Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Using Ontologies for Solving Cross-Domain Collaboration Issues

Matthieu Perin * **Laurent Wouters** *

* *CEA, LIST, Lab. of Model Driven Engineering for Embedded Systems, F-91191 Gif-sur-Yvette, France (e-mail: matthieu.perin@cea.fr, laurent.wouters@cea.fr).*

Abstract: In a Model-Driven Engineering methodology, the design phase of a product involves the creation of models in multiple domains. A current industrial issue is the integration of domain-specific models in order to homogeneously design the product from the different perspectives. For example, testing a product (e.g. a plane) based on its model is easier and cheaper, but requires the domain-specific representations to be integrated. This paper proposes to solve this integration issue through the use of OWL ontologies. It emphasizes the importance of taking care of the behavioral semantics of the domain-specific models and provides a solution based on the xOWL language, used for the expression of behaviors within ontologies. This paper is supported by a case study, extracted from the industry, about the multiple domains of expertise involved in the design of an aircraft's cockpit.

1. INTRODUCTION

Domain-Specific Modeling Languages (DSMLs) are widely used by experts to describe concepts, objects and even behaviors during the early design phase of a product. Due to historical choices, domain-specific languages are generally well adapted for their own domain but lack possibilities of exchange with other DSMLs (Venkatesh et al. [2003]). As the complexity of design increases, some issues of communication and collaboration across related domains also appear more frequently. Previous works have proposed methodologies and languages, sometimes based on ontologies, to allow the translation of models from a domain to another (Herdebolle [2008], Baudry et al. [2011]).

When faced with multiple domain-specific languages, the heterogeneous modeling approaches in Figure 1 focus on the integration of models expressed in the languages. This integration can be difficult and has to be repeated each time we need a global model integrating all the models expressed in the different domain-specific languages. Instead, we can focus on the integration of the languages themselves so that all models would be expressed at the same time in compatible languages. This approach called the syntactic and semantic merge of the languages is summarized in Figure 2. In a syntactic and semantic merge approach, it is possible to express a single model that readily spans all the domains addressed by the original DSMLs. A consequence of a syntactic and semantic merge approach is that it will avoid the issue of heterogeneous modeling because there is only one kind of model, always expressed in the syntactically and semantically merged modeling language spanning multiple domains. In a sense, a syntactic and semantic merge approach brings the heterogeneous modeling issue at the language level in order to avoid it afterwards at the level of the models.

The term "syntactic and semantic merge" hints at the fact that the approach is really two-folds: First, all the models produced by the experts must be linguistic instances of the same modeling language. Otherwise, this is equivalent to having heterogeneous models that need to be integrated. This issue is called the **syntactic merge** of the DSMLs. Second, the DSMLs must be semantically aligned so that concepts in one can match concepts in another. This is required as the DSMLs will have connections between each other.

Theses solutions have proven to be useful despite the behavioral side of DSMLs generally being comments. The work presented here is based on an extension of the OWL ontology language named xOWL (Wouters and Gervais [2011]). It allows behavioral interactions between models built with different DSMLs. The methodology is presented in Figure 4.

This paper presents an example of behavioral interactions between domain-specific models to solve a cross-domain issue in the avionic domain. Section 2 first briefly presents

