fly over harbour basin to search for missing persons using sector search ... **Remaining time:** 4 min. **Speed:** 780 km/h **Altitude:** 203 m **Pitch:** - 10° **Next available:** 15:30
C	**D**
	Q39 Video feed 

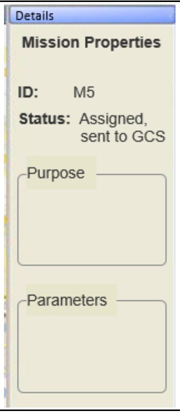
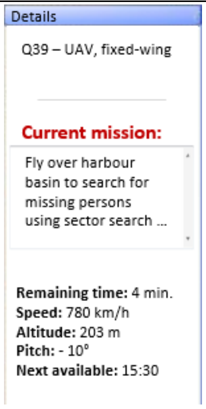
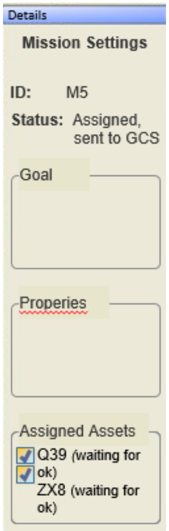
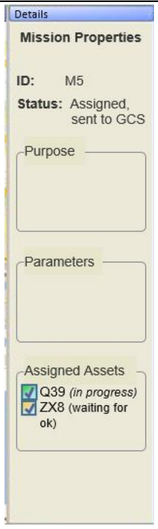
Task 3.1.2

In this task, one pair of FLUIDE-A and FLUIDE-D specifications (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A and FLUIDE-D specification. Identify the user interface by its letter.



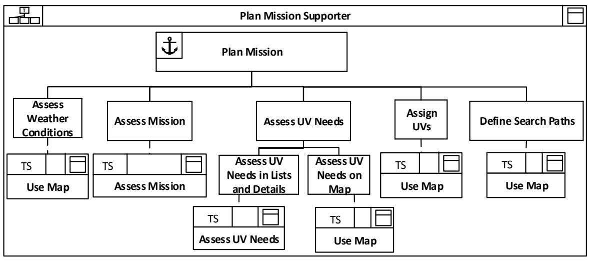


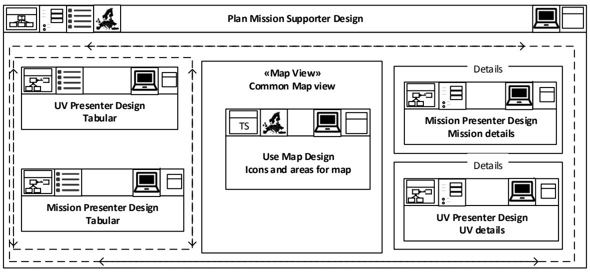
The user interface that best matches the specification is: A / B / C / D

A	B
 Details Mission Properties ID: M5 Status: Assigned, sent to GCS Purpose Parameters	 Details Q39 – UAV, fixed-wing Current mission: Fly over harbour basin to search for missing persons using sector search ... **Remaining time:** 4 min. **Speed:** 780 km/h **Altitude:** 203 m **Pitch:** - 10° **Next available:** 15:30
C	D
 Details Mission Settings ID: M5 Status: Assigned, sent to GCS Goal Properies Assigned Assets ☑ Q39 (waiting for ok) ☑ ZX8 (waiting for ok)	 Details Mission Properties ID: M5 Status: Assigned, sent to GCS Purpose Parameters Assigned Assets ☑ Q39 *(in progress)* ☑ ZX8 (waiting for ok)

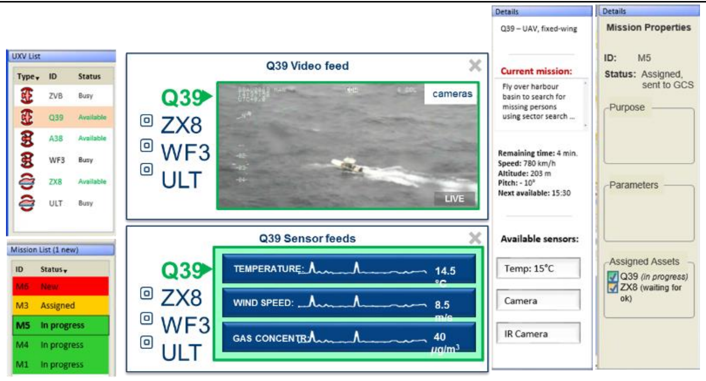
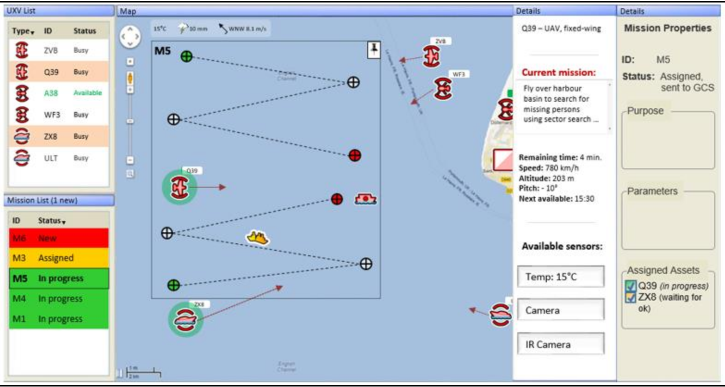
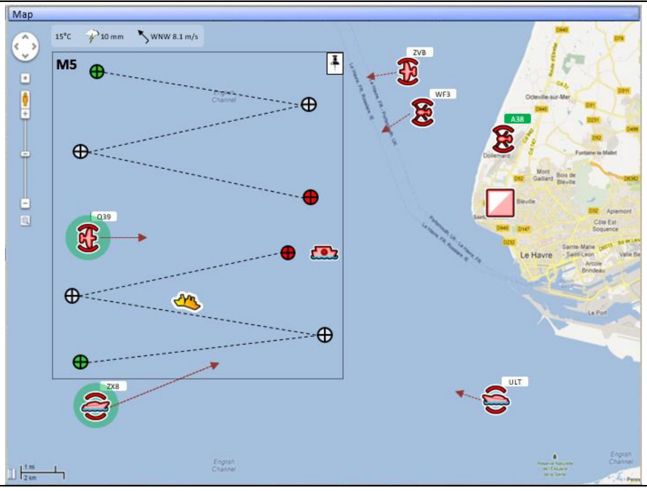
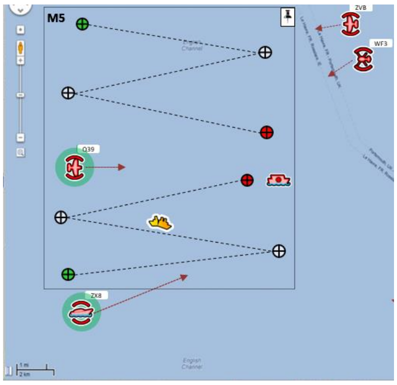
Task 3.1.3

In this task, one pair of FLUIDE-A and FLUIDE-D specifications (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A and FLUIDE-D specification. Identify the user interface by its letter.



