Enterprise Modeling in Support Of Transparency in the Design and Use of Software Systems

Mario Nolte(✉) and Monika Kaczmarek-Heß

University of Duisburg-Essen, Essen, Germany
{mario.nolte,monika.kaczmarek-hess}@uni-due.de

Abstract. Transparency is increasingly perceived as a relevant requirement for the design and use of software in general, and for systems using machine learning (ML) algorithms in particular. The existing approaches to ensuring software transparency however, among others, often follow only a one-sided perspective on transparency and, at the same time, neglect the organizational context of software design and use. Since enterprise modeling (EM) allows to analyse enterprise information systems (EIS) and organizational aspects in tandem, in this paper we focus on how EM can support transparency while designing and using software. To this aim, we propose an interactive understanding of transparency, which has the collaboration of different stakeholders at its core. Based on this understanding, we derive a set of requirements, and use them to extend a selected EM approach. We evaluate the extended approach two-fold: against requirements and using an exemplary scenario.

Keywords: Transparency · Enterprise modeling · Machine learning

1 Introduction

In recent years, the demand for transparency has become central part of many debates. On the one hand, it seems to be caused by striving for democracy and equality, which may be put at risk by information asymmetries [38,58]. On the other hand, it seems to be raised by the increasing usage of software systems in private and professional contexts. For instance, as software systems support business processes, those systems determine the processes execution paths and the decisions being made, however, at the same time, they often remain black boxes to involved stakeholders [58,75]. Considering it, many (legal) institutions and organizations have become aware of the importance of transparency with respect to software systems [1,37,55], leading to transparency of some systems, e.g., those relying on machine learning (ML), being required by law [19,20]. Here the transparency of algorithms [1,31], models [23], and data [19], is called for.

Subsequently, various initiatives emerged that focus on transparency of software systems. Examples include the provision of source code or pseudo code [40], using transparency audits [11,33,52], or, with respect to ML, explainable artificial intelligence (XAI) [32]. While existing approaches relate to domain-specific

A. Augusto et al. (Eds.): BPMDS 2022/EMMSAD 2022, LNBIP 450, pp. 157–172, 2022.
https://doi.org/10.1007/978-3-031-07475-2_11

views and come with specific strengths, the practical implementation of transparency of software systems causes several challenges. Aside from harmonizing partly juxtaposing notions of transparency, different, rather simplistic conceptions, e.g., considering transparency only as a provision of (one-sided) information that relates to specific views, can be dysfunctional to each other, and result in unintended effects like information overload or resignation [3,44,75].

Therefore, based on our analysis of different understandings of transparency, cf. Sect. 2, we argue for an *interactive understanding*, which allows to align transparency-demands and corresponding activities. Specifically, we argue that an instrument to support transparency of software should not only be understood as a mere form of information disclosure in one-way communication to interested parties, but should also account for: (1) different perspectives of involved stakeholders, (2) their views on individual and shared objects, (3) organizational context, and (4) enabling critical analysis and interactions among stakeholders. Consider, e.g., usage of ML: not only there is a diverse understanding of artifacts that are subject of related discourses [6,67], but also, a broad knowledge of the application domain and usage context seems to be of relevance to evaluate and challenge the results of ML to avoid domain-specific pitfalls [17,63].

Considering different facets of transparency of software systems, an instrument is needed that would reduce complexity, increase understanding, and enable multi-perspective analyses. A promising instrument seems to be the application of conceptual modeling (CM), which can be roughly defined as "the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication" [53]. We deem it as a promising since (1) different modeling languages applied together may offer a multi-perspective view of a software system, and the way it is used by an organization, (2) application of a modeling language forces one to be concrete, which seems to be beneficial with a contested term such as transparency, and (3) application of CM fosters communication among stakeholders, thus promoting a shared understanding of features of software systems used. From the vast field of CM, especially enterprise modeling (EM) seems to be a promising instrument to support our interactive understanding of transparency, by accounting for EIS and an organizational action system in tandem, cf. [25]. Such a multi-perspective model may not only enable a cross-disciplinary exchange and collaboration (e.g., by outlining differences and similarities between conceptual notions of different software artifacts), but also models of usage context of software systems (e.g., process or goal models) support the corresponding evaluation and reflection.

Although EM has been already explicitly [66,72] or implicitly related to transparency, e.g., by referring to related terms like clarity [76, p. 277], or comprehensibility and understandability of models [53, p. 52]; elaborated or more explicit conceptions of transparency are not considered. Particularly, while some works focus on transparency of EM and related activities [10,22], others are focusing on conceptual models only, ignoring the demand for cross-disciplinary analysis and collaboration, resulting in more one-sided representations [36,45].

Further contributions like, e.g., [16], acknowledge the need for different kinds of models to foster transparency, but focus on specific stakeholders only.

Against this background, we follow two interrelated aims: (1) to propose a broader conception of transparency while designing and using software systems that (i) comprises different conceptions of transparency, and (ii) supports different activities and collaboration; and based on it, (2) to investigate how enterprise modeling can be used in order to support it. This contribution follows the design-oriented research paradigm [56]. Furthermore, we adopt the paradigm of constructivism. Thus, in line with, among others [65], we essentially understand models as means of representation of socially constructed knowledge. The modeling process is understood as the process of constructing, representing and sharing this knowledge between the involved participants.