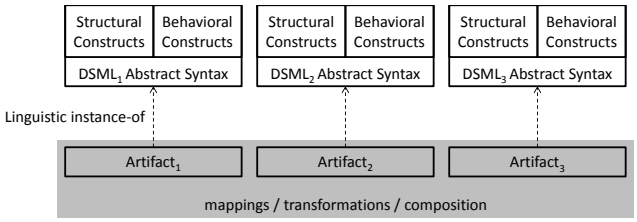

Fig. 1. Heterogeneous Modeling

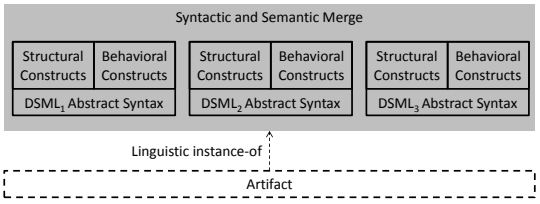

Fig. 2. Syntactic and semantic merge of DSMLs

the possible methodologies for this issue, as well as how it is solved using the xOWL approach. Section 3 then presents is application to a case study, extracted from the aeronautic industry, about the multiple domains involved in the design of an aircrafts cockpit. Section 4 finally concludes and gives some perspectives on the present proposal.

2. METHODOLOGIES

As presented in the previous section, this paper focuses on solving the cross-domain collaboration issue through the integration of the Domain-Specific Modeling Languages themselves, as opposed to the integration of the models expressed in them. The expected benefits are 1) a gain in formal correctness by focusing on the semantics of the actual languages and 2) a trivial communication between the domain-specific models, which will have to be repeated throughout the design phase of a product.

2.1 Syntactic and Semantic Merge of DSMLs

The semantic alignment of the DSMLs is called the **semantic merge**. It means that all the DSMLs share common semantics ensuring the consistency of models spanning multiple domains. A good starting point to achieve this objective is to represent the DSMLs as ontologies. This reduces the issue to the one of ontology mapping.

The mapping of ontologies is defined in Kalfoglou and Schorlemmer [2002] as "the task of relating the vocabulary of two ontologies that share the same domain or discourse in such a way that the mathematical structure of ontological signatures [...] are respected." The **semantic** merge of DSMLs using ontology mappings then consists in expressing the abstract syntaxes of the considered DSMLs in an ontological language, for example OWL, and finding the adequate transformations. An important point of ontology mapping approaches is that the behavioral aspects of the different DSMLs are ignored in that the complete abstract syntaxes of the DSMLs are expressed using an ontological language that only maps to declarative semantics. Two broad classes of approaches address the mapping of ontologies:

Ontology Merging, which consists in merging all the considered ontologies into a single one Fernández-Breis and Martínez-Béjar [2002], Calvanese et al. [2001], Stumme and Maedche [2001], Kalfoglou and Schorlemmer [2002], Noy and Musen [2000], Doan et al. [2004]. The approaches in this category perform a **syntactic** and **semantic** merge by resulting in a single ontology representing the integrated abstract syntaxes of all the considered DSMLs. Semantic Bridges, which consists in the definition of semantic mappings (or bridges) between the considered ontologies Maedche et al. [2002], Kiryakov et al. [2001], Mitra and Wiederhold [2002], Compatangelo and Meisel [2002]. The approaches in this category only perform a **semantic** merge because they only focus on the semantic alignment of the DSMLs. In the Model-Driven Engineering community, several works also have identified and addressed the problem of the **syntactic** and/or **semantic** merge of multiple DSMLs. The earliest work in this regard is the introduction of Semantics Units in Chen et al. [2005]. In this approach, the authors define the notation of Semantic Unit as a gateway for the specification of formal declarative and operational semantics for DSMLs. Semantics Units are always mapped to the same semantic domain, thus realizing their semantic merge. Given a set of DSLMs, the approach consists in building a set of Semantic Units, which semantics can be merged.

The approach introduced in Kappel et al. [2006], Kramler et al. [2006] goes a bit further than the previous one. With the same problem in mind, the authors also define a mapping of the DSMLs' abstract syntaxes onto ontologies in a process called "ontology lifting". The corresponding ontologies are then bound to an upper ontology called the "generic ontology" in this context. Hence, the semantic merge of the DSMLs is achieved through the mapping of their abstract syntaxes first onto ontologies for their respective domain and then to a generic ontology, which is given formal semantics through the ontology modeling language (OWL for example).

