A Systematic Evaluation of Enterprise Modelling Approaches on Their Applicability to Automatically Generate ERP Software

D.M.M. Schunselaar*, J. Gulden†, H. van der Schuur‡, H.A. Reijers*
* VU University Amsterdam, The Netherlands
† University of Duisburg-Essen, Germany
‡ AFAS Sofware, The Netherlands,
Email: d.m.m.schunselaar@vu.nl, jens.gulden@uni-due.de, h.vdschuur@afas.nl, h.a.reijers@vu.nl

***Abstract*— Customising Enterprise Resource Planning (ERP) software to the enterprise's needs is still a technical endeavour often involving enabling/disabling modules, modifying configuration files, etc. This is quite surprising given the large body of work on Enterprise Modelling and Model-Driven Software Engineering. Ideally, one models one's enterprise and, with a press on a button, the ERP software supporting the enterprise is generated. In this paper, we present a systematic evaluation of Enterprise Modelling approaches for their applicability to automatically generate ERP software supporting an enterprise. Inspired by the work of an ERP vendor, which, next to generating ERP software to support an enterprise, also wants to incorporate the common-sense[1] of an enterprise in the ERP software, we also evaluate Enterprise Ontologies since current Enterprise Modelling languages lack semantics on the instance level. None of the existing approaches are tailored towards automated (ERP) software generation without the need for programming. However, the approaches possess valuable aspects which can aid in the ERP software generation process. Therefore, next to the evaluation, we present take-home points from the approaches, e. g., reasoning capabilities to make common-sense suggestions when something is not possible.**

I. Introduction

Nowadays, enterprises cannot operate without software. Often, enterprises are supported by Enterprise Resource Planning (ERP) software. Often this ERP software is not directly tailored towards a specific enterprise. In order to tailor the ERP software, often configuration files need to be modified, components need to be enabled / disabled, etc. All of this happens at the technical (IT) level (left-hand side of Fig. 1). At the same time, if certain functionality is not supported, extensions are made to the ERP software (mis)using existing functionality / concepts.

With the large body of work on Enterprise Modelling and Model-Driven Software Engineering, it is surprising that we are still configuring IT-artefacts. In the ideal world, one models an enterprise and, with a press on the button, the ERP software supporting the enterprise is generated (right-hand side of Fig. 1). This is the goal AFAS, a Dutch ERP vendor, has set itself.

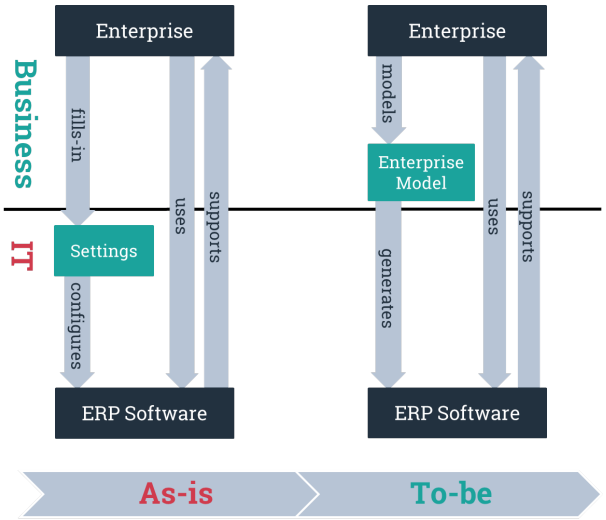

Figure 1. Currently, enterprises use IT-artefacts like configuration files and other kinds of settings to configure the ERP software to the customer's needs. In the future, an enterprise models its enterprise and, based on this, the ERP software is generated perfectly capable of supporting the enterprise.

AFAS is developing a *Ontological Enterprise Model* (OEM)[2]. This OEM will be expressive enough to fully describe the real-world enterprise of virtually any customer, and as well form the main foundation to automatically generate the ERP software supporting the enterprise. This means that (a) the OEM is formally defined, and (b) every construct in the OEM has precise semantics. At the same time, this OEM should be able to use the *common-sense* of an enterprise. Using this common-sense, it becomes possible to deduce new facts. For instance, it does not make sense to suggest a meeting room with a capacity of 8 people for a meeting with 10 participants. This means that the OEM should "know" the concept of capacity. The knowledge of the OEM does not stop at these

This is an AMUSE paper. See amuse-project.org for more information.

[1]Common-sense is used to denote fundamental/basic enterprise rationale, e. g., impossibilities due to capacity constraints (like the fact that a meeting for 10 people cannot be scheduled in a room with 8 seats), computations of payment slips, collective labour agreements, etc.

[2]See vision AFAS for the complete vision (in Dutch).

simple checks. It should be familiar with business rules, laws and regulations, time dependencies, etc. In essence, common-sense are those aspects one would not consider modelling explicitly when describing an enterprise.

Within this paper, we present a systematic evaluation of related work on Enterprise Modelling approaches for the applicability to automatically generate ERP software. This means that (a) the approaches under consideration need to have formal (execution) semantics, and (b) need to allow for modelling aspects of an enterprise that are relevant for the generation of ERP software capable of supporting an enterprise. Inspired by AFAS to capture the real world combined with the common-sense of an enterprise, we also evaluate whether the Enterprise Modelling approaches have semantics on the instance level to reason over the intention of the model to enrich it with the common-sense. As we will see, most Enterprise Modelling approaches lack semantics on the instance level, e. g., all approaches have a notion of an activity, however, *what* the activity does often still needs to be encoded in source code by the enterprise (e. g., within ARIS [1, p. 50] in case the specification is not detailed enough or in CIMOSA [2, p. 261] where one can use Pascal-like procedural statements). Therefore, we also have incorporated Enterprise Ontologies in our comparison. For these Enterprise Ontologies, we evaluate if they can add the common-sense of the aspects of an enterprise under consideration. Next to the evaluation of the various approaches, we also present take-home points from the approaches for the OEM in particular and generating ERP software to support an enterprise in general.

This paper is structured as follows: We first present the requirements for the Enterprise Modelling / Enterprise Ontology approaches in Sect. II. In Sect. III, we present our comparison between the Enterprise Modelling / Enterprise Ontology approaches. We present the take-home points for each approach which could be valuable in Sect. IV. We conclude the paper with an overview of related work on comparing Enterprise Modelling / Enterprise Ontology approaches (Sect. V) as well as our conclusions and future work (Sect. VI).