To reach our aims, first, based on the review of the current use of the term, we derive an interactive understanding of transparency (Sect. 2). Then, by analyzing its main features and contemplating a use scenario, we derive a set of requirements that an EM approach should fulfill (Sect. 3). As none of the existing approaches addresses the identified requirements to the full extent, we select an enterprise modeling approach Multi-Perspective Enterprise Modeling (MEMO) [25], and extend an already existing domain-specific modeling language (DSML) focusing on modeling IT infrastructures, called ITML [27,34], with additional concepts and properties (Sect. 4). To perform the desired extensions, we follow the language development method proposed by [24]. We evaluate the proposed artifact twofold: (1) against the identified requirements to check consistency and comprehensibility, and (2) using an exemplary case scenario (Sect. 5).

2 Towards an Interactive Understanding of Transparency

As already indicated, transparency is a contested term, especially when it comes to the use of software and ML [41,49,71]. In this section, we first discuss the term transparency in general, and then in the context of software design and use. Finally, we propose the *interactive understanding of transparency*.

Transparency in Organizations. Originally coined in a physical context [54, 62], the term was adopted in a figurative manner to social contexts to, e.g., hold members of governments and other organizations legitimate, accountable, or to derive an inter-subjective truth and knowledge about their behavior and actions [35]. Considering it, the term today is widely recognized for its ameliorating potential [7]. Although the early conceptions of transparency were based on the direct observability of actors, inline with the physical sense, the intended motives mentioned above presumed a critical public, e.g., in form of a public-opinion tribunal, to challenge deceptive self-representations [8, p. 158].

While this classical form of transparency still can be found in grassroots democratic initiatives, nowadays the term is widely brought down to information-disclosure, e.g., on financial or social affairs of organisations [48,71]. Even if this understanding seems to be intuitive and widely accepted [2,74], it causes several problems. E.g., as information does not equal facts and often

results in self-interested representation, this might lead rather to obfuscation than legitimate knowledge or accountability [15,40,70]. In a similar vein, it is argued that transparency reduced to information disclosure might result in an information overload, hindering a proper assessment [3,44], or that the skills and legibility of a transparency-requester have to be considered, so that information is understood in the intended way [21,49]. As a remedy some propose to view *transparency as a process*, where stakeholders look actively into an organization by evaluating, if the information provided meets their needs and seems relevant [2,44,70].

In-/Transparency of Software. Even if software and related terms, such as algorithms or models, seem to be easy to grasp, a closer look reveals that software can be represented in several ways, e.g., code, documentation, or metrics, as used especially by data scientists to evaluate ML software [14,47,75]. While these views often correspond to transparency-demands in intra-disciplinary settings, such narrow technical understanding of software is of little use, when it comes to transparency-demands of other stakeholders [6,67]. As stated above, also in case of software, provision of narrow and one-sided information might not be sufficient to satisfy related goals of transparency, or even worse, might be dysfunctional to the intended motives of transparency-demands [43,75].

For example, [28] argues that transparency as proposed by the *General Data Protection Regulation* (GDPR) [19] will more likely result in self-interested representations, than in what was intended by the regulation, namely gaining insights and knowledge about the use of personal data for the data-subject. Similarly, but related to Algorithmic-Decision Making (ADM) Systems, [4] argue that "[t]o ask to 'look inside the black box' is perhaps too limited demand and ultimately an ill-fitting metaphor" (p. 982) to gain knowledge. By referring to [50, p. 6], [4] stress that the creation of knowledge as well as individual understanding, both need many views, especially when it comes to such complex systems as ADM. In particular, when transparency should improve accountability, narrow-technical views might intentionally occlude [4, p. 980][18, p. 1830] therefore, to foster accountability, it is demanded to consider responsible persons as well.

With respect to ML systems, apart from the problem that the access to software might not be possible for good reasons (e.g., intellectual property rights, security reasons), or require specialist knowledge [12], in some cases (e.g., artificial neural networks) parts of the software are rather complex and are difficult to be interpreted and explained also by experts [12, p. 4][63, p. 206]. While for data scientists several metrics are proposed to estimate the behavior of the model [32], for (potential) users of such a software system they are of little help, since they are hard to interpret and do not at all explain reasons for decisions [63]. Therefore, to give potential users at least a chance to gain knowledge about a system, other notions of transparency have been proposed. For instance, under the label of *practical transparency*, it is demanded to inform users about assumptions and potential risks, and to enable their interactions with a system to learn how it behaves [59,61]. Furthermore, several questionnaires have been proposed to help users evaluate, if a system is appropriate for the intended context [29,51].

