REAlist—A Tool Demo

Bernhard Wally[1], Alexandra Mazak[1], Bernhard Kratzwald[1], Christian Huemer[1], Peter Regatschnig[2], and Dieter Mayrhofer[1]

[1] Vienna University of Technology, Institute of Software Technology and Interactive Systems, Business Informatics Group, Favoritenstraße 9–11, 1040 Vienna, Austria,
{wally, mazak, kratzwald, huemer, mayrhofer}@big.tuwien.ac.at,
http://www.big.tuwien.ac.at/

[2] eventus Marketingservice GmbH,
Gumpendorferstraße 21/Top 12, 1060 Vienna, Austria,
peter.regatschnig@eventus.at,
http://www.eventus.at/

Abstract. REAlist is a prototypical web-based ERP system built on top of a generic REA core system. It is implemented as a multi-tenant aware software-as-a-service. We intend to give a live demo of REAlist and show how various REA concepts are used to model the required ERP artefacts.

Keywords: Tool Demo, REA, ERP System Implementation, Model-Driven Engineering, Software-as-a-Service

1 Introduction

With *REAlist* [6], a prototypical Enterprise-Resource-Planning (ERP) system was built that is based on the Resource-Event-Agent (REA) ontology [7] as its core meta-model. The developed solution thus consists of two software layers: (i) an implementation of the REA ontology and (ii) an ERP abstraction layer based thereon that allows driving a web based software-as-a-service. See Fig. 1 for a more complete visualization of the software stack.

The REA core implementation has been designed with genericity in mind, allowing to create and manipulate arbitrary business models at runtime—for that, a generic data model has been designed. This implementation concern follows a current trend in software engineering that is *model driven software engineering* [2,8,9,4]. In order to accelerate the definition of business models, reference modeling is applied [3], i.e. existing models can be used as a basis for further model refinement.

The ERP implementation follows the Handels-H (engl. "Trading H") approach [1], which visualizes (i) purchase processes as a vertical bar to the left, (ii) sales processes as a vertical bar to the right, and (iii) links both by a horizontal bar which is the warehouse. A set of key performance indicators (KPI) is implemented in order to provide condensed information gathered from quantifiable data of business events. Also, inter-organizational benchmark indicators can be calculated due to the multi-tenant nature of the developed solution.

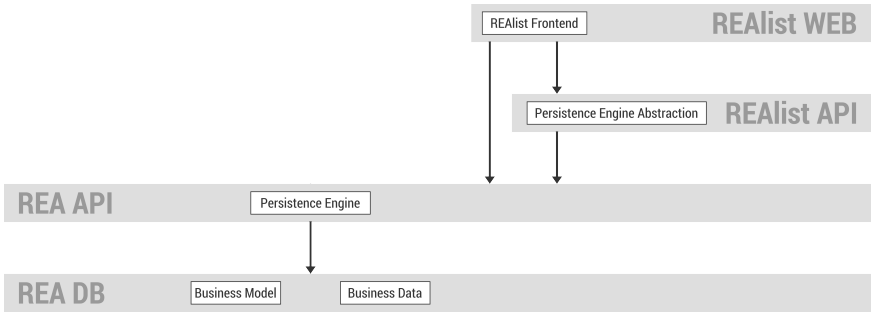

Fig. 1. Layers of the software architecture implemented in the REAlist prototype: the two bottom layers (REA DB and REA API) depict the generic REA core implementation, comprising a database implementation (DB) and an application programming interface (API). Based thereon an ERP application is built consisting of the REAlist API (which encapsulates ERP functionality on top of REA concepts) and REAlist WEB, a web frontend (and the visible part of the tool we are presenting here).

2 Tool Demo

The REAlist tool features two distinct user interfaces: (i) a "REA browser" that enables the manipulation of raw REA entities (on both the type and instance layer), and (ii) an ERP frontend that abstracts REA details where necessary and provides a condensed interface for purchasing, storing and selling articles. The REA browser is not explained here any further with the exception of the business model editor (see below).