II. Requirements towards Enterprise Modelling approaches

The requirements stem from the fact that we would like to let an end-user model her enterprise. Afterwards, with the press of a button, the ERP software is automatically generated. This brings three requirements on which we evaluate the current approaches: Firstly, the capability of an approach to capture the various aspects of an enterprise. Secondly, tool support for the approach. Thirdly, the accessibility of the approach for an end-user. The following sub-sections further elaborate on these requirements.

A. Enterprise aspects

For modelling the enterprise, we use the perspectives from [2], i. e., views on an enterprise with respect to its workflows, functions, information, resources, and its organisational structure. The reason why we focus on these five perspectives is that these contain relevant facts about the enterprise as a socio-technical action system, which can be interpreted as the requirements descriptions for desired functionality of software used for supporting the enterprise. As a consequence, these perspectives are best suitable for (ERP) software development [3]. In short, the various perspectives represent the following:

- Workflow: knowledge about when which actions can / will be executed;
- Functional: descriptions of how input is transformed into output and under which conditions at the various actions;
- Information: explication of knowledge as input / output of the enterprise's actions;
- Resources: descriptions of resources within the enterprise together with their capabilities, e. g., competences of a human resource;
- Organisation: structural relationships of organisational units together with responsibilities and authorities.

These perspectives do not take other aspects of an enterprise into account, e. g., modelling the goals of an enterprise [4] or how value flows through an enterprise [5, Ch. 3].

An approach needs to support each of these perspectives. Furthermore, these perspectives need to have *formal (execution) semantics* and *ontological information*. The first, formal semantics, is necessary to be able to generate code. The latter, ontological information, is necessary to be able to interpret the model to understand what the modeller is modelling. This allows the generated ERP software to reason over the model, e. g., to deduce if certain activities are permitted given the business rules.

B. Tool support

The tool support is two-fold. Both requirements on the tools allow the approach under consideration to be used to go from an Enterprise Model to running code. First, there needs to be support to create Enterprise Models and keep them updated regularly by means of efficiently usable diagrams and form based editors. But besides mere visual diagram editing capabilities, a suitable Enterprise Modelling tool should provide guidance for the modeller in keeping different views semantically consistent, and make sure by the offered edit functionality that model elements across perspectives correctly reference each other.

Next to this, there needs to be support to interpret the model or to generate ERP software. This means, besides handling the visual aspects of Enterprise Models for human interaction, the tool must also provide the modelled knowledge in an efficiently queryable structure for further automatic processing. The Eclipse Modeling Framework (EMF) [6] and its additions for visual modelling have proven to offer a flexibly extendable components that allow to further transform models to executable ERP software. Both a code generation approach, and runtime-interpretation, are equally expressive options for this.

C. Accessibility

An enterprise can be modelled at various abstraction levels with varying complexities. For instance, a programming language is already capable of capturing an enterprise to generate for instance ERP software. After all, the ERP software itself is written in a programming language. At the same time, offering a programming language to an end-user will not make the approach very accessible since an end-user does not necessarily possess knowledge of programming. Furthermore, someone familiar with programming does not necessarily possess knowledge of the enterprise.

AFAS' aim is to let the enterprise expert model the enterprise. At the same time, AFAS wants to keep the modelling as simple as possible. For an approach, we require that no programming (expert) is required.

III. Comparison of Enterprise Modelling approaches

In this section, we present how the approaches have been found as well as how they score against our requirements.

A. Finding the approaches

The approaches have been found by starting from a selection of known Enterprise Modelling approaches, i. e., ARIS, CIMOSA, and DEMO. From these approaches, we have applied an explorative search to find other Enterprise Modelling approaches referring to these works. This resulted in a total of 12 Enterprise Modelling approaches. Note that the Zachman framework [7] is not considered since this is not aimed at Enterprise Modelling, i. e., it presents a set of views on the information systems and to a lesser extent on the enterprise. Furthermore, it does not contain any explications of domain-specific modelling languages.

Of these 12 approaches, 4 were discarded since hardly any publications are available, which holds true for scientific work about the Dynamic Enterprise Modeling (DEM) and Extended Enterprise Modeling Language (EEML). The Semantic Object Model (SOM) was discarded since the majority of work is in German and available publications in English primarily focus on the business process model. As a result, the complete SOM approach could not be evaluated.

The above resulted in the following Enterprise Modelling approaches for our evaluation: ArchiMate, Architecture of Integrated Information Systems (ARIS), Computer Integrated Manufacturing Open Systems Architecture (CIMOSA), Design & Engineering Methodology for Organizations (DEMO), Multi-Perspective Enterprise Modelling (MEMO), Model-driven Entity Relationship Object-oriented DEvelopment (MERODE), and Unified Enterprise Modelling Language (UEML).

For the enterprise ontologies, there are two main approaches: the Enterprise Ontology [8], and the TOronto Virtual Enterprise (TOVE) [9]. Next to this, for UEML, an ontology extension has been made namely the Unified Enterprise Modelling Ontology (UEMO). Note that we view the work on DEMO [10] primarily as an enterprise modelling approach and not an enterprise ontology.

B. Approaches found

For each of the approaches found, we briefly introduce them in the upcoming sub-sections together with their scores on our requirements. Within the discussion of the approaches, we use *perspective* to denote the perspectives introduced in Sect. II, e. g., resource perspective, and we use *view* or *part* to denote the perspective in the approach.

1) ArchiMate: ArchiMate [11] is a modelling language to provide a uniform representation for diagrams that describe enterprise architectures. Within ArchiMate, there are three different layers: business layer, application layer, and the technology layer. In this paper, we primarily focus on the business layer. Within the business layer, there are *active structure elements*, *behaviour elements*, and *passive structure elements*.