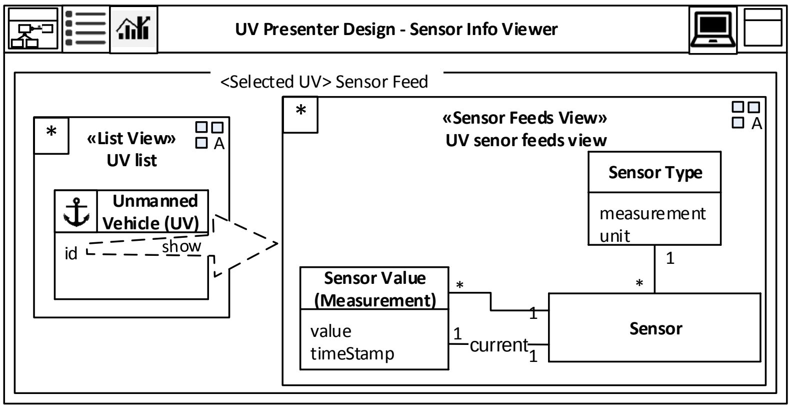


The user interface that best matches the specification is: A / B / C / D

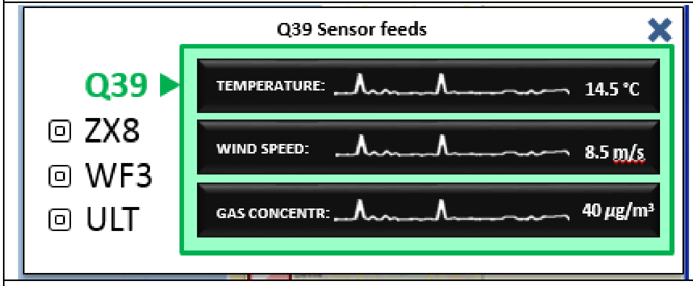
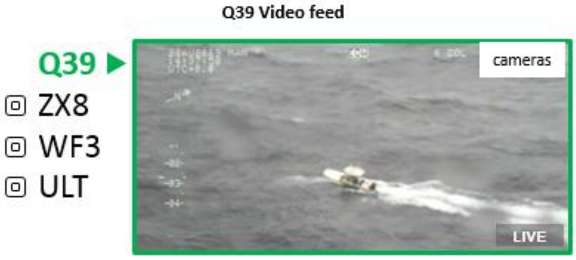
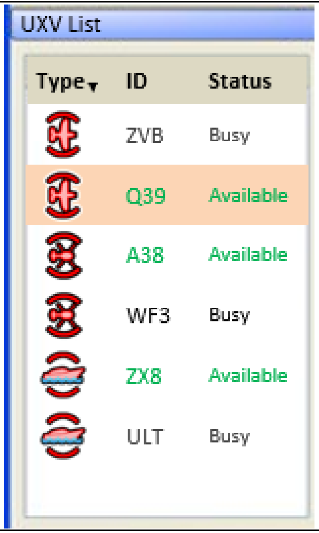
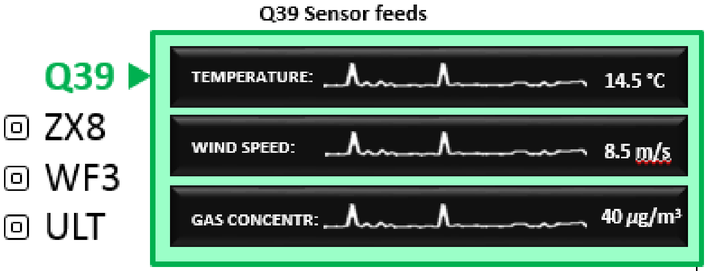
A	B
	
C	D
	

Task 3.2.1

In this task, one FLUIDE-D specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-D specification. Identify the user interface by its letter.

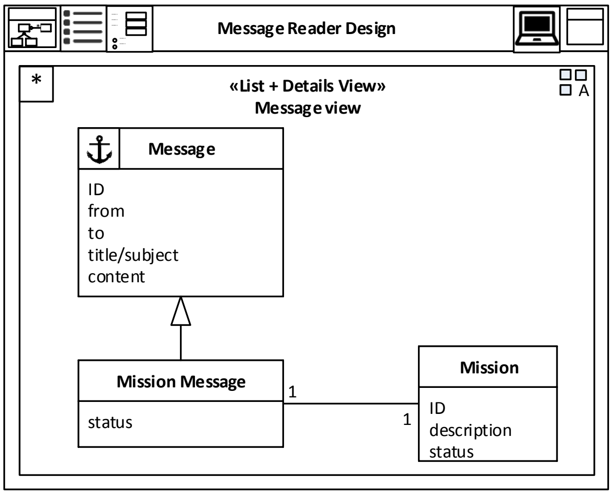


The user interface that best matches the specification is: A / B / C / D

A	B
 Q39 Sensor feeds Q39 ZX8 WF3 ULT TEMPERATURE: 14.5 °C WIND SPEED: 8.5 m/s GAS CONCENTR: 40 µg/m³	 Q39 Video feed Q39 ZX8 WF3 ULT cameras LIVE
C	D
 UXV List Type ID Status ZVB Busy Q39 Available A38 Available WF3 Busy ZX8 Available ULT Busy	 Q39 Sensor feeds Q39 ZX8 WF3 ULT TEMPERATURE: 14.5 °C WIND SPEED: 8.5 m/s GAS CONCENTR: 40 µg/m³

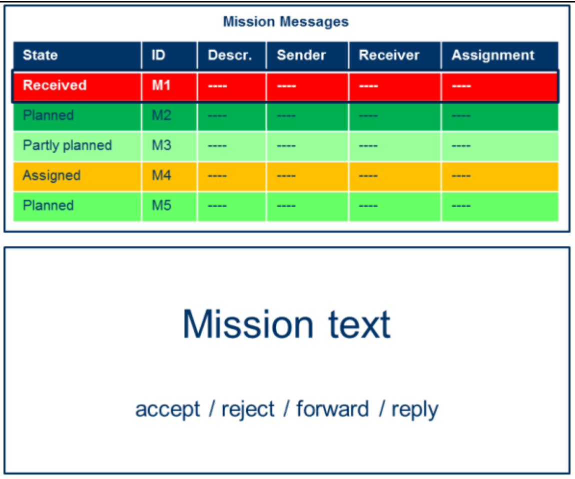
Task 3.2.2

In this task, one FLUIDE-D specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-D specification. Identify the user interface by its letter.



