Applied Ontology 13 (2018) 201–224
DOI 10.3233/AO-180198
IOS Press

REA^2: A unified formalisation of the Resource-Event-Agent ontology

Wim Laurier [a,b,c,*], Jesper Kiehn [a] and Simon Polovina [d]

[a] *University Saint Louis, Bruxelles, Boulevard du jardin Botanique 43, 1000 Bruxelles, Belgique*
E-mails: wim.laurier@usaintlouis.be, jkiehn@hotmail.com
[b] *Ghent University, Faculty of Business Administration, Department of Business Informatics and Operations Management, Tweekerkenstraat 2, 9000 Gent, Belgium*
[c] *inno.com Institute, Heiststeenweg 131, 2580 Beerzel, Belgium*
[d] *Conceptual Structures Research Group, Communication and Computing Research Centre, Department of Computing, Sheffield Hallam University, Sheffield S1 2NU, UK*
E-mail: S.Polovina@shu.ac.uk

Abstract. Through a proof of concept in SWI-Prolog, this paper demonstrates a business transaction model by which the trading partners can derive their own, personal perspective from shared data. The demonstration is an innovative formalisation of the Resource-Event-Agent (REA) ontology as it allows for switching viewpoints in real-time between one trading-partner's perspective and that of a trading-partner with an opposing view (i.e., customer or supplier), or a trading-partner independent perspective (e.g. trusted third-party). The business transaction model is achieved by uniting REA with the Open-EDI Business Transaction Ontology (OeBTO). The resulting unified formalisation of the REA ontology (REA^2) also highlights implications for the future development of a) enterprise information systems (EIS) in the cloud, b) social-media-based EIS, c) blockchain EIS, and d) EIS interoperability across business paradigms. The EIS interoperability such as between traditional EIS (which typically uses a trading-partner perspective), and EIS for the collaborative economy (which typically uses a trading-partner independent perspective) is particularly highlighted as it becomes much more transparent than previously.

Keywords: Resource, event, agent, REA, ontology, prolog, dependent view, independent view, Open-edi Business Transaction Ontology (OeBTO), collaborative economy, blockchain

Accepted by: Michael Gruninger

1. Introduction

Businesses collaborate with partners to create value and compete with those same partners to earn the larger part of the value they created together (Brandenburger and Nalebuff, 1996). Collaboration requires sharing information with trading-partners, whereas information sharing might hamper maximisation of earnings in a competitive environment (Letaifa, 2014). Meanwhile, third parties (such as governments, trusted third parties) request access to trade data so that they can process this information to reduce health and safety risks through food-traceability, e-customs, or in novel forms of business interactions such as blockchains (Laurier and Poels, 2012; Steiner and Baker, 2016; Tan et al., 2010). Traditional

*Corresponding author. E-mail: wim.laurier@usaintlouis.be.

1570-5838/18/$35.00 © 2018 – IOS Press and the authors.

This article is published online with Open Access and distributed under the terms of the Creative Commons Attribution Non-Commercial License (CC BY-NC 4.0).

enterprise resource planning (ERP) and accounting information systems (AIS) are limited in their capability to provide this third-party or value network perspective, due to their internal focus (Value Network Software (NRP), 2016).

To address these issues, business ontology needs to be in a form that is concise and simple enough to be implemented in the multitude of modelling and programming languages that reflect the technologies that the trading-partners or third parties may be using. At the same time the ontology has to permit an interoperable, simultaneous representation of the opposing perspectives of collaborating trading-partners as well as a neutral third-party perspective on business transactions and operations to support, for example, traceability (Value Flows vf vocabs, 2016).

The paper is structured as follows: Section 2 outlines the capability of the Resource-Event-Agent (REA) ontology for formalising both a trading-partner dependent and trading-partner independent (or neutral) perspective. Section 3 presents a unified REA-ontology called REA^2 that formalises both the dependent and independent dimensions in one view. Section 4 addresses the research methodology and related research. Section 5 introduces the meta-model that extends the traditional interpretation of the REA ontology. Subsequently, Section 6 introduces the semantic pattern formalised as a data-model. Sections 7 and 8 present an example that is used as the exemplar that acts as the proof of concept for REA^2. Section 9 discusses the limitations and implications of the proof of concept. This discussion is complemented with directions for future research, and concludes with the value that REA^2 brings to REA.

2. Resource-Event-Agent ontology

The Resource-Event-Agent (REA) is an ontology that has already been used to document both the perspective of a trading-partner and that of a third-party (Geerts and McCarthy, 2002; ISO/IEC, 2007). REA thus remains as the focal ontology. This paper argues however that the REA ontology has the capability of modelling a trading-partner dependent and independent (or neutral) perspective simultaneously if and only if the REA primitives are assigned an identifiable trading-partner.

Given REA's origin in accounting, REA's dependent view, which is also called the trading-partner view, is enterprise-centric. In one dimension, REA focusses on the semantics required for describing the perspective of a single organisation (McCarthy, 1979; 1982). In another dimension, REA's independent view, which is also called helicopter view, is supply-chain centric and focusses on the semantics for describing business from the perspective of a neutral third-party (Haugen, 2007). The validity of the independent view has been recognised by the International Organisation for Standardisation (ISO) as a standard for effective electronic data interchange (EDI) (ISO/IEC, 2007). In practice, the dependent view of REA has many implementations in numerous fields. It has been mapped to other enterprise-centric ontologies, and has been identified as a recurring pattern in the market-leading SAP ERP software (Gailly and Poels, 2009; Geerts, 2011; O'Leary, 2004). There is thus a need to bring an independent view to REA that than elegantly co-exist with these existing enterprise-centric implementations, bringing them into line with the ISO standard.

3. REA^2

To meet the above-identified need, this paper presents a unified REA-ontology called REA^2 that formalises both the dependent and independent views. Figure 1 visualises the intended application context

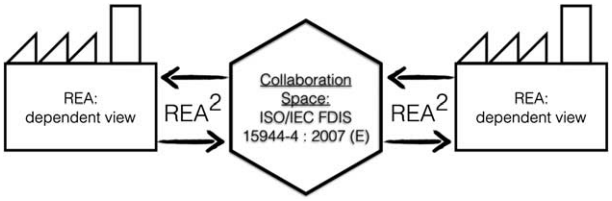

Fig. 1. REA^2 view integration: application context.

of the REA^2 formalisation, in which data available in ISO/IEC FDIS 15944-4 or a compatible view-independent data format supports interoperability between the information systems of trading-partner sharing a collaboration space. Such a *collaboration space* is defined by the ISO OeBTO standard as a business activity space in which an economic exchange of valued resources is viewed independently of the perspective of any trading-partner (ISO/IEC, 2007). ISO/IEC FDIS 15944-4 is one of the earlies and still an essential source for REA's independent view.

In the intended application context, the trading-partners would continue to use REA's dependent view (or a compatible view-dependent data format such as SAP (O'Leary, 2004), since data in an ISO/IEC FDIS 15944-4 compatible format would require no or a trivial transformation). REA^2 will be shown to allow for an automated transformation of view-dependent data into view-independent data and vice versa. This automated transformation will allow trading-partners from their dependent view perspective to share their data in the collaboration space, where the transformation allows them to reuse the data shared by their counterpart trading partners, which maintain their counterpart dependent view.

To avoid readability issues with the visualisations, the REA^2 formalisation has been decomposed in and presented as 3 separate models: 1) a meta-model (M2), 2) a data-model (M1), and 3) an example (M0) that acts as the exemplar that demonstrates the value of REA^2. To assist understanding, the instantiation semantics that relate these models to each other accord with the M2-M1-M0 semantics of the Object-Management Group's (OMG) Meta-Object Facility (MOF) (OMG, 2016).

The meta-model (M2) uses OntoUML stereotypes, to present a minimal set of REA constructs that allow for a simultaneous representation of the opposing perspectives of trading-partners – i.e., dependent view – and the viewpoint of an independent third-party – i.e., independent view. The generalisation semantics in the REA^2 meta-model will be used to derive the independent view from the dependent view. OntoUML is of interest as it has been developed to counter silo-based information architectures, which hamper answering critical questions that can only be solved by connecting information that is scattered over these silos (Guizzardi, 2014). These critical questions include the traceability, profitability and equitability questions that require a view-independent perspective. As this paper focusses on ontological aspects, all the models below abstract from context-specific aspects such as cardinalities that model business rules (e.g. the upper and lower boundaries for the number of economic-agents that can be related to an economic event). Consequently, no cardinalities will be shown on the diagrams in this paper. As the REA ontology addresses a pattern in conceptual modelling, it requires us to focus on the role and relationship semantics of the pattern, inherently abstracting from the kinds of things that can play the resource, event and agent roles. This abstraction and subsequent focus on roles is essential to the contribution of this paper.

Assisted by MOF, the data-model (M1) implements the meta-model to show a semantic pattern that addresses trading-partner dependent and independent semantics in a collaboration space as described earlier. This semantic pattern can in turn be used to map the dependent view to the independent view and vice versa. Extensions of, and variations on, this archetypal data-model can be found in model-driven

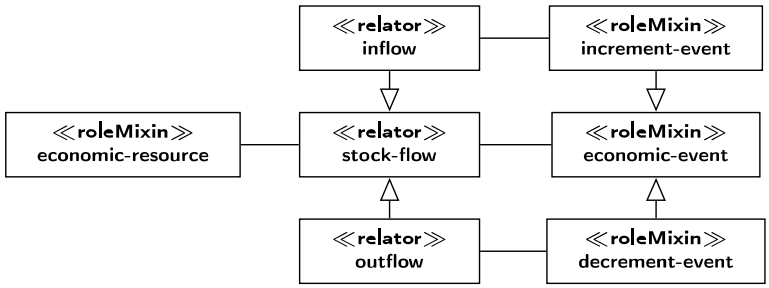

Fig. 2. The REA^2 meta-model for stock-flows.