Within the active structure, there are business actors. These can be, for instance, humans or organisational units. As a result, business actors correspond to the resource and organisation perspective. The active structure is formalised by means of a meta-model. There is no ontological information. The behaviour elements are used to define a unit of activity performed by one or more active structure elements [11]. The behaviour can be specified by means of business processes, business functions, interactions, events, and services. The business processes correspond to our workflow perspective. The business functions within ArchiMate do not correspond to our functional perspective since a business function in ArchiMate is defined as: "a behaviour element that groups behaviour based on a chosen set of criteria (typically required business resources and / or competences)" [11]. As a result, ArchiMate does not have a functional perspective. The execution semantics of the workflow perspective are only partially defined, i. e., "[..] It does not, however, list the flow of activities in detail. During business process modeling, a business process can be expanded using a business process design language; e. g., BPMN [..]" [11, p. 20]. At the same time, it is possible to denote causal relationships between processes, functions, interactions, and events by means of a *Triggering relationship* [11, p. 63]. The behaviour elements within ArchiMate do not possess any ontological information.

The passive structure elements consist, amongst others, of the business objects. These can be used to represent important informational elements. As a result, they coincide with our information perspective. For the business objects, it is possible to define multiple representations, e. g., the invoice business object which is represented as a paper bill. The business objects are part of the meta-model of ArchiMate. Within the passive structure, it is possible to have a *meaning* associated with all other elements, e. g., with a business process, business object, or a business actor. The meaning is not further defined. As such, it does not need to possess ontological information.

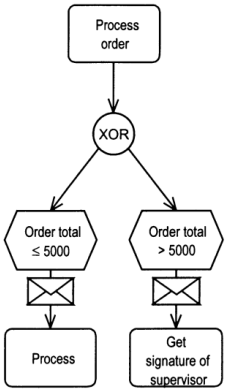

Figure 2. Example Event-driven Process Chain (EPC) for order processing. Dependent on whether the order total is more than 5000, the signature of the supervisor has to be requested. Figure taken from [1].

For ArchiMate, there is a modelling tool[3]. Since ArchiMate is focussed on modelling, there is no execution environment. There is no programming expert necessary for ArchiMate.

2) ARIS: ARIS [1] provides a set of modelling formalisms to model organisations. ARIS has been developed for the comprehensive description of information systems [1, p. VIII]. Within ARIS, there are the following views: *Control*, *Function*, *Data*, *Organisation*, and *Output*. We exclude the output view from our comparison as this does not correspond to any of our perspectives.

The control view of ARIS corresponds to the workflow perspective and partially to the resource perspective. Within the control view, the other perspectives are connected, e. g., which resource can execute a particular step. The workflow is modelled using Event-driven Process Chains (EPCs). In Fig. 2, an example EPC is depicted. Various papers have been published on formally defining the execution semantics of EPCs, e. g., [12]. The EPCs themselves do not possess any ontological information, i. e., the steps (or functions in an EPC) merely have a label.

The function view corresponds to our functional perspective. Functions are defined as operations applied to objects for the purpose of supporting one or more goals [1]. Within the function view, functions can be nested by means of sub-functions. Functions can have execution semantics if the original specification are detailed enough, else an implementation step can be performed [1, p. 50]. The function view does not possess any ontological information.

The data view corresponds to the information perspective. The data view can be modelled by means of an Entity Relationship model. As a result, there is a formal definition but there is no ontological information.

The organisation view corresponds to the organisation and resource perspectives. With the organisation view, one can define the organisational structure of the enterprise. A meta-model is presented which lists the organisation unit and type, as well as the roles and their qualifications. Apart from the meta-model, no ontological information is provided.

For ARIS, there is modelling and execution software available from Software AG[4]. The functional perspective might require some programming skills.

3) CIMOSA: The CIMOSA approach has as aim to provide the industry with (1) an Enterprise Modelling Framework (EMF), which can accurately represent business operations, support their analysis and design, and lead to executable enterprise models; (2) an Integrating Infrastructure (IIS), used to support application and business integration as well as execution of the implementation model to control and monitor enterprise operations; and (3) a methodology to be used along the System Life Cycle (SLC) to assist in applying CIMOSA principles [2]. The CIMOSA approach defines 5 different languages which correspond to our perspectives. These are: *workflow language*, *functional language*, *information language*, *resource language*, and an *organisation language*.

The workflow language is specified by means of a set of rules, e. g., when a particular event occurs, do a particular action. Using these rules, also sequentiality and alternative paths can be encoded [2]. In [2], a mapping is presented to Petri nets giving execution semantics. Next to this work, in [13], the formalisation as well as a different graphical notation is presented. The workflow language does not possess any ontological information.

The functional implementation can be specified using Pascal-like procedural statements [2]. As a result, the execution semantics in CIMOSA are clear but it does not possess any ontological information.

For the information language, one can use an extended entity-relationship (ER) model [2] to express the information and data objects which are present within the enterprise. As a result of using ER models, similar to the functional language, the semantics are clear but there is no ontological information, e. g., terms used in the ER model are not necessarily shared amongst organisations.

The resource language allows one to define the resources, their capabilities, and the set of functional operations that a resource understands and can execute (for instance for a machine). Within [2], the formalisation of the resources is presented by means of adding capabilities and function operations to a resource. However, the contents of these capabilities and functional requirements are not presented. As a result, in their simplest form, they are labels which do not possess any ontological information.

Finally, the organisation language entails specifying the organisation units (OU) and organisation cells (OC) [2]. On both, one can define responsibilities and authorities. Furthermore, an OC is an aggregation of OUs. The organisation language does not possess any ontological information, nor is there a complete formalisation.

[3]http://www.archimatetool.com/

[4]http://www.softwareag.com/corporate/products/az/aris/default.asp

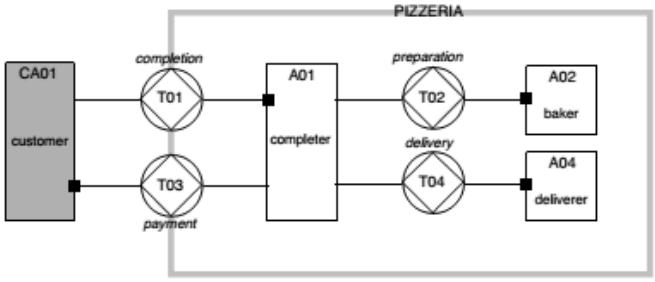

Figure 3. Part of a construction model of a pizzeria [10].

