Modeling heterogeneous IT infrastructures: a collaborative component-oriented approach

Benjamin Somers, Fabien Dagnat, Jean-Christophe Bach

Modeling heterogeneous IT infrastructures: a collaborative component-oriented approach

Benjamin Somers[1,2][0000−0002−0359−0902], Fabien Dagnat[1][0000−0002−2419−7587], and Jean-Christophe Bach[1][0000−0001−6986−1093]

[1] IMT Atlantique, Lab-STICC, UMR 6285, F-29238 Brest, France
[2] Crédit Mutuel Arkéa, 29480 Le Relecq-Kerhuon, France

Abstract. The advent and growing sophistication of modern cloud-native architectures has brought into question the way we design IT infrastructures. As the architectures become more complex, modeling helps employees to better understand their environment and decision makers to better grasp the "big picture". As the levels of abstraction multiply, modeling these infrastructures is becoming more difficult. This leads to incomplete, heterogeneous views difficult to reconcile. In this article, we present a collaborative approach focused on improving the accuracy of IT infrastructure modeling through the involvement of all stakeholders in the process. Our approach, applied in an incremental manner, is meant to increase confidence, accountability and knowledge of the infrastructure, by assigning responsibilities early in the process and leveraging the expertise of each stakeholder. It is suited for both *a priori* and *a posteriori* modeling at a low adoption cost, through adaptation of existing workflows and model reuse. Building collaborative models in such a way aims at bridging the gap between different areas of business expertise. As a result, we believe that our approach allows to perform analyses and use formal methods on larger scale models and cover wider technical domains.

Keywords: IT infrastructure modeling · Collaborative infrastructure design · Heterogeneous models · Multi-viewpoint design · Model federation

1 Introduction

The evolution of IT infrastructures has followed a tendency towards abstraction. This has enabled the development and deployment of enterprise-wide systems with greater flexibility and scalability, in order to fulfill a wider variety of needs. These advances come however at the cost of increased architectural complexity [30]. In some domains such as banking, healthcare and defense, the lifetime of IT infrastructures can span over decades, combining the old and the new. Modern cloud-native IT infrastructures are therefore often built upon so-called *legacy* systems, with which they need to interact to perform business-critical operations.

Yet, IT infrastructures do not depend exclusively on the proper operation of hardware and software: the human factor and the good performance of business

processes must be taken into account. This is where Enterprise Modeling comes into the picture [32], with two major challenges. First, technical infrastructures tend to be stacked in technical layers (such as hardware, software, network...) while human organizations tend to be divided in business domains (financial, cloud operations, support...). These technical layers, however, do not necessarily match the business ones, and vice versa; this disalignment must be taken into account in the modeling. Second, all the stakeholders should be included in the modeling process [33], to properly capture the interactions within the infrastructure and the related responsibilities. The different stakeholders, however, have different skill sets, reflected in the use of different tools and a different jargon from other employees [3], making model alignment process even more complex.

It is important to gather this knowledge around a common model to have a good overview of IT infrastructures and to conduct analyses covering several technical and administrative domains. This article proposes a component-oriented metamodel that takes into account the various perspectives of such infrastructures. We argue that a complete and correct model is achieved by considering all of these viewpoints and better integrating the responsibilities of each stakeholder.

To this end, we present the theoretical background of our study in section 2. Section 3 describes the metamodel we use. In section 4, we propose a collaborative methodological framework to encode business processes and technical jobs in this metamodel. We present a case study in section 5 to illustrate our approach by reasoning on a model. Finally, section 6 concludes this article.

2 Related work

Models convey domain-specific knowledge in a broad range of fields through abstraction of concepts. From technical to administrative areas, with various levels of granularity, they are an integral part of today's businesses [16]. IT infrastructure modeling combines the concepts of enterprise modeling and IT models, linked together to provide a complete picture.

2.1 Enterprise modeling

Enterprise modeling has undergone major evolutions since its inception [32]. Models have become "active" [10], to manage the complexity of interactions across diverse business domains. Combined with formal methods, high-fidelity models can bring great value to companies adopting them [5]. However, due to the wide range of professions and tools used within these companies, such models are complex to produce and discrepancies occur [2].

Standards such as ISO 19439 [12] propose to model enterprises in (at least) four views (process, information, infrastructure and organization) on three modeling levels (requirements, design and implementation). Such a layered approach is found in many modeling frameworks, among which we can mention RM-ODP [1] and Archimate [29]. These frameworks provide a high-level way to model enterprises and allow to represent the infrastructures on which businesses lie. However,

the large amount of concepts and the lack of precise semantics regarding IT infrastructures make the frameworks difficult to use in IT domains for people whose main job is not enterprise modeling [17]. Moreover, the matricial aspects of their approaches and their division into layers are not always suitable [14].

2.2 IT models

Many languages and representations exist to describe IT infrastructures, from hardware to software, including networks and processes. A datacenter can be described by rack diagrams, illustrating the layout of servers and network components. A software can be represented using UML diagrams [23], to show its structure and the different interactions at work, or can be described by its code. A network topology can be seen as mathematical objects [27,24], described by switch configurations, or as code in *Software-Defined Networking* [18]. These areas also benefit from many contributions from the formal methods communities (*e.g.* Alloy [13] to verify specifications or Petri nets [20] to model complex behaviors).