The user interface that best matches the specification is: A / B / C / D

A



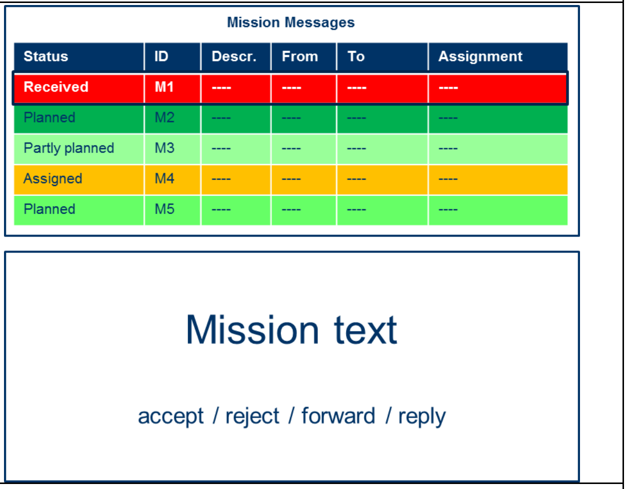
Mission Messages

State	ID	Descr.	Sender	Receiver	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

Mission text

accept / reject / forward / reply

B



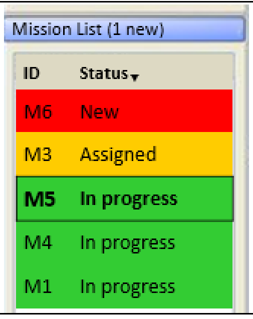
Mission Messages

Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

Mission text

accept / reject / forward / reply

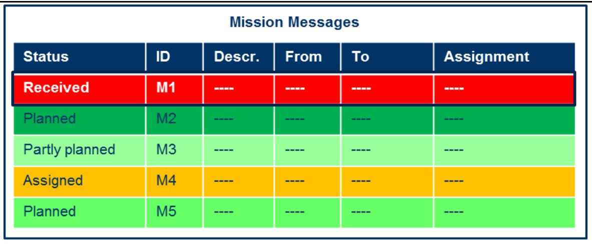
C



Mission List (1 new)

ID	Status
M6	New
M3	Assigned
M5	**In progress**
M4	In progress
M1	In progress

D

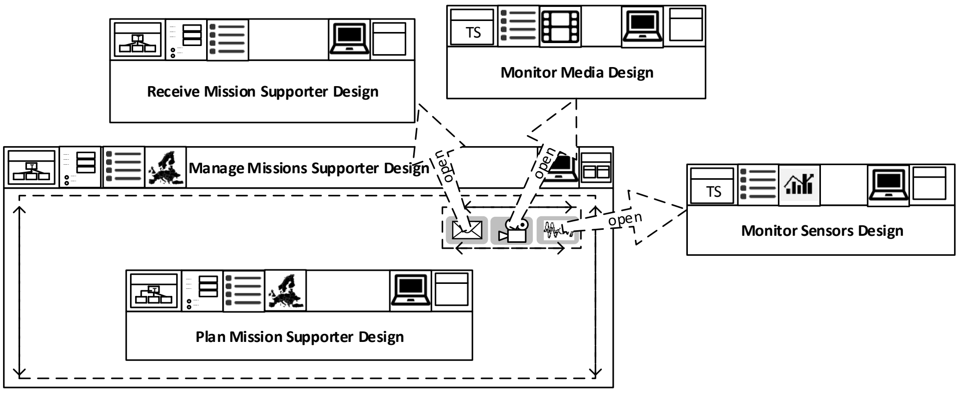


Mission Messages

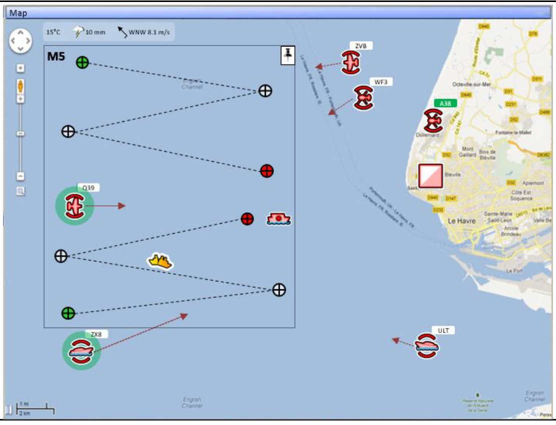
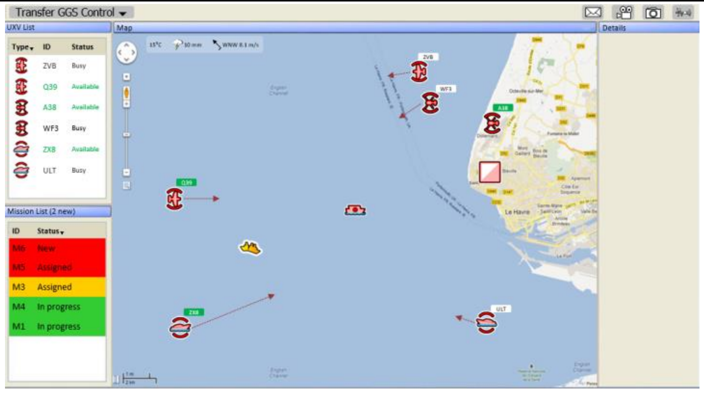
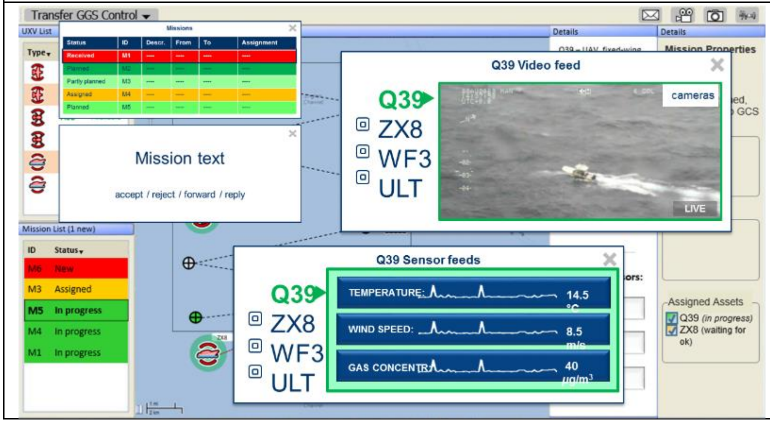
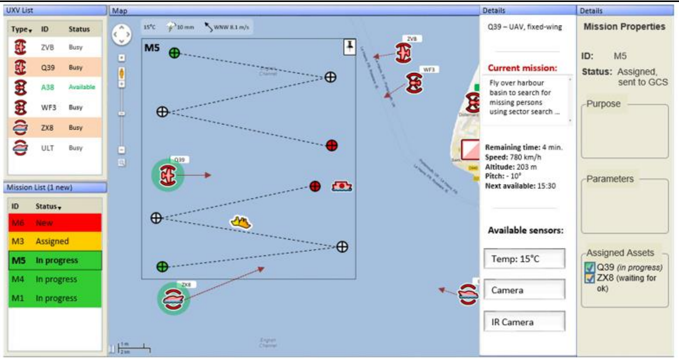
Status	ID	Descr.	From	To	Assignment
Received	M1	----	----	----	----
Planned	M2	----	----	----	----
Partly planned	M3	----	----	----	----
Assigned	M4	----	----	----	----
Planned	M5	----	----	----	----

Task 3.2.3

In this task, one FLUIDE-D specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-D specification. Identify the user interface by its letter.

