Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 37 (2014) 473 – 480

Procedia Computer Science

1st International Workshop on Metamodelling for Healthcare Systems (MMHS–2014)

Co-ordination of Multiple Metamodels, with Application to Healthcare Systems

Fazle Rabbi[*,a,b], Yngve Lamo[a], Wendy MacCaull[c]

[a]*Bergen University College, Bergen, Norway*
[b]*University Of Oslo, Oslo, Norway*
[c]*St. Francis Xavier University, Antigonish, Canada*

Abstract

Due to their complexity and the plethora of requirements placed upon them, healthcare systems so far have not been adequately modeled for the purpose of software development. As a result, the healthcare software suffers from high development costs and lack of flexibility. Model driven software engineering (MDSE) is an emerging methodology for software development, targeting productivity, flexibility and reliability of systems; metamodelling is at the core of most MDSE approaches. In previous work, we proposed a multi metamodelling approach that captures the complexity of these systems by using a metamodelling hierarchy, built from individually defined metamodels, each capturing different aspects of a healthcare domain, namely, user access modelling, health process modelling, process monitoring, user interface modelling and modelling of the data sources. Here, we formalize the co-ordination among metamodels, using a linguistic extension of the metamodelling hierarchy. This linguistic extension is an added metalevel which models the integration of two or more different aspects of the system. We focus on two features essential to the co-ordination of healthcare metamodels, namely the integration of process and data, modelling data-aware processes, and the integration of process and user, modelling both user access as well as inheritance of user access to tasks.

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.

Keywords:
multi metamodelling; co-ordination among metamodels; linguistic extension; model driven software engineering; healthcare information systems

1. Introduction

Rising costs, ageing populations and increased expectations are making the current healthcare systems in the developed world unsustainable. Information technology has the potential to support healthcare but its application to support the continuum of care has not nearly reached its full potential. Barriers include the proliferation of systems even within one hospital, often which do not support interoperability; the fact that systems must be highly customized to adequately serve local situations (usually an expensive and error prone process); the fact that the systems are constantly evolving due to new medications, and changing protocols and management strategies; the fact that software engineering itself for such safety critical systems as healthcare needs new strategies to ensure that systems behave

* Corresponding author. *Email address:* Fazle.Rabbi@hib.no

Peer-review under responsibility of the Program Chairs of EUSPN-2014 and ICTH 2014.
doi:10.1016/j.procs.2014.08.071

correctly in every possible scenario; and the fact that in many situations, healthcare is a team process involving many players, each with specific requirements from the technology. Taken altogether, we may say that the current healthcare information systems can be characterized as follows: they are hard to analyze to ensure correctness of process; they have sophisticated tracking and interoperability requirements due to numerous silo-ed data sources; they have high development and maintenance costs; they lack flexibility to adapt to changes and for customization required by different users with different needs; and, finally, the ability to customize different subsystems with different access policies is limited.

Model driven software engineering (MDSE) is an emerging and promising methodology for software development, targeting challenges in software engineering relating to productivity, flexibility and reliability of systems. The construction of various kinds of models (e.g., blueprints, mockups etc.) is a well-known approach in the more traditional engineering fields; these models are used as artifacts to enable engineers to describe designs and validate whether a proposed design has desired qualitative and quantitative properties. We propose to use MDSE in an analogous manner for the development of reliable and robust software systems supporting the diverse, evolving and often safety critical requirements of healthcare. We note that many complex systems are characterized by the features discussed above and as such, the discussion here may be generalized to numerous processes involving complex information systems, such as transportation systems, communication systems, etc.

Many different MDSE technologies automatically generate code from models[1,2]: these technologies are particularly suited for specifying the structural aspects of software systems generally, whereas the actual behavior is programmed manually. Some technologies for behavioural modelling in MDSE exist[3,4], but current approaches are often at a low level of abstraction and lack domain concepts for specifying behavior[5].

A collaborative group of researchers in Norway and Canada have been working on various issues relating to these problems. We describe a formal approach to workflow modelling in[6], based on the Diagram Predicate Framework (DPF)[7] which provides a formalism of (meta) modelling and model transformations based on category theory and graph transformations[8,9,10]. The dynamic semantics of models were described by a transition system where the states are instances and transitions are applications of coupled transformation rules. Our long term goal is to promote automatic generation of complex software systems from models, which can be easily adapted as the need arises; the formal approach permits a variety of formal validation techniques which go a long way toward ensuring correctness.