However, most of these frameworks cannot interact with one another. This limit appears if we ask questions that cross several domains, such as "which services become non-operational if we unplug this cable?" [21] or "is my business domain impacted by this router vulnerability?".

2.3 Collaborative modeling

Domain-specific languages are adapted to their respective domains [6] and can represent in detail things that holistic frameworks cannot. But these languages are sometimes not understandable by other parties. It leads to many metamodels [15], often sharing the same core concepts [4], being used to model enterprises.

The modeling process must come from a need and be undertaken by including all the professional disciplines concerned. However, due to a lack of modeling skills, some stakeholders are not able to participate in such a process [25]. Work in the model federation community [9] (where we maintain links between models expressed in different metamodels) is a step towards including the expertise of such stakeholders. Other approaches, such as composition [8] (where we build a common metamodel to align models) and unification [31] (where we build a single model) are described in the literature.

3 Our proposal

In this article, we advocate a collaborative approach to IT infrastructure modeling. In this section, we present our metamodel (represented in figure 1) and detail its characteristics, before presenting its differentiating aspects.

3.1 Presentation of the metamodel

Our first contribution combines a responsibility-oriented metamodel (though simpler than what can be found in [7]) and a component-oriented metamodel, all

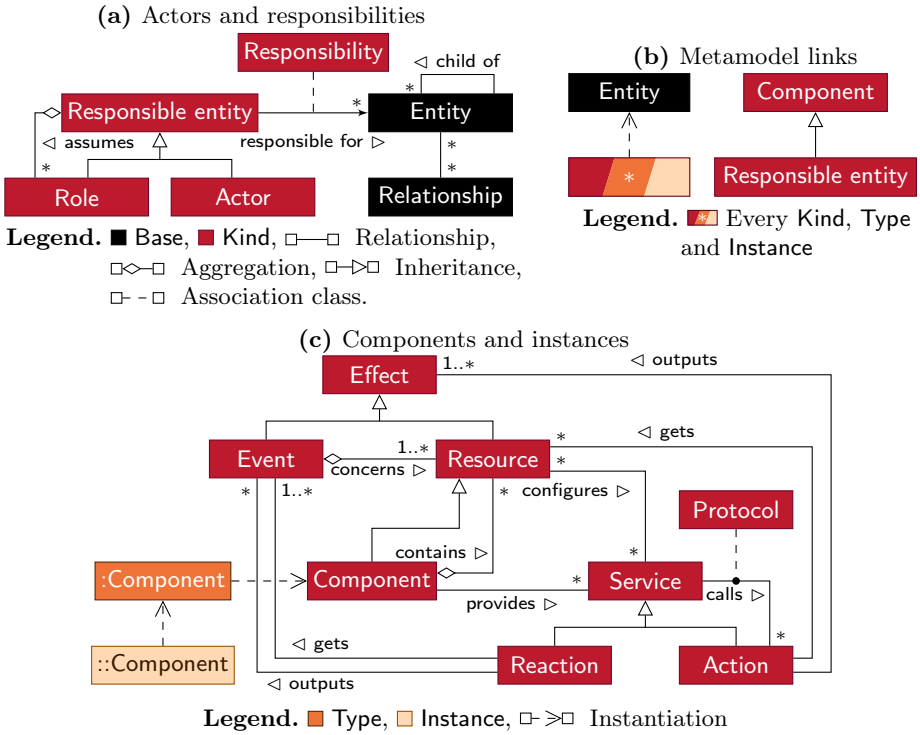

Fig. 1: Component-oriented metamodel

within a reflexive unifying metamodel. Our metamodel is divided in three parts: the actors and their responsibilities (on figure 1a), the components and their instances (on figure 1c), and the additional links in the metamodel (on figure 1b).

Actors and responsibilities. In our metamodel, *responsible entities* have *responsibilities* over *entities* and can assume *roles*. *Roles* represent generic sets of *responsibilities*. They can be used to encode access rights on an information system or positions in a company's organizational chart for example. *Actors* represent the actual *entities* which can assume *roles* and have *responsibilities*. They can be used for example to encode users, allowed to access specific servers because of their positions, or even a whole company, responsible for the proper functioning of the products it sells. *Relationships* represent all lines and arrows in models (and in the metamodel itself).

Components and instances. Our metamodel is focused on *components*. They provide *services* (*actions* or *reactions* to *events*) that can in turn use other *components*' *actions* through *procotols*. They can also contain *resources*, that may be *entities* providing *services* (*components*, such as a web server) or not (such as configuration files, web resources, or even the models themselves). Both

resources and *events* can be the result of an *action*, so we decided to unify them under the *effect* kind. Three layers appear here:

- The Kind layer (■), representing the core concepts of the metamodel;
- The Type layer (■), representing "types of" these concepts. For example, "physical server" is a type of *component*. To follow the UML notation, a *component* type is represented here as :Component.
- The Instance layer (■), representing "instances of" these concepts and types. For example, *a* physical server is an instance of "physical server" and "check webpage availability" is a *service* instance. In our work, we have not encountered the need for *service* types. An instance of :Component is represented here as ::Component.