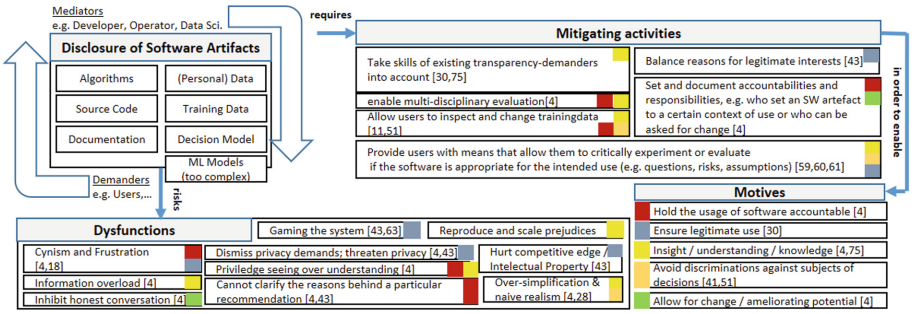

Fig. 1. Interactive understanding of transparency in the design&use of software

Interactive Understanding. Based on the conducted analysis, we propose to consider transparency in the design and use of software in organizations not as a state, but as an interactive process, that comprises various activities between stakeholders, and that depends on the motives of transparency-demand, which often will go beyond the ordinary provision of information, cf. Fig. 1. While in intra-disciplinary settings the provision of a view demanded might satisfy the transparency-demand (e.g., source code for programmers or algorithms and hyperparameters for data scientists), especially in those settings where different stakeholders with different professions strive for transparency, other activities might be of relevance too. In particular, if the demand for transparency is related to social ambiguous concepts like accountability or legitimacy, other activities get relevant to capture the ameliorating potential of transparency. We term these activities *mitigating activities* and present them in Fig. 1, where they are related to potential motives of transparency by colored squares. In addition, we also list dysfunctions that are discussed in literature when transparency is understood as a pure disclosure. Please note that due to space limitations, neither is the list of motives comprehensive, nor is the list of mitigating activities complete. Nevertheless, this selection allows us to show in following sections, how EM can be used to support this conception, while avoiding dysfunctional effects.

3 Goals, Requirements and Existing Approaches

We argue that CM can foster the introduced interactive understanding of transparency in various ways, e.g., by capturing domain-specific knowledge [46], or by documenting information exchanges [36]. In line with the proposed understanding of transparency on the one hand, and the specifics of software design and use on the other hand, we focus here on those scenarios and requirements (denoted 'Rx') that facilitate the interaction between stakeholders of software systems, e.g., user, programmer, data scientists, while providing support for specific views, and related analysis. The requirements have been systematically derived in line with [24]. Due to space restrictions we introduce the identified requirements on a high-level only.

Goals and Requirements. The main goal of the targeted approach is to provide support for the interaction of different stakeholders that are related to design and/or use of software, so that they can satisfactorily fulfill their transparency needs. To this aim, the modeling approach should provide systematic support for different domain-specific perspectives (R1), while at the same time capture relations between those perspectives to support interactions (R2). The targeted approach is to be used in an inter-disciplinary setting, where different stakeholders assign and assess transparency-demands related to a software and its usage. Since software (i) can be quite complex, and (ii) is amorphous and may be represented in various views (e.g., source code for programmers or user interface for users), the approach should provide means to relate these various representations to abstract notions that matter in relevant discourses, i.e., on a language level. With respect to complexity, it should also provide (i) means for decomposition, and (ii) differentiated information on its parts (R3). At the same time, the approach should relate different views to a view-independent, rather abstract software concept, that is subject of the overarching discussion (R4).

To align views on a software with corresponding competencies, e.g., code-literacy, of stakeholders, the following questions should be considered: (Q1) What stakeholders are related to a software and its usage? (Q2) What are the competences of a stakeholder and what domain-specific views are related to them? (Q3) Which views on a software artifact are available? (Q4) Does a stakeholder have access to the available views? (Q5) Who can grant access, if a view is already available? (Q6) Who is responsible for a software artifact and might support the construction of a view? In line with these questions, the approach should support the representation of stakeholders and their competencies, as well as views onto a software that fits those competencies (R5). Additionally, stakeholders with transparency-demands can be manifold, and range from specific individuals to specific types, e.g., programmers. Similarly, also transparency-demands can be assigned to individuals, or types of stakeholder. Therefore, the approach should provide dedicated abstractions differentiating among stakeholder groups and accounting for individual stakeholders (R6).

In line with the proposed interactive understanding, it is important to not only provide information, e.g., in the form of certain views, but also to consider the purpose(s) of transparency demands, e.g., to avoid unintended or dysfunctional effects. While these purposes can be manifold and need specific considerations that cannot be discussed here in detail, e.g., for transparency and accountability, cf. [4,43], we point here to the purpose of legitimacy due to its specific relevance. Namely, acknowledging that stakeholders may reject to work with an organization due to a perceived lack of legitimacy [30], it is of central relevance to strive for legitimacy that can be understood as "a generalized perception or assumption that the actions of an entity are desirable, proper, or appropriate within some socially constructed system of norms, values, beliefs and definitions" [68, p. 574]. For the software itself and its use, this means that even if access to information is granted and well-understood by a stakeholder, they might consider the circumstances that information expresses as illegitimate,

risking frustration [18] and turning away. To support the discussion of a legitimate use of a software, the approach should provide concepts that document reasons, e.g., for decisions made, or for rejecting transparency-demands (R7), as well as the state of legitimacy perceived by stakeholders (R8). In addition, it is necessary to document the purpose of a transparency-demand to provide a basis for discussions of unintended or dysfunctional effects (R9).

With respect to the development and use of, e.g., ML, the modeling approach can support what was introduced as *practical transparency* (cf. Sect. 2). Apart from providing diagrams that allow to answer different questions, by capturing assumptions and (not-)intended use cases of (ML) software, the approach might help potential users evaluate, if the software is appropriate for their use (R10). In addition, questionnaires already included in diagrams and directly associated with specific software artifacts might be of help (R11). Even if the different questionnaires are already in use and provide a good orientation for users [60], we propose that CM might foster reuse, if questions and assumptions can be collected during implementation (R12), while being evaluated in diagrams of their context of use (R13). Finally, considering the risk of inappropriate and deflective diagrams or models [13, p. 164][42, p. 2], the information on software provided should be linked to its actual implementation (or its model). It should be indicated whenever the information might be outdated (R14).