In his thesis[11], Baarah proposed an application framework for care process monitoring that collects and integrates events from event sources, maintains the individual and aggregate state of the care process and populates a metrics datamart to support performance reporting. He presented a UML-style metamodel for the care process monitoring application that had 3 main components: a process model, a performance model and an enterprise model. This work motivated us to take a more modular approach to our metamodelling, in order to more adequately model the various components involved in a healthcare process. In[12] we proposed that the separation of concerns via the use of multimodelling was an appropriate methodology for designing healthcare information systems. The use of multi metamodels for modelling different aspects of a system facilitates abstraction and require less coupling among the models; this modularity gives flexibility as it permits the independent remodelling of parts of the system. It also reduces development and maintenance costs. The paper outlined several metamodels, each of which comes equipped with its own domain specific language which supports one aspect of the system and outlined where some of the links were needed for co-ordination. We focused on user aspects and monitoring, and the need for flexibility in design of user interfaces to suit the needs of different participants in the healthcare process. Users may interact with parts of the workflows, and parts of the datasource and receive and acknowledge alerts from the monitoring system.

In this paper, we begin the formalization of the co-ordination among the metamodels necessary to integrate two or more aspects during different phases of modelling. We use a linguistic extension of the metamodelling hierarchy, in the form of a metalevel which sits at the top of the multi metamodelling hierarchy and which provides co-ordinating edges which link two different metamodels together with constraints relating these co-ordinating edges to edges in the existing metamodels. This metamodelling formalism can be used to automatically generate program code from models to implement software systems. In section 2, we briefly outline the DPF modelling approach, and use it to describe two metamodels, namely, the process metamodel and the datasource metamodel. In section 3, we describe the enrichment of DPF by defining the linguistic extension of the metamodelling hierarchy which permits the integration of metamodels. We discuss the co-ordination of data, by the integration of the process and datasource metamodels via a co-ordinating edge between them, subject to a commute constraint, ensuring that the data is assigned to the

appropriate patient. We also discuss the co-ordination of user privileges, by integrating the process and user access metamodels by means of a co-ordinating edge between them which models user access and also models inheritance of user access to workflow tasks. In that section, we also give an overview of five aspects of healthcare information systems and some required co-ordinating edges between them. In section 4, we briefly discuss related and future work. Section 5 concludes the paper.

2. Metamodeling in DPF

A traditional modelling hierarchy consists of 4 levels, meta-metamodels, metamodels, models and instances of models[13]. A meta-metamodel represents a modelling formalism, modelling languages are represented by metamodels, software systems are represented by models and possible instances of a software system are represented by instances of models. The metamodel describes the syntax of the modelling language, i.e., it describes the types and relations between the types. Each model must conform to the language's metamodel, i.e., it must respect the typing and other constraints of the language. Finally, instances must conform to models. In contrast to traditional hierarchies, DPF[7] has a potentially unbounded number of metalevels[14]. A model is represented by a diagrammatic specification $\mathfrak{S} = (S, C^{\mathfrak{S}} : \Sigma)$ consisting of an underlying graph S together with a set of *atomic constraints* $C^{\mathfrak{S}}$ specified by a *predicate signature* Σ. A predicate signature consists of a collection of predicates, each having a name and an arity (or shape graph). A predicate (name) imposes a constraint on a portion of the graph S, of the associated arity. DPF provides a formalisation of multi level metamodelling by defining the conformance relation between models at adjacent levels of a metamodelling hierarchy. There are two kinds of conformance: *typed by* and *satisfaction of constraints*.