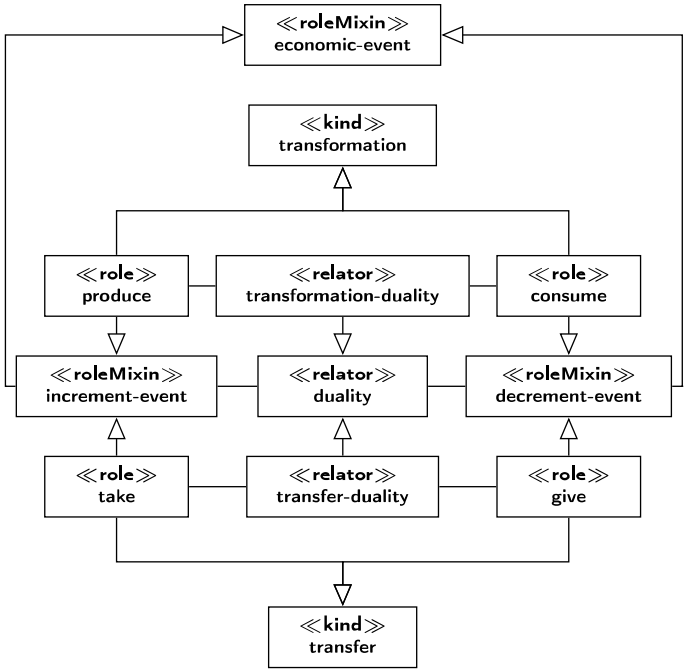

Fig. 3. The REA^2 meta-model for duality.

business patterns as documented by Hruby et al. (2006). This M1 model is an archetypal implementation of the REA pattern described by the meta-model (M2). This implementation does not require a particular syntax. Consequently, it uses a standard UML syntax, instantiating the stereotyped classes of the meta-model as associations with an REA association stereotype, being classes with an REA role stereotype and a name referring to a kind. The principle is also applied to the classes in the M1 diagram (i.e., Fig. 5), which are assigned M2 level REA role stereotypes (i.e., instances of Figs 2–4) where their names refer to instances of a kind.

The exemplar illustrates the archetypal implementation scenario with example data (M0) that were developed for testing the statements that address the semantic changes induced by the switching between the dependent and independent viewpoints. The exemplar will be discussed in detail shortly. For now, we can explain that the test scenario generates independent view statements from the data of a single trading-partner taking the dependent view i.e., the trading-partner 104 in Fig. 6. The generated independent view

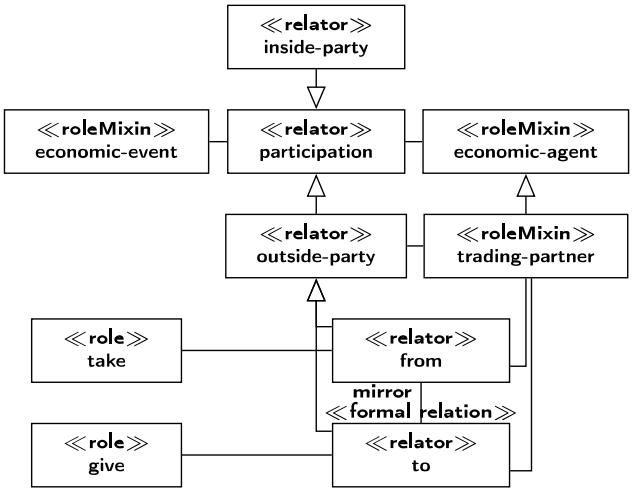

Fig. 4. The REA^2 meta-model for participations.

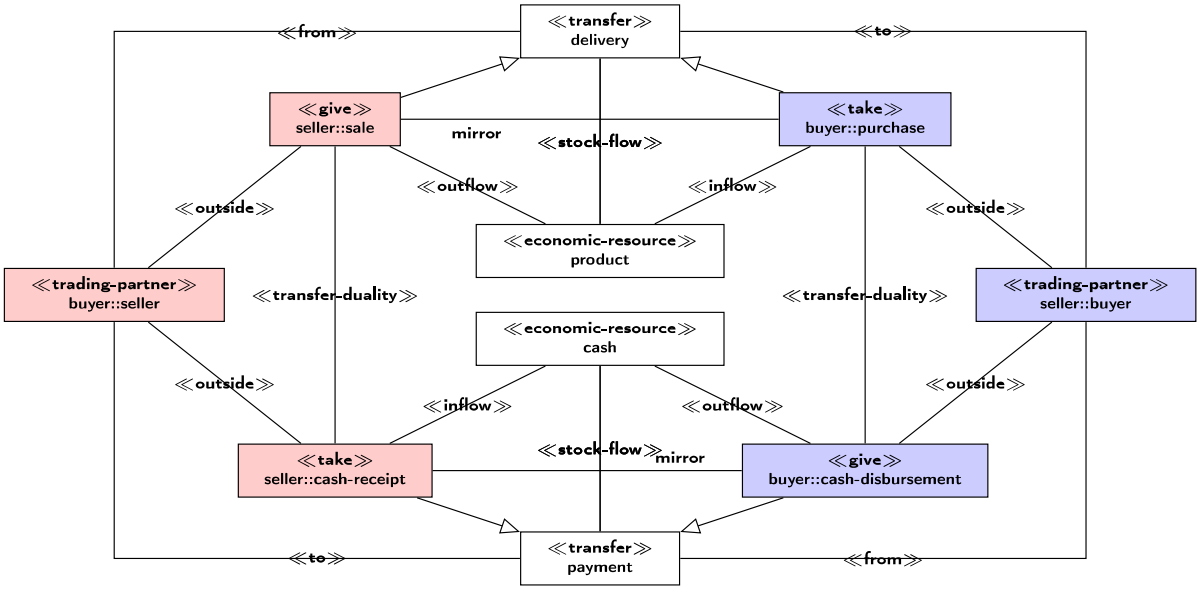

Fig. 5. Collaboration space semantics.

statements are then transformed back into dependent view statements for trading-partners with a view opposing that of the trading-partner of which the data were used to generate the independent view data i.e., the trading-partner 106 – 104's supplier – and 107 – 104's client – in Fig. 6.

4. Research methodology and related research

To the best of our knowledge, this paper is the most recent in a history of papers that endeavour to elucidate the finer semantics of the REA ontology. The best-known originate from the analysis of REA

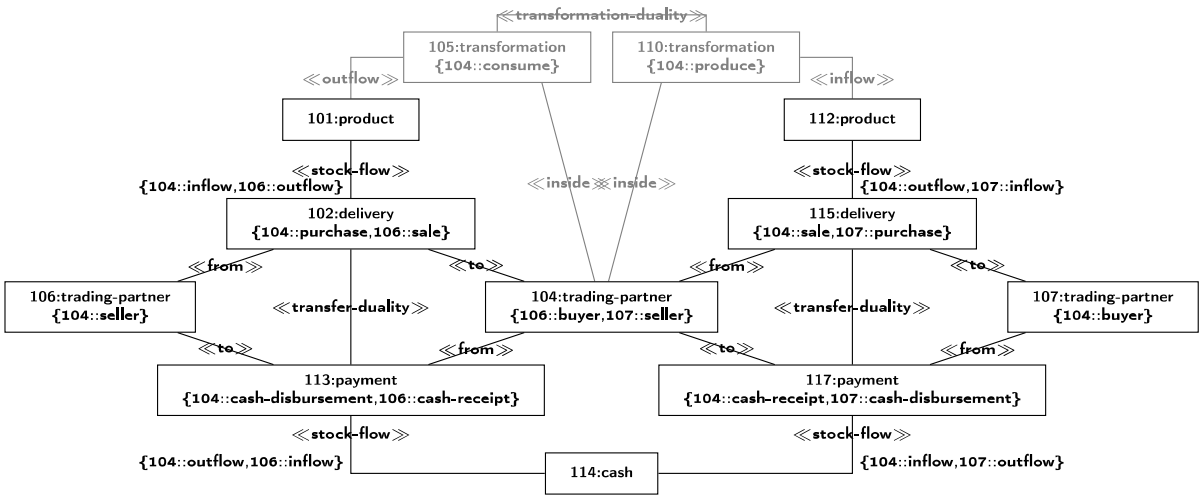

Fig. 6. The evaluation scenario.

in the light of Sowa's foundational ontology (Sowa, 2000), with one further study that supplements Sowa's ontology with abstraction mechanisms in REA (Geerts and McCarthy, 2000b, 2002, 2005, 2006; Sowa, 2000). A number of authors have formalised REA to make it computer-readable, whilst numerous others have tried to find the best definitions for REA concepts that most help the human endeavour in applying REA (Gailly and Geerts, 2013; Gailly, Laurier and Poels, 2008; Hruby and Kiehn, 2006; Ito and Vymětal, 2013; Jaquet, 2006; Laurier and Poels, 2014; McCarthy, Geerts and Gal, 2016a,b; Zdravkovic, Zikra and Ilayperuma, 2011).