Metamodel links. Every Kind, Type and Instance of the metamodel is an instance of *entity*; it means that *responsible entities* can have *responsibilities* over them. For example, an *actor* can be responsible for the development of a software *component* (on the Type layer) and another can be responsible for its configuration and deployment (on the Instance layer).

A core feature of our metamodel is that *responsible entities* can be responsible of *responsibilities* themselves. It makes sense in a context of access management where a person may be responsible for the access right given to another person. As models can themselves be *resources* in the metamodel, it is easy to represent situations like an employee responsible for modeling a particular *component*. We think that both characteristics are distinctive features of our approach.

For consistency, *responsible entities* are also *components* and their behaviors can be encoded as *services*.

3.2 Collaborative modeling

Our second contribution is a collaborative modeling framework using this metamodel relying on three principles: non-intrusiveness, refinability and correctness.

Non-intrusiveness. Most technical IT domain have their own sets of tools and representations to convey information and model systems [3]. Two persons in the same domain can understand one another thanks to this common jargon, but may struggle to interact with people in areas with which they are not familiar.

Working on small models whose boundaries are clearly defined enables experts to work collaboratively and still preserve the integrity and coherence of each model. Furthermore, these experts can take advantage of the most appropriate tools and techniques for their particular domains. Our metamodel intends to provide a framework adapted to linking these tools together, rather than replacing them. By allowing employees to work locally with their peers on models, we avoid the issues raised by [25] in the first phases of modeling.

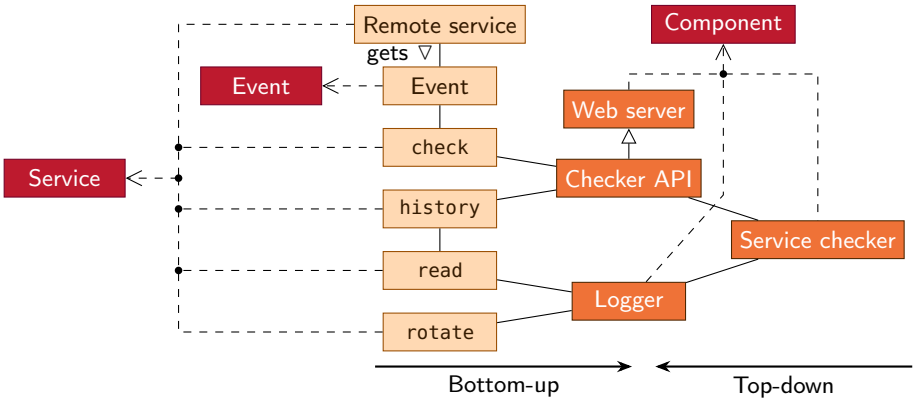

Fig. 2: Model for a generic service checker

Refinability. To reflect the actual design process of its components, people involved in infrastructure modeling should be able to gradually refine their models. This can be done either in a top-down (where one adds details) or bottom-up (where one abstracts them) approach, or a combination of both, as shown in figure 2. In this example, a top-down approach would describe what a "service checker" is, by dividing it into sub-components (Service checker, made of Checker API and Logger) and then refining their services (`check`, `history`...). A bottom-up approach would be to describe the services wanted for a "service checker" and combining them into components and super-components providing them.

Iterative conception goes through a succession of incomplete models. When working on a new piece of software, a common approach consists in letting the end-users describe their needs and iteratively producing code that meets these needs [26]. The initial need may be very imprecise and high-level and may require several refinement steps during the project's lifespan.

"Holes" in models can also arise from *blackbox* software or hardware, or even legacy components whose knowledge has been lost, for example due to employee turnover. Even though the knowledge of an IT infrastructure is partial, properties can still be deduced. By allowing imprecision, the benefit is threefold:

- Coherence: instead of making wrong assumptions, modelers can express their lack of knowledge, limiting the number of inconsistencies between models;
- Reconciliability: employees should not attempt to refine a component they are not responsible for, simplifying the reconciliation phase and ensuring that responsibilities are respected;
- "Fail-early"-ness: as properties can be proved early in a project modeling, safety and security issues can be addressed from the first stages of development.

Correctness. In order to get a detailed view of an IT infrastructure, it is crucial to involve all its stakeholders in the process. Indeed, the sum of local viewpoints is not enough to produce an overall model: the reconciliation and the resulting links

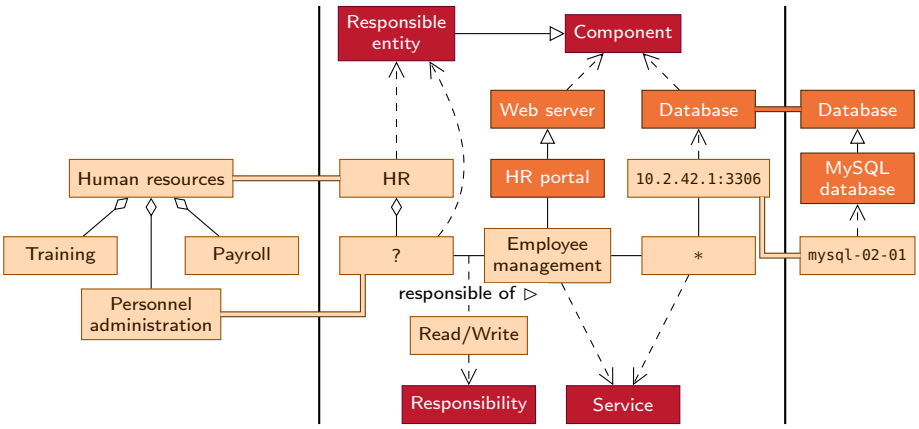

Fig. 3: Model reconciliation with three points of view
Legend. Reconciliation

between models are essential. As the literature shows [22], model reconciliation is a complex task, this is why we advocate to start the collaborative modeling process as soon as possible.