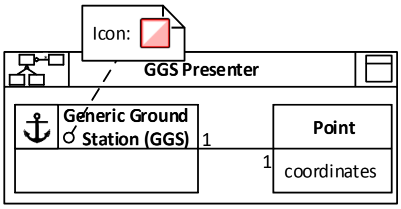

Receive Mission Supporter Design
TS
Monitor Media Design
Manage Missions Supporter Design
open
open
open
TS
Monitor Sensors Design
Plan Mission Supporter Design

The user interface that best matches the specification is: A / B / C / D

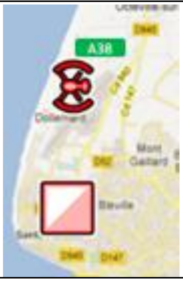
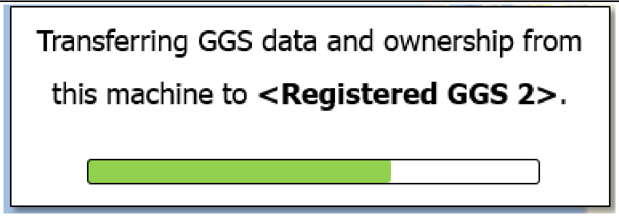
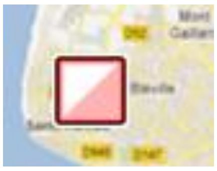
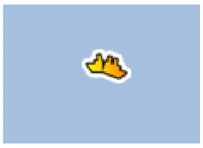
A	B
	
C	D
	

Task 3.3.1

In this task, one FLUIDE-A specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A specification. Identify the user interface by its letter.

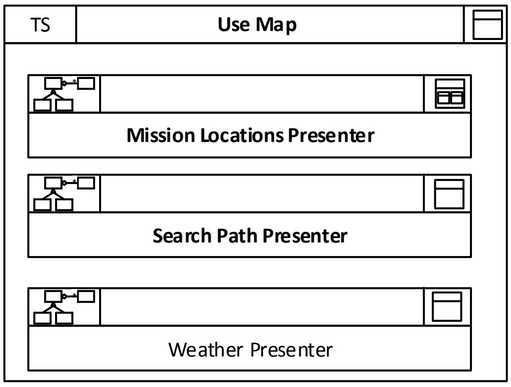


The user interface that best matches the specification is: A / B / C / D

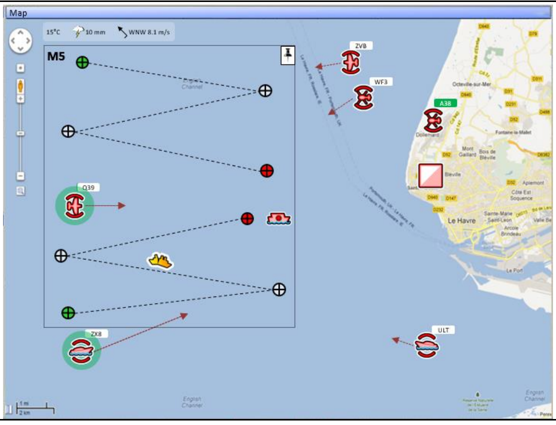
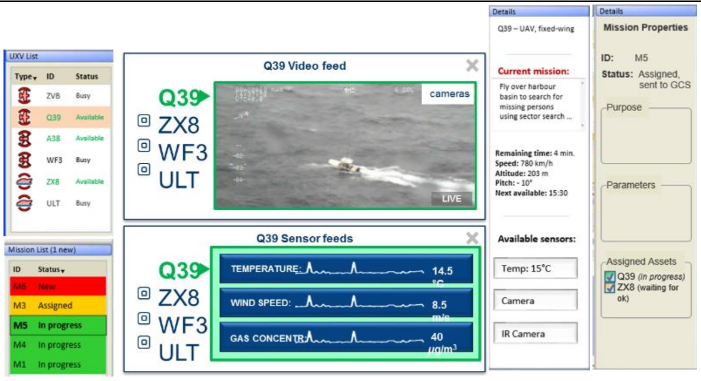
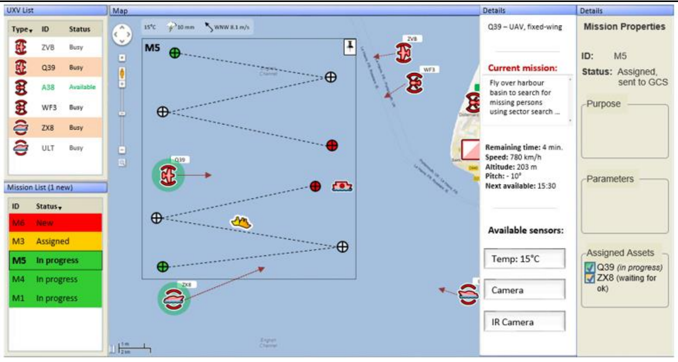
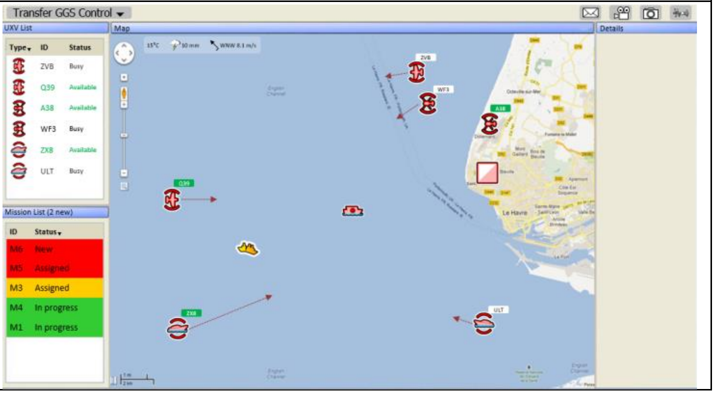
A	B
	 Transferring GGS data and ownership from this machine to **<Registered GGS 2>**.
C	D
	

Task 3.3.2

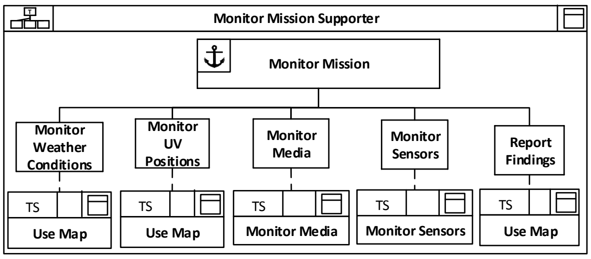
In this task, one FLUIDE-A specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A specification. Identify the user interface by its letter.