Existing Approaches. Various (standalone) modeling approaches exist that support understanding of selected business-related and IT aspects. However, as these standalone modeling approaches focus on selected aspects of an enterprise only, they do not allow for a comprehensive, integrated analysis accounting for multiple perspectives (cf. R1&R2). Such an integrated perspective is offered, as already mentioned, by enterprise modeling approaches. Several EM approaches exist that support modeling of IT infrastructure (cf. R3) in the context of an enterprise action system, e.g., ArchiMate [69], Architecture of Integrated Information Systems (ARIS) [64], and Multi-Perspective Enterprise Modeling (MEMO) [25] with the IT Modeling Language (ITML) [27,34]. Each of these approaches has been designed with a set of intended scenarios in mind [9], supporting transparency analyses, as discussed in this paper, not being one of them. Therefore, to support our vision some extensions to those approaches would be required. Although these approaches exhibit similarities, cf. [9], they also differ substantially in terms of the domain coverage and semantic richness of offered concepts, which is necessary to address the identified requirements (cf. e.g., R3). While ArchiMate and ARIS favor a concise language design by focusing on a small set of essential enterprise (architecture) concepts, MEMO provides domain stakeholders with elaborate reconstructions of the (technical) concepts. Particularly, while ArchiMate, ARIS and MEMO offer means to describe IT infrastructure, they do so at different levels of granularity. And so, ArchiMate provides a set of generic concepts where attributes can only be specified per instance, but not on a language level, which would be however required to differentiate various software artifacts (cf. R3). Similarly, although ARIS offers an extensive set of diagram types, its individual diagram types offer generic

concepts with few attributes and relations only. In contrast, MEMO ITML offers a set of more fine-grained concepts with a rich set of attributes (cf. R3&R4). Considering the above, MEMO seems to be a promising approach to support our aims. However, it lacks the ability to, among others, express different views on software and relate them to different competencies of stakeholders, support stating transparency demands or documenting results of analysis, cf. R7 and R8. It also falls short, when it comes to supporting analysis of suitability of a system to certain scenarios (R10), transparency questionnaires (R11–R13), or linking the information to software implementation (R14). Therefore, we take MEMO as point of departure and propose corresponding extensions.

4 Extensions to MEMO in Support of Transparency

Several means of defining a modeling language exist. However, the one frequently used, also in case of MEMO, is by specifying a meta model. As we extend already existing DSMLs, we use the MEMO method's common Meta Modeling Language (MML) [25], and thus, integrate the extensions made into the MEMO method's language architecture. Compared to 'traditional' meta modeling languages, MML provides additional language constructs for expressing: (a) intrinsic attributes and relations, and (b) language-level types. Intrinsic attributes and relations are instantiated only on the instance level, but not on the type level. They are visualized with a white letter 'i' on a black background. In turn, language-level types are instantiated on the type level only, but not further. They are visualized with a grey-background of the concept's name [25].

In terms of the employed language design method, cf. [24], it is notable that: (1) we consider the use scenarios as the first class citizens that drive the design of the language, cf. previous section; and (2) we employ the guidelines for concept inclusion from [24]. Extensions as well as new concepts are shown in Fig. 2. Please note that due to space restriction only selected concepts, attributes, relations and Object Constraint Language (OCL) constraints are shown.

The core concept of interest is `Software`, cf. Fig. 2, originally defined in the ITML, characterized through a rich set of attributes (e.g., version, documentation, source code) and associated with other concepts as, e.g., programming language implementing it, libraries used, functions provided and used, or `UseCase`s it is supposed to support (R10). A software may be represented and stored as a `File`. A software can be used in various usage contexts (`UseContext`), e.g., in processes (`AnyProcess`, defined in OrgML [25]), or to satisfy certain goals (`AbstractGoal`, part of MEMO GoalML [57]). In a given usage context, a software provides a `SpecificSupport` with such attributes as IT artifact relevance or support quality. This allows for instance to express, whether a given process type can be also realized without the support of a given type of software artifact.

A `Software` can be decomposed (R3) via a part-of relation. Thus, it is possible to model, e.g., an ERP System (as `ApplicationAndSystemSoftware`), and to decompose it into its different modules (e.g., HR management, financial management) down to the level of sub-routines, if of relevance for transparency

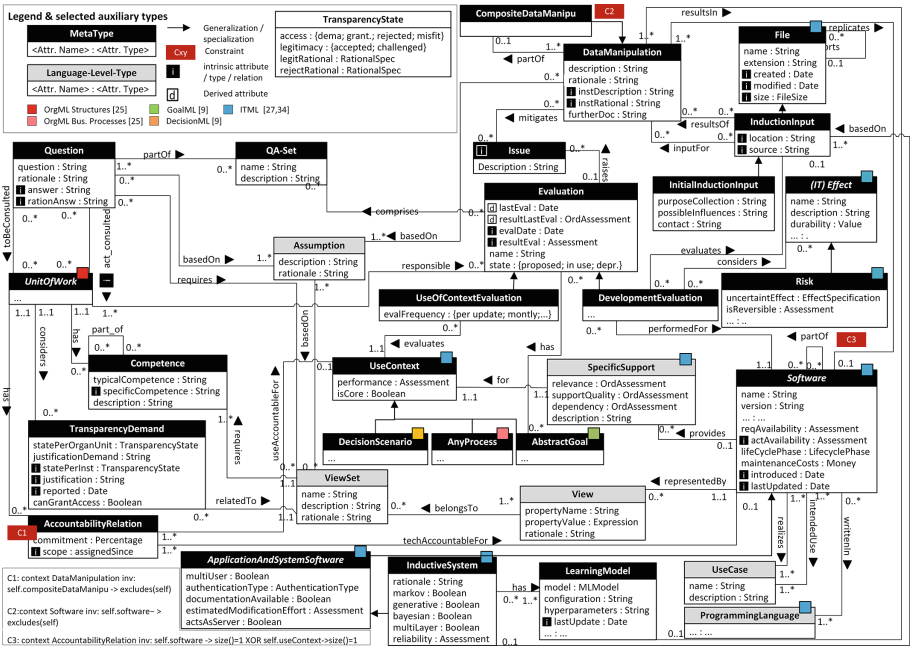

Fig. 2. Meta model excerpt: extended ITML & integration with other DSMLs.

discussions. The software and corresponding modules can be represented by different `Views` (R4) encompassing a property of interest and a way it should be derived/calculated. Those views can belong to a `ViewSet`, which considers the `TransparencyDemand` of different stakeholders. In addition, each `ViewSet` requires some `Competency` to be understood.