2.1 REA Business Model Editor

Fig. 2 depicts the REAlist business model editor (part of the raw REA browser), where specific entities of a business are defined following a MOF[1] and type object [5] approach: REA entities (M2) are instantiated on a M1 "type" layer, after which operational instances of those types can be instantiated on a M0 layer. In the business model editor only the M1 layer is regarded—the markings in Fig. 2 are explained as follows:

① depicts the top level agent types defined for REAlist,
② are the top level resource types,
③ reciprocity type `ORDER` (ordering products from a supplier for later selling) comprises increment (`ARTICLE_RECEIPT`) and decrement (`PAYMENT`) event types (realized on MOF layer M0 by commitment instances),
④ duality type `PURCHASE` displays which events are allowed for that duality, namely increment event types `ARTICLE_RECEIPT`, `PURCHASE_CASH_DISCOUNT`, and `PURCHASE_CREDIT` and decrement event types `PAYMENT` and (in case articles are sent back to the supplier) `PURCHASE_ARTICLE_RETURN`,

[1] See `http://www.omg.org/mof/`

⑤ shows which claims are relevant for the purchase business case, namely `INCOMING_INVOICE` and `PREPAYMENT` (e.g. if we have to pay before receiving the goods).

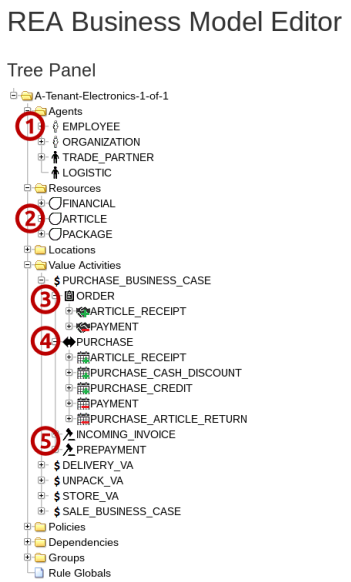

Fig. 2. The REA business model editor is implemented as a tree, where each REA entity (MOF layer M2) can be instantiated as a type entity on M1.

2.2 ERP Frontend

For demo purposes we present the main screens of the purchase business case and explain the underlying REA concepts that are abstracted therein. Fig. 3 shows the overview page of a product order from a supplier. The markings in Fig. 3 are explained below:

① the address of the supplier (information gathered from a `dependency` relation from the supplier agent to an address location point),
② the supplier number is an instance property of agents of type "supplier",
③ the internal agent responsible for the current order—the list is generated from the available agent instances of type "clerk",
④ a line item of the current order showing the amount of items ordered (combined view of the resource and the stockflow with which the resource is bound to the current commitment),
⑤ a scaled discount for line item 2—the scaled discount is defined in a policy that relates a supplier agent to a certain resource while providing minimum/maximum limits and the value of the discount in percent,