The ontology mapping approaches enables the semantic merge of the structural aspects of multiple DSMLs but do not address the behavioral aspects. The metamodel-based approaches presented here address the problem in the case where the DSMLs' abstract syntaxes are expressed in the MOF or UML languages. Aside from the approach based on Semantic Units, the others also rely on the translation or the mapping of these abstract syntaxes to ontologies, but somehow fail to use the ontology mapping techniques from the Semantic Web community. These approaches all have drawbacks that prevent them to be applicable in their current form to the problem at hand.

To conclude, an approach for the semantic merge of multiple DSMLs with both structural and behavioral aspects could be achieved through the reuse of an ontology merging approach extended for dealing with behaviors. The ontology merging approach would readily take care of the syntactic and semantic merge problems for the structural aspect.

2.2 DSMLs Integration with xOWL

We then aim at filling in the gap by extending the OWL language for the expression of ontologies with constructs for the expression of behaviors. Doing so we can 1) rely on the existing Ontology Merging approaches and 2) address the complete range of DSMLs with behavioral semantics. For this purpose we propose the xOWL language introduced in Wouters and Gervais [2011].

The OWL2 language's abstract syntax defines what can be expressed in an *ontology*, which is named by an IRI and contains *axioms*. It is critical to understand that OWL2 is only able to express these axioms, which states what is true in the domain about the entities represented by their name (IRI). These axioms are units of information within an ontology. It is not possible in OWL2 to describe finer grain information than axioms, even though axioms themselves can be expressed in complex manners. The OWL2 language proposes a wide range of axioms for expressing complex hierarchies of classes and properties, as well as complex classes using so-called class expressions.

The xOWL language extends OWL2 with an action language designed for the expression of algorithms. For

this purpose it exposes classical imperative statements such as branch instructions and loops, with the addition of some functional features, lambda expressions in this case.For example, the xOWL language is able to express the algorithm for the simulation of state machines that will be used in this paper.

The state machine used in this work are close to the UML Ones (OMG [2004]), complete semantic will not be presented here but a short presentation is given below. State machine are based on States -including an initial one with an incoming non-sourced transition- and Transitions that can be fired when: their source state is active, their guard -in brackets, using | as a logical or and & as a logical and- are satisfied -equation evaluated to true- and the incoming signal is present. The results of a transition firing are the active state change -from source state to target state- and the outputs -after the / symbol, separated with a coma- application: assignations are taken into account -described using the := symbol- and output signal are emitted. In the figure 3, the initial states of state machines A and B are respectively states 1 and A. To take transition from State 1 to state 2, the Signal sig1 must me present (emitted by another state machine) and ,as no guard is present, the outputs will be applied: Boolean variable A will be set to true and signal Sig2 will be emitted, leading to the firing of transition from state A to State B in state machine B.