The meta class `TransparencyDemand` is central, as it allows to capture the current view of a stakeholder on a software, which helps derive the specific state of transparency in an interactive setting where various users participate (cf. also R7). For example, if a particular user demands on a certain date access to the source code (view) (R4) of a software, e.g., (R9, via justification) to learn about its behavior, e.g., in form of if-else statements, then this demand can be expressed with the attributes on the instance level and the auxiliary type `TransparencyState` in this case is 'demanded'. If all users, e.g., in the position of HR recruiters, demand this access, this can be expressed on the type level.

To account for different stakeholders and their groups, we use the abstract meta class `UnitOfWork` from MEMO OrgML [25], specialized into other organizational concepts (e.g., Organizational Unit), to express information on the type level, and through intrinsic attributes and relations, on the instance level (R6). The `UnitOfWork` can be related to `Competence`s, which allows to analyze, if the access onto views, if granted, can be of use for the stakeholders (R5).

In order to capture whether the state of social affairs realized by a software (e.g., a hetero-normative view in a registration software) is considered legitimate by the stakeholders, `TransparencyState` provides a corresponding attribute to capture it (R8). However, when it comes to the use of a software, we propose that (il)legitimacy can be also the result of a `UseOfContextEvaluation` that can be conducted several times (captured by intrinsic attributes), but where most relevant seems to be the result of the last evaluation (derived from the intrinsic attributes). These evaluations can be based on *QA-Set*s (R11), i.e., to guide the evaluation per `UseContext` as specific as possible, and also support *practical transparency* in the case of ML systems. Even if those `QA-Set`s can be independently defined, they can also stem from a `DevelopmentEvaluation` that is performed during the design phase of a software, independent from its context of use. Here also assumptions and potential risks can be collected (R12), via dedicated concepts, that developers have in mind when publishing a software.

When it comes to the development of ML systems, we consider induction from a `DataInput` as a central characteristic of class of software using ML (`InductiveSystem`). The induction can be based on various ML Models (e.g. CART/C 4.5, Artificial Neural Networks) that come with specific configurations and hyper-parameters an `InductiveSystem` is based on. Important is however, that the process of building such a model depends on various activities, among others, e.g., data cleansing or preparation. We capture such activities with the meta class `DataManipulation` that can be part of `CompositeDataManipulation`. Since these activities can be used to mitigate `Issues` that stem from evaluation of `InductionInput`, a relation to `DevelopmentEvaluation` has been defined.

5 Exemplary Application

As we have pointed to the fulfillment of identified requirements already while describing the extended meta model, here we illustrate how the extended approach may be used in support of transparency analysis.

Figure 3 shows three integrated diagrams supporting interactive transparency. At the very bottom, we present a *ML Development diagram* that is used to document activities, assumptions and rationale during the development of a specific (inductive) software system. The content of the diagram is inspired by a dataset provided in Kaggle [39], a platform for data scientists. It shows the development of a software called *leaveCompPrediction* (LCP), which should support the `UseCase` of predicting the probability of a job change. Developers can use this diagram for an intra-disciplinary form of transparency, i.e., to document what data is taken as `InductionInput`, and how it is processed (`DataManipulation` steps). In addition, inline with the discussion about the kaggle dataset [39,46], several `issue`s, activities and `rationale`s are documented, which allows developers of the software not only to provide information about narrow technical software artifacts, but also, e.g., to behave responsibly (in the sense of the capacity to respond [73]) towards users of the software. Next, this