According to [14], ENTERPRISE PROCESS CENTER by INTERFACING TECHNOLOGIES[5] has a CIMOSA flavour. Using this product, running software can be deduced. For full use of the functional perspective, programming skills are required.

4) DEMO: One of the goals of the DEMO approach is to understand the essence of the construction and operation of complete systems; more specifically, of enterprises [10, p. 10]. This means that the approach wants to abstract from IT related aspects and purely focuses on how the enterprise works. Within the DEMO approach, an enterprise is viewed from 4 related models: *construction model*, *process model*, *state model*, and the *action model*.

The construction model corresponds to the resource perspective and the organisation perspective. Within the construction model, the emphasis is on the interaction (or transaction) of actor roles within the organisation. In Fig. 3, we have taken an excerpt of the construction model for a pizzeria [10]. In Fig. 3, *customer* and *completer* are two actor roles. The elements *T01 – T04* are transactions. Finally, a line *without* a black box indicates the initiator, and the line *with* the black box indicates the executor, e. g., the *payment* is initiated by the pizzeria and executed by the customer. With respect to the payment, it is irrelevant whether this happens electronically or in cash. As one can see, the names used are simply labels. At the same time, every transaction corresponds to a fixed set of internal steps, i. e., *request*, *promise*, *execute*, *state*, and *accept*. Apart from the correspondence of a transaction with the internal steps, no ontological information can be deduced, i.e., the label of a transaction is in essence free text.

The process model corresponds to the workflow perspective and to the information perspective. In the process model, links between transactions can be formulated. As mentioned, these transactions follow a predetermined sequence of steps. Between steps in different transactions, causal links can be added. Next to the causal links, conditional links can be formulated. These conditional links can be used to formulate that a part of the process model has to wait until some other part of the process model has been executed. The constructs are stated within [10]. However, the exact execution semantics are not entirely clear, especially since the causal / conditional links can have a weight to allow for multiple instantiations. See for instance [15] where a transformation is presented into Petri nets, but the case for multiple instances is left up to interpretation. Furthermore, see [16], where a tool is presented to execute DEMO models. However, in this work, the models seem to possess a certain structure where from the construction model the order of execution is clear. Furthermore, [16] states that "[..] This strict interpretation may be subject of discussion because in DEMO modeling practise sometimes deviations are found [..]" [16, p. 90]. Finally, in [16], the axioms underlying the theory are the starting point and not the modelling method. By incorporating basic patterns of interactions with a transactional nature, DEMO defines a basic set of ontological characteristics of enterprise descriptions on a high level of abstraction. These concepts can particularly be incorporated into a practically applicable ontology-based Enterprise Modelling approach. However, the set of ontological statements incorporated in DEMO does not reach a degree of domain-specificity as it would be suggested by an ontology of common-sense knowledge on enterprises.

The state model specifies, amongst others, the state space of the object classes, result types, and the coexistence rules [10]. It corresponds to the information perspective. The state model consists of an Object Fact Diagram, which is based on the World Ontology Specification Language (WOSL) [10], and an Object Property List (OPL). The OPL consists of a property type, e. g., date of birth, an object class on which the property type is defined, e. g., person. Finally, there is a scale indicating the domain from which a property type takes its values, e. g., Julian Date. OPLs can also be deduced from other OPLs, e. g., the age of a person can be deduced from her date of birth and the current date (i. e., today). The state model does not possess any ontological information, but is formally defined.

The action model specifies the action rules which serve as guidelines for the actors [10]. The actor has the option to deviate from the action rules. The action model is within the functional perspective and it is close to the syntax of a programming language, i. e., given a certain event, a condition is evaluated and, based on this evaluation, an action is performed. Within [16], an instruction set is presented for the action rules but neither a claim for completeness nor correctness is made [16, p. 222]. As a result, we do not consider this formalised with regard to execution semantics. Finally, it possesses no ontological information.

For the modelling environment and execution environment, there is a company FORMETIS[6] which has taken the DEMO approach and built a tool around it based on the work of [16].

The action model is with respect to syntax close to a programming language. As a result, a programming expert is necessary.

5) MEMO: The MEMO approach is presented in [4]. The aim of MEMO is, amongst others, to present a particular method for multi-perspective enterprise modelling. To this end, a number of requirements on the enterprise model are postulated, e. g., an approach to enterprise modelling should

[5]http://www.interfacing.com/

[6]http://www.formetis.nl/

offer semantically integrated model perspectives and should be supplemented by corresponding modelling tools that ensure the semantic integration. MEMO consists of a framework of views (technically called perspectives but this term might lead to confusion in our paper) on an enterprise, offers domain-specific modelling languages, and proposes accompanying methods and tools. The views entail the *information system*, the *organisation*, and the *strategy*. Next, for each of these views, there are various aspects, i. e., *resource*, *structure*, *process*, and *goal*.

The process aspect [17] corresponds to our workflow perspective. Various constructs are listed which can be used to create the workflow, e. g., sequence and exclusive choice. Next to presenting the constructs, also the semantics of the constructs are presented. As a result, the process aspect has execution semantics. Next to these constructs, MEMO also allows aggregate processes in order to decompose a process into smaller steps. MEMO offers a fundamental ontological distinction between automatic, semi-automatic, and manual processes. On top of this, processes do not possess any ontological information, i. e., they have just a label with no further meaning.

The functional perspective is not present within MEMO.

The object modelling aspect [18] corresponds to the information perspective. It allows one to model the objects present within the enterprise. For this, the MEMO meta-modelling language MML [19] is part of the MEMO language family, which allows to create meta-models of objects. Within object models, there is no ontological information. They are formally defined by means of the meta-models and their meta-meta-model.

In [20], the meta-model for modelling resources within MEMO is presented. Within the meta-model, resources can be, for instance, humans or machines. Furthermore, human resources can have various skills, e. g., a diploma issued by a certification authority, which can be regarded as a simple incorporation of ontological relationships into the semantics of the language constructs. The resource perspective is formalised by means of the meta-model.

The organisation perspective is presented in [21]. Within the organisation perspective, there are the organisation itself, organisational units, and positions to name a few. The organisation perspective is formalised by means of a meta-model.