Our approach for this paper inscribes itself in the design science tradition, which has been the dominant tradition in REA research (Geerts, 2011). Design science involves a relevance, a rigor and a design cycle. The *relevance cycle* contextualizes the design artefact by referring to a practical problem in the real world. This practical problem informs the evaluation of the artefact. The *rigor cycle* addresses the foundations and innovative aspects of the design artefact referring to the state of the art. The state of the art is defined by scientific and professional literature and presented as the expertise of specialists involved in the artefact's design. The *design cycle* produces design artefacts through an iterative process of radical innovation and continuous improvement, informed by the relevance and rigor cycle (Hevner and Chatterjee, 2010).

As the design, rigor and relevance cycle are typically not addressed simultaneously in the REA literature (Dunn, Gerard and Grabski, 2016), our paper will focus on the rigor cycle addressing the current state of the REA ontology knowledge. To a minor extent, we will also focus on the design cycle by presenting an artefact – namely the exemplar – that will serve as our proof of concept and allow for a scenario-based analysis of our approach presented shortly. This paper addresses research rigour by refactoring the Prolog[1] code found in Geerts and McCarthy (2000a) with notions from the REA formalisation developed by Gailly, Laurier and Poels (2008). That formalism is stereotyped in turn with OntoUML stereotypes (Guizzardi, 2014) instead of OWL stereotypes, and notions from the original Resource-Event-Agent (REA) data model published by McCarthy (1982) and Hruby et al.'s (2006) book on REA. The proof of concept addresses the design cycle through a demonstration of the approach through the

[1]Prolog is a logic-based programming language (Colmerauer and Roussel, 1993).

exemplar being the archetypal example. The exemplar is a set of statements written in Prolog able to transform dependent view Prolog statements into independent view Prolog statements and vice-versa. To emphasise this paper's embedding in the REA literature, the code in Appendix C and E also shows how an opposing and independent perspective could be derived from Geerts' and McCarthy's (2000a) code. These reference older REA artefacts but nonetheless illustrate this paper's embedding of REA's design science tradition; the illustration is accordingly reinforced by an informed argument.

5. The meta-model

The meta-model extends the traditional interpretation of the REA ontology. For readability, the meta-model will be visualised as three separate figures abstracting from cardinality constraints, each focusing on a relator defined by a REA axiom. A *relator* mediates between the roles in a *material relation*. A material relation relies on the relata, which are the entities that are being related, irrespective of their intrinsic properties (e.g. Bob likes Alice). A formal relation is reducible to intrinsic properties of the relata (e.g. Bob is taller than Alice) (NEMO, 2015).

Figure 2 addresses the stock-flow axiom – i.e., *"At least one inflow event and one outflow event exist for each economic resource; conversely inflow and outflow events must affect identifiable resources*." (Geerts and McCarthy, 2005). *Stock-Flows* are shown as relators. The relata in Fig. 2 – i.e., *economic-resource*, *economic-event* and its subtypes – have all been stereotyped as *rolemixins*. A rolemixin is a generalisation of a set of roles, and is depicted in the Fig. 2 as *roleMixin*. A *role* is defined as an anti-rigid sortal type that is connected to a characterising relation and specialises a unique kind (Guizzardi, 2014). Through this generalisation, rolemixins can be assigned to instances of different kinds through a joint specialisation of the kind and rolemixin, where roles can only be assigned to instances of a single kind though specialisation of this kind. The rolemixin *economic-resource* can be assigned to an entity (or thing) that is scarce and is controlled by a trading-partner, to whom it has value (e.g., a right, services and a good can all be economic-resources) (ISO/IEC, 2007). By defining an economic-resource as a rolemixin, this paper defends the opinion that things can only be considered an economic-resource when they are valuable to a trading-partner. For example, a car only has value to the person that can drive it. So, my car is a resource to me, the car next to it in the parking lot is not a resource to me – despite being a resource to its owner - even if its kind (e.g. VW Golf) identical to that of my car. This interpretation of the term economic resource differs significantly from the traditional interpretation as the traditional interpretation can be read as: an economic resource is anything that has the potential of being considered an economic resource by at least one person. Consequently, any object of the kind VW Golf is an economic resource in the traditional interpretation. The rolemixin *economic-event* can be assigned to a phenomenon that reflects changes[2] in the value or the quantity of economic resources (McCarthy, 1982). By defining economic-events as rolemixins, this paper argues that some kinds of event can potentially have an economic impact without having to. For example, a storm can have a considerable economic impact when it damages goods. However, a storm of the exact same kind can have no economic impact at all (e.g., when crossing land with no economic value). Moreover, events that have an economic impact by definition do not need to have an economic impact on every person that can observe them. For example, looking at a cash transfer (e.g., from a third-party perspective) does not make me any richer

[2]In the accounting literature this change is always instantaneous. Even when this change is part of a lengthy process, a single temporal part – typically the first or last – of this process is labelled as the change. For transfers this implies that ownership is transferred either at the start or at the end of the transport.

or poorer, although the trading-partners involved will observe the economic impact of the cash transfer. This person-bound interpretation of the notion economic event is also related to this paper's contribution as the traditional interpretation of economic event is absolute and should be read as "an event that has the potential of having an economic impact on at least one person".

Figure 2 reads the "inflow event" and "outflow event" in the stock-flow axiom as an event playing the increment event role in an inflow and an event playing the decrement event role in an outflow respectively, as inflow and outflow are considered subtypes of stockflow by Gailly, Laurier and Poels (2008). As stated by the stock-flow axiom, the economic event rolemixin has two sub-types. The *increment-event* rolemixin is connected to an inflow relator, where the *decrement-event* rolemixin is connected to an outflow relator, both inflow and outflow are sub-types of the stock-flow relator. The increment and decrement roles are considered mutually exclusive in a single perspective – i.e., in the universe of discourse of a single trading-partner an economic-event is assigned either an increment or decrement role but not both.

Within the context of this paper, the second REA axiom – i.e., *"All events effecting an outflow must be eventually paired in duality relationships with events effecting an inflow and vice-versa*." (Geerts and McCarthy, 2005) – has been modelled as increment and decrement event rolemixins that are connected to a *duality* relator. By modelling duality as a relator connecting two events, this paper subscribes to the interpretation of a duality as a social construct (i.e., a material relationship), which prevails in accounting (Fisher-Pauzenberger and Schwaiger, 2017). For example, when stealing a car we are not defeating the laws of physics, but merely breaking civil law, which is a social construct, as it is physically possible to take a car without having to give anything in return. In the universe of discourse of a single trading-partner, a duality pairs an increment-event to a decrement-event. This unilateral definition of duality as perceived by a single trading-partner relates to the definition of a value-interface in the e3value ontology, which models the duality concept from the perspective of each actor – i.e., trading-partner (Gordijn, 2002). As increment and decrement events are rolemixins, they can be assigned to different kinds of events (e.g., transfers and transformations), shown in Fig. 3. Moreover, as dualities add value for each of the trading-partners involved, the added value for each trading partner can be considered a "qua individual". This added value inheres in the recipient and depends on the provider. For example, in an exchange of cookies for money between Elmo and Cookiemonster, for Cookiemonster the added value of the exchange depends on Elmo transferring the cookies and inheres in Cookiemonster valuing the cookies more than the money that is spent. From Elmo's perspective, the added value of the exchange depends on Cookiemonster transferring the money and inheres in Elmo valuing the money more than the cookies that are transferred. In the case of a transformation duality – as opposed to the transfer duality described above – this added value inheres in the owner valuing the produced resources higher than the consumed resources.

The kinds of things (entities) that the economic resource and agent rolemixins could be assigned to would be an interesting side-discussion. However, to maintain the focus of this paper (and as reflected in Fig. 3) we centre our attention on the kinds of events that the economic-event rolemixin could be assigned, as it is relevant for formalising trading-partner viewpoints. The increment and decrement event rolemixins can be assigned to either events of the transformation or transfer kind. A *transformation* adds value by changing resource properties, whereas a *transfer* adds value in a market transaction between trading-partners (McCarthy, Geerts and Gal, 2016a,b). The *produce* role specialises both the kind of transformation and the increment-event rolemixin. The *consume* role specialises both the transformation kind and decrement-event rolemixin. As a result, only transformation events can be assigned produce and consume roles. When transformations are instantaneous, produce and consume roles can be assigned

to the same event. For example, when blending fruits for making a smoothie, all ingredients lose their identity simultaneously (i.e., consume) and the immediate result (i.e., produce) of the blending process can be identified as a smoothie. However, when describing a complex process with discrete steps, produce and consume roles can be assigned to different events. For example, when adding raisins to dough to make raisin bread, the raisins lose their identity when being added (i.e., consume) although they only change the properties of an intermediate state (i.e., dough) of the product, while the final product (i.e., raisin bread) will only be obtained at the end of the baking process (i.e., produce). The same reasoning applies to adding extra salt or butter to the dough, or seasoning it. In REA, such a complex process is modelled as a configuration of produce and consume events interlinked with transformation dualities. The interlinked consume and produce events can be diverse in nature (e.g. a number of instantaneous consume events combined with a production process with a considerable duration, and an instantaneous production of a by-product at the end).

Continuing, the *transformation-duality* relator specializes the duality relator, connecting a produce role to a consume role. On the other hand, a *transfer-duality* relator specialises the duality relator by connecting a *take* to a *give* role. The *take* role specialises both the transfer kind and increment-event rolemixin, whereas the *give* role specialises the transfer kind and the decrement-event rolemixin.