If the modeling is done according to these principles, model reconciliation is mainly a matter of refining black boxes in other models and linking them together, as illustrated in figure 3. Here, we have three points of view, from three teams. On the left, the organizational structure of a human resources department is represented. In the middle, we have the design of a web application using a remote database (not modeled) allowing some HR people (unknown at modeling time) to manage employees. On the right, there is a simple model of said database.

During model reconciliation, the teams align their vocabularies (Human resources and HR), specify black boxes (? becomes Personnel administration) and combine knowledge (`10.2.42.1:3306` refers to `mysql-02-01`). There is no universal method to solve modeling conflicts, but we think that modeling in incremental steps avoids solving them on larger models. The reconciliation process itself may be modeled using our metamodel, by assigning responsibilities to the employees performing the reconciliation. When a model is updated, it becomes easy to know who performed the reconciliation and notify them to review whether the change invalidates their work or not. This idea, which to our knowledge has not been explored, ensures that the overall model remains correct in the long term.

4 Encoding business processes and technical jobs

Modern IT companies have a combination of business processes, which are more administrative in nature, and very detailed technical workflows. Choosing a holistic modeling framework that can cover all these aspects in detail seems unrealistic. A federated approach [11] enables to take advantage of everyone's skills, while

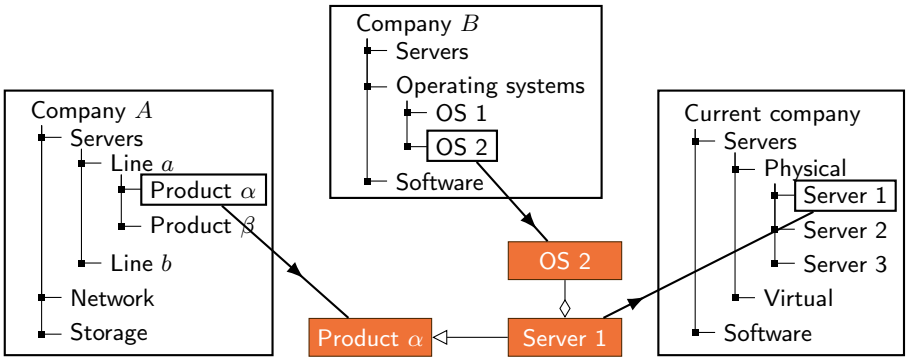

Fig. 4: An example of model reuse

achieving a more thorough modeling. Such an approach keeps the metamodels and models of each stakeholder and proceeds by establishing semantics links between models. Yet, model interdependencies and inconsistencies can arise and hinder collaboration, especially when changes are made in one model that affect other models. As already stated in section 3.2, our framework makes it possible to detect and react to these changes.

In this section, we propose a methodological framework adaptable to concrete business processes, to build thorough yet accurate models using our metamodel.

4.1 Component catalogs

Companies usually design their systems by using external components. For hardware systems, most of them include *commercially available off-the-shelf* components sold by other companies. In the software domain, third-party libraries and packages are an integral part of modern systems.

This decentralized aspect of system design can be applied to infrastructure modeling: manufacturers can produce models for their systems and users can integrate these models into their infrastructure models. Such models can be made available in catalogs available internally within companies and externally to clients or for public use. The benefit is twofold. First, responsibilities and knowledge are better distributed: the models are produced by system designers, not the users. Second, the modeling process is sped up: model reuse, as would code reuse in software development, allows designers to build systems faster.

The two modeling steps, design and use, are illustrated in figure 4, where a company's Server 1 is build from other companies' Product α and OS 2. One can then instantiate this Server 1 architecture in their models without redesigning it.

4.2 *A posteriori* modeling

Understanding the orchestration of a company's business processes can help optimize the existing, as well as build the new, in a better controlled way. In the