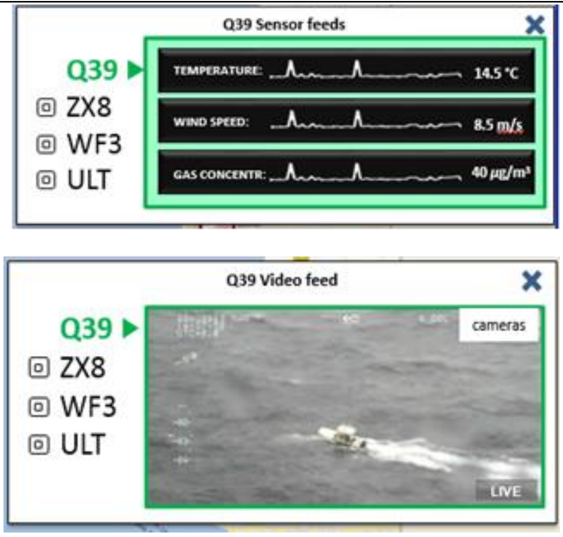
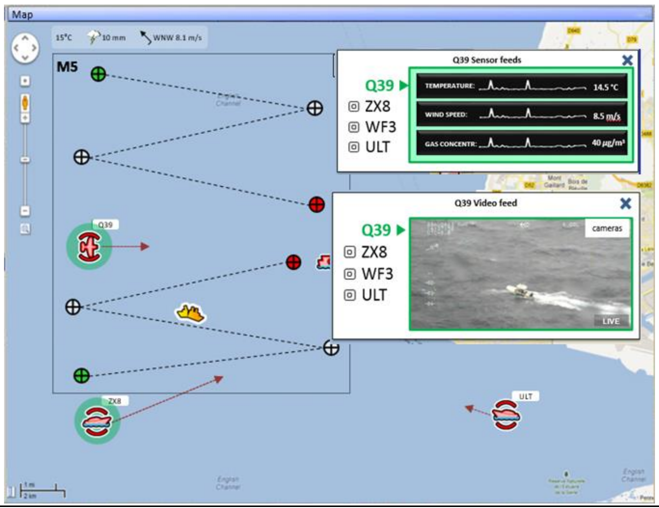
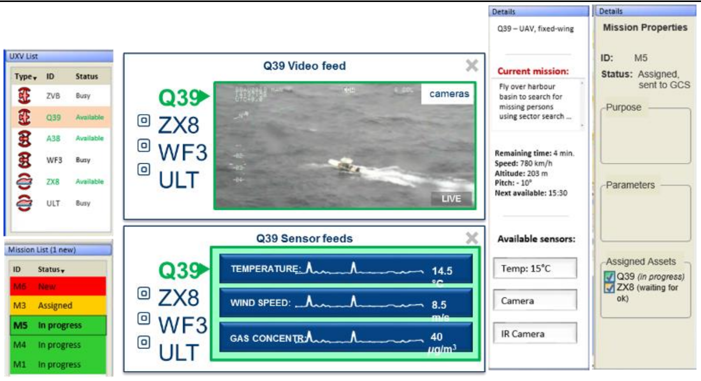
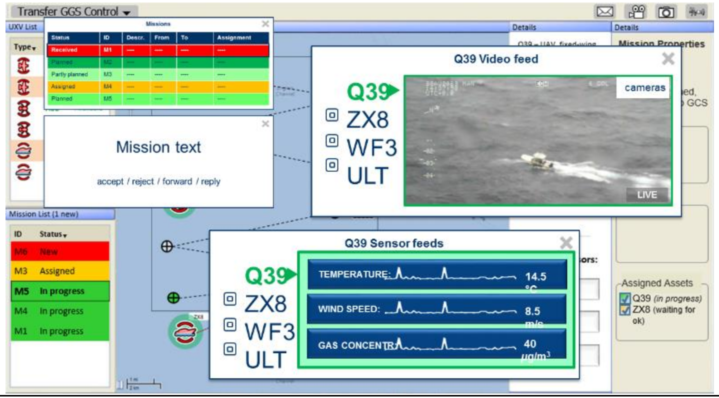
The user interface that best matches the specification is: A / B / C / D

A	B
	
C	**D**
	

Task 3.3.3
In this task, one FLUIDE-A specification (on the top) and four user interfaces (on the bottom) are shown. Your task is to identify which of the user interfaces that best matches the FLUIDE-A specification. Identify the user interface by its letter.



The user interface that best matches the specification is: A / B / C / D

A	B
 Q39 Sensor feeds Q39 ZX8 WF3 ULT TEMPERATURE: 14.5 °C WIND SPEED: 8.5 m/s GAS CONCENTR.: 40 µg/m³ Q39 Video feed cameras LIVE	 Map M5 Q39 Sensor feeds Q39 Video feed ZX8 WF3 ULT cameras LIVE
C	D
 UXV List Mission List (1 new) Q39 Video feed Q39 Sensor feeds Details Mission Properties Current mission: Available sensors: Temp: 15°C Camera IR Camera Assigned Assets	 Transfer GGS Control UXV List Missions Mission text accept / reject / forward / reply Mission List (1 new) Q39 Video feed Q39 Sensor feeds Details Mission Properties Assigned Assets

Questions about the specifications

Were the specifications easy to understand?

Impossible	Very difficult	Somewhat difficult	Quite easy	Very easy	Trivial

Was it easy to grasp the essential part of the specifications?

Impossible	Very difficult	Somewhat difficult	Quite easy	Very easy	Trivial

Was it easy to understand the connections between the specifications and the user interfaces?

Impossible	Very difficult	Somewhat difficult	Quite easy	Very easy	Trivial

How did you find the size of the specifications?

Too large	Appropriate size	Too small

How did you find the levels of details in the specifications?

Too detailed	A bit too detailed	Appropriate	Should have some more details	Should have a lot more details

Some specifications mix UML class models with FLUIDE specific notation. How did you react to this?

It was very confusing	It was a bit confusing	I did not react to it	It enhanced my understanding a bit	It enhanced my understanding very much

Which tasks were easiest to solve?

The ones with both FLUIDE-A and FLUIDE-D specifications	The ones with only FLUIDE-D specifications	The ones with only FLUIDE-A specifications

Which tasks were most difficult to solve?

The ones with both FLUIDE-A and FLUIDE-D specifications	The ones with only FLUIDE-D specifications	The ones with only FLUIDE-A specifications

Which specifications were easiest to understand?

The ones containing UML class models	The ones containing task models	The ones containing neither

Which specifications were most difficult to understand?

The ones containing UML class models	The ones containing task models	The ones containing neither

Do you have any suggestions for improving the FLUIDE specifications (write below)?