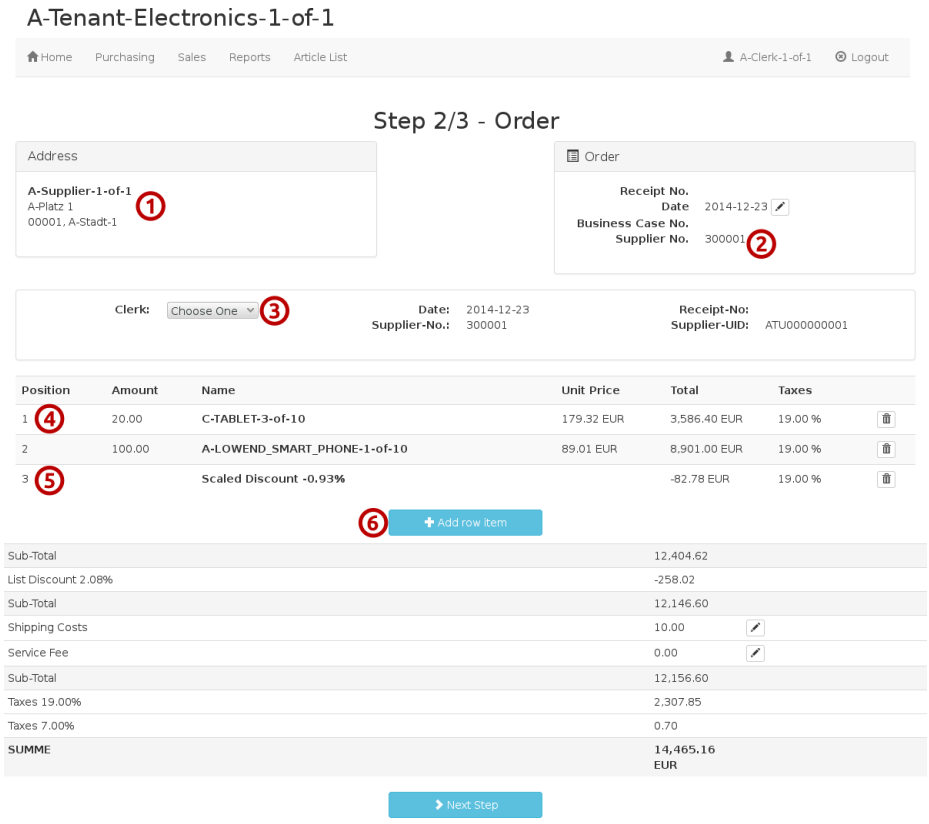

Fig. 3. ERP frontend for an order. The address of the supplier is retrieved from a corresponding dependency, shipping costs and service fees are defined in their respective policies, taxes are calculated based on the provided tax policies.

⑥ clicking on “Add row item” leads to the interface presented in Fig. 4.

Fig. 4 depicts the adding of a new line item to the list of products to order in the current order process:

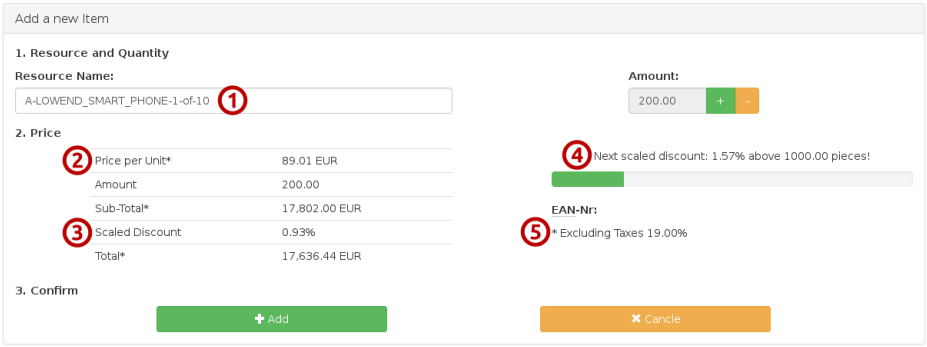

Fig. 4. ERP frontend for adding items to an order. The availability and base price of a product from a certain supplier is defined by corresponding policies, and so are the scaled discounts for the selected product.

① an auto-complete textfield supports the finding of products that are availalbe from the given supplier (a `LIST_PRICE` policy defines whether a certain product can be ordered from a supplier or not),
② based on the beforementioned policy, the net list price per unit is extracted,
③ a scaled discount of 0.93% is defined in a `PURCHASE_SCALED_DISCOUNT` policy when ordering between 100 and 1000 units,
④ an additional scaled discount policy of 1.57% is defined for orders exceeding 1000 units—the progress bar visualizes the fulfillment grade of the next scaled discount,
⑤ based on the address of the supplier (here: Germany) the corresponding tax value is retrieved from the respective tax policy.