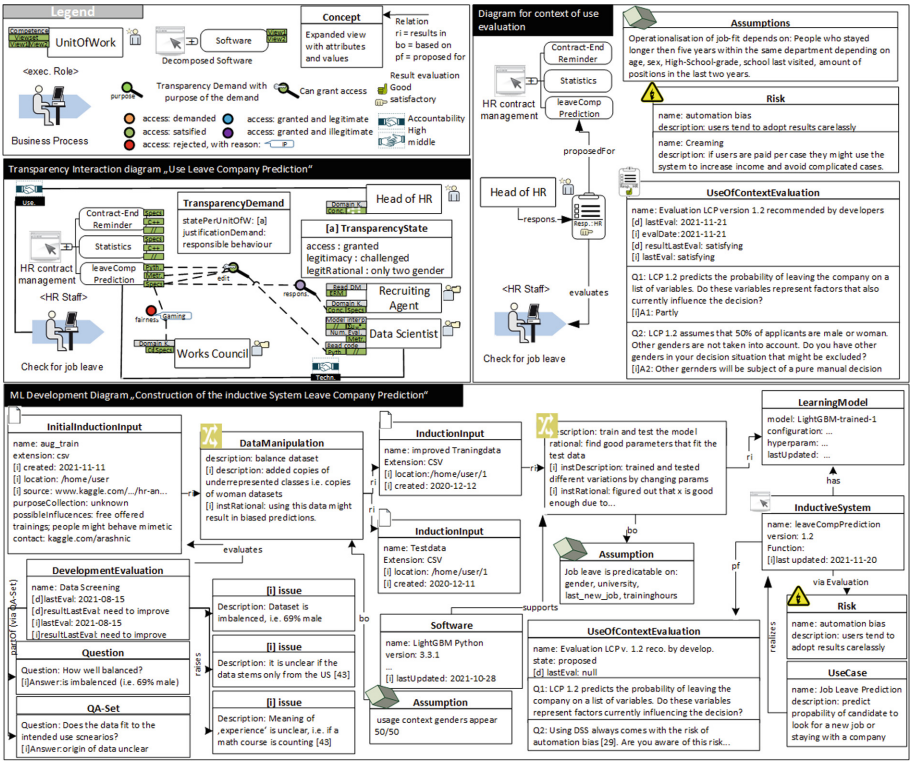

Fig. 3. Diagrams in support of an interactive transparency

diagram captures also `assumptions`, potential `risks` and questions, which should be considered during the use of the LCP. For instance, the LCP is based on the `assumption` that the risk of a job leave can be predicted only via the variables such as gender, university, duration of last job and hours of training within the current company. All assumptions, risks and potential questions can be bundled by the developers as a `UseOfContextEvaluation` with a state `proposed`, and provide a basis for a critical reflection of the software in support of a *practical transparency*. This critical reflection can be supported via the *Diagram for the Context of use evaluation*, where a `Software` and its related `Risks`, `Assumptions` are presented. Answers to the `UseOfContextEvaluation` can be captured per use case, and engage a discussion on the appropriateness of the specific software in this `UseContext`. In this case the LCP is used as part of an *HR contract management* `Software` for a business process in a specific company. Answers to the `UseOfContextEvaluation` are provided by various stakeholders (not shown here), and the `Head of HR` as responsible `UnitOfWork` that seems at least to be satisfied. However, during the use of the LCP the *Work Council* of our case company has a `TransparencyDemand` to clarify complaints about discrimination during contract renewals. Via a *business process model* (not shown here) the

Work Council identifies that the LCP is associated with this process, and that an evaluation was conducted by the Head of HR. To get a first impression they ask the Data Scientist who is technically responsible for specifications of the software. The Data Scientist rejects this decision however, due to the risk of gaming the system. The Work Council considers this reason as legitimate and asks the Recruiting Agents that use the LCP. To behave responsible to the Work Council, the Recruiting Agents ask for an access to the metrics, which is granted. However, by discussing the assumptions, i.e., the factors used for the prediction, the Recruiting Agents come to the conclusion that the software is not legitimate. By considering that the transparency demand relates to discrimination, the Head of HR starts a discussion about fairness. The *Transparency Interaction diagram* captures this situation, and allows to answer questions about responsibilities or available views and their accessibility, and whether stakeholders might make.

6 Conclusions

In this paper, based upon the conducted analysis, we propose an interactive understanding of transparency and identify requirements that an EM approach should fulfill to support this understanding. As none of existing approaches fulfills all requirements, we extend MEMO, in particular the ITML, to support transparency analysis of software design and use. Then, we show how the *extended* ITML can be applied to an exemplary scenario.

The extensions introduced into ITML enhance the set of available analysis scenarios, among others, assessing legitimacy of software design and use. Please note however, that while most of the requirements are being fulfilled through dedicated concepts and relations, some of the aspects have been only superficially addressed, e.g., the concept of competencies and cognitive skills of involved stakeholders related to ideas of views and perspectivity, or not at all, e.g., linking the information on a software artifact to its actual implementation (R14). In addition, due to space limitations, we have focused here on a selected class of software systems only, namely induction-based systems by taking more pragmatic considerations into account. We acknowledge also that a process model guiding the use and adoption of the extended MEMO might be needed. Currently, its usage requires specific skills, and the judgment of transparency measures is dependent on those involved. Finally, while the application of MML allowed us to take advantage of the intrinsic features and relations, and thus, to refer to the instance level, we have faced numerous challenges pertaining to the restrictions given by the type/instance dichotomy or the semantic differences between instantiation and specialization, cf. [26]. As in conventional meta modeling, there is no 'perfect' solution to the mentioned challenges, cf. [5,26], for our future research the application of multi-level modeling [5,26], seems promising. In addition, as some multi-level modeling approaches support integrated modeling and programming [26], also R14 could be in this way fulfilled.