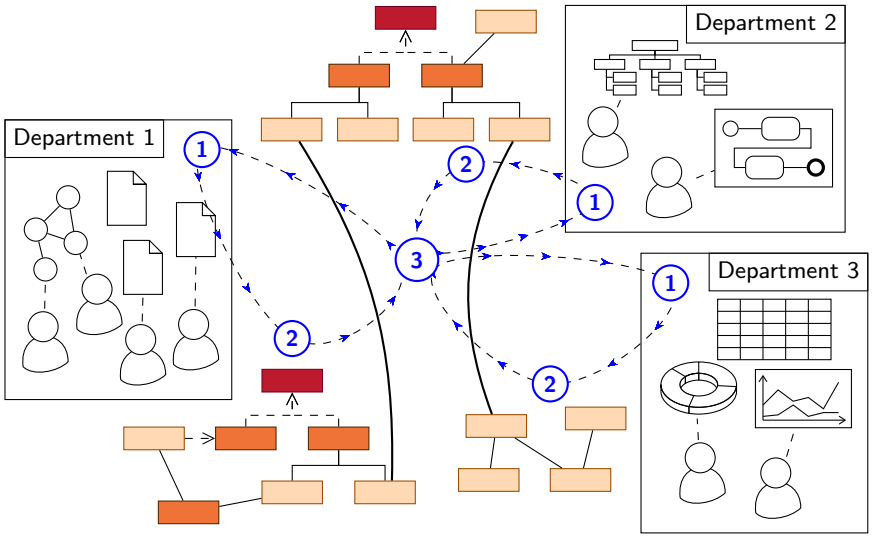

Fig. 5: *A posteriori* modeling. Step 1 corresponds to the inventory, step 2 is the modeling process and step 3 is the reconciliation.

banking industry for example, the use of legacy systems imposes technical choices that cannot be made without a good knowledge of existing architectures. Within the company's departments, this knowledge exists in diagrams, source code, configuration files... which first need to be identified (step 1 of figure 5). This step comes along with hardware inventories, if needed. Then, the identified elements are mapped onto our metamodel. Care must be taken to assign responsibilities to the model entities in the early stages of modeling (who *owns* which product, who develops which service, who is in charge of modeling which component...). Step 2 of figure 5 illustrates this process. In an iterative way, business processes interacting with the modeled elements must be identified. Agile collaboration frameworks should be used to implement this process, as they promote collaborative, quick and iterative changes. This is represented in step 3 of figure 5.

4.3 *A priori* modeling

Accurate and complete models allow to better evaluate the financial and technical costs of projects, to optimize infrastructure dimensioning and to create *safe and secure by design* systems. Throughout the life of a project, it is important to ensure that a system does not deviate from its specification, for example due to a lack of communication, a misunderstanding or an urge to move too quickly. Verification of expert-defined properties on these models allows to ensure the conformity of such systems before their realization and helps to select a technical solution rather than another. For example, in a banking infrastructure, we could check that only certified personnel can access sensitive cardholder data.

A specification is seen in our approach as a model interface that technical proposals must implement. This extends the concepts from Object-Oriented Programming to infrastructure modeling. This concept of model typing is explored in [28]. Our metamodel ensures the syntactic conformance of the models to the specification, but the semantic conformance must be verified by domain experts.

5 Case study

To illustrate our approach, let us now consider a fake banking company, called eBank. eBank provides banking and payment services to consumers and businesses. One of its flagship product, ePay, acts as a payment processor for companies and as an instant payment and expense sharing tool for consumers. The employees of the company want to have a better understanding of its overall processes and decided to use our approach to this end. In this section, we first make an inventory of the company's models. Then, we link these models together into our metamodel. Finally, we use the resulting big picture for a cross-model case study.

5.1 Heterogeneous models...

The company decided to start its modeling by focusing on ePay's environment, namely the company's organizational structure, the business processes around the product and the technical architecture. Each department uses domain-specific modeling tools, leading to different views of the overall infrastructure.

Organizational structure. eBank is structured in two directions: Technical and Administrative, each subdivided into structures, divided themselves into departments. An organizational chart of the company is given in figure 6.

Business processes. The company's activities are guided by various business processes. For the sake of brevity, we consider here only the equipment purchase process, represented in figure 7.

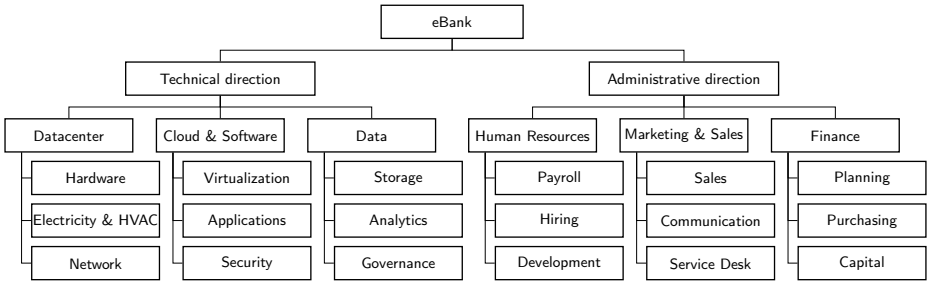

Fig. 6: eBank's organizational structure

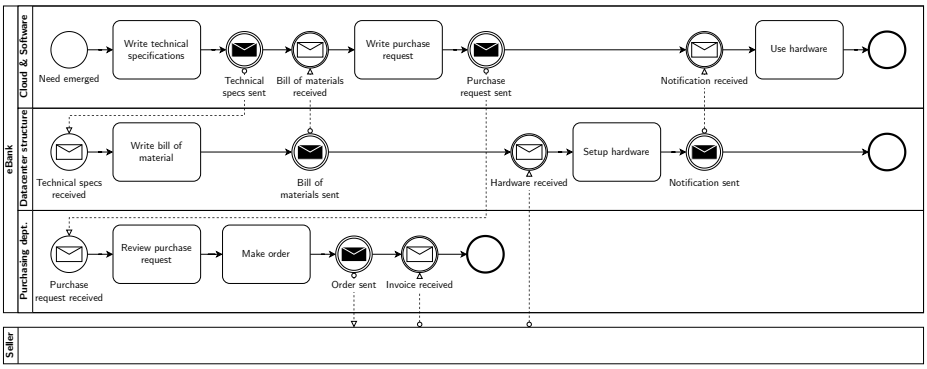

Fig. 7: BPMN diagram for purchasing new hardware. For clarity, we do not show the exclusive gateways and assume requests to be automatically accepted.