In[6] we detailed the DERF modeling hierarchy developed to represent processes. Traditional DPF models were used to represent structural (static) aspects of a software system. The DERF modeling hierarchy extended modelling to include behavioural aspects (dynamic semantics); behaviour was defined by a coupled transition system. As an illustration we briefly discuss the process formalizing the Clinical Practice Guideline for the Management of Hypertension[15]. In the management of hypertension, an initial blood pressure (BP) measurement is made; if the BP is normal, the patient is "Safe", i.e., the process stops. If the BP is high, the patient is scheduled for Visit 1, the first visit in the treatment process. An overview of the model of the treatment process is found in the lower portion of Fig. 1, the reader will note the *[XOR SPLIT]* predicate indicating that exactly one of the paths is followed at the Initial BP Measure task. This model uses predicates *[XOR SPLIT]*, *[AND SPLIT]*, *[OR SPLIT]*, etc., to describes the control flow of the process model. The metamodel level provides the abstract modelling language for workflow, and consists of two types, Task, and Flow; the model level details a specific workflow; the instance level gives instances of specific enactments of a workflow. Fig. 1 shows two meta level specifications $\mathfrak{S}_{W_1}$ and $\mathfrak{S}_{W_2}$ for this process (the instance level is not shown). Arrows from the bottom layer to the top layer indicate the typing constraints. In the examples of the DPF modelling hierarchies given here, predicates at each level are described by their name and a "shape graph", for visualization. The metamodel specification $\mathfrak{S}_{W_2}$ of Fig. 1 is similar to a workflow metamodel specification presented in[6]. Space does not permit us to discuss this modelling hierarchy and its dynamic behaviour further; the reader can refer to[6,12] for more details.

Fig. 1 provides a process flow modelling hierarchy for the hypertension managment system but it does not specify the data model of the system. Let us consider the following data requirements from the healthcare domain which we use to describe the datasource modelling hierarchy: (We remark that these requirements are overly simplified from real life applications, but suffice to illustrate the features we wish to discuss here.)

Requirement 1 *An employee (e.g., nurse, doctor) must work for a department.*

Requirement 2 *A department may have zero or more employees.*

Requirement 3 *A ward must be under a department.*

Requirement 4 *An employee who is involved in a ward, must work in the controlling department.*

Requirement 5 *A patient's systolic and diastolic blood pressure records are numbers.*

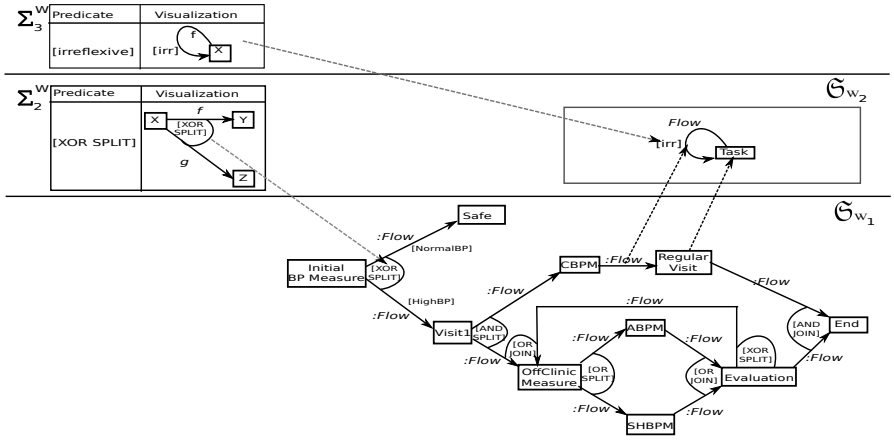


Fig. 1. Hypertension Management Workflow (Overall)