Finally, the third REA axiom – i.e., *"Each exchange needs an instance of both the inside and outside subsets*." (Geerts and McCarthy, 2005) – addresses how a *participation* relator[3] connects an economic-agent to an economic-event. The *economic-agent* rolemixin *can be* assigned to a natural or legal person participating in economic events. Economic-agents such as employees can participate in events as inside-party on behalf of themselves or another economic agent. In an inside-party relation, economic agents deal with another person (legal or natural) in a *trading-partner* role. In a trading-partner role, an economic agent experiences the effect of an economic event. All economic agents that relate to an economic-event through an outside-party relator are trading-partners. The event kind in which they participate is a transfer between trading-partners. From and to are the subtypes of outside-party relator. A trading-partner connected to a *from* relator assigns a take role to a transfer event. A trading-partner connected to a *to* relator assigns a *give* role to a *transfer* event. Figure 4 summarises the above.

As with REA's economic-resource and economic-event primitives, this paper argues that REA's economic-agent should be modelled as rolemixins assigned to a person (i.e., a kind). A discussion concerning the kinds of persons that can or cannot be assigned an economic-agent role is considered outside the scope of this paper. A consequence of modelling economic-agents as roles is that the same or different persons can play multiple inside and outside roles. For example, when I buy pizza to Joe's Pizza, I can act on behalf of myself as inside and outside agent. In that case, I am Joe's trading-partner and involved in both transfer events (i.e., payment and delivery) through an inside- and outside-party relation. I could also ask my daughter to pick up the pizza and pay Joe. In that case, I am still Joe's trading-partner, but the delivery and payment are executed by my daughter on behalf of me. Consequently, I'm linked to both transfers through an outside-party association and my daughter through an inside-party association. Similarly, Joe could deliver the pizza directly to me or decide to send a pizza-boy. In the former case, Joe is both my trading-partner (i.e., outside-party) and the person participating in the payment and delivery (i.e., inside-party). In the latter case, Joe is still my trading-partner (i.e., outside-party) by the transfer events are executed by the pizza-boy (i.e., inside party) on behalf of Joe.

The REA^2 ontology formalisation as indicated earlier allows for a simultaneous documentation of both – i.e., seller and buyer – perspectives on an exchange. As REA^2 also accounts for the independent

[3]This participation relator represents an REA participation, not an OntoUML participation, which is a formal relationship that holds between an event (economic or not) and all objects involved in it.

view, besides both dependent views, the symmetry between the universes of discourse of both trading-partners needs to be captured by the formalisation. In the meta-model, this symmetry is captured by the formal relation named *mirror*, akin to a mirror image, which is a reflected duplication but is reversed in direction just like in a mirror. A mirror is shown in Fig. 4. A *formal relation* is a relation that can hold directly between its relata and is reducible to intrinsic properties of the relata, such as 'older than'. The mirror relation addresses the fact that a transfer always involves a stock-flow from one trading-partner to another. The formal mirror relation formalises that from and to relators are two sides of the same coin, the coin being the transfer. For example, when Joe sends pizza to a customer, it is a transfer from him to the customer. No pizza can be sent by Joe without going to a customer, nor can any customer receive pizza without it being shipped (i.e., from). The same logic applies to money transfers. No payment can occur without a person spending the money (i.e., from) and a person receiving (i.e., to).

6. Collaboration space: A semantic pattern

Whereas the meta-model in Figs 2, 3 and 4 formalises the concepts of the REA Ontology using OntoUML stereotypes, the semantic pattern in Fig. 5 simultaneously visualises a buyer, seller and independent perspective.[4] The acquisition cycle in blue[5] shows the buyer perspective on an economic exchange, which typically involves a purchase of raw materials from a vendor paired in duality with a cash disbursement to the vendor. The revenue cycle in red,[6] on the other hand, shows the seller perspective on an economic exchange, which typically involves a sale of final products to a customer paired in duality with a cash receipt from that customer.

REA also supports the conversion cycle, which transforms products (i.e., raw materials) acquired through the acquisition cycle to products that can be sold through the revenue cycle (McCarthy, 2003). Besides the presence of a transformation-duality in the meta-model, the conversion cycle is another side-discussion that is thus not addressed in this paper. The stereotypes in Fig. 5 relate data-model constructs to the meta-model. To embed Fig. 5 in the REA literature, events follow the naming conventions of the ISO OeBTO standard[7] on the independent view and the trading-partner perspective on these events follows the naming conventions of McCarthy's work.[8]

The formal relation *mirror* predicates that an opposing view exists on each *stock-flow* connected to a transfer in a *give* or *take* role. It should be noted that the independent view does not allow for these simultaneous inflow-outflow semantics, as the same stock-flow is an inflow for one trading-partner and an outflow for the other. Consequently, view independent generalisations of stock-flows have been included in Fig. 5. A *delivery* has a *sale* and *purchase* role that cannot be seen from an independent perspective, and cannot be seen simultaneously from a dependent perspective.

From the buyer's perspective, the delivery event is an inflow of products; in the seller's eyes this same delivery is an outflow of the same products. The delivery role connected to a product inflow is called a purchase, whereas the delivery role connected to a product outflow is called a sale. A *payment* has a *cash-disbursement* and *cash-receipt* role that are invisible from an independent perspective, and

[4]Each of these perspectives should be seen an additional viewer-specific semantic layer on top of a shared objective reality, where this paper explicitly abstracts from modelling the objective reality.

[5]The blue refers to Cookiemonster in W.E. McCarthy's iconic slides on exchanges (McCarthy, 2017).

[6]The red refers to Elmo in W.E. McCarthy's iconic slides on exchanges (McCarthy, 2017).

[7]Figure 11 of ISO/IEC (2007).

[8]Figure 7 of McCarthy (1982), McCarthy, Geerts and Gal (2016a,b).

cannot be seen simultaneously from a dependent perspective. The payment is an inflow of cash from the perspective of the seller – i.e., cash-receipt – and an outflow of cash from the perspective of the buyer – i.e., cash-disbursement.

Figure 5 highlights two distinct transfer-duality relators between a delivery and a payment in their respective give and take roles – i.e., one for each perspective. From the buyer perspective, the delivery is perceived as an inflow of products – i.e., purchase – that is dual to the payment, which is perceived as an outflow of cash – i.e., cash disbursement – while the seller perceives them as a dual sale – i.e., outflow – and cash receipt – i.e., inflow. Since the duality axiom requires inflow and outflow semantics, it depends on the perspective of a trading-partner. The compliance of data with the duality axiom thus cannot be verified without reference to the perspective of a specific trading-partner (Geerts and McCarthy, 2000a; 2005).

In the independent view, the semantics of the relators are independent of the perspective of the trading-partners. Rather, they represent the perspective of the entire trade community or an independent third-party not taking part in the transaction (such as a supply chain manager, government agency, regulatory body or – indeed – an entire trading community). Where the dependent view focuses on the direction – i.e., in or out – of resource flows for a single trading-partners, the independent view distinguishes two kinds of participation relators – i.e., *to* and *from* – that signal the direction of the stock-flows connected to transfers, which are consequently deprived of their inflow or outflow semantics compared to the dependent perspective (ISO/IEC, 2007). Accordingly, we need to switch between both perspectives.

7. The exemplar

The exemplar that will be used to demonstrate how the models introduced above enable computers to produce the independent, buyer and seller perspective is now presented in detail. Although switching between the buyer, seller and third-party scripts is intuitive and easy for humans, it is less obvious for computers and algorithms. The exemplar will reveal how this challenge is overcome. For this purpose and ease of understanding, we will refer to the data of a single trading-partner in the supply chain. The example data are assembled into the exemplar that is the paper's example and proof of concept.

The exemplar and proof-of-concept application were coded in SWI-Prolog. The code can be found in the Appendix. Since the code covers the entire value chain, it is more elaborate than the Prolog code in Geerts and McCarthy's (2000a) intensional reasoning paper, which exclusively covers the revenue cycle. The test scenario will be introduced and visualised in the remainder of this section.

The exemplar shown by Fig. 6 covers the data of trading-partner 104, who assigns REA rolemixins as modelled in Fig. 5 to things (entities) 101 to 120. The test scenario contains acquisition, revenue and conversion cycle data. The acquisition cycle consists of trading-partner 106, delivery 102, payment 113, cash account 114 and product 101, all seen from the perspective of agent 104. The revenue cycle concerns product 112, delivery 115, trading-partner 107, payment 117 and cash account 114. The conversion cycle consists of consume event 105, product 101, produce event 110, and product 112.

The newly introduced formal relation mirror and the semantic pattern presented in Fig. 5 allow for deriving the vendor perspective on trading-partner 104's acquisition cycle from the data shared by 104. Figure 6 reveals that trading-partner 106 will perceive trading-partner 104 as a buyer, which is formalised as the tag "`106::buyer`" (i.e., trading-partner 104 is a buyer to trading-partner 106). The semantic pattern also predicates that event 102, which is a purchase to trading-partner 104 – i.e., tag "`104::purchase`" – is a sale to trading-partner 106 – i.e., tag "`106::sale`" – whereas event 113

will be categorised as a cash-receipt by trading-partner 106 – i.e., tag "`106::cash-receipt`" – as it is a cash-disbursement to trading-partner 104 – i.e., tag "`104::cash-disbursement`".

Figure 6 also shows the perspective of trading-partner 107 on trading-partner 104's revenue cycle. Trading-partner 107 will perceive trading-partner 104 as a seller – i.e., tag "`107::seller`". Sale 115 as perceive by trading-partner 104 – i.e., tag "`104::sale`" – will be perceived as a purchase by trading-partner 107 – i.e., tag "`107::purchase`". Economic event 117 will be perceived as a cash-disbursement by trading-partner 107 – i.e., tag "`107::cash-disbursement`" – as it is a cash-receipt for trading-partner 104 – i.e., tag "`104::cash-receipt`".