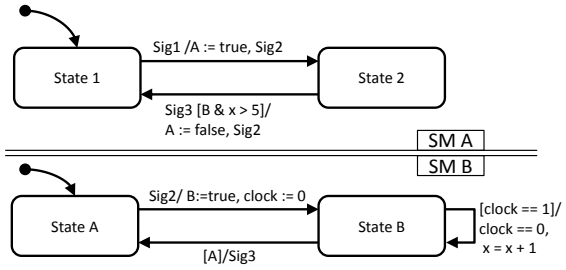

Fig. 3. Example of state machine A and B

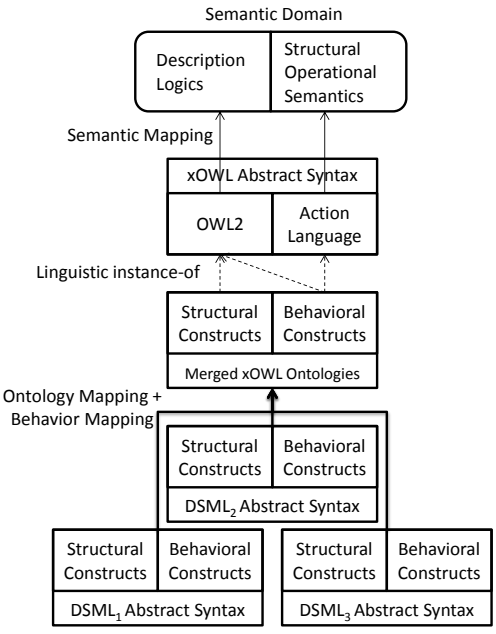

Fig. 4. Semantic integration of DSMLs with xOWL

In the present approach, summarized in Figure 4, the syntactic and semantic merge of the considered DSMLs is achieved with the following steps. First, the software engineers need to express the abstract syntaxes of all the DSMLs as xOWL ontologies, including the behavioral elements using the xOWL action language. Second, the semantic merge of the structural features, i.e. pure OWL2, of the ontologies is realized. This includes the structural representation of conceptually behavior-related concept. At this point, it is possible to rely on any of the ontogy mapping approaches identified above. xOWL is agnostic regarding the particular method that is used for this purpose. Third, the semantic merge of the behavioral features is realized. This is usually achieved through the definition of new pieces of behavior that will make bridges between the behaviors of the different domain. Fourth, the syntactic merge of the DSMLs is achieved by considering that the representation of all the abstract syntaxes as xOWL ontologies, plus the bridges, define a coherent and integrated abstract syntax of a single modeling language that spans all the original domains.

In addition, this approach does not use ontologies only as a representation of the abstract syntaxes of the DSMLs, but also as a representation of the models conforming to them. Each domain model expressed in one of the DSML is also in fact expressed as an ontology. The linguistic relationships between the models and the DSMLs are materialized in the ontologies as ontological instantiations. Doing so, this approach has the following properties: First, the syntactic merge of the DSMLs is easy to realize because all models will be expressed as ontologies and the merged xOWL ontologies of the DSMLs' abstract syntaxes are simply "upper" ontologies. Second, the syntactic merge in this approach ensures that models coming from different domains are still separated and can still be used separately with only the language for their respective domain. This is due to the fact that the xOWL ontologies for the DSMLs are not modified by the merge process, only new information (the bridges) is added. These properties will be made more explicit in the following Section that validates this approach by applying it on the use case presented in Section 3.

3. CASE STUDY: AN AVIONIC COCKPIT

An important property of aircrafts is that they are designed to be used by professionals with adequate training. The professionals (i.e. the pilots) have to interact with the aircrafts through specific procedures. The procedures are designed, validated and certified to ensure the safety of both the aircraft and its operators. A striking property of the procedures is that they are themselves safety-critical. In this point of view, even the most simple action -like pushing a button- has to be considered a part of a complete safety-related context. The completion of any procedure also depends of the level of stress of the pilots, which is also related to the easiness of manipulating the buttons.

The case study in this paper is how such a simple button, one of the core elements of the Cockpit Interfaces Domain, has to be taken into account in the Pilot's Procedures Domain and the Human Sciences Domain due to the level of stress induced by its manipulation. The three domains presented below are about different aspects of

the same central concern, the representation of human-machine interactions in a cockpit. The objective of the domain experts is to put together their work so that it is possible to have cross-domain analysis of the human-machine interactions. In this paper, the evaluation of a procedure in term of stress impact for the pilot will be the cross-domain chosen issue. For the sake of readability, all the behaviors presented in this section are using a state machine formalism although it may not be the source behavior model used by expert in each domain.