For the MEMO approach, the MEMOCENTERNG modelling environment [22] provides tooling support for all of the introduced perspectives. However, there is no environment for the execution of MEMO models. The MEMO approach does not require the programming expert.

6) MERODE: The MERODE approach is presented in [23]. One of the goals of the approach is to equip modellers with a set of sound basic principles and a deep understanding of enterprise modelling techniques [23]. Next to this, within [23], transformations are presented to transform the model into code. The MERODE approach consists of *existence-dependency graphs*, *object-event tables*, and *finite-state machines*.

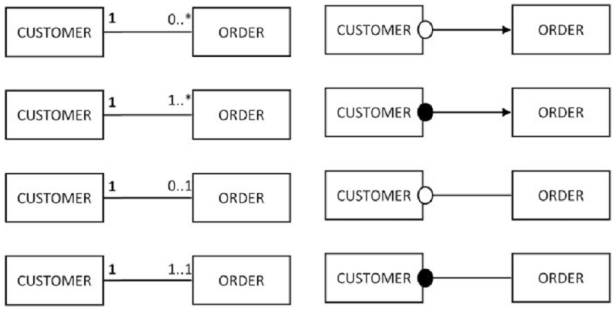

Figure 4. Duality between UML class diagrams (left) and MERODE's existence-dependency graph (right). Figure taken from [23].

The existence-dependency graph can be used to model the data model and corresponds to our information perspective. The notation used within the existence-dependency graph is related to UML class diagrams. The various multiplicities are graphically differently encoded, e. g., in Fig. 4, we have depicted how some of the relations in an UML class diagram can be transformed to the existence-dependency graph. For all other relations in the UML class diagram also transformations are presented. There is a formal definition for the existence-dependence graph. However, there is no ontological information.

The object event table is designed to model the participation of object types to business event types. Within the object event table, one can indicate if an object is created, modified, or ended. In a sense, this can be mapped onto the functional perspective as the object event table indicates the input / output of events. A formalisation of the object event table is presented. It does not possess any ontological information.

The finite-state machines in MERODE are used to encode the behavioural aspects. It corresponds to our workflow perspective. Using a finite-state machine, one can encode the sequences of business events in which a business object type participates. Finite-state machines have a formal definition (in [23] it is given in terms of regular languages). They do not have any ontological information.

On top of the aforementioned, MERODE allows for the addition of attributes and constraints.

Both the resource and organisation perspective are not present in MERODE.

There is tool support for MERODE[7] for both modelling and generating the code. The constraints are on the programming level.

7) UEML + UEMO: The Unified Enterprise Modelling Language (UEML) approach has been presented as a common user language [14]. Within UEML, the following views are listed: *function / process*, *information / object*, *resource / agent*, and *organisation / decision*. Next to UEML, also the Unified Enterprise Modelling Ontology (UEMO) exists [24]. One of the goals of UEMO is to develop precise semantic definitions of a wide variety of enterprise modelling languages and to use

[7] http://merode.econ.kuleuven.be/

the semantic definitions to facilitate integrated use of models expressed in those languages [24].

The views within UEML correspond to the perspectives presented earlier, i. e., the function / process view corresponds to the functional and workflow perspectives. The information / object view corresponds to the information perspective, the resource / agent view corresponds to the resource perspective. Finally, the organisation / decision view corresponds to the organisation perspective. In all presented perspectives, a formalisation is presented but on a very high level of abstraction, e. g., within the process view, a notion of *behaviour* is introduced which specifies the workflow without diving deeper into what is possible [14]. In [24], a formalisation is presented of the process by defining a notion of states and transformations between both. For the function view, [24] provides formalisation by means of defining a notion of transforming the input of an activity into its output. Resources are not formalised in [24], i. e., "[..] we have left resources undefined for now [..]" [24, p. 72]. No formalisation is presented for the organisation in [24].

Next to [14], there is also [25]. Within [25], similar to [24], the goal is to support integrated use of enterprise and IS models expressed using different languages. To this end, similar terminology is presented as in [24], e. g., *MutableProperty*. There is no subdivision presented into different perspectives nor any formalisation of the introduced terminology.

For UEML, modelling tools are listed in [25]. Finally, the approach does not require a programming expert since the focus primarily lies on modelling languages.

8) Enterprise Ontology: The enterprise ontology in [8] describes and defines a large set of terms relevant to business enterprises. The goal of the approach is to provide a shared understanding of the relevant aspects of a business enterprise [8]. The enterprise ontology is divided into the following parts: *activity*, *organisation*, *strategy*, *marketing*, and *time*. In this paper, we primarily focus on the activity and organisation parts as these correspond to our chosen perspectives.

Within the activity part, there is a process specification to denote the repeated execution of an activity specification with an intended purpose [8]. This can be seen as the *workflow perspective* (note that within the activity specification already causality between subactivities can be encoded). Next to this, the *activity specification* can be seen as the *functional perspective*. The process specification does not possess any execution semantics since the activity specification does not necessarily have any execution semantics, i. e., "[..] An ACTIVITY SPECIFICATION need not be EXECUTABLE; possible reasons are: [..] it is underspecified and / or ambiguous [..]" [8, p. 47-48]. Due to this, there is possibly no ontological information.

Next to the activity and process specifications, the activity part also consists of resources and capabilities, as well as the allocation of resources to activities. Within the organisation part, there are legal entities and organisational units. The organisational units may contain persons, resources, and other organisational units. As such, it corresponds to the organisation perspective. Both perspectives provide a description of what it *is* but not what it *does*. As such, it does not provide any ontological information.

The information perspective is not present within the enterprise ontology.

Within [8], various references are made to an editor at the KSL lab at Stanford. However, this page is no longer available. There is a page related to the KSL lab[8] however the most recent ontology there is from the mid 90s and most links are no longer working.

The accessibility is not applicable since no modelling approach is presented, i. e., it is a collection of definitions.

9) TOVE: The TOronto Virtual Enterprise (TOVE) has, amongst others, as aim to create a shared representation (ontology) of the enterprise, and to automatically deduce the answer to many "common sense" questions about the enterprise [9]. Within TOVE, there are the following parts [26]: *activity, causality, time* [27], *resources* [28], *cost*, *quality*, *organisation structure* [29], *product* [30], and *agility*. Of these parts, we focus on the activity, resources, organisation, and products.