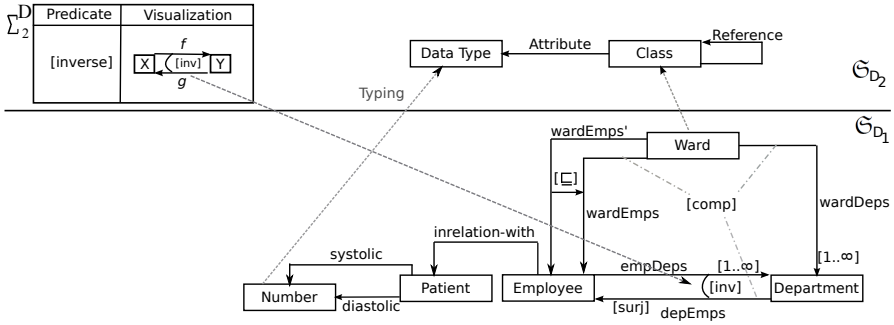


Fig. 2. Example of a datasource metamodel

Table 1. Predicate constraints of the signature Σ_2^D

p	$\alpha^{\Sigma_2^D}(p)$	Proposed Vis.	Semantic Interpretation
[mult(n,m)]	1 —f→ 2	X —f [n..m]→ Y	$\forall x \in X : m \leq \lvert f(x)\rvert \leq n, 0 \leq m \leq n, n \geq 1$
[irreflexive]	f: 1 → 1	[irr] f: X → X	$\forall x \in X : x \notin f(x)$
[surjective]	1 —f→ 2	X —f [surj]→ Y	$\forall y \in Y, \exists x \in X, f(x) = y$
[inverse]	1 ⇄ 2 (f, g)	X —f [inv] g→ Y	$\forall x \in X, \forall y \in Y : y \in f(x)$ iff $x \in g(y)$
[image-inclusion]	1 ⇄ 2 (f, g)	X —f [⊑] g→ Y	$\forall x \in X : f(x) \subseteq g(x)$
[composition]	1 —f→ 2, 1 —h→ 3, 2 —g→ 3	X —f→ Y, X —h [comp] g→ Z	$\forall x \in X : h(x) = \bigcup\{g(y) \mid y \in f(x)\}$

In Fig. 2, we outline two meta level specifications, $\mathfrak{S}_{D_2}$ and $\mathfrak{S}_{D_1}$ of the datasource metamodelling hierarchy that we developed from the above mentioned requirements. In this specification we used the signature $\Sigma_2^D = (P^{\Sigma_2^D}, \alpha^{\Sigma_2^D})$ (Σ_2^D indicates the signature at meta–level 2) from Table 1 to define the set, $C^{\mathfrak{S}_{D_1}}$, of atomic constraints where the first column of Table 1 shows the names of the predicates; the second, third and fourth columns show the arities of

predicates, possible visualizations, and semantic interpretations, respectively (Fig. 2 does not show all of Σ_2^D). The *predicate constraints* are added into the specification by a set of *atomic constraints*; this is depicted in the specification by using the proposed visualization. Requirement 1 is encoded in $\mathfrak{S}_{D_1}$ by the [*surjective*] *predicate constraint* over the morphism *depEmps*; the fourth requirement "an employee who is involved in a ward, must work in the controlling department" is encoded in $\mathfrak{S}_{D_1}$ by two *predicate constraints* [*composition*] and [*image − inclusion*] over morphisms *depEmps*, *wardDeps*, *wardEmps* and *wardEmps*′ where $wardEmps := wardDeps; depEmps$ (the composition of morphisms *wardDeps* and *depEmps*). The atomic constraint for [*composition*] predicate in $\mathfrak{S}_{D_1}$ indicates a morphism $z \xrightarrow{wardEmps} x$ exists *iff* morphisms $z \xrightarrow{wardDeps} y$ and $y \xrightarrow{depEmps} x$ exist, and the atomic constraint for [*image−inclusion*] predicate in $\mathfrak{S}_{D_1}$ indicates that for any Employee x working under a Ward z (i.e., $z \xrightarrow{wardEmps'} x$) there is a morphism $z \xrightarrow{wardEmps} x$. In this situation, *wordEmps* morphisms are explicitly defined while *wordEmps*′ morphisms are derived from the existance of *wordDeps* and *depEmps*.

3. Co-ordination of metamodels

In this section we describe a linguistic extension to the DPF metamodelling hierarchy; this extension is an added metalevel which models the integration of two or more different aspects of the system. The formalization of the linguistic extension, that is, the top layer of the hierarchy with two different types of edges, is inspired by the notion of E-Graph [16]. To illustrate co-ordination, we present two features essential to the co-ordination of healthcare metamodels, namely, the integration of process and data, modelling data-aware processes, and the integration of process and user, modelling user access to tasks in a process as well as inheritance of user access.