In an REA independent view script, which takes the perspective of an independent third-party, trading-partners cannot be assigned customer or vendor semantics as those depend on a trading-partner's perspective. Consequently, in the semantic model derived from agent 104's data, agents 104, 106 and 107 are to be categorised as trading-partners. Trading-partner is the stereotype assigned to buyer and seller in Fig. 5. Similarly, event 102 and 115 are categorised as deliveries where their dependent view semantics are sale and purchase, which are subtypes of delivery in Fig. 5. Correspondingly, event 113 and 117 are assigned payment semantics in the independent view, while they are cash receipts and disbursements in their respective dependent views.

Equivalently, it is impossible to assign inflow and outflow semantics to stock-flows connected to transfer roles – i.e., give and take – as, in the independent view, a single transfer is connected to an inflow for one trading-partner and an outflow for the other.

Since the independent view also requires information about the flow of resources (typically oppositely directed products and money flows) between the *from* and *to* participations, it can convey the inflow and outflow semantics that characterise the dependent view. If a trading-partner role connects a transfer to an inflow in his/her dependent view, resources are flowing towards (to) him/her, where resources will be flowing away from him/her when he/she relates the transfer to an outflow.

Consequently, delivery 102 entails a stock-flow of product 101 from trading-partner 106 to trading-partner 104. The dual event is a payment involving a cash transfer from trading-partner 104 to trading-partner 106. Then delivery 115 results in a stock-flow of product 112 from trading-partner 104 to trading-partner 107. The requiting payment is cash transfer 117 from trading-partner 107 to trading-partner 104. Were the case exemplar shows traditional product-for cash exchanges, the same logic can be applied to barter trade (i.e., product for product exchange) and money for money exchanges, since transfer dualities also apply to delivery-delivery and payment-payment pairs.

Table 1 summarises the case examplar visualising all semantics assigned to an object in different views. It shows that in the case exemplar, resource semantics are universal for the entire trade community. Consequently, no specific dependent view semantics were assigned to resources 101, 112 and 114. Although view dependent semantics exist (e.g. raw-material, final-product for products) a discussion was irrelevant for this paper's contribution. As a result of this decision, resources 101 and 112 are recognised as products and resource 114 as cash by the entire trade community including trading-partners 104, 106 and 107. On the other hand, economic-event semantics differ according to the trading-partner perspective taken. For example, delivery 102 is a delivery for the entire trade community, but also a purchase for trading-partner 104 alone and a sale for trading-partner 106 alone. Economic-agents also receive perspective dependent semantics. For example, trading-partner 104 is seen as a trading-partner by the entire trade community, but additionally as a buyer by trading-partner 106 and as a seller by trading-partner 107. In turn, trading-partner 104 will perceive 106 and 107 as a trading-partners like the entire the trade-community, but also as a seller and buyer respectively in his particular perspective. In REA's dependent perspective trading-partners are not aware of themselves as they are never modelled

Table 1
Summary of the semantics of the evaluation scenario

Object	Independent	104's view	106's view	107's view
101	Product			
102	Delivery	Purchase	Sale	
104	Trading-partner		Buyer	Seller
105	Transformation	Consume		
106	Trading-partner	Seller		
107	Trading-partner	Buyer		
110	Transformation	Produce		
112	Product			
113	Payment	Cash-disbursement	Cash-receipt	
114	Cash			
115	Delivery	Sale		Purchase
117	Payment	Cash-receipt		Cash-disbursement

explicitly. However, since 104 is part of the trade community, he knows that he's a trading-partner. When applying the logic presented above, 104 can infer that he is a buyer to trading-partner 106 in the context of exchange 103–112 and a seller to trading-partner 107 in the context of exchange 115–117. The missing object numbers were assigned to inside agents in a more elaborate version of this case exemplar. They were omitted in this version to focus on the actual contribution of this paper.

8. The proof of concept: Prolog code

The Prolog code in Appendix formalises the inference rules (Appendix A), tests (Appendix D) and counterparts of the exemplar introduced above (Appendix B). We now introduce the statements and inference rules that are required for deriving the independent view from dependent view data and an (opposing) dependent view from independent view data. The translation process to trading-partner view to the perspective of a trading-partner with an opposing view is mediated by the independent view, which is semantically equivalent. In the Prolog code, this translation process is decomposed in two phases. First, trading-partner view data are translated to independent view data. In this case, trading-partner 104's data (Appendix B) are made view independent by the inference rules in Appendix A. Appendix A contains a set of `rea_lattice` predicates that is equivalent to the inheritance and instantiation semantics in Figs 2 to 5. These predicates are then used to assign independent-view semantics to the dependent view data in Appendix B through `is_a` statements. Through this process, the objects of the exemplar are assigned both trading-partner and independent-view semantics. Subsequently, the trading-partner view semantics are hidden and the independent-view semantics shared as `oebto_role` and `oebto_relator` predicates. The filtering technique that is used to hide trading-partner view semantics could be generalised to protect information that provides competitive advantage (e.g., by sharing the components of a product, without sharing the steps of the production process). The independent-view oebto statement are then converted to trading-partner view statements by `dependent_role`, `dependent_relator`, `agent_pattern` and `event_pattern` statements. The opposing views of trading-partner 106 and 107 obtained through the statements described above are then tested in Appendix D.

The is-a semantics – i.e., inheritance and stereotypes – found in Figs 2–4 have been summarised in a semantic lattice (i.e., a partially ordered set of categories (Sowa, 2000)). The lattice is used to derive

independent view semantics from dependent view data. The is-a relations are formalised as predicates, of which a subset is shown in Code Snippet 1, the complete set of statements can be found in Section A.4 and A.5 of Appendix A. The is-a argument in the rea_lattice statements has been added for clarity. It has no computational value. The first line in Code Snippet 1 specifies that every increment-event is an economic event. The second line specifies that every produce is an increment event, and hence an economic event. This statement is indented because it models a second layer of the semantic lattice. The indentation has no computational value.

The inference rules in Section A.1, shown in Code Snippet 2, result in is-a statements. Section A.1.1 contains the code that processes the transitivity of is-a semantics at the type level, which can be found in the in the semantic lattice. For example, if a rea_lattice statement predicates that a purchase is a delivery (i.e., `rea_lattice(purchase,is_a,delivery).`) and another statement states that a delivery is a transfer (i.e., `rea_lattice(delivery,is_a,transfer).`), then a purchase is also a transfer (i.e., `is_a(purchase,transfer)`). The first inference rule under Section A.1.1 is

```
rea_lattice(increment-event,is_a,economic-event).
    rea_lattice(produce,is_a,increment-event).
    rea_lattice(produce,is_a,transformation).
    rea_lattice(take,is_a,increment-event).
    rea_lattice(take,is_a,transfer).
rea_lattice(delivery,is_a,transfer).
    rea_lattice(sale,is_a,delivery).
    rea_lattice(sale,is_a,give).
    rea_lattice(purchase,is_a,delivery).
    rea_lattice(purchase,is_a,take).
```

Code Snippet 1. As-is statements in the semantic lattice.

```
/*A.1. Inference Rules*/
/*A.1.1. Navigating the REA Ontology Primitives and their Super- and
Sub-Types*/
is_a(X,X).
is_a(X,Y):-
     rea_lattice(X,is_a,Y).
is_a(X,Z):-
     is_a(X,Y),
     rea_lattice(Y,is_a,Z).
/*A.1.2. Inherited Semantics*/
is_a_role(X,is_a,Z,to,P):-
     rea_role(X,is_a,Y,to,P),
     is_a(Y,Z).
is_a_relator(X,is_a,Z,to,P):-
    rea_relator(X,is_a,Y,to,P),
    is_a(Y,Z).
```

Code Snippet 2. Processes the is-a statements in the semantic lattice.

```
/*A.2. The Open-edi Business Transaction Ontology*/
/*A.2.1. The OeBTO vocabulary as a subset of REA*/
oebto_vocabulary([product,cash,delivery,payment,trading-partner,
stock-flow,transfer-duality,from,to]).
oebto_interface(X):-oebto_vocabulary(L),
      member(X,L).
/*A.2.2. Defining the visibility of OeBTO compliant REA roles and
relators*/
oebto_role(X,is_a,Y,to,Z):-
      is_a_role(X,is_a,Y,to,P),
      oebto_interface(Y),
      access(Z,to,P,data).
oebto_relator(X,is_a,Y,to,Z):-
     is_a_relator(X,is_a,Y,to,P),
     oebto_interface(Y),
     access(Z,to,P,data).
```

Code Snippet 3. Derives the independent view from the dependent view.

the identity (e.g. every economic-event is an economic-event). The second statement operationalises is-a semantics with the statements of which Code Snippet 1 is a subset.

Section A.1.2 shows the code that combines these type-level is-a semantics with the is-a semantics of instantiation such as shown by Fig. 6. For example, if it is given that 102 is a purchase (i.e., `rea_role(102,is_a,purchase,to,104)`), then 102 must also be a delivery (i.e., `rea_role(102,is_a,delivery,to,104)`), and a transfer (i.e., `rea_role(102,is_a, transfer,to,104)`). The latter two predicates can be derived from the first rea_role statement with the help of the inference rules under Section A.1.2, which combine the first rea_role statements with is-a statements. For computational reasons this assignment had to be operationalised as two separate statements (i.e., one for roles one for relators).