The activity part corresponds to the workflow and functional perspectives. Both are an extension of the situation calculus [27], [31]. As a result, there is execution semantics. Next to this, it is possible to reason over possible execution sequences. However, the actions themselves do not possess semantics, i. e., they are just labels. As a result, there is no ontological information which is shared amongst enterprises.

The resource part corresponds with the resource perspective [28]. One important aspect mentioned is that being a resource is not an innate property of an object but it is a property that is derived from the role an object plays [28]. Within [28], various questions are raised which the TOVE model should be able to answer, e. g., can the resource be divided, what is the stock level at time t, or what are the subparts of a resource. For the questions, reasoning capabilities are introduced in First-Order Logic and Prolog. Most of the ontological information still needs to be encoded within the model, e. g., whether something is a division of a resource.

The organisation structure corresponds with the organisation perspective [29]. In [29], the organisation ontology is very similar to the work of [8], i. e., mainly definitions of concepts are presented possibly with some formalisation. No ontological information is present.

The product part corresponds to the information perspective [30]. This ontology bears some resemblance with the resource perspective, e. g., where a resource can consist of subparts, a product can consist of components. Within [30], also units of measurements, e. g., kilograms, and conversion between them is presented, e. g., from pound to kilogram. The product part is formally defined and, as with the resource part, most ontological information still needs to be encoded.

For the TOVE model, no specific editor could be found. At the same time, the models lack graphics and only consist of formulae in Prolog and First-Order Logic. As a result, a Prolog editor can be used to model and execute the model.

[8] http://www.sigchi.org/chi96/proceedings/papers/Rice/jpr_ht75.htm

Table I
APPROACHES UNDER CONSIDERATION AND THEIR SCORES ON OUR REQUIREMENTS.

Approach	Enterprise aspects															Tool supp.		Access.
	Workflow			Function			Information			Resource			Organisation					
	Name	F	O	Name	F	O	Name	F	O	Name	F	O	Name	F	O	M	E	P
ArchiMate	PM.	±	−	N/A			BO.	+	−	BA.	+	−	BA.	+	−	+	−	
ARIS	Control	+	−	Function	+	−	Data	+	−	Control/ Org.	+	−	Org.	+	−	+	+	✗
CIMOSA	Workflw./ Process	+	−	Function	+	−	Informa- tion	+	−	Resource	+	−	Org.	±	−	±	±	✗
DEMO	PM.	±	−	AM.	−	−	PM./SM.	+	−	CM.	+	−	CM.	+	−	+	+	✗
MEMO	Control Flow	+	−	N/A			Obj. Mod. & MML	+	−	Resource	+	−	Org.	+	−	+	−	
MERODE	FSM	+	−	OET	+	−	EDG	+	−	N/A			N/A			+	+	✗
UEML,UEMO	Process	+	−	Function	+	−	Enterprise Object	+	−	Resource	−	−	Org.	−	−	+	−	
The Enterprise Ontology	Activity/ Proc. spec.	−	−	Activity	−	−	N/A			Resource	+	−	Resource/ Org.	+	−	−	−	N/A
TOVE	Activity	+	−	Activity	−	−	Product	+	±	Resource	+	±	Org.	+	±	+	+	✗

Finally, the modelling level is on the programming level since the formulae are in essence programming statements.

C. The scores

The scores of the nine approaches on our requirements are summarised in Table I. In the first column, we list the approaches. Thereafter, the scores on our requirement for being able to model the enterprise are listed. This requirement is subdivided into the perspectives. For each of these perspectives, we list its name for the respective approach as well as whether it has formal (execution) semantics (indicated by an *F*) and ontological information (indicated by an *O*). After the requirement for modelling the enterprise, we have the requirement for tool support. There are two columns for this requirement: *M* for modelling, and *E* for execution. Finally, we have the requirement for the accessibility. We use *P* for programming expert.

In the cells for the perspectives, we put $+$ if it has formal (execution) semantics / ontological information. In case we put $-$, this means that it does not have formal (execution) semantics / ontological information. Finally, if we put $\pm$ it means there is some formal (execution) semantics, but this is not sufficient to automatically generate the ERP software. For the TOVE approach, we have $\pm$ in the ontological information columns since one still needs to encode the common-sense of one's enterprise. For tool support, we have $+$ if there is a modelling / execution tool. We use $-$ if there is no modelling / execution tool. Finally, for CIMOSA, we set $\pm$ since there only is a tool with a CIMOSA flavour. For the accessibility, we use ✗ in case a programming expert is required. If something is not applicable, we have denoted this with N/A, e. g., there is no information perspective in the Enterprise Ontology.

Within Table I, we use PM. for *process model*, BO. for *business object*, BA. for *business actor*, Org. for *organisation*, AM. for *action model*, SM. for *state model*, CM. for *construction model*, Obj. Mod. for *object model*, FSM for *Finite-state machine*, OET for *Object-event table*, EDG for *Existence-dependency graph*, and Proc. Spec. for *process specification*.

IV. TAKE HOME POINTS

As shown in Table I, none of the existing approaches can fulfil all requirements without a programming step/ someone familiar with programming. At the same time, the various approaches have specific qualities which could be valuable for creating an Enterprise Modelling language for the automated generation of ERP software supporting an enterprise. In this section, we review the various approaches and present the take-home points (THP for short) for anyone interested in creating/extending an Enterprise Modelling language for the automated generation of ERP software.

A. CIMOSA

Within the CIMOSA approach, various aspects of an enterprise are discussed in quite some detail, e. g., capability sets for resources but also the causality of activities by means of Allen's temporal logic [32]. Therefore, our take-home points from the CIMOSA language are:

***THP** 1:* Use the details, e. g., capabilities of resources, identified by CIMOSA as a starting point for defining relevant details for multiple organisations.