References

1. ACM Public Policy Council: Statement on algorithmic transparency and accountability (2017. https://www.acm.org/. Accessed 1 July 2021
2. Albu, O.B., Flyverbom, M.: Organizational transparency: conceptualizations, conditions, and consequences. Bus. Soc. **58**(2), 268–297 (2019)
3. Alloa, E.: Transparency: a magic concept of modernity. In: Alloa, E., Thomä, D. (eds.) Transparency, Society and Subjectivity, pp. 21–55. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77161-8_3
4. Ananny, M., Crawford, K.: Seeing without knowing: limitations of the transparency ideal. New Med. Soc. **20**(3), 973–989 (2016)
5. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. SOSYM **7**(3), 345–359 (2008). https://doi.org/10.1007/s10270-007-0061-0
6. Barocas, S., Hood, S., Ziewitz, M.: Governing algorithms: provocation piece, SRRN (2013). http://dx.doi.org/10.2139/ssrn.2245322
7. Baume, S.: Publicity and transparency: the itinerary of a subtle distinction. In: Alloa, E., Thomä, D. (eds.) Transparency, Society and Subjectivity, pp. 203–224. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77161-8_10
8. Bentham, J.: Constitutive authority. In: Bowring, J. (ed.) The Works of Jeremy Bentham, pp. 155–160. Russell and Russell, New York (1962)
9. Bock, A., Kaczmarek, M., Overbeek, S., Heß, M.: A comparative analysis of selected enterprise modeling approaches. In: Frank, U., Loucopoulos, P., Pastor, Ó., Petrounias, I. (eds.) PoEM 2014. LNBIP, vol. 197, pp. 148–163. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45501-2_11
10. Bork, D., Roelens, B.: A technique for evaluating and improving the semantic transparency of modeling language notations. Softw. Syst. Model. **20**(4), 939–963 (2021). https://doi.org/10.1007/s10270-021-00895-w
11. Brown, S., Davidovic, J., Hasan, A.: The algorithm audit: scoring the algorithms that score us. Big Data Soc. **8**(1) (2021). https://doi.org/10.1177/2053951720983865
12. Burrell, J.: How the machine 'thinks': understanding opacity in machine learning algorithms. Big Data Soc. **3**(1) (2016). https://doi.org/10.1177%2F2053951715622512
13. Christensen, L.T.: Corporate communication: the challenge of transparency. Corp. Commun. Int. J. **7**(3), 162–168 (2002)
14. Chun, W.H.K.: Programmed Visions: Software and Memory. MIT Press, Cambridge (2011)
15. Deetz, S., Mumby, D.: Metaphors, information, and power. In: Ruben, B.D. (ed.) Information and Behavior, pp. 369–385. Transaction Inc., New Brunswick (1985)
16. do Prado Leite, J., Cappelli, C.: Software transparency. BISE **2**(3), 127–139 (2010)
17. Dobbe, R., Dean, S., Gilbert, T., Kohli, N.: A broader view on bias in automated decision-making. FATML, Stockholm (2018). https://doi.org/10.48550/arXiv.1807.00553
18. Draper, N.A., Turow, J.: The corporate cultivation of digital resignation. New Med. Soc. **21**(8), 1824–1839 (2019)
19. European Parliament and the Council of European Union: Regulation (EU) no 679/2016 (GDPR) (2016). https://eur-lex.europa.eu. Accessed 1 July 2021
20. European Parliament and the Council of European Union: Regulation (EU) no 1150/2019 (2019). https://ec.europa.eu. Accessed 1 July 2021
21. Fenster, M.: The opacity of transparency. Iowa L. Rev. **91**, 885 (2005)

22. Fill, H.G.: Abstraction and transparency in meta modeling. In: Schweighofer, E., Kummer, F., Hötzendorfer, W., (ed.) Transparency, pp. 435–442. Österreichische Computer Gesellschaft, Salzburg (2014)
23. Fleischmann, K.R., Wallace, W.A.: Ensuring transparency in computational modeling. Commun. ACM **52**(3), 131–134 (2009)
24. Frank, U.: Outline of a Method for Designing Domain-Specific Modelling Languages. ICB Research Report 42, University of Duisburg-Essen, Essen (2010)
25. Frank, U.: Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Softw. Syst. Model. **13**(3), 941–962 (2012). https://doi.org/10.1007/s10270-012-0273-9
26. Frank, U.: Multilevel modeling - toward a new paradigm of conceptual modeling and information systems design. BISE **6**(6), 319–337 (2014)
27. Frank, U., Kaczmarek-Heß, M., de Kinderen, S.: IT infrastructure modeling language. ICB Research Report 72, Essen (2021)
28. Fuster, G.G.: Transparency as translation in data protection. In: BEING PROFILED, pp. 52–55. Amsterdam University Press (2018)
29. Gebru, T., et al.: Datasheets for datasets. CACM **64**(12), 86–92 (2021)
30. Goad, D., Gal, U.: Understanding the impact of transparency on algorithmic decision making legitimacy. In: Schultze, U., Aanestad, M., Mähring, M., Østerlund, C., Riemer, K. (eds.) IS&O 2018. IAICT, vol. 543, pp. 64–79. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04091-8_6
31. Goldenfein, J.: Algorithmic transparency and decision-making accountability. In: Closer to The Machine: Technical, Social and Legal Aspects of AI, pp. 41–61 (2019)
32. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. CSUR **51**(5), 1–42 (2018)
33. Guszcza, J., Rahwan, I., Bible, W., Cebrian, M., Katyal, V.: Why we need to audit algorithms. HBR (2018)
34. Heise, D.: Unternehmensmodell-basiertes IT-Kostenmanagement als Bestandteil eines integrativen IT-Controllings. Logos, Berlin (2013)
35. Hood, C., Heald, D.: Transparency in historical perspective. In: Hood, C., Heald, D. (ed.) Transparency: the Key to Better Governance? Oxford University Press (2006)
36. Hosseini, M., Shahri, A., Phalp, K., Ali, R.: Engineering transparency requirements: a modelling and analysis framework. Inf. Syst. **74**, 3–22 (2018)
37. IEEE - Institute of Electrical and Electronics Engineers Inc: IEEEP7001 - Transparency of autonomous systems (draft) (2020)
38. Jobin, A., Ienca, M., Vayena, E.: The global landscape of AI ethics guidelines. Nat. Mach. Intell. **1**(9), 389–399 (2019)
39. Kaggle: HR analytics: Job change of data scientists (2020). https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists
40. Kitchin, R.: Thinking critically about algorithms. Inf. Comm. Soc. **20**(1), 14–29 (2017). https://doi.org/10.1080/1369118X.2016.1154087
41. Kohli, N., Barreto, R., Kroll, J.A.: Translation tutorial: a shared lexicon for research and practice in human-centered software systems. In: 1st Conference on Fairness, Accountability, and Transparency, New York, NY, USA, vol. 7 (2018)
42. Krogstie, J.: Model-Based Development and Evolution of Information Systems: A Quality Approach. Springer Science & Business Media, London (2012). https://doi.org/10.1007/978-1-4471-2936-3
43. de Laat, P.B.: Algorithmic decision making based on ML from big data. Philos. Technol. **31**(4), 525–541 (2018)