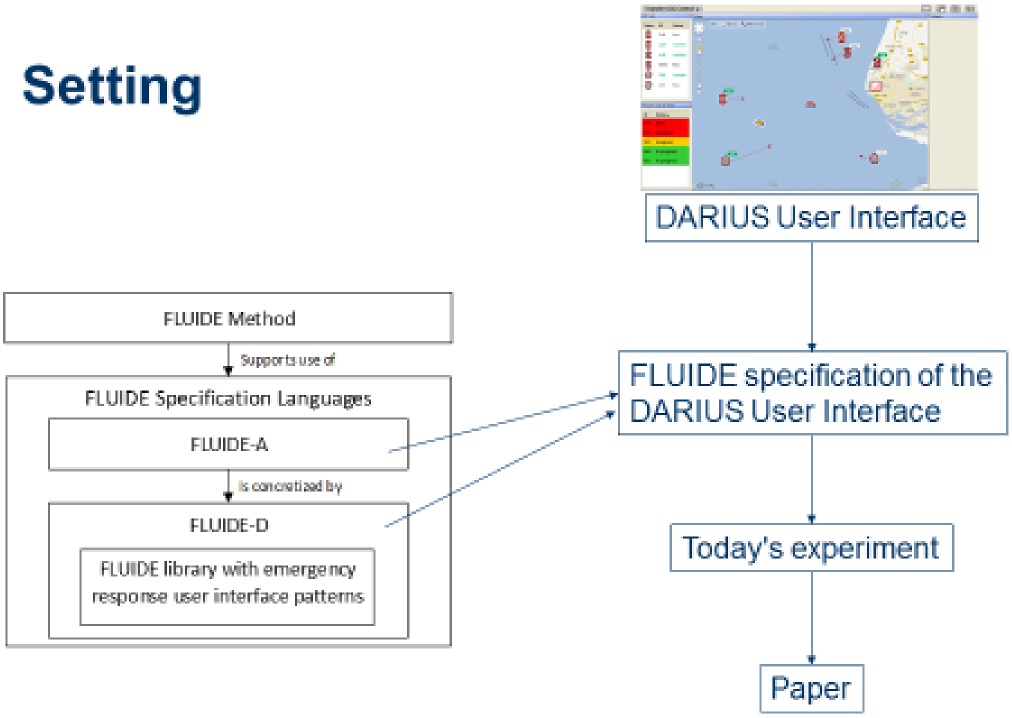
Appendix C – Introductory material for participants in the experiment

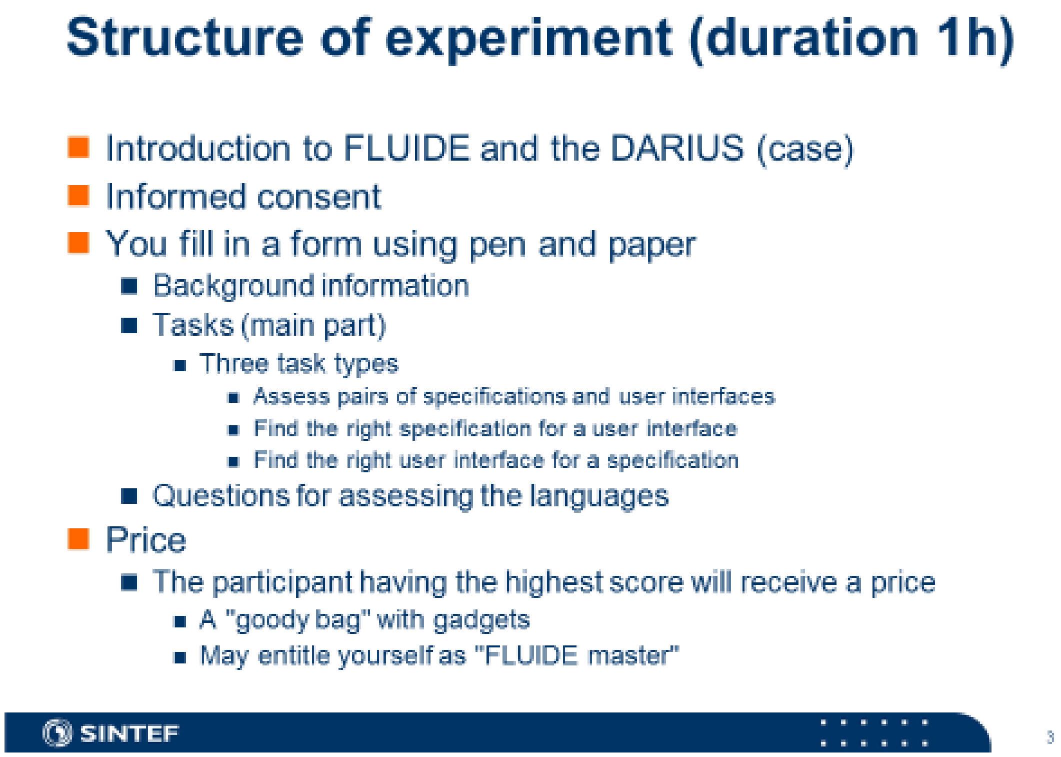
This appendix contains the presentation used as a short introduction to FLUIDE, the search and rescue case (with focus on the users and their tasks) and the goal and execution of the experiment. A paper copy of Slide 6 was given to the participants as a guide to the FLUIDE specification languages.

Experiment assessing the FLUIDE Framework

Erik G. Nilsson
Senior scientist, SINTEF
PhD Candidate, IFI

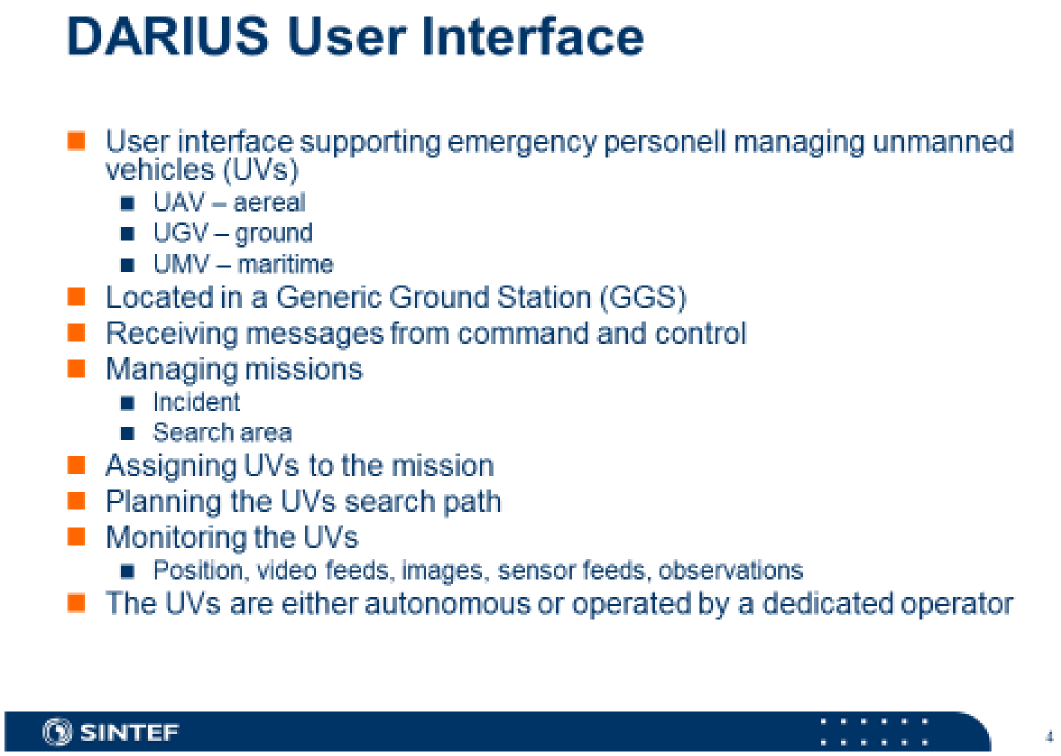
Setting


DARIUS User Interface
FLUIDE specification of the DARIUS User Interface
Today's experiment
Paper
FLUIDE Method
Supports use of
FLUIDE Specification Languages
FLUIDE-A
Is concretized by
FLUIDE-D
FLUIDE library with emergency response user interface patterns