eBank has been using a task management solution for many years to track how many person-hours are needed for which projects. The solution is also used to know who is working on what at a given time. By reusing this software's database, employees created a catalog of common company tasks to predict their durations and help project planning. This catalog ranges from technical tasks, for example "commission a server", to administrative ones, for example "open a position". These two tasks are represented in figure 8.

Lastly, all financial transactions are managed by the finance department.

Technical architecture. eBank manages a datacenter hosting the hardware necessary for its activities. Some services are hosted on dedicated machines and others are on an internal cloud infrastructure. Due to time constraints, ePay has not been migrated to a modern cloud infrastructure yet. The service follows an active/passive architecture, where only one node operates at a time.

To check the proper functioning of its services in real time, the company has a monitoring infrastructure that measures availability and several key performance

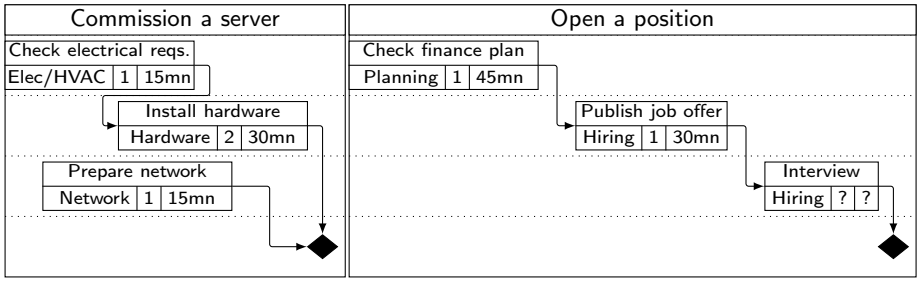

Fig. 8: Task catalog

Legend. $\frac{a}{b|c|d}$ Sub-task (a: title, b: department, c: people, d: duration), ◆ End

indicators. For business clients, ePay must process its requests within three seconds 99.9% of the time and must pay penalties in case of non-compliance.

5.2 ... linked together

After several rounds of modeling, eBank's employees came up with the representation shown in figure 9. First, the organizational structure (figure 6) is partially mapped to the eBank component and its four sub-components representing structures and departments. The BPMN diagram (figure 7) adds the Hardware component type, along with the Maintenance and Usage responsibilities that the Cloud & Software structure and Hardware department have on this Hardware. The task catalog (figure 8) adds knowledge about the Hiring department and its Hiring responsibility. The task management solution (Task manager) keeps track of the time spent on the Usage and Maintenance of the Hardware and on the Hiring process, highlighting the particular nature of responsibilities in our metamodel. Finally, the Finance department is responsible for employees' Wage payment, for the Invoice payment of Hardware and the company's Financial obligation regarding its Payment processing's SLAs.

5.3 Exploiting the model

The company's real time monitoring has recently identified slowdowns in ePay on peak hours. Following our model, we can see that there is a potential impact on the Finance department because of its Financial obligations. Some employees have suggested scaling the infrastructure before such slowdowns violate SLAs. One way to do so is to setup new Physical servers and change the overall architecture of the services. This new architecture is expected to mobilize part of the Cloud & Software structure for several months. The Human Resources structure proposes to either ignore the potential problem or to assign its teams on the scaling project (by hiring new staff, outsourcing some of the work or reassigning staff without additional hiring or outsourcing).

In figure 10, we explore our model to trace the paths between the slowdown and the Finance department, to identify potential financial impacts. To make its decision, the company has to compare (a) the financial impact if nothing is done (resulting from the Financial obligation) to (b) the financial impact of the improved architecture (resulting from Invoice payments and the personnel cost). The company does not have an outsourcing process, so the analysis of the Outsource branch, represented by "?" in the figure, cannot be performed. We have not detailed the Reassign branch because it is outside the scope of this article. However, its analysis is valuable to the company since the reassignment of staff would change the time allocated in the Task manager. This would consequently have indirect impacts on the Finance department (for example, failure to deliver new features to clients due to a lack of time, leading to increased customer churn).

By calculating the cost of each decision branch, which is partly automatable, the company can make a decision regarding this particular problem.