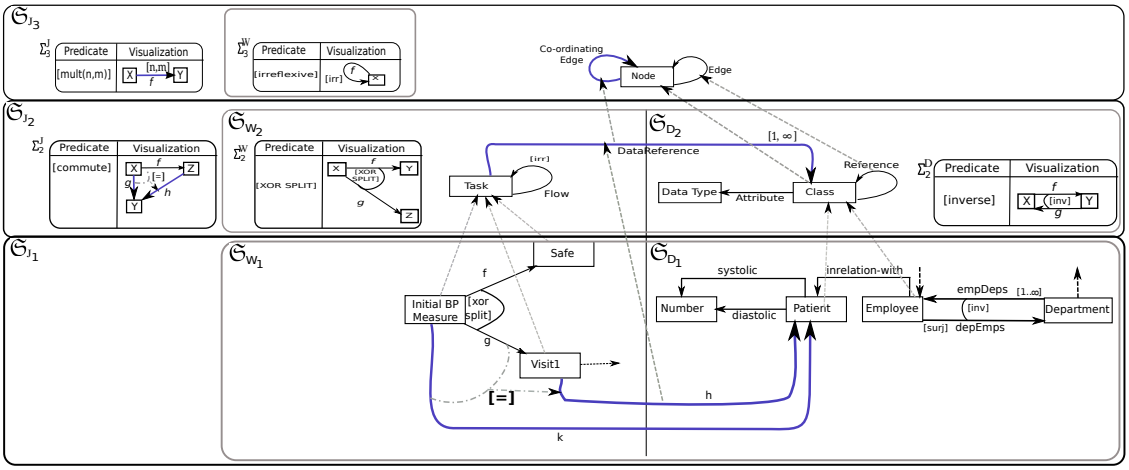


Fig. 3. Example integration of a process metamodel and a datasource metamodel

3.1. Co-ordination of a process model and a datasource model

The process model shown in the previous section is not data-aware. In order to develop a data-aware process model we need to co-ordinate the process model with a datasource model. In Fig. 3 we present the integrated modelling hierarchy where $\mathfrak{S}_{J_1}, \mathfrak{S}_{J_2}, \mathfrak{S}_{J_3}$ are specifications of metalevels representing the co-ordination of the process metamodel and the datasource metamodel. Our linguistic extension is based on ideas from [17]. The formalization of the specification $\mathfrak{S}_{J_3}$ (similar to that in [17]) uses, in addition an "ordinary" edge (called Edge), a so-called "Co-ordinating Edge", subject to constraints. The specification $\mathfrak{S}_{J_3}$ contains a signature Σ_3^J with a predicate called $mult(n,m)$. The semantics of the predicates containing "Co-ordinating Edge" is defined in a fibred manner as in [7]. That is, the semantics of a predicate p is given by the set of its instances. $\mathfrak{S}_{J_2}$ consists of specification $\mathfrak{S}_{W_2}$ and $\mathfrak{S}_{D_2}$ and a signature Σ_2^J with one predicate [*commute*]. Moreover it has a co-ordinating edge, DataReference from Task in W_2 (underlying shape graph

of $\mathfrak{S}_{W_2}$) to Class in D_2 (underlying shape graph of $\mathfrak{S}_{D_2}$). In specification $\mathfrak{S}_{J_1}$, the edges h, k are co-ordinating edges and the three edges g, h and k respect the commute constraint: i.e., g followed by h equals k. This is to indicate that the patient who is assigned an initial BP measurement is the same patient who is assigned a Visit1 task.

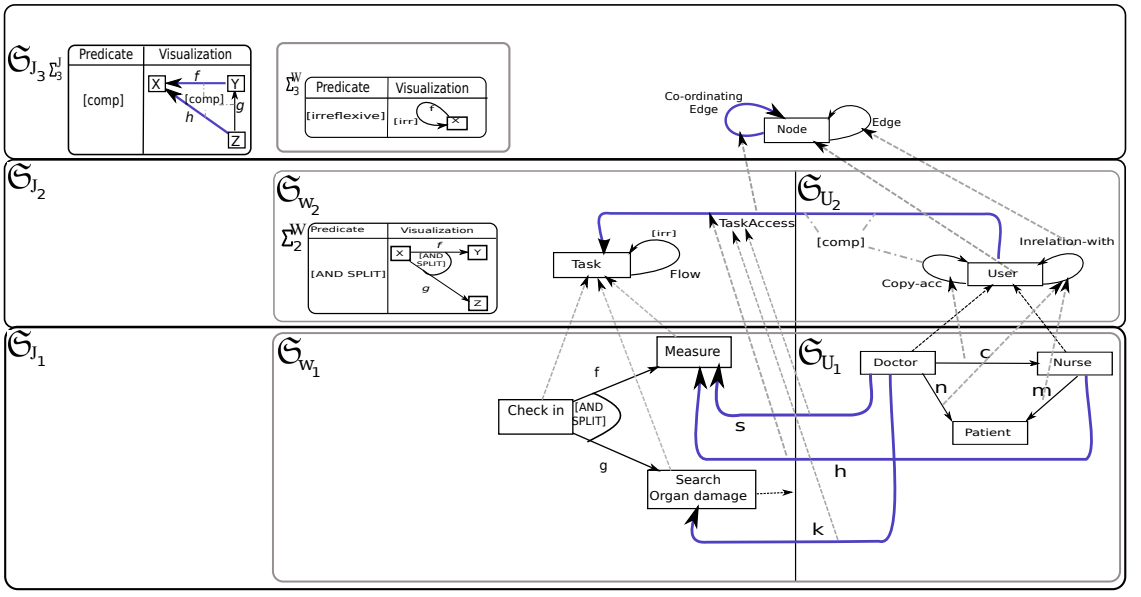
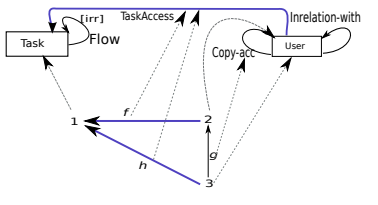