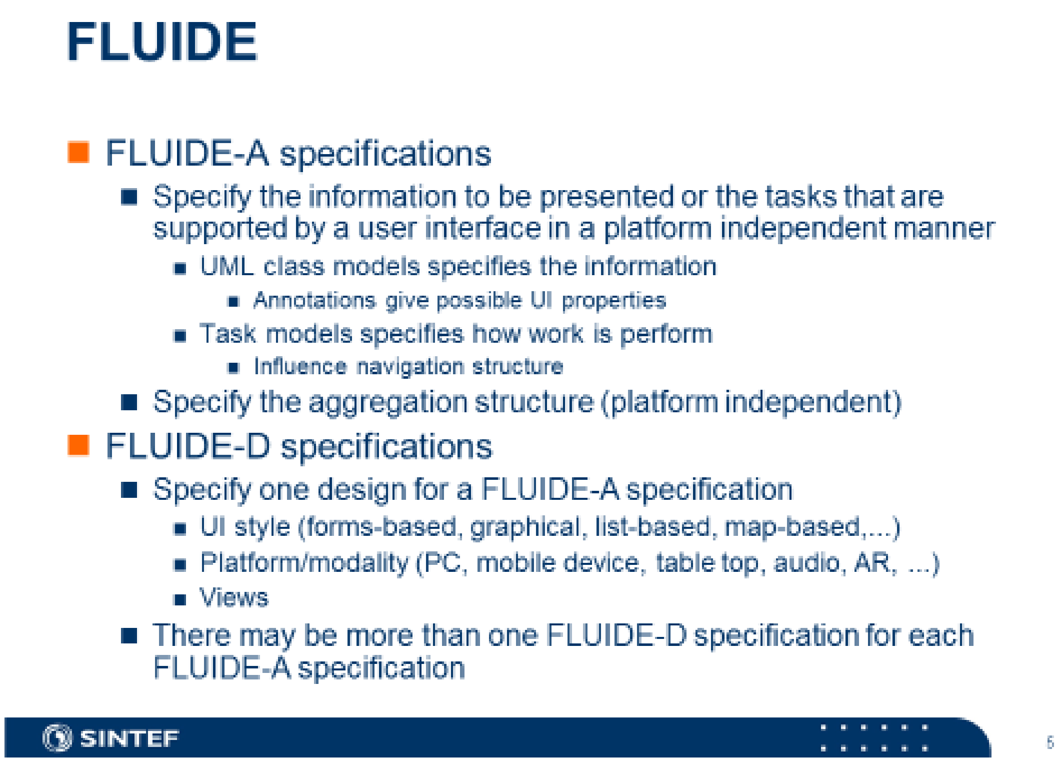
Structure of experiment (duration 1h)

- Introduction to FLUIDE and the DARIUS (case)
- Informed consent
- You fill in a form using pen and paper
  - Background information
  - Tasks (main part)
    - Three task types
      - Assess pairs of specifications and user interfaces
      - Find the right specification for a user interface
      - Find the right user interface for a specification
  - Questions for assessing the languages
- Price
  - The participant having the highest score will receive a price
    - A "goody bag" with gadgets
    - May entitle yourself as "FLUIDE master"



DARIUS User Interface

- User interface supporting emergency personell managing unmanned vehicles (UVs)
  - UAV – aereal
  - UGV – ground
  - UMV – maritime
- Located in a Generic Ground Station (GGS)
- Receiving messages from command and control
- Managing missions
  - Incident
  - Search area
- Assigning UVs to the mission
- Planning the UVs search path
- Monitoring the UVs
  - Position, video feeds, images, sensor feeds, observations
- The UVs are either autonomous or operated by a dedicated operator