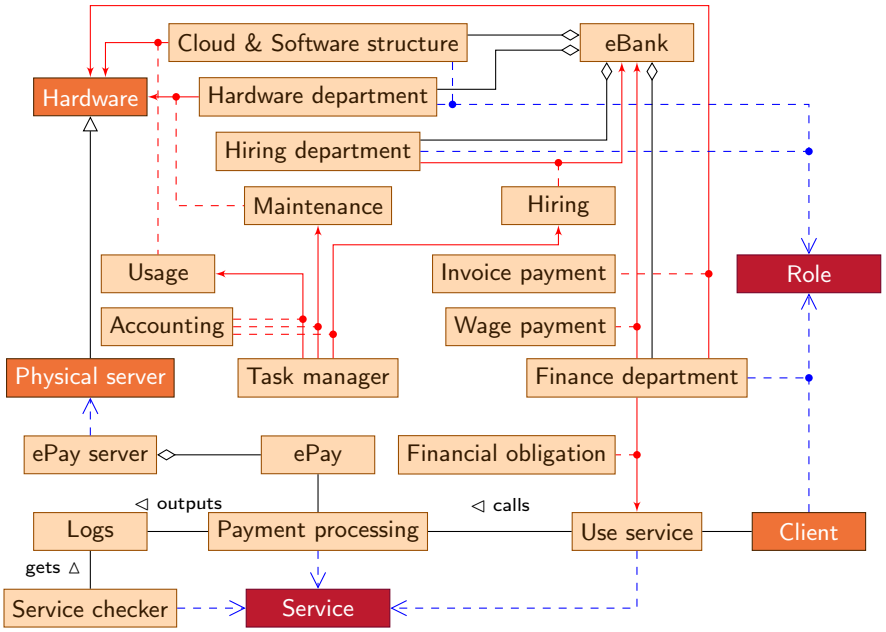

Fig. 9: ePay's big picture. To help the reader, instantiations are blue (□-⩾□) and responsibility links and their association classes are red (□→□ and □- -□).

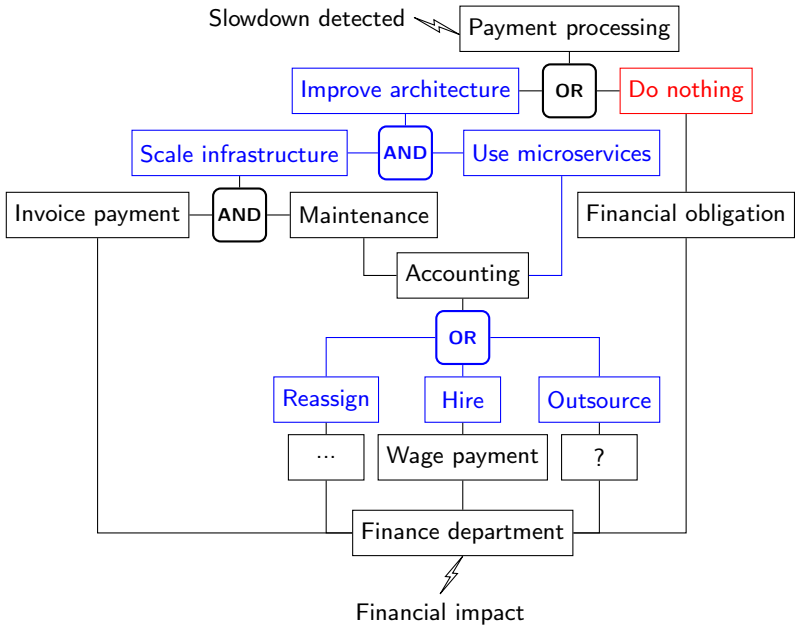

Fig. 10: Impact tree from application slowness to financial impact
Legend. □ Default branch, □ Employee suggestions, □ Model elements

6 Conclusion and future work

In this article, we have presented a collaborative approach for IT infrastructure modeling. This approach consists of a generic metamodel aimed at linking models together, a better control of each stakeholder's responsibilities and methods inspired by software engineering to guide the modeling process. Our approach is not intended to replace established methods within companies, but rather to allow for analyses spanning across multiple models. Through model federation, we think that business modeling can include more stakeholders, while preserving the tools and models they are used to working with.

The validation of our approach is still a work in progress. To this end, we have developed an infrastructure modeling language and its compiler that can be linked to model checkers such as Z3 [19]. While we have been able to verify consistency (for example, whether a set of configuration constraints is satisfiable) and safety properties (for example, whether a system is fault-tolerant) on small models, we are now focusing on scaling our tool to larger models. An important step in the validation of our approach is an industrial experiment, covering broad business domains, that we are currently drafting.