Fig. 4. Example integration of a process metamodel and a user metamodel

3.2. Co-ordination of a process model and a user metamodel

The access policy of a task may be specified by an integrated metamodelling hierarchy shown in Fig. 4. The Copy-acc edge of the User Access specification $\mathfrak{S}_{U_2}$ copies access from one user to another. The specification $\mathfrak{S}_{J_2}$ has a co-ordinating edge, TaskAccess, which co-ordinates the user access to a task. Both the TaskAccess and Copy-acc arrows are labelled with the composition constraint, [*comp*], which ensures the inheritance of task access between users. The arrow h at the bottom level is a co-ordinating edge and copy-acc arrow c from Doctor to Nurse indicates that the Doctor inherits the access that the Nurse has to the tasks. The co-ordinating edge s indicates that the Doctor has access to the task Measure.

Note that the [*comp*] predicate is associated with the integrated shape graph of $\mathfrak{S}_{J_2}$ by a graph homomorphism. The arity of the [*comp*] predicate consists of 2 co-ordinating edges and 1 (ordinary) edge. The diagram to the right shows the graph homomorphism from the arity of [*comp*] to the underlying shape graph of $\mathfrak{S}_{J_2}$, where f and h are co-ordinating edges.



More complex user access rules may be specified co-ordinating the datasource metamodel with the process and user access metamodel. E.g., if we want to specify that when a user has copy access from a second user then the first user is allowed to see only the data the second user has access to. Table. 2 illustrates an example where Doctor Bryan is in relation with Patient 1 and therefore has access to all Patient 1's data. Doctor Bryan can see only Patient 2's BP-related information since he copied (inherited) the access of Nurse Jessica who has access only to Patient 2's BP information. Other fine-grained access policies may be defined; e.g., only allowing a doctor to copy (inherit) access regarding a patient from a nurse, if the patient the nurse is treating is one of the doctor's patients.

Table 2. Example of a user access policy

User	Patient 1		Patient 2	
	BP	More	BP	More
Jessica : Nurse	✓	*X*	✓	*X*
Bryan : Doctor	✓	✓	✓	*X*

3.3. Co-ordination of different aspects

In Fig. 5 we show 5 aspects of Health Information Systems, namely, a Monitor aspect, a User Interface View aspect as well as the Process, Datasource and User Access aspects discussed here. See [12] for details on the metamodels for the two aspects not discussed here as well as an example of the co-ordination among 3 metamodelling hierarchies (User Access, Process and Monitor) required for the Monitor to give alerts. In Fig. 5 we show a number of co-ordinating edges: Trigger, Sends, Displays, Alert UI, View Access, as well as other examples of Data Access, which are required to co-ordinate the 5 aspects displayed.