***THP** 2:* Use Allen's temporal logic to (more precisely) describe the causal relationship between activities.

B. DEMO

In the DEMO methodology, the idea is to abstract from IT related information, i. e., the focus is on *what* the organisation does instead of *how* it is implemented. By abstracting from IT and focussing on the business, the methodology becomes accessible to business experts. Therefore, our take-home point from the DEMO methodology is:

***THP** 3:* Make the enterprise modelling language independent from the IT domain.

C. MEMO

Within MEMO, various requirements are listed for Multi-perspective Enterprise Modelling. For instance, requirement HR3 in [4] states: "Enterprise models should include concepts that can be mapped to implementation-level concepts according to clear transformation rules.". This requirement fits very well with the ideas of generating ERP software. Therefore, as take-home point based on MEMO:

***THP** 4:* For each of the requirements in MEMO, verify if they also apply to the enterprise modelling approach. If so, verify if the ideas behind the MEMO approach can be transferred. If not, verify that nothing is missed by not incorporating a requirement from MEMO.

D. UEML + UEMO

One of the main goals of the UEML and UEMO approaches is to map the concepts of various enterprise modelling languages onto each other. In [24], a list is presented of languages for which (partly) mappings exist to UEMO, e.g., ARIS, BPMN, Petri nets, and UML. By mapping an enterprise modelling language onto UEML and UEMO, it becomes possible to (a) check for completeness of the language relative to the conceptual framework suggested by UEML and UEMO, and (b) to present modelling constructs in another language with which the modeller is possibly more familiar.

***THP** 5:* Create a mapping of the enterprise modelling language to and from UEML and UEMO.

E. Enterprise Ontology

The main advantage of the Enterprise Ontology is the fact that every term has a definition. One of the important aspects of generating code from a model is to have unambiguous terms. The Enterprise Ontology contains a large number of terms / concepts within the enterprise domain. As a result, the Enterprise Ontology is a promising starting point for the concepts present within an enterprise.

***THP** 6:* Precisely define a concept once in the language and use it consistently everywhere even between enterprises.

***THP** 7:* Take the concepts identified by the Enterprise Ontology, besides details from the CIMOSA approach (see *THP 1*), as a starting point for the concepts within the enterprise modelling language.

F. TOVE

The reasoning capabilities within TOVE provide a means to reason about the enterprise. This allows a system to possibly make suggestions in case invalid or untypical states occur. For instance, take the earlier example of a meeting room with a capacity of 8 seats, which "naturally" cannot hold a meeting of 10 people. In this case, the system should be able to suggest an alternative room with enough space, and should prevent to allocate the wrong room in the first place.

***THP** 8:* Provide enough information to the model such that common-sense suggestions can be made when invalid or critical system states are reached.

V. Related work

Various work has been conducted on comparing Enterprise Modelling languages and Enterprise Ontologies. None of the comparisons have the goal to automatically generate ERP software. At the same time, various comparisons touch upon the aspects we are interested in. In this section, we list some of these works.

In [33], a comparison is presented between different Enterprise Modelling approaches, e. g., ARIS and CIMOSA. The comparison consists of the perspectives similar to ours, e. g., organisation, and resources. Their comparison coincides with ours on the overlapping aspects. However, their focus is on the mapping between perspectives, e. g., the data view within ARIS coincides with the information view within CIMOSA.

A comparison between the concepts in, amongst others, ArchiMate, DEMO, and MEMO is presented in [34]. This comparison is one level deeper than our comparison in the sense that the elements in the perspectives are compared, e. g., instead of the *organisation* perspective, they list *organisational units, Superior, Composite Actor*, etc. This comparison can be used to detail the OEM with relevant aspects of an enterprise.

The work in [4] compares the MEMO approach with related work. In this comparison, the focus is on domain-specific modelling languages and their meta-models. As such, the focus is different from ours. At the same time, this work shows the current state of the approaches, e. g., whether there are tools and if the approach is still under active research.

In [35], a comparison is made between Enterprise Modelling approaches based on their formal semantics. The focus in the paper is on the behavioural aspect, while we focus on all aspects of an Enterprise Modelling approach. Furthermore, our focus is on formal execution semantics in the behaviour perspective.

In [36], a classification is made between Enterprise Ontologies. To this end, a matrix is presented to classify if an ontology is tightly / loosely specified and applicable in multiple domains / a single domain. An Enterprise Ontology used for generating software to support an enterprise should be both tightly specified as well as applicable in multiple domains. The work by [36] shows that all ontologies considered in their paper do not fall in this category.

The work in [37] lists requirements for incorporating ontologies in Enterprise Modelling approaches. Furthermore, the work shows that none of the ontologies adhere to all requirements. It would be interesting to use these requirements for defining the ontology of an Enterprise Modelling approach.

VI. Conclusion and future work

In the paper, we have presented a systematic evaluation of Enterprise Modelling languages with respect to their application for automatic generation of ERP software which can support an enterprise. Next to generating this ERP software, inspired by the Dutch ERP vendor AFAS, we have evaluated well-known Enterprise Ontologies for their applicability to encode the common-sense of an enterprise. Neither the Enterprise Modelling languages nor the Enterprise Ontologies

are directly applicable. At the same time, they possess aspects which are valuable to an Enterprise Modelling language tailored towards the automated generation of ERP software capable of supporting the enterprise. Among these are formal definitions for each concept, and reasoning capabilities often present within ontologies. To this end, we have presented take-home points to emphasise these aspects.

The OEM is still under development. This paper is a first step in comparing the many ideas present in Enterprise Modelling approaches and Enterprise Ontologies. Reviewing these approaches already showed a large body of (related) work which could also prove valuable for generating ERP software from an Enterprise Model to support the enterprise.

Our goal with the OEM is not a one-size-fits-all approach to enterprise modelling (such an approach might not even exist). Our goal is to have an Enterprise Modelling approach rich enough to capture all peculiarities of an enterprise which are of importance to generate ERP software capable of supporting the enterprise. We hope this research gives some new insights into existing approaches as well as new opportunities to advance the field of Enterprise Modelling and Enterprise Ontologies.

Acknowledgment This research was supported by the NWO AMUSE project (628.006.001): a collaboration between Vrije Universiteit Amsterdam, Utrecht University, and AFAS Software in the Netherlands.

REFERENCES

[1] A.-W. Scheer, *Aris–Business Process Modeling*, 2nd ed. Springer-Verlag New York, Inc., 1999.

[2] F. Vernadat, *Handbook on Architectures of Information Systems*. Springer, 2006, ch. The CIMOSA Languages, pp. 251–272.