The inference rules in Section A.2, shown in Code Snippet 3, use the is-a statements of Section A.1.2 to derive the independent view from given dependent view data. The code in Section A.2.1 defines a vocabulary for the independent view as a subset of the rolemixins and relators included in the rea_lattice. This vocabulary is also a subset of the terms found in the OeBTO standard (ISO/IEC, 2007).

Section A.2.2 contains the oebto statements that gives persons (such as customer, supplier, government) access to a subset of a trading-partner's data. This subset is constrained by the vocabulary defined in Section A.2.1 and the access is restricted through the access statement. In absence of an access statement, a person has no access to a trading-partner's data. When a person has access, he only sees the semantics he has access to. For example, event 102 is a purchase to 104, since purchase is not part of the OeBTO vocabulary defined in Section A.2.1, the term delivery will be shared with a trading-partner that has access to 104's data (i.e., `oebto_role(102,is_a,delivery,to,106)`), as delivery is a super-type of purchase (i.e., `is_a_role(102,is_a,delivery,to,104)`) and part of the vocabulary defined in Section A.2.1 (i.e., `oebto_interface(delivery)`). The third and final condition constrains the access of trading-partners to data. Although the delivery semantics of entity 102 can be shared in the collaboration space, trading-partner 007 will not have access to the data in absence of an

explicit access statement (i.e., `access(007,to,104,data)`). As with Appendix Section A.1.2 the filters for roles and relators have been operationalised separately for computational reasons.

The statements in Section A.2.3, shown in Code Snippet 4, add *from* and *to* statements involving the modeller, since the modeller is not self-aware in the dependent view, and required in the independent view. The added from and to statements comply with the formal relation from-mirrors-to shown in Fig. 4. The Code in Section A.2.3. mirrors all from and to statements identified by the modeller. For example, if trading-partner 106 is connected to purchase 102 through a *from* relator according to modeller 104 (i.e., `rea_relator([102,106],is_a,from,to,104)`), then 104 will be connected to 102 through a *to* relator (i.e., `oebto_relator([102,104],is_a,from,to,106)`). For a view independent example, entity 106 can be replaced with entity 000. The statements in Section A.2.3.1 formalise the formal relation mirror. The statements under Section A.2.3.1 define the symmetric nature of the mirrors relation.

The inference rules in Section A.3, shown in Code Snippet 5, derive dependent view from given independent view data. Since the proof of concept opts for a one version of the truth approach, dependent view statements are derived from OeBTO statements, which are in their turn derived from the dependent view data of the modeller as shown above. In the independent view, *from* and *to* statements convey the give and take (inflow and outflow) semantics that are characteristic to the dependent view. The statements in Section A.3.1 use the information conveyed by to and from statements to assign inflow and outflow semantics to stock-flows. An outflow is a stock-flow connected to a transfer from the modeller. An inflow is a stock-flow connected to a transfer to the modeller. The first statement under Section A.3.1 transforms from semantics, which are typical for the independent view perspective, to outflow semantics, which are typical for the dependent view perspective. If the relator between 102 and 106 is a from (i.e., `oebto_relator([102,106],is_a,from,to,106)`) and there is a stock-flow relator 101–102

```
/*A.2.3. Mirror (Formal relation)*/
oebto_relator([X,Y],is_a,Z,to,P):-
    access(P,to,Y,data),
    rea_relation(Alpha,mirrors,Z),
    rea_relator([X,_],is_a,Alpha,to,Y).
/*A.2.3.1. Opposing Semantics in Opposing Views*/
rea_relation(from,mirrors,to).
rea_relation(to,mirrors,from).
```

Code Snippet 4. Adds the modeller in the independent view.

```
/*A.3. Deriving a Trading-partner view from the Independent view*/
/*A.3.1. inflow and outflow for transfers*/
dependent_relator([X,Y],is_a,outflow,to,P):-
    oebto_relator([X,Y],is_a,stock-flow,to,P),
    oebto_relator([Y,P],is_a,from,to,P).
dependent_relator([X,Y],is_a,inflow,to,P):-
    oebto_relator([X,Y],is_a,stock-flow,to,P),
    oebto_relator([Y,P],is_a,to,to,P).
```

Code Snippet 5. Derives inflow and outflow relators from to and from semantics.

```
/*A.3.2.2. Event semantics*/
dependent_role_event(E,is_a,X,to,P):-
    oebto_role(E,is_a,Y,to,P),
    oebto_relator([E,P],is_a,Z,to,P),
    event_pattern(X,is_a,Y,Z).
/*A.3.2.2.1. Event Patterns*/
event_pattern(sale,is_a,delivery,from).
event_pattern(purchase,is_a,delivery,to).
event_pattern(cash-receipt,is_a,payment,to).
event_pattern(cash-disbursement,is_a,payment,from).
/*A.3.2.3. Agent semantics*/
dependent_role_agent(A,is_a,Z,to,P,in,[D,A]):-
    oebto_relator([D,A],is_a,Y,to,P),
    oebto_role(D,is_a,X,to,P),
    agent_pattern(X,Y,Z).
/*A.3.2.3.1. Agent patterns*/
agent_pattern(delivery,to,buyer).
agent_pattern(delivery,from,seller).
```

Code Snippet 6. Derives event and agent roles from to and from statements.

according to 106 (i.e., `oebto_relator([102,101],is_a,stock-flow,to,106)`), then this stock-flow must be an outflow to 106 (i.e., `dependent_relator([102,101],is_a,outflow,to,106)`). The second statement under Section A.3.1 does the same for inflow and to semantics.

Similarly, buyer and seller semantics and sale, purchase, cash-receipt and cash-disbursement semantics can be derived from the *from* and *to* statements in the independent view. The statements in Section A.3.2.3, shown in Code Snippet 6, assign seller and buyer semantics. If a trading-partner is connected to a delivery through a *to* relator, it must be a buyer, it a trading-partner is connected to a delivery through a *from* relator, it must be a seller. The statements under Section A.3.2.3 operationalise the agent patterns under Section A.3.2.3.1. If entity 115 is a delivery to 107 (i.e., `oebto_role(115,is_a,delivery,to,107)`) and relator 115–104 is a from to 107 (i.e., `oebto_relator([115,104],is_a,from,to,107)`), 104 must be a seller to 107 in the context of relator 115–104 (i.e., `dependent_role_event(104,is_a,seller,to,107,in,[115,104])`).

The statements in Section 3.2.2, shown in Code Snippet 6, assign sale, purchase, cash-receipt and cash-disbursement semantics to deliveries and payments. If a trading-partner is connected to a delivery through a *from* relator, the delivery must be a sale from his viewpoint. If a trading-partner is connected to a delivery through a *to* relator, the delivery must be a purchase from his viewpoint. If a trading-partner is connected to a payment through a *to* relator, the payment must be a cash-receipt from his viewpoint. If a trading-partner is connected to a payment through a *from* relator, the payment must be a cash-disbursement from this viewpoint. The statement under Section A.3.2.2 operationalises the semantic patterns defined under Section A.3.2.2.1. For example, entity 102 is a sale to entity 106 (i.e., `dependent_role_event(102,is_a,sale,to,106)`) if 102 is a delivery (i.e., `oebto_role(102,is_a,delivery,to,106)`) and 102–106 is a from according to 106 (i.e., `oebto_relator([102,106],is_a,from,to,106)`).

The rea_role and rea_relator statements in Appendix B identify the roles and relators as they are assigned by trading-partner 104. The statements in Section B.1 of Appendix B are equivalent to trading-partner 104's perspective in Fig. 6. For example, 113 is a cash-disbursement to 104, 102 is a purchase to 104, and the relator between them is a transfer duality in 104's perspective. Since trading-partner 104 is not self-aware in the exemplar, trading-partner 106 and 107 identify 104 as a trading-partner. Appendix B also contains the access statements that give trading-partner 106 and 107, as well as an independent third person – i.e., 000 – access to 104's data.

Appendix D, contains the tests that evaluate whether the opposing perspectives of trading-partner 106 and 107, and an independent perspective (as visualised in Fig. 6) can be derived from trading-partner 104's perspective as formalised by the statements in Appendix B. Section D.1. tests the perspective of trading-partner 106, and Section D.2. test the perspective of trading-partner 107. Section D.3 tests the perspective of an independent third-party. Section D.4 tests a full-disclosure approach to the independent view. This alternative independent view has no access restrictions, which means that all persons can access the information, and includes transformation roles and relators in its vocabulary, which means that traceability of product components is enabled through the publication of production transformation processes. Finally, in order prove that the approach presented above can be applied to traditional REA data, Appendix E tests whether an independent and opposing dependent view can be derived from the Prolog code found in Geerts and McCarthy's (2000a) intensional reasoning paper. The relevant[9] part of Geert's and McCarthy's Prolog code can be found in Appendix C. Appendix E demonstrates that the logic introduced in this paper can transform dependent view data into independent view data that can in turn be transformed into the dependent view of a trading-partner with an opposing perspective (i.e., the other trading-partner). To achieve this, it is only required to identify the trading-partner to which the dependent view belongs and map the vocabulary of the proprietary data format to that of the dependent view model presented above. Since the model presented above is based on role semantics, it does not require interfering with the semantics of the proprietary data model.