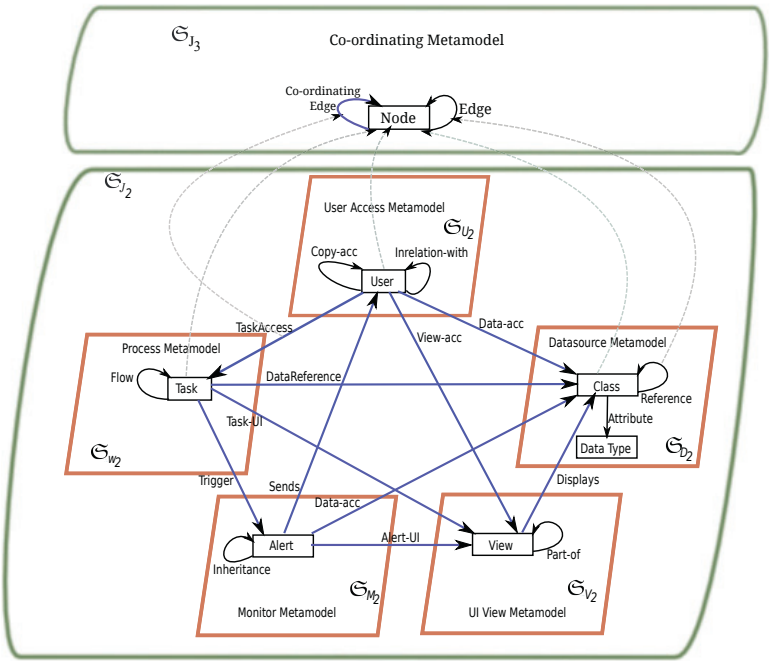


Fig. 5. Multi metamodel for healthcare information systems

4. Related and Future Work

Many workflow languages such as YAWL [18], ADEPT2 [19] focus on a single aspect of a system, usually the control flow. YAWL is data-aware workflow language which supports XML-based models for data definition and manipulation based on XML Schema. These languages are not flexible enough to customize to a different syntax or semantics. In this article we enhanced the DPF framework using an integrated metamodelling technique which can be customized according to different needs. The approach makes the framework modular and flexible yet structured as an integrated model specification has to satisfy typing and conformance to its metamodel specification.

An OCL-based domain specific language, MoScript, was presented in [20] for model-based task and workflow automation. The language is based on the metadata contained in a Megamodel that allows querying a Megamodel for the required modelling artefacts and tools. It supports model manipulations, such as loading and storing modelling artefacts, invocation of a model transformation engine, etc,. MoScript is able to perform some validation at run-time such as to check if a model transformation is applied to a model that conforms to the right meta-model. Whereas in our work, we enhanced the software modelling with the co-ordination of multiple metamodels.

Diskin et al. proposed a megamodeling framework based on graphs and graph mappings, and operations over them [21]. They presented *model mapping* which is a structured set of links between models, specified a library of elementary building blocks, and presented how to combine them into a complex workflow. In that paper, the authors focused on a single aspect of a system with multiple views. Different views of the entity model in [21] would correspond to different views of the datasource model in our paper. They presented a query mechanism and, correspondingly, q-mappings (Kleisli morphisms), essential for megamodeling such views. In this paper, we focused on several aspects of a system (datasource aspect, monitor aspect, etc.) and presented a metamodelling approach to integrate them by co-ordination. Our approach can be further improved by the use of a query mechanism which should incorporate more expressiveness in the DPF metamodelling language. In future we also plan to metamodel other aspects of healthcare information system (eg., Scheduling, Resources, and Service Orientation) and co-ordinate them. This may require an extension of the metamodelling language with time [22], and compensation [23] and co-ordinating them; including such features as time and compensation are essential for the enactment of real life healthcare workflows. The implementation of the editor requires abstraction and projection facilities in order to accommodate many models in the display devices.

5. Conclusion

Separating different concerns (aspects) of a system into several metamodels gives us flexibility, allowing us to modify one aspect of a system described by a particular metamodel without affecting other metamodels. However, to realistically capture the requirements of complex systems such as healthcare systems we need to integrate the metamodels. In this paper, we proposed a linguistic extension to the DPF metamodelling hierarchy, consisting of co-ordinating edges together with constraints on these edges, this allows us to integrate metamodels, thereby modelling features that cannot be modelled by the multi metamodels in isolation. We were able to capture data-aware processes through the use of co-ordinating edges together with the constraint represented by the commute predicate, and the inheritance of user access using a co-ordinating edge together with the constraint represented by the composition predicate. We believe that multi metamodeling with linguistic extensions as described here, together with MDSE principles in general, can be used as the main methodology in the development process of software for care processes.