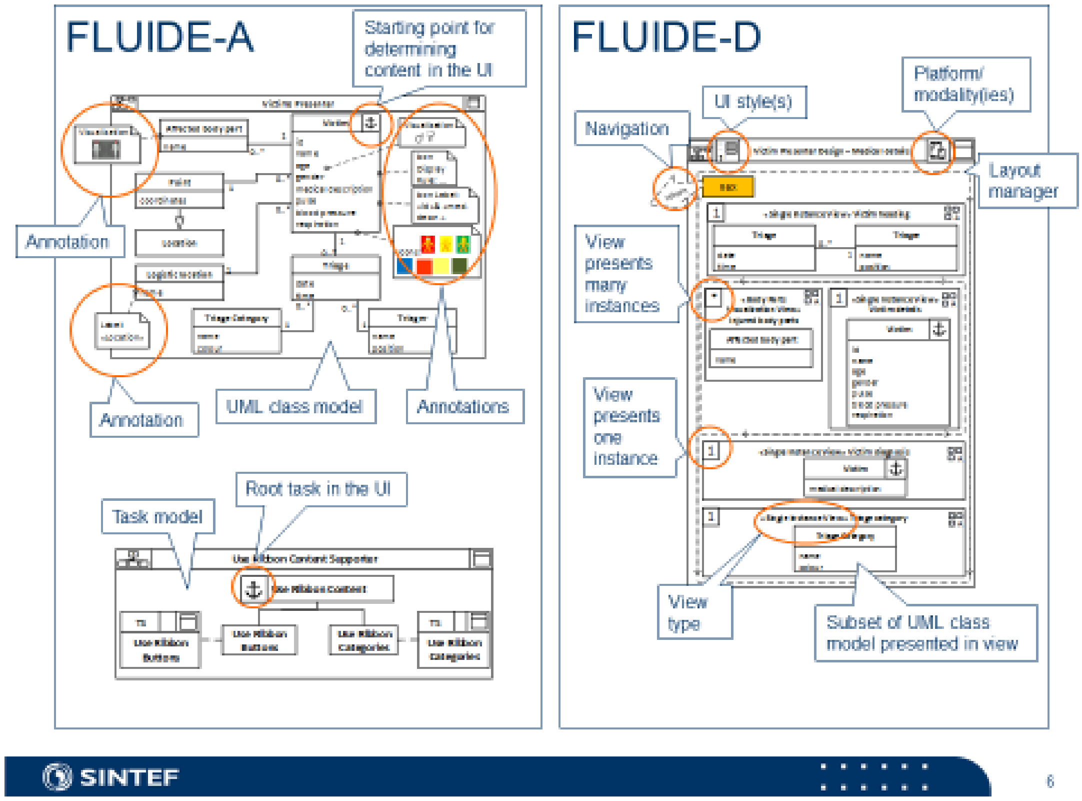


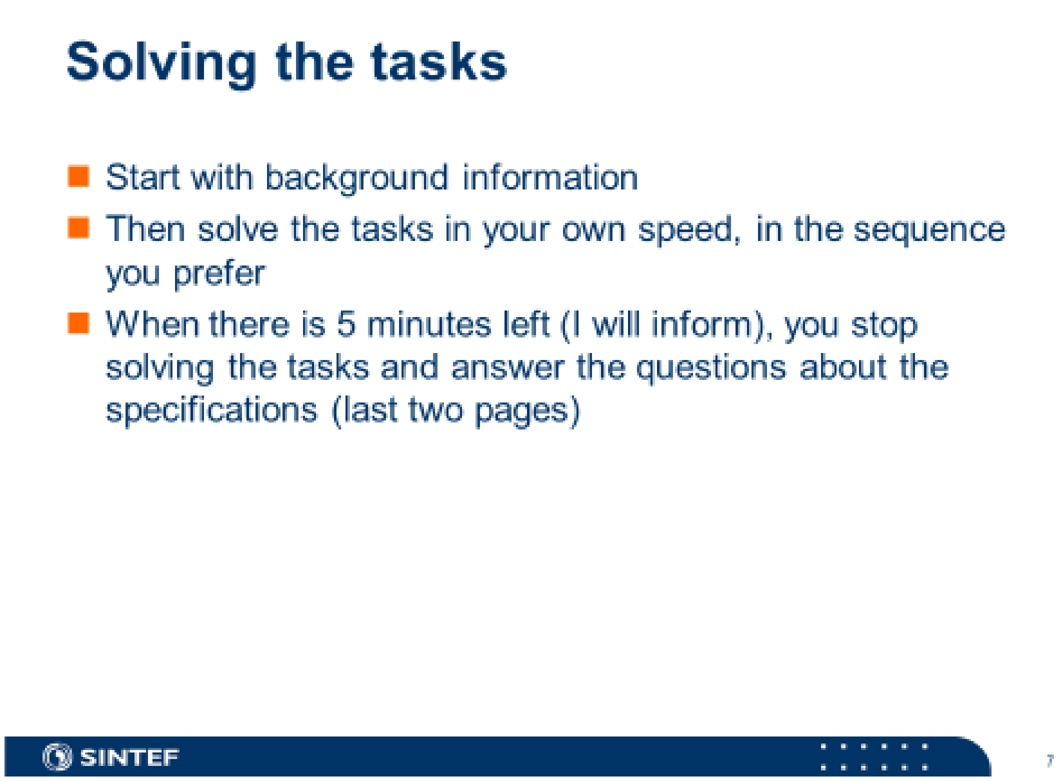
FLUIDE

- FLUIDE-A specifications
  - Specify the information to be presented or the tasks that are supported by a user interface in a platform independent manner
    - UML class models specifies the information
      - Annotations give possible UI properties
    - Task models specifies how work is perform
      - Influence navigation structure
  - Specify the aggregation structure (platform independent)
- FLUIDE-D specifications
  - Specify one design for a FLUIDE-A specification
    - UI style (forms-based, graphical, list-based, map-based,...)
    - Platform/modality (PC, mobile device, table top, audio, AR, ...)
    - Views
  - There may be more than one FLUIDE-D specification for each FLUIDE-A specification

SINTEF

5

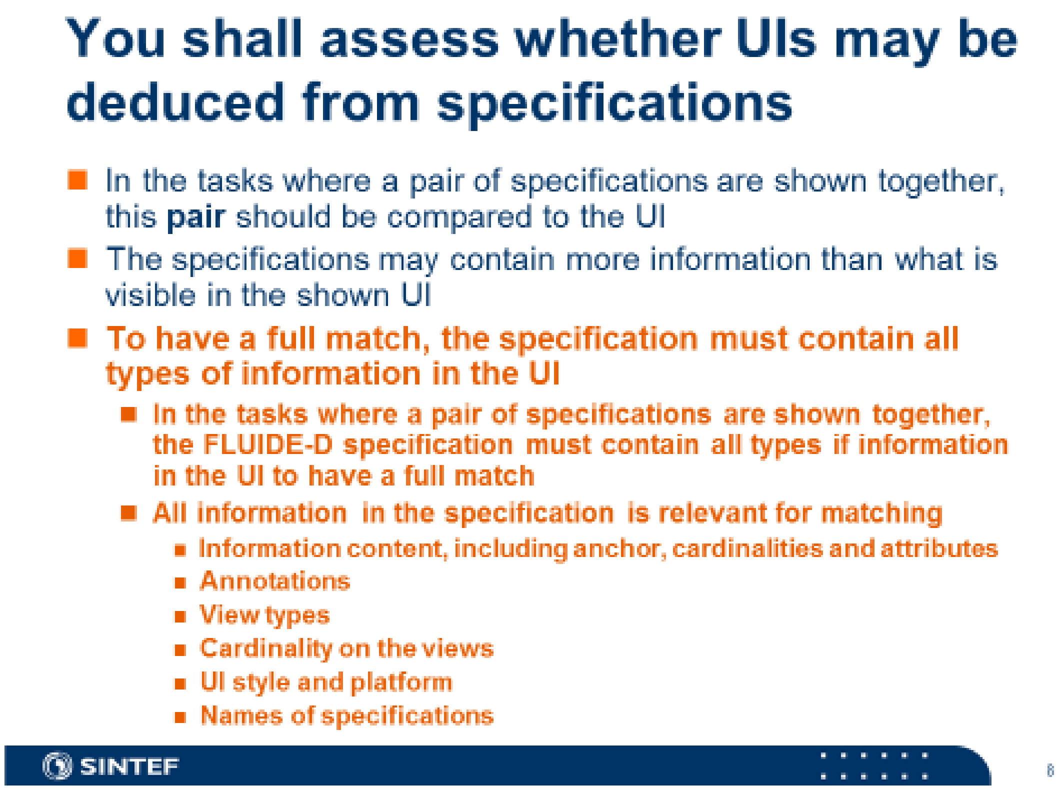




Solving the tasks

- Start with background information
- Then solve the tasks in your own speed, in the sequence you prefer
- When there is 5 minutes left (I will inform), you stop solving the tasks and answer the questions about the specifications (last two pages)

7



You shall assess whether UIs may be deduced from specifications

- In the tasks where a pair of specifications are shown together, this **pair** should be compared to the UI
- The specifications may contain more information than what is visible in the shown UI
- **To have a full match, the specification must contain all types of information in the UI**
  - **In the tasks where a pair of specifications are shown together, the FLUIDE-D specification must contain all types if information in the UI to have a full match**
  - **All information in the specification is relevant for matching**
    - Information content, including anchor, cardinalities and attributes
    - Annotations
    - View types
    - Cardinality on the views
    - UI style and platform
    - Names of specifications

8

SINTEF

Technology for a better society

www.sintef.no