9. Discussion

The innovation in the formalisation that we have presented in this paper is the identification of a trading-partner's central role in REA models, as trading-partners define their own universe of discourse by assigning REA-roles and relators to instances of different kinds. Through this formalisation of REA constructs as roles and relators, the constructs become context dependent tags assigned to instances of a kind. A *kind* is a rigid sortal, which means that their instances cannot change their kind during their lifespan. For example, a natural person, which is an instance of a kind, becomes an agent in the context of an exchange, or land (kind) is a resource when it is cultivated. This distinction between a subjective and objective reality is relevant to all human interaction. This distinction is important in many economic disciplines (such as contracts) and taken to an extreme in inductive game theory, in which it is possible to abstract from the meaning assigned to an objective reality (Kaneko and Kline, 2008, 2010). The ability to model an economic reality without the need for an objective reality might be extremely useful for modelling REA's unhappy paths. For example, an expected sale that did not go through. Modelling such an unhappy path can be very difficult if we assume that value is an intrinsic (objective) attribute of things irrespective of their utility to a specific person. However, when working with subjective realities, it is

[9] As we are only interested in the data, all Prolog statements relating to the definition of the conceptual schema in the intensional reasoning paper have been omitted.

possible to account for a sudden loss of utility – and hence value – of a product for a person without observing any change in the objective properties of a product. When modelling resources, events and agents as roles instead of kinds, we can account for the subjective and contextual character of value. Such subjective economic models should allow us to model and simulate both rational and irrational customer behaviour.

This paper also demonstrates that it is possible to formalise the dependent view as a specialisation of the independent view. Consequently, is-a semantics operationalised in object-oriented programming could easily implement the logic that obtains the independent view by stripping the view-dependent semantics from the original data.

The paper also shows that the *from* and *to* semantics in the independent view are meaningful enough to derive the perspective of one or more trading-partners from an independent view.

This discovery opens new perspectives for blockchain accounting in which the block-chain takes an independent perspective and individual trading-partners derive their own perspective from the block-chain. Such a blockchain would need a clear definition of its scope, which could be limited to transfers only or could involve transformations as well. The filtering principle shown in Section A.3 could be used in both situations. The vocabulary in Section A.3 that is limited to transfers, and also in Section A.6 allows for sharing information about transformations. In a more restricted implementation, the sharing could be limited to deliveries or payments only. Consequently, this filtering principle might be useful if trading-partners want to share enough information for collaboration without sharing too much information about processes that provide competitive advantage.

Given that REA^2 uses the exact same primitives as the REA ontology that maps it to other interpretations and implementations of REA, ontologies that have been mapped to the original interpretation of the REA ontology should be straightforward. However, alternatives for or extensions of the REA nomenclature used in this paper exist. For example, Hruby et al. (2006) define provider and recipient for participation relators related to increment or decrement events. That stated, Laurier and Poels (2014) explain how these provider-recipient semantics relate to the inside-outside semantics used in this paper.

9.1. Limitations

This paper refrains from addressing the constraints on assigning REA roles to instances of kinds. For example, can a computer play the role of economic agent? Whilst of interest, such discussions would distract from the central contribution of this paper. The paper also limits itself to a set of core REA concepts. It does not address commitments, types, groups, responsibility relators and many other REA constructs, so as to focus on event roles and their relators. In a collaboration space, a single transfer can be assigned different semantics by trading-partners and third-parties. The same applies to commitments, which are like events. As a result, an REA expert should be able to apply the principles shown above to commitments. Next to addressing only core REA concepts, the paper also refrains from addressing the similarity between REA roles and relations and thematic roles and relations. The discussion on roles and relationships above is solely focused on an economic context, abstracting from linguistic aspects. A thorough analysis of the interaction between the linguistic and economic aspects of REA could contribute to natural language processing in commerce.

Where the produce and consume role and the transformation duality are shown in Fig. 3, their semantics in the dependent and independent perspective have not been discussed. The main reason for their present exclusion is that their semantics are underspecified in the REA literature, and that the formalisation would involve a considerable amount of speculation (McCarthy, Geerts and Gal, 2016a,b). Given these remarks, a thorough discussion on transformations is considered out of scope.

Although visualised in Fig. 4, a discussion on inside-party relators is also considered out of scope. In the formalisation presented above, a transfer duality is modelled by one trading-partner. Subsequently, the duality is shared in the independent view and needs to be accepted by the trading-partner with an opposing view. Future research should address the presence and absence of interactions between one trading-partners subjective reality, the objective reality and the subjective reality of a trading-partner with an opposing view. This research should mitigate the current dominance of the modelling trading-partner in the triangle subjective reality, objective reality and opposing subjective reality.

As there is – to the best of our knowledge – no modelling language that allows for a multi-layer observant-dependent semantics, we have chosen OntoUML to formalise our models as its semantics closely relate to the semantics required for showing a multi-perspective reality with layered semantics. However, we are fully aware that the use of OntoUML as demonstrated in this paper goes well-beyond its intended use. For example, the definition of name spaces inside tags is a novel practice that was required for showing multiple perspectives in a single model (i.e., the perspectives of trading partner 104, 106, 107 and 000). The use of OntoUML roles, rolemixins and relators as social constructs part of an observer's subjective reality is not included in OntoUML's intended use. Consequently, this paper introduces a first use case – and hence the need for – modelling languages that support multiple (subjective) realities simultaneously.

9.2. Implications

The REA^2 formalisation introduced above allows for an automated transformation from dependent view information to independent view information and vice versa. This approach is expected to facilitate the integration between traditional ERP and AIS, of which most take the dependent view, and Network Resource Planning (NRP) systems, which take the independent view (Value Network Software (NRP), 2016). Based on these two types on enterprise information systems (EIS), REA^2 thus addresses EIS interoperability between traditional EIS (which typically uses a trading-partner perspective), and EIS for the collaborative economy (which typically uses a trading-partner independent perspective).

Through a unified formalisation of the REA ontology, the model and meta-model presented above bridge the gap between (accounting) information for obtaining a competitive advantage that is classified and cannot be shared such as the company's cost structure, business process layouts, employee expertise, and unclassified trade information that is essential for collaborative value creation. In such a setting, classified information could be documented using REA's dependent view, where the unclassified information could be filtered and shared in the collaboration space using REA's independent view derived automatically from the dependent view by statements like those in Section A.2. The vocabulary that defines the filtering could be extended or reduced according to the requirements and preferences of the trading-partners involved.

Since the REA ontology finds its origin in accounting the view integration technique shown above could allow for an integrated type of accounting that could take advantage of the cloud technology that is widely successful. The ability to share accounting data automatically, which is expected to decrease the cost of compliance and without the risk of exposing information of strategic importance might increase an organisations' willingness to share trade information. Such information can help implement 'e-customs' and implement supply-chain monitoring systems such as with blockchains. Similarly, the hitherto hidden information explicated by our approach could increase food safety, fight counterfeit and money laundering.

9.3. Future research

This paper has mainly focussed on construction rigor and design. Whilst having the ambition to be part of the REA research embedded in the design science paradigm, relevance evaluation will follow in a later phase of this project. The relevance would be tested by generating and testing a prototype in a commercial programming language (e.g., C#, Java, Python, Ethereum). The test scenarios of this initial prototype will also include tracking and tracing (Laurier and Poels, 2012). After testing this initial prototype, commitments, types and groups will be added to the prototype. After testing this prototype for cloud-based and blockchain accounting, we intend to add cross-company planning to the prototype. Cross-company planning is the rationale for the view integration technique introduced above and is expected to advance the state-of-the-art considerably, since it is our goal to build a prototype that plans an entire value network (supply chain) while the processing should be local and decentralised. After building and testing the cross-company planning prototype, it is our goal to build libraries, prototypes and test the commercial prototypes in a real-world setting. We plan to draw on existing experiences. For example, an REA-based intra-company planning system that has already been built in Java (Buysse and Jonnaert, 2012).

This paper primarily uses (McCarthy, 1982) and ISO OeBTO 15944-4 as a source for it's REA vocabulary. In the light of the future work above, more general or more specific terms might be required. For example, it might be convenient to have more elegant but perspective independent generally acccepted terms for seller and buyer, vendor and client. It would also be convenient to find generally accepted terms for sales and purchases, and cash-receipt and cash disbursement in goodsdominant logic, service-dominant logic, rightsdominant logic, barter trade and finance (i.e., money-for-money exchanges).

10. Conclusions

This paper has introduced an innovative formalisation of the REA Ontology called REA^2, which allows for view integration across trading-partners and third parties. REA^2 uses REA constructs as rolemixins assigned by an explicitly identified trading-partner as the modeller instead of rigid kinds assigned by an implicit modeller. *View integration* is defined as the integration of local views into a global model, specifying how each of these local views can be derived from the global view (McCarthy, 1982). In contrast to McCarthy's definition of view integration, the global model that is REA^2 and presented here is collaboration-space wide and not limited to the enterprise, whereas the local views can remain dependent on the perspective of a trading-partner.

The REA^2 formalisation also allows us to distinguish between a data format for information that is relevant to the entire trade community and third parties – i.e., the independent view – and a data format for information that is only relevant to members of the organisation – i.e., the dependent view. Furthermore, a technique is demonstrated that can directly or indirectly generate information relevant to an independent third-party or a trading-partner with an opposing perspective from the information shared by a single trading-partner. This technique shows that the omission of the perspective defining trading-partner impedes an automated transformation of the dependent to independent view, or from one trading-partner's view to the opposing view of another trading-partner as computers cannot deal with the ambiguity. In Fig. 6, the perspective defining trading-partner defines a namespace in the tags such as "`106::sale`". In Appendix A, this perspective of the defining trading-partner is shown as variable P in the statements. Humans can easily deal with this ambiguity as they are aware of the trading-partner that

defines the perspective taken by an information system. Humans are also familiar with the reality that, what is given away by one trading-partner, must be received by a trading-partner with an opposing view. Given computers cannot do the same, we have brought computer productivity to this otherwise purely human endeavour. REA^2 raises awareness that making this view-determining trading-partner explicit in dependent view models allows for automated transformations between opposing trading-partner views and between one or more trading-partner views and the independent view.