[3] J. Gulden, "Methodical support for model-driven software engineering with enterprise models," Ph.D. dissertation, University of Duisburg-Essen, Berlin, 2013.

[4] U. Frank, "Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges," *Software and System Modeling*, vol. 13, no. 3, pp. 941–962, 2014.

[5] J. Gordijn, "Value-based requirements engineering: Exploring innovative e-commerce-commerce ideas," Ph.D. dissertation, Vrije Universiteit Amsterdam, 2002.

[6] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *Eclipse Modeling Framework*, 2nd ed. Addison Wesley, 2009.

[7] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 38, no. 2/3, pp. 454–470, 1999.

[8] M. Uschold, M. King, S. Moralee, and Y. Zorgios, "The enterprise ontology," *Knowl. Eng. Rev.*, vol. 13, no. 1, pp. 31–89, 1998.

[9] M. S. Fox, *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Springer, 1992, ch. The TOVE project towards a common-sense model of the enterprise, pp. 25–34.

[10] J. L. G. Dietz, *Enterprise ontology - theory and methodology*. Springer, 2006.

[11] Open Group, *Archimate 2.1 Specification*. Van Haren Publishing, 2013.

[12] J. Mendling, "EPCs," in *Modern Business Process Automation - YAWL and its Support Environment*. Springer, 2010, pp. 369–383.

[13] G. Berio, "A formal approach to CIMOSA concepts semantics." LGIPM 98-01, ENIM/University of Metz, Ile du Saulcy, 57045 Metz, France, Tech. Rep., 1998.

[14] F. Vernadat, "UEML: Towards a unified enterprise modelling language," *International Journal of Production Research*, vol. 40, no. 17, pp. 4309–4321, 2002.

[15] Z. Vejrazkova and A. Meshkat, "Translating DEMO models into petri net," in *Enterprise and Organizational Modeling and Simulation*, ser. LNBIP, vol. 153. Springer, 2013, pp. 57–73.

[16] S. J. H. van Kervel, "Ontology driven enterprise information systems engineering," Ph.D. dissertation, TU Delft, 2012.

[17] U. Frank, "Memo organisation modelling language (2): Focus on business processes," ICB-Research Report, No. 49, Tech. Rep., 2011.

[18] ——, *Arbeitsberichte des Instituts für Wirtschafts- und Verwaltungsinformatik*. Koblenz-Landau: Universität Koblenz-Landau, 1998, ch. The MEMO Object Modelling Language (MEMO-OML).

[19] ——, "The MEMO Meta Modelling Language (MML) and Language Architecture. 2nd Edition," Institut für Informatik und Wirtschaftsinformatik (ICB), Universität Duisburg-Essen, Campus Essen, Essen, Tech. Rep. 43, 2011.

[20] J. S. Jung, "Entwurf einer sprache für die modellierung von ressourcen im kontext der geschäftsprozessmodellierung," Ph.D. dissertation, Universitt Duisburg-Essen, 2007.

[21] U. Frank, "Memo organisation modelling language (1): Focus on organisational structure," ICB-Research Report, No. 48, Tech. Rep., 2011.

[22] J. Gulden and U. Frank, "MEMOCenterNG – a full-featured modeling environment for organization-modeling and model-driven software development," in *Proceedings of the CAiSE Forum 2010*, ser. CEUR Workshop Proceedings, vol. 592. CEUR, 2010, pp. 76–83.

[23] M. Snoeck, *Enterprise Information Systems Engineering - The MERODE Approach*, ser. The Enterprise Engineering Series. Springer, 2014.

[24] A. L. Opdahl, G. Berio, M. Harzallah, and R. Matulevičius, "An ontology for enterprise and information systems modelling," *Appl. Ontol.*, vol. 7, no. 1, pp. 49–92, 2012.

[25] V. Anaya, G. Berio, M. Harzallah, P. Heymans, R. Matuleviius, A. L. Opdahl, H. Panetto, and M. J. Verdecho, "The unified enterprise modelling languageoverview and further work," *Computers in Industry*, vol. 61, no. 2, pp. 99 – 111, 2010, integration and Information in Networked Enterprises.

[26] M. S. Fox and M. Grüninger, "Enterprise modeling," *AI Magazine*, vol. 19, no. 3, pp. 109–121, 1998.

[27] M. Grüninger and J. A. Pinto, "A theory of complex actions for enterprise modelling," AAAI Technical Report SS-95-07: Extending Theories of Action: Formal Theory and Practical Applications, Tech. Rep., 1995.

[28] F. G. Fadel, M. S. Fox, and M. Gruninger, "A generic enterprise resource ontology," in *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1994, pp. 117–128.

[29] M. S. Fox, M. Barbuceanu, and M. Gruninger, "An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour," in *Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1995, pp. 71–81.

[30] L. Jinxin, M. S. Fox, and T. Bilgic, "A requirement ontology for engineering design," *Concurrent Engineering*, vol. 4, no. 3, pp. 279–291, 1996.

[31] R. Reiter, "The frame problem in situation the calculus: A simple solution (sometimes) and a completeness result for goal regression," in *Artificial Intelligence and Mathematical Theory of Computation*. Academic Press Professional, Inc., 1991, pp. 359–380.

[32] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[33] K. Kosanke, *Information Infrastructure Systems for Manufacturing*. Springer US, 1997, ch. Comparison of Enterprise Modelling Methodologies, pp. 115–127.

[34] A. Bock, M. Kaczmarek, S. Overbeek, and M. Heß, "A comparative analysis of selected enterprise modeling approaches," in *The Practice of Enterprise Modeling*, ser. LNBIP, vol. 197. Springer, 2014, pp. 148–163.

[35] D. Bork and H. Fill, "Formal aspects of enterprise modeling methods: A comparison framework," in *47th Hawaii International Conference on System Sciences*. IEEE Computer Society, 2014, pp. 3400–3409.

[36] D. E. O'Leary, "Enterprise ontologies: Review and an activity theory approach," *International Journal of Accounting Information Systems*, vol. 11, no. 4, pp. 336 – 352, 2010.

[37] M. Kaczmarek, "Ontologies in the realm of enterprise modeling - A reality check," in *Formal Ontologies Meet Industry*, ser. LNBIP, vol. 225. Springer, 2015, pp. 39–50.