References

1. Fowler, M.: Domain-specific languages. Addison Wesley Signature Series. Addison Wesley; 2011.
2. Kroiss, C., Koch, N., Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. In: Gaedke, M., Grossniklaus, M., Daz, O., editors. *Web Engineering*; vol. 5648 of *LNCS*. Springer Berlin Heidelberg. ISBN 978-3-642-02817-5; 2009, p. 493–496.
3. *Object Management Group: Semantics of a Foundational Subset for Executable UML Models (FUML)* ; 2011. http://www.omg/spec/FUML/1.0/.
4. *Fujaba Development Team: The Fujaba Tool Suite*; 2012. Http://www.fujaba.de/.
5. Kindler, E.: Model-based software engineering: The challenges of modelling behaviour. In: *Proceedings of the Second International Workshop on Behaviour Modelling: Foundation and Applications*; BM-FA '10. New York, NY, USA: ACM. ISBN 978-1-60558-961-9; 2010, p. 4:1–4:8.
6. Rutle, A., MacCaull, W., Wang, H., Lamo, Y.: A Metamodelling Approach to Behavioural Modelling. In: *Proceedings of BM-FA 2012: 4th Workshop on Behavioural Modelling: Foundations and Applications*. ACM. ISBN 978-1-4503-1187-8; 2012, p. 5:1–5:10.
7. Rutle, A.: *Diagram Predicate Framework: A Formal Approach to MDE*. Ph.D. thesis; Department of Informatics, University of Bergen; Norway; 2010.
8. Barr, M., Wells, C., editors. *Category Theory for Computing Science, 2nd Ed.* Hertfordshire, UK, UK: Prentice Hall International (UK) Ltd.; 1995. ISBN 0-13-323809-1.
9. Diskin, Z., Wolter, U.: A diagrammatic logic for object-oriented visual modeling. *Electr Notes Theor Comput Sci* 2008;**203**(6):19–41.
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer; 2006. ISBN 978-3-540-31187-4, 978-3-540-31188-1.
11. Baarah, A.H.: An application framework for monitoring care processes. PhD thesis, University of Ottawa; 2013.
12. Rabbi, F., Lamo, Y., MacCaull, W.: A Flexible Metamodelling Approach for Healthcare Systems. *2nd European Workshop on Practical Aspects of Health Informatics (PAHI), Accepted* 2014.
13. OMG, . *Meta Object Facility (MOF) Core Specification Version 2.0*; 2006. URL: `http://www.omg.org/cgi-bin/doc?formal/2006-01-01`.
14. Rutle, A., Rossini, A., Lamo, Y., Wolter, U.: A Diagrammatic Formalisation of MOF-Based Modelling Languages. In: *TOOLS (47)*; vol. 33 of *Lecture Notes in Business Information Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-02570-9; 2009, p. 37–56.
15. *The Chinook Primary Care Network*; Last accessed, May 2014. www.chinookprimarycarenetwork.ab.ca/extranet/docs/guides/7.pdf.
16. Ehrig, H., Prange, U., Taentzer, G.: Fundamental theory for typed attributed graph transformation. In: Ehrig, H., Engels, G., Parisi-Presicce, F., Rozenberg, G., editors. *Graph Transformations*; vol. 3256 of *LNCS*. Springer Berlin Heidelberg. ISBN 978-3-540-23207-0; 2004, p. 161–177.
17. Klokkhammer, O.: *A Diagrammatic Approach To Deep Metamodelling*. Master's thesis; Department of Informatics, University of Bergen; Norway; 2014.
18. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. *Information Systems* 2005;**30**(4):245–275.
19. Reichert, M., Rinderle, S., Kreher, U., Acker, H., Lauer, M., Dadam, P.: ADEPT2 - Next Generation Process Management Technology. In: *4th Heidelberg Innovation Forum*. D.punkt Verlag; 2007.
20. Kling, W., Jouault, F., Wagelaar, D., Brambilla, M., Cabot, J.: Moscript: A dsl for querying and manipulating model repositories. In: *Proceedings of the 4th International Conference on Software Language Engineering*; SLE'11. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-28829-6; 2012, p. 180–200.
21. Diskin, Z., Kokaly, S., Maibaum, T.: Mapping-aware megamodeling: Design patterns and laws. In: Erwig, M., Paige, R.F., Wyk, E.V., editors. *SLE*; vol. 8225 of *LNCS*. Springer; 2013, p. 322–343.
22. Wang, H., Rutle, A., MacCaull, W.: A Formal Diagrammatic Approach to Timed Workflow Modelling. In: *Proceedings of TASE 2012: 6th International Conference on Theoretical Aspects of Software Engineering*; vol. 0. IEEE Computer Society. ISBN 978-1-4673-2353-6; 2012, p. 167–174.
23. Rutle, A., Wang, H., MacCaull, W.: A formal diagrammatic approach to compensable workflow modelling. In: Weber, J., Perseil, I., editors. *FHIES*; vol. 7789 of *LNCS*. Springer. ISBN 978-3-642-39087-6; 2012, p. 194–212.