In its widest sense, this paper highlights REA^2's implications for the beneficial future development of a) enterprise information systems (EIS) in the cloud, b) social-media-based EIS, c) blockchain EIS, and d) EIS interoperability. The modest contribution demonstrated by REA^2 thus far moves REA closer towards that aim.

Supplementary data

Appendices A–E are available at: http://dx.doi.org/AO198.

Acknowledgements

We would like to thank W.E. McCarthy, who helped us realize that the work on integrating REA's dependent and independent view we did earlier could be simplified while making it more powerful. We would also like to thank Robert Haugen, Pavel Hruby, Paul Johannesson and an anonymous reviewer for their valued comments, and Karl Dawson from Amazon.com for the inspiring discussion on potential use cases.

References

Brandenburger, A. & Nalebuff, B. (1996). *Co-opetition* (1st ed.). New York: Doubleday.

Buysse, F. & Jonnaert, T. (2012). Het ontwikkelen van een MRP(II) toepassing op basis van het REA referentiemodel. Master thesis, Ghent University, Ghent. Available at: http://lib.ugent.be/fulltxt/RUG01/001/893/296/RUG01-001893296_2012_0001_AC.pdf.

Colmerauer, A. & Roussel, P. (1993). *The Birth of Prolog*. Paper presented at the Second ACM SIGPLAN Conference on History of Programming Languages, Cambridge, Massachusetts, USA.

Dunn, C.L., Gerard, G.J. & Grabski, S.V. (2016). Resources-Events-Agents design eory: A revolutionary approach to enterprise system design. *Communications of the Association for Information System*, *38*(5), 554–595. doi:10.17705/1CAIS.03829.

Fisher-Pauzenberger, C. & Schwaiger, W.S.A. (2017). The OntoREA accounting model: Ontology-based modeling of the accounting domain. *Complex Systems Informatics and Modeling Quarterly*, *54*(11), 20–37. doi:10.7250/csimq.2017-11.02.

Gailly, F. & Geerts, G.L. (2013). Ontology-driven business rule specification. *Journal of Information Systems*, *27*(1), 79–104. doi:10.2308/isys-50428.

Gailly, F., Laurier, W. & Poels, G. (2008). Positioning and formalizing the REA enterprise ontology. *Journal of Information Systems*, *22*(2), 219–248. doi:10.2308/jis.2008.22.2.219.

Gailly, F. & Poels, G. (2009). Using the REA ontology to create interoperability between E-collaboration modeling standards. In P. Eck, J. Gordijn and R. Wieringa (Eds.), *Advanced Information Systems Engineering* (Vol. 5565, pp. 395–409). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-02144-2_32.

Geerts, G.L. (2011). A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems*, *12*(2), 142–151. doi:10.1016/j.accinf.2011.02.004.

Geerts, G.L. & McCarthy, W.E. (2000a). Augmented intensional reasoning in knowledge-based accounting systems. *Journal of Information Systems*, *14*(2), 127–150. doi:10.2308/jis.2000.14.2.127.

Geerts, G.L. & McCarthy, W.E. (2000b). *The Ontological Foundation of the REA Enterprise Information Systems*. Available at: http://www.msu.edu/user/mccarthy4/Alabama.doc.

Geerts, G.L. & McCarthy, W.E. (2002). An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems*, *3*(1), 1–16. doi:10.1016/S1467-0895(01)00020-3.

Geerts, G.L. & McCarthy, W.E. (2005). *The Ontological Foundation of REA Enterprise Information Systems*. Newark: University of Delaware. East Lansing: Michigan State University. Available at: http://www.msu.edu/user/mccarth4/Tulane.doc.

Geerts, G.L. & McCarthy, W.E. (2006). Policy-level specifications in REA enterprise information systems. *Journal of Information Systems*, *20*(2), 37–63. doi:10.2308/jis.2006.20.2.37.

Gordijn, J. (2002). *e3-Value in a Nutshell*. Amsterdam: Vrije Universiteit.

Guizzardi, G. (2014). Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In E. Yu, G. Dobbie, M. Jarke and S. Purao (Eds.), *Proceedings of 33rd International Conference on Conceptual Modeling, ER 2014*, Atlanta, GA, USA, October 27–29, 2014 (pp. 27–29). Cham: Springer.

Haugen, R. (2007). Beyond the Enterprise: Taking REA to Higher Levels.

Hevner, A. & Chatterjee, S. (2010). *Design Science Research in Information Systems Design Research in Information Systems: Theory and Practice* (pp. 9–22). Boston, MA: Springer. doi:10.1007/978-1-4419-5653-8_2.

Hruby, P. & Kiehn, J. (2006). *Revised Classification of the Enterprise Information Architecture Elements*. Paper presented at the 3rd International Conference on Enterprise Systems and Accounting (ICESAcc'06), Santorini Island, Greece. Available at: http://www.itu.dk/people/hessellund/REA2006/papers/HrubyKiehn.pdf.

Hruby, P., Kiehn, J. & Scheller, C.V. (2006). *Model-Driven Design Using Business Patterns*. Berlin: Springer.

ISO/IEC (2007). *Information technology – Business Operational View Part 4: Business transaction scenario – Accounting and economic ontology*. ISO/IEC FDIS 15944-4: 2007(E).

Ito, S. & Vymětal, D. (2013). The formal REA model at the operational level. *Applied Ontology*, *8*(4), 275–300. doi:10.3233/AO-140129.

Jaquet, M. (2006). A property driven approach towards describing semantics of REA entities. In *Proceedings of the 2nd International REA Technology Workshop*.

Kaneko, M. & Kline, J.J. (2008). Inductive game theory: A basic scenario. *Journal of Mathematical Economics*, *44*(12), 1332–1363. doi:10.1016/j.jmateco.2008.07.009.

Kaneko, M. & Kline, J.J. (2010). Two dialogues on epistemic logics and inductive game theory. In *Advances in Mathematics Research* (Vol. 12, pp. 199–238). New York: Nova Science Publishers.

Laurier, W. & Poels, G. (2012). Track and trace future, present, and past product and money flows with a Resource-Event-Agent model. *Information Systems Management*, *29*(2), 123–136. doi:10.1080/10580530.2012.662102.

Laurier, W. & Poels, G. (2014). An enterprise ontology based conceptual modeling grammar for representing value chain and supply chain scripts. *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*, *2*(1), 18–35. doi:10.4018/ijcssa.2014010102.

Letaifa, S.B. (2014). The uneasy transition from supply chains to ecosystems: The value-creation/value-capture dilemma. *Management Decision*, *52*(2), 278–295. doi:10.1108/MD-06-2013-0329.

McCarthy, W.E. (1979). An entity-relationship view of accounting models. *Accounting Review*, *54*(4), 667–686.

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

McCarthy, W.E. (2003). The REA modeling approach to teaching accounting information systems. *Issues in Accounting Education*, *18*(4), 427–441. doi:10.2308/iace.2003.18.4.427.

McCarthy, W.E. (Producer) (2017). An REA model of an economic exchange [MS Powerpoint presentation]. Available at: https://msu.edu/user/mccarth4/cookie--elmo--basic%20REA.ppt.

McCarthy, W.E., Geerts, G.L. & Gal, G. (2016a). *Congruent en Meronymic Constellations in the REA Ontology*. Paper presented at the 10th International Workshop on Value Modeling and Business Ontologies, Trento, Italy. Available at: http://www.loa.istc.cnr.it/vmbo2016/wp-content/uploads/2016/02/VMBO2016_paper_15.pdf.

McCarthy, W.E., Geerts, G.L. & Gal, G. (2016b). *The Economic Structures of Exchanges vs. Conversions in the REA Enterprise Ontology*. Paper presented at the Understanding the Notion of Value in the Service Economy, Trento, Via S. Croce 77. Available at: http://www.loa.istc.cnr.it/vmbo2016/wp-content/uploads/2016/02/conversion-paper-trento-22-FEB.-embedded-figures.pdf.

NEMO (2015). OntoUML Specification 1.0: da Fonseca, Lucas Bassetti R.

O'Leary, D.E. (2004). On the relationship between REA and SAP. *International Journal of Accounting Information Systems*, *5*(1), 65–81. doi:10.1016/j.accinf.2004.02.004.

OMG (2016). Meta Object Facility. Needham, MA 02494, USA: Object Management Group.

Sowa, J.F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove: Brooks/Cole.

Steiner, J. & Baker, J. (2016). Blockchain: The solution for transparency in product supply chains. Available at: https://www.provenance.org/whitepaper.

Tan, Y.-H., Niels, B.-A., Klein, S. & Rukanova, B. (Eds.) (2010). *Accelerating Global Supply Chains with IT-Innovation: ITAIDE Tools and Methods*. Berlin: Springer.

Value Flows vf vocabs (2016). Available at: https://valueflo.ws.

Value Network Software (NRP) (2016). Available at: http://mikorizal.org/software.html.

Zdravkovic, J., Zikra, I. & Ilayperuma, T. (2011). An MDA method for service modeling by formalizing REA and open-edi business frameworks with SBVR. In J. Ralyté, I. Mirbel and R. Deneckère (Eds.), *Engineering Methods in the Service-Oriented Context, ME 2011*, IFIP Advances in Information and Communication Technology (Vol. 351, pp. 219–224). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-19997-4_20.