3.1 Cockpit Interfaces Domain

Experts in this domain have to define the behavior of the components within a cockpit. They only consider the independent behavior of each category of components. In this case study, only the behavior of *Guarded Button* will be used, it is described as follow: The default state is Guarded and Inactive. When a user Lift the Protection, it becomes Unguarded. When Unguarded a user can Push the button, which makes it cycle through the Inactive and Active states. This kind of behavior is best represented with state machines. For example the button for navigation Lights is shown in Figure 3.1. This state machine has four states (NavLightsGuardedInactive, NavLightsUnguardedInactive, NavLightsUnguardedActive, NavLightsGuardedActive), reacts to three input signals (P_NavLights_LIFT, P_NavLights_LOWER -P stands for protection here-, and B_NavLights_PUSH -B stands for Button here-). Depending on the state, the light associated to the button is visible -L_NavLights for Light of Navigation Lights- and the button is protected -P_NavLights for Portection of Navigation Lights- or not. This behavior essentially focuses on the physical low level description of the button and is based on concept such as **Physical action**, **Light**, **Button** and **Protection**.

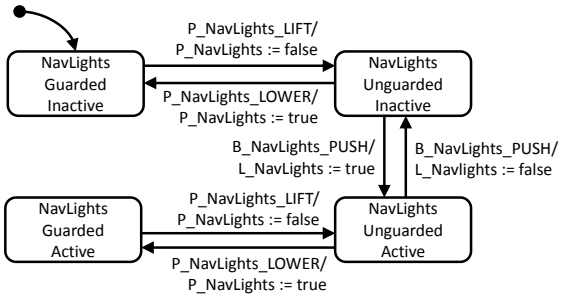

Fig. 5. Example of state machine for Navigation Lights

3.2 Pilot's Procedures Domain

Experts in this domain build procedures with varying levels of details. Some low-level procedures describe how a pilot can access a certain value in the embedded software. These procedures specify in details which cockpit components the pilot has to activate in order to complete a procedure. Low-level procedures will be reused by the experts to define more high-level procedures describing the achievement of high-level goals in the cockpit. Pilot's procedures may not be unique to achieve a high-level goals: domain experts need to represent this variety as the chosen one may vary from the safest to the quickest depending on the situation. Therefore, domain experts need to evaluate the time consumption and the stress induced by a procedure to choose the best suited.

In this case study the procedure to respond to a wing fire will be used. It is expressed as follow: Switch OFF Navigation Lights, Switch OFF Strobes Lights, Switch OFF Pilot Heat (heating of pilot probes in the wings), Ti side Slip away from fire and to Land ASAP. This procedure has been described using the Domain-Specific Language and then translated into the state machine presented in Figure 6 using xOWL. In this state machine, the seven states describe the several steps of the procedure (starting from ProcedureStart, trough Step1Complete, Step2Complete, Step3Complete, Step4Complete until the last step ProcedureComplete closing to finally ProcedureExited) and emit signals to act represents needed action from the pilot (NavLights_OFF, Strobes_OFF, PilotHeat_OFF, SideSlip, ELandingReq -for Emergency Landing Request-) and to declare starting and ending of the procedure (EmerProc_Start -for Emergency Procedure Start- and Proc_Exit). This behavior focuses on high level interactions between pilots and the aircraft, concepts such as **Procedure** and **Pilot action** are used.

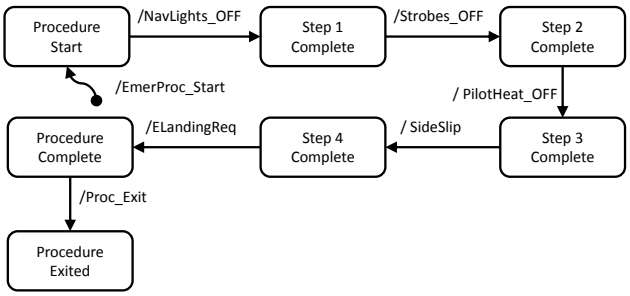

Fig. 6. Pilot Procedure

3.3 Human Sciences Domain

Human Sciences do not form a single and consistent domain. However, in the context of this use case, the term refers to a small cluster of fields: physical and cognitive ergonomics. Physical ergonomics focuses on the tangible aspects of the human-machine interfaces and uses studies to validate that all the interfaces are easily accessible to the pilots from their seat.