References

1. Reference Model of Open Distributed Processing (RM-ODP), http://rm-odp.net/
2. van der Aalst, W.M.P.: Business Process Management: A Comprehensive Survey. ISRN Software Engineering (2013). https://doi.org/10.1155/2013/507984
3. Amaral, V., Hardebolle, C., Karsai, G., Lengyel, L., Levendovszky, T.: Recent Advances in Multi-paradigm Modeling. In: Models in Soft. Eng. pp. 220–224 (2010)
4. Breton, E., Bézivin, J.: An Overview of Industrial Process Meta-Models. In: Int. Conference on Software & Systems Engineering and their Applications (2000)
5. Cohn, D., Stolze, M.: The rise of the model-driven enterprise. In: IEEE International Conference on E-Commerce Technology for Dynamic E-Business. pp. 324–327 (2004). https://doi.org/10.1109/CEC-EAST.2004.65
6. van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: An annotated bibliography. SIGPLAN Notices (2000). https://doi.org/10.1145/352029.352035
7. Feltus, C., Petit, M., Dubois, E.: ReMoLa: Responsibility model language to align access rights with business process requirements. In: International Conference on Research Challenges in Information Science. pp. 1–6 (2011). https://doi.org/10.1109/RCIS.2011.6006828
8. Fleurey, F., Baudry, B., France, R., Ghosh, S.: A Generic Approach for Automatic Model Composition. In: Models in Software Engineering. pp. 7–15 (2008)
9. Golra, F.R., Beugnard, A., Dagnat, F., Guérin, S., Guychard, C.: Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation. In: Companion Proceedings of the International Conference on Modularity (MoMo). ACM (2016). https://doi.org/10.1145/2892664.2892701
10. Greenwood, R.M., Robertson, I., Snowdon, R.A., Warboys, B.: Active Models in Business. In: Annual Conference on Business Information Technology (BIT) (1995)
11. International Organization for Standardization: ISO 14258:1998 — Industrial automation systems and integration – Concepts and rules for enterprise models (1998), https://www.iso.org/standard/24020.html

12. International Organization for Standardization: ISO 19439:2006 — Enterprise integration – Framework for enterprise modelling (2006), https://www.iso.org/standard/33833.html
13. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press, second edn. (2011)
14. Jørgensen, H.D.: Enterprise Modeling – What We Have Learned, and What We Have Not. In: The Practice of Enterprise Modeling. pp. 3–7 (2009)
15. Kaczmarek, M.: Ontologies in the realm of enterprise modeling – a reality check. In: Formal Ontologies Meet Industry. pp. 39–50 (2015)
16. Kulkarni., V., Roychoudhury., S., Sunkle., S., Clark., T., Barn., B.: Modelling and enterprises - the past, the present and the future. In: International Conference on Model-Driven Engineering and Software Development. pp. 95–100 (2013). https://doi.org/10.5220/0004310700950100
17. Lantow, B.: On the Heterogeneity of Enterprise Models: ArchiMate and Troux Semantics. In: IEEE Int. Enterprise Distributed Object Computing Conf. Workshops and Demonstrations. pp. 67–71 (2014). https://doi.org/10.1109/EDOCW.2014.18
18. Masoudi, R., Ghaffari, A.: Software defined networks: A survey. Journal of Network and Computer Applications (2016). https://doi.org/10.1016/j.jnca.2016.03.016
19. de Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340 (2008)
20. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989). https://doi.org/10.1109/5.24143
21. Neville-Neil, G.: I Unplugged What? Communications of the ACM **65**(2) (2022). https://doi.org/10.1145/3506579
22. Nuseibeh, B., Kramer, J., Finkelstein, A.: ViewPoints: meaningful relationships are difficult! In: International Conference on Software Engineering. pp. 676–681 (2003). https://doi.org/10.1109/ICSE.2003.1201254
23. OMG: Unified Modeling Language (UML), Version 2.5.1 (December 2017), https://www.omg.org/spec/UML/2.5.1
24. Park, K., Willinger, W.: Self-Similar Network Traffic: An Overview, chap. 1, pp. 1–38. John Wiley & Sons, Ltd (2000). https://doi.org/10.1002/047120644X.ch1
25. Renger, M., Kolfschoten, G.L., de Vreede, G.J.: Challenges in Collaborative Modeling: A Literature Review. In: Advances in Enterprise Eng. I. pp. 61–77 (2008)
26. Ruparelia, N.B.: Software Development Lifecycle Models. SIGSOFT Software Engineering Notes **35**(3), 8–13 (2010). https://doi.org/10.1145/1764810.1764814
27. Salamatian, K., Vaton, S.: Hidden Markov Modeling for Network Communication Channels. SIGMETRICS Perform. Eval. Rev. **29**(1), 92–101 (2001). https://doi.org/10.1145/384268.378439
28. Steel, J., Jézéquel, J.M.: On model typing. Software & Systems Modeling **6**(4), 401–413 (Dec 2007). https://doi.org/10.1007/s10270-006-0036-6
29. The Open Group: ArchiMate® 3.1 Spec., https://publications.opengroup.org/c197
30. Urbach, N., Ahlemann, F.: Transformable IT Landscapes: IT Architectures Are Standardized, Modular, Flexible, Ubiquitous, Elastic, Cost-Effective, and Secure, pp. 93–99 (2019). https://doi.org/10.1007/978-3-319-96187-3_10
31. Vernadat, F.: UEML: Towards a unified enterprise modelling language. International Journal of Production Research **40**(17), 4309–4321 (2002). https://doi.org/10.1080/00207540210159626
32. Vernadat, F.: Enterprise modelling: Research review and outlook. Computers in Industry **122**, 103265 (2020). https://doi.org/10.1016/j.compind.2020.103265
33. Voinov, A., Bousquet, F.: Modelling with stakeholders. Environmental Modelling & Software **25**, 1268–1281 (2010). https://doi.org/10.1016/j.envsoft.2010.03.007