44. Lee, T.H., Boynton, L.A.: Conceptualizing transparency: propositions for the integration of situational factors and stakeholders' perspectives. Public Relat. In. **6**(3), 233–251 (2017)
45. Lukyanenko, R., Castellanos, A., Parsons, J., Tremblay, M.C., Storey, V.C.: Using conceptual modeling to support machine learning. In: Cappiello, C., Ruiz, M. (eds) Information Systems Engineering in Responsible Information Systems. CAiSE 2019. Lecture Notes in Business Information Processing, vol. 350, pp. 170–181. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21297-1_15
46. Maass, W., Storey, V.C., Lukyanenko, R.: From mental models to machine learning models via conceptual models. In: Augusto, A., Gill, A., Nurcan, S., Reinhartz-Berger, I., Schmidt, R., Zdravkovic, J. (eds.) BPMDS/EMMSAD -2021. LNBIP, vol. 421, pp. 293–300. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79186-5_19
47. Margetts, H.: The internet and transparency. Political Art. **82**(4), 518–521 (2011)
48. Meijer, A.: Understanding modern transparency. Int. Rev. Admin. Sci. **75**(2), 255–269 (2009)
49. Michener, G., Bersch, K.: Conceptualizing the quality of transparency. Polit. Concepts **49**, 1–27 (2011)
50. Minsky, M.: The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind. Simon and Schuster, New York (2007)
51. Mitchell, M., et al.: Model cards for model reporting. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 220–229 (2019)
52. Mittelstadt, B.: Automation, algorithms, and politics— auditing for transparency in content personalization systems. Intl. J. Commun. **10**, 12 (2016)
53. Mylopoulos, J.: Conceptual Modelling and TELOS. Conceptual Modelling, Databases, and CASE: An Integrated View of is Development, pp. 49–68 (1992)
54. Newton, I.: Opticks, or, A Treatise Of The Reflections, Refractions, Inflections & Colours of Light. Courier Corporation (1952)
55. Organisation for Economic Co-operation and Development: Recommendation of the council on artificial intelligence (2021)
56. Österle, H., et al.: Memorandum zur gestaltungsorientierten Wirtschaftsinformatik. ZfBF **62**(6), 664–672 (2010)
57. Overbeek, S., Frank, U., Köhling, C.: A language for multi-perspective goal modelling: challenges, requirements and solutions. CSI **38**, 1–16 (2015)
58. Pasquale, F.: The Black Box Society. Harvard University Press, Cambridge (2015)
59. Paßmann, J., Boersma, A.: Unknowing algorithms: on transparency of unopenable black boxes. In: Schäfer, M., van Es, K. (eds.) The Datafied Society, pp. 139–146 (2017)
60. Raji, I.D., et al.: Closing the AI accountability gap: defining an end-to-end framework for internal algorithmic auditing. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 33–44 (2020)
61. Resnick, M., Berg, R., Eisenberg, M.: Beyond black boxes: bringing transparency and aesthetics back to scientific investigation. J. Learn. Sci. **9**(1), 7–30 (2000)
62. Rey, A. (ed.): Dictionnaire historique de la langue française. Le Robert, Dictionnaires Le Robert, Paris, nouv. éd edn. (1995)
63. Rudin, C.: Stop explaining black box ML models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. **1**(5), 206–215 (2019)
64. Scheer, A.W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen, 4th edn. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-642-97731-2

65. Schuette, R., Rotthowe, T.: The guidelines of modeling – an approach to enhance the quality in information models. In: Ling, T.-W., Ram, S., Li Lee, M. (eds.) ER 1998. LNCS, vol. 1507, pp. 240–254. Springer, Heidelberg (1998). https://doi.org/10.1007/978-3-540-49524-6_20
66. Schwarzer, B., Krcmar, H.: Wirtschaftsinformatik?: Grundlagen betrieblicher Informationssysteme, 4th edn. Schäffer-Poeschel, Stuttgart (2010)
67. Seaver, N., Vertesi, J., Ribes, D.: Knowing algorithms. In: digitalSTS, pp. 412–422. Princeton University Press (2019)
68. Suchman, M.C.: Managing legitimacy: strategic and institutional approaches. Acad. MGM Rev. **20**(3), 571–610 (1995)
69. The Open Group: ArchiMate 2.1 Specification: Open Group Standard. The Open Group Series, Van Haren, Zaltbommel (2013)
70. Timothy Coombs, W., Holladay, S.J.: The pseudo-panopticon. Corp. Commun. Int. J. **18**(2), 212–227 (2013)
71. Turilli, M., Floridi, L.: The ethics of information transparency. Ethics Inf. Technol. **11**(2), 105–112 (2009)
72. Voß, S.: Informationsmanagement : mit 25 Tabellen. Springer, London (2001)
73. Waldenfels, B.: The Question of the Other. Chinese University Press, Hong Kong (2007)
74. Wehmeier, S., Raaz, O.: Transparency matters: the concept of organizational transparency in the academic discourse. PR In. **1**(3), 337–366 (2012)
75. Weller, A.: Transparency: motivations and challenges. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.-R. (eds.) Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. LNCS (LNAI), vol. 11700, pp. 23–40. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28954-6_2
76. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. **26**(3), 276–292 (1987)