The actuation sub-domain describes how humans make changes to their environment. The experts in this sub-domain represent the human limbs' capabilities. For example, human hands are able to PUSH, PULL, ROTATE, GRIP and so on. In this case study, this domain-specific language will explain the link between an pilot action a and the physical and logical actions that as to be performed -and associated stress-, in relation with the chosen technology in the cabin. The result is presented here in the form of a state machine in Figure 7. It is limited to the actions to be performed when the pilots have to check that Navigation lights are OFF (input signal NavLights_OFF), states represents the state of mind of the pilot in relation with the asked action (start at Idle, then evolve to NavLightsToCheck, NavLightsStateKnowed, and then depending on the status of the Navigation Lights Button: NavLightsInactive, NavLightsUnguardedInactive, NavLightsUnguardedActive, and

NavLightsGuardedActive). The pilots will perform actions related to the global environments (H_Obs_NavLights -stands for Human Observe Navigation Lights button status- and H_MovTo_NavLights -stands for Human Move Hands to reach the Navigation Lights button-) and the button manipulation (NavLights_P_LIFT, NavLights_P_LOWER, and NavLights_B_PUSH). When observing the Navigation Lights button, pilots will determine is status using human related variables NavLights_A -Navigation Lights Active-, NavLights_NA -Navigation Lights Non Active-, NavLights_B_P -navigation Lights Button Protected-, and NavLights_B_NP -navigation Lights Button Non Protected-. For each action to perform, a stress level increase (S_A_LVL -Stress Addition Level-) is set and and output (ActionPerfom) is send to signal that an action as been taken (and thus some stress addition level have to be taken into account). This part of the domain mainly use concepts around **Human physical action**, **Human observation**, and **Stress**.

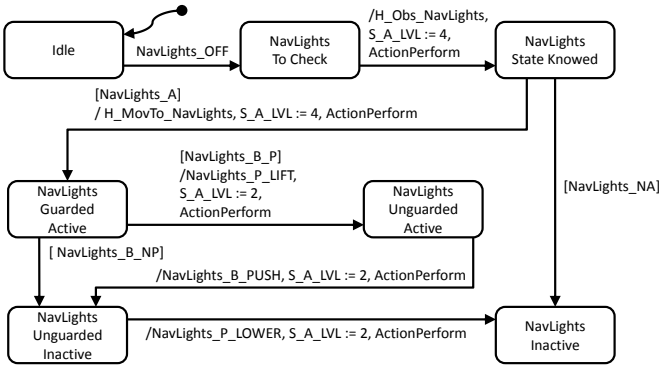

Fig. 7. Human action related to the switching of a button StateMachine.

The cognitive ergonomics field focuses on the analysis of factors such as stress, fatigue and workload on the pilot' performances: The heavier the workload is; the likelier pilots are to commit errors. The cognitive ergonomics ensure that the workload is still reasonable for the pilots and is evenly distributed between the flying and non-flying pilots. The state machine shown in Figure 8 represents the evolution of the stress level of a pilot performing actions without any break time. The stress of the pilot during the execution of a procedure is modeled by states (NoProc when no procedure are used, ProcStart when a procedure has started and from NoStress, LowStress, High Stress to CriticalStress), the stress levels are defined using a stress level variable (S_LVL, using guard with limits from 25, 60 to 80) incremented each time an action is performed (input signal ActionPerform, the stress addition Level is added to the stress Level). If the stress level reaches a maximum level, a critical stress state of mind is reached and the safety is no more guaranteed (forbidden state). The stress level decreases with time when the pilot is idle (S_LVL lose S_R_LVL -for Stress remove Level- any DT time delay). Starting an emergency procedure is itself a stressful action (has something visibly go wrong during the flight) and thus will rise the stress level (when input signal EmerProc_Start arrive, the stress level is raised by 25). This other part of the domain uses on concepts such as **Stress Level**, **Human rest behavior**, and **Human reaction**.