Fig. 5 depicts the storing activity that takes place after the ordered products have arrived. This screen is only shown if the business activity `STORE_VA` is defined in the business model of the respective tenant. The markings in Fig, 5 have the following meaning:

① the first line item has been stored in Bin-1 of Shelf-1 in Warehouse-1,
② the second line item (received amount: 13 units) has been stored in two different bins of two different shelves in the same warehouse,
③ the third line item is about to be stored in Bin-3 of Shelf-2 in Warehouse-1—with a click on `Store` that intent is persisted, i.e. the corresponding dependency is created or an existing dependency is updated with the additional amount of items to be stored.

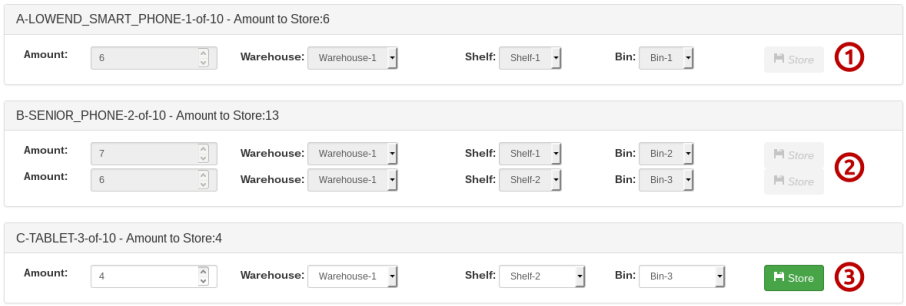

Fig. 5. ERP frontend for storing items of an order into a warehouse. The available warehouses, shelves and bins are defined by dependency entities.

3 Conclusion

With REAlist we have protoypically implemented (i) a generic REA core comprising a business model independend database and corresponding data access layer, as well as (ii) an ERP application which is built on top of that REA core. We have shown that it is feasible to implement business applications on top of REA and with our internal test data we have set up three different business branches with different requirements for mainly products and employees that can all be served from a single REAlist instance thanks to its multi-tenant support.

4 Acknowledgements

This work was supported as part of the BRIDGE program of the Austrian Research Promotion Agency (FFG) under grant number 841287—a joint research effort of Vienna University of Technology and eventus Marketingservice GmbH.

References

1. Becker, J., Schütte, R.: Handelsinformationssysteme. MI Wirtschaftsbuch, 2 edn. (2004)

2. Bézivin, J.: On the unification power of models. Software & Systems Modeling 4(2), 171–188 (May 2005), http://dx.doi.org/10.1007/s10270-005-0079-0

3. vom Brocke, J.: Design principles for reference modeling: reusing information models by means of aggregation, specialisation, instantiation, and analogy. Reference Modeling for Business Systems Analysis pp. 47–75 (2007)

4. Hofreiter, B., Huemer, C., Kappel, G., Mayrhofer, D., vom Brocke, J.: Inter-organizational reference models—may inter-organizational systems profit from reference modeling? In: Ardagna, C.A., Damiani, E., Maciaszek, L.A., Missikoff, M., Parkin, M. (eds.) Business System Management and Engineering, Lecture Notes

in Computer Science, vol. 7350, pp. 32–47. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-32439-0_3

5. Johnson, R., Woolf, B.: Type object. In: Martin, R.C., Riehle, D., Buschmann, F. (eds.) Pattern Languages of Program Design 3, chap. Type Object, pp. 47–65. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997), http://dl.acm.org/citation.cfm?id=273448.273453

6. Mayrhofer, D., Mazak, A., Wally, B., Huemer, C., Regatschnig, P.: REAlist: Towards a business model adapting multi-tenant ERP system in the cloud. In: 8th International Workshop on Value Modeling and Business Ontology (VMBO 2014) (March 2014)

7. McCarthy, W.E.: The REA accounting model: A generalized framework for accounting systems in a shared data environment. The Accounting Review 57(3), 554–578 (July 1982)

8. Selic, B.: MDA manifestations. The European Journal for the Informatics Professional IX(2), 12–16 (April 2008)

9. Stahl, T., Völter, M., Efftinge, S., Haase, A.: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt. verlag (2012)