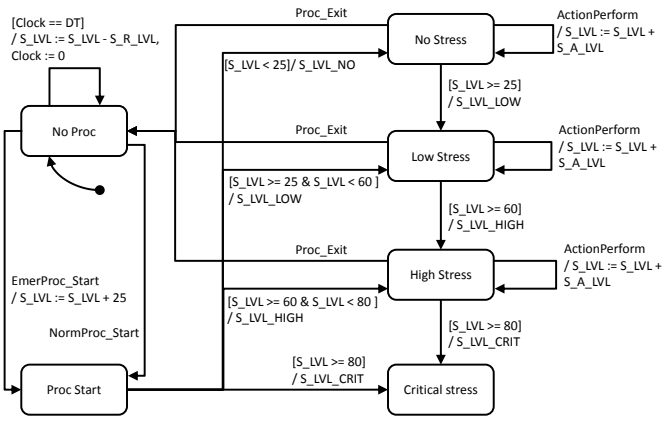

Fig. 8. Modeling a pilot's stress level

3.4 Domains Integration

For each domain, a xOWL ontology representing the abstract syntax of the corresponding DSML is built. Up to now, the three xOWL ontologies are independent and represent domains that are not related (in their ontological representations). The semantic merge of these domains consists in their integration into a single consistent xOWL ontology. In this example this is achieved by the description of the relations between concepts from different domains. For more complex applications, the use of an existing approach to ontology merging would be useful. Because there are only three domains in this application case, their semantic integration has been achieved pairwise.

Pilot's Procedures and Cockpit Interfaces The Pilot's Procedures domain does not go too deep into the details of what exactly are the entities that are interacting. It is clear that there should be a relation between the **Pilot action** concept and the **Physical action**. In practice, based on the technology of the **Button** and **Protections**, the **Pilot action** will be detailed into sets of possible **Physical action**. The fact that **Pilot action** also consider the state of the **Button** (switching OFF a button included the knowledge of the state of the button) implies a link between the **Pilot action** concept and the **Light** concept. In our case study, the **Pilot action** NavLights_OFF will be related to the NavLights **Button**.

Cockpit Interfaces and Human Sciences The Cockpit Interfaces and Human Sciences domain are nearly disjoint. Nevertheless, in order to be able to decide what to do, pilots base their **Human physical action** on their **Human observation**, which in fact depends on the **Light** and **Protection** state of the **Buttons**. In our case study, for example the **Human observation** NavLights_A is related to the value of variable L_NavLights of the **Light** of the NavLights **Button**. The **Human physical action** NavLights_P_LIFT is to be linked to the **Physical action** P_NavLights_LIFT of the **Protection** concept.

Pilot's Procedures and Human Sciences On one hand, the Pilot's Procedure domain defines the **Procedure** as a set of **Pilot action** that as to be performed. On the other hand, the Human Sciences domain defines the **Human physical action** and **Human observation** humanly possible. In the context of the application case, the human pilots will

uses both physical actions and observations to perform the high level actions described in the procedure. Based on the set of low level physical and logical actions taken by the pilots, the stress level increase associated to a procedure may be calculated using the concepts of **Stress**, **Stress Level**, **Human rest behavior**, and **Human reaction**. The **Pilot action** NavLights_OFF is in fact the complete behavior of **Human physical action** and **Human observation** represented in Figure 7.

Solving stress level issue Once the links between all concept are effective, the whole set of behaviors are simulated to determine the stress level increase due to the emergency procedure for wing fire. The impact of using a guarded button may be measured, as for the order of the procedure in regard to the movement -and associated stress increase- needed to go from one button to another.

4. CONCLUSION

The approach presented in this paper allows software engineers to integrate multiple DSMLs so that experts from different domains can build models conceptually expressed in all of them at the same time. Doing so we will avoid having to integrate heterogeneous models (expressed in different languages) when we need to run tests and/or simulations. The main advantage of this approach is that it builds on the existing ontology merging approaches with the added features for behaviors. The integration of the DSMLs is a one-time effort whereas the integration of heterogeneous models each time a global artefact is needed can be more time and/or resource consuming.

Fortunately, by leveraging the expressiveness of the OWL2 language, software engineers are able to express complex relations between different DSMLs. For example, a class in domain A can be equivalent to the intersection of two other classes in domains B and C. Still some relations cannot be expressed. For example, a single class in domain A that corresponds to the aggregation of two other classes in domain B.

The main drawback is then that the integration of the DSMLs has to be achieved before any model can be expressed. This means that it is difficult to integrate in this approach models that have already been expressed in one of the DSMLs because this is going back to the heterogeneous modeling approach. Hence, it is often not possible to reuse models that have been expressed in a DSML prior to its integration with the other considered DSMLs. In an industrial context where this approach is applied, this can become problematic. However, bridging the gap between the approach proposed here and the heterogeneous modeling approaches is a research objective.

REFERENCES

B. Baudry, B. Combemale, J. DeAntoni, and F. Mallet. Study on Heterogeneous Modeling, Action Spcifique 2011 du GDR GPL. Technical report, Centre National de la Recherche Scientifique, 2011.

D. Calvanese, G. De Giacomo, and M Lenzerini. Ontology of Integration and Integration of Ontologies. In *International Description Logics Workshop*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2001.

K. Chen, J. Sztipanovits, S. Abdelwahed, and E. K. Jackson. Semantic Anchoring with Model Transformations. In *ECMDA-FA*, 2005.

E. Compatangelo and H. Meisel. Intelligent Support to Knowledge Sharing Through the Articulation of Class Schemas. In *In Proc. of the 6th Intl. Conf. on Knowledge-Based Intelligent Information & Engineering Systems*. IOS Press, 2002.

A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies*, International Handbooks on Information Systems. Springer, 2004.

J. T. Fernández-Breis and R. Martínez-Béjar. A Cooperative Framework For Integrating Ontologies. *International Journal on Human-Computer Studies*, 56:665–720, 2002.

Cécile Herdebolle. *Composition de Modles pour la Modlisation Multi-Paradigme du Comportement des Systmes*. PhD thesis, Universit Paris-Sud, 2008.

Y. Kalfoglou and W. M. Schorlemmer. Information-Flow-Based Ontology Mapping. In *CoopIS/DOA/ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1132–1151. Springer, 2002.

G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger, and M. Wimmer. On models and ontologies - a semantic infrastructure supporting model integration. In *Modellierung*. GI, 2006.

A. Kiryakov, K. Ivanov Simov, and M. Dimitrov. OntoMap: Portal for Upper-Level Ontologies. In *FOIS*, 2001.

G. Kramler, G. Kappel, T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger. Towards a semantic infrastructure supporting model-based tool integration. In *Proceedings of the 2006 international workshop on Global integrated model management*. ACM, 2006.

A. Maedche, B. Motik, N. Silva, and R. volz. MAFRA - A MApping FRAmework for Distributed Ontologies. In *Knowledge Engineering and Knowledge Management*, volume 2473 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2002.

P. Mitra and G. Wiederhold. Resolving Terminological Heterogeneity in Ontologies. In *ECAI Workshop on Ontologies*, 2002.

N. Fridman Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 150–455. AAAI Press / The MIT Press, 2000.

OMG. Unified Modeling Language. Technical Report 2.0, OMG, 2004.

G. Stumme and A. Maedche. Ontology Merging for Federated Ontologies on the Semantic Web. In *Proceedings of the International Workshop for Foundations of Models for Information Integration*, 2001.

V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis. User acceptance of information technology: Toward a unified view. *MIS Quaterly*, 27:425–478, 2003.

L. Wouters and M.-P. Gervais. xOWL an Executable Modeling Language for Domain Experts. In *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2011.