Formal Correctness of Supply Chain Design

Joerg Leukel [a], Vijayan Sugumaran [b,c,*]

[a] University of Hohenheim, Schwerzstr. 35, 70599 Stuttgart, Germany, joerg.leukel@uni-hohenheim.de

[b] Oakland University, 306 Elliott Hall, Rochester, MI 48309, USA, sugumara@oakland.edu

[c] Sogang University, Seoul 121-742, Republic of Korea

[*] Corresponding author, phone +1 (248) 370-2831, fax +1 (248) 370-4275

Abstract. Many companies use supply chain models for designing the flow of goods and services from their suppliers all the way up to the final customers. Over the past 15 years, the Supply Chain Operations Reference Model (SCOR) has become a widespread modeling technique for designing such supply chains and sharing design information with supply chain stakeholders. However, neither the syntax nor the semantics of SCOR are well defined. This limitation has important consequences for its usage: Supply chain models may be ambiguous and their correctness cannot be verified. We address this problem by mapping SCOR supply chains onto graphs and formalize the semantics of SCOR. The mapping is driven by constructs from the supply chain management literature. The proposed artifact is a supply chain grammar, which we apply to a set of SCOR models taken from industry sources. We show the grammar's usefulness by verifying the correctness of these models using analytical techniques.

Highlights:

- We deduce a grammar of the SCOR modeling technique from SCM literature.
- We propose correctness properties for SCOR-based designs.
- We demonstrate the usefulness of the grammar for detecting errors in existing models.

Keywords: conceptual modeling, process modeling, supply chain management, supply chain design, model verification, SCOR

1 Introduction

Supply chain design is a critical business problem. For many industries, supply chains have become an important focus for competitive advantage. With the increasing global division of labor, the performance of a single company depends more and more on its ability to maintain effective and efficient relationships with its suppliers and customers. Thus, managerial decisions are moving from an organizational scale to a supply chain scale [20]. Supply chain design is the task of determining the basic, long-term structure of the supply chain by defining its elements, objectives, locations, and key organizations [37]. The role of Information Systems (IS) to support this task has recently been the subject of inquiry.

In general, supply chain design faces two difficulties. First, the design space contains a vast number of alternatives, which makes it hard for designers to evaluate and select the best alternative. Second, designing a supply chain incorporates stakeholders from the supply and demand side, which requires sharing and understanding design information by various parties. These two difficulties can be mitigated through reference models that: (1) restrict the design space by providing core constructs that can be configured under certain design constraints, and (2) define a common terminology for sharing designs across organizations. Supply chain management (SCM) has adopted this idea in the form of the *Supply Chain Operations Reference Model* (SCOR) [34, 35]. Over the past 15 years, SCOR has become a widespread modeling technique for supply chain design. It is promoted by a stellar group of firms from various industries and can be regarded as a best practice. Research has made use of SCOR for designing both descriptive and analytical methods for various supply chain problems, in particular, performance management [23, 41], configuration [32], and market-based balancing of demand and supply [27].

The main disadvantage of SCOR is that neither its syntax nor its semantics is well defined. A formal specification of SCOR in the form of a grammar does not exist. The modeling technique is only described in a handbook [36], which provides a reference to model elements with simple example models that don't provide much explanation. The lack of well-defined syntax and semantics has severe consequences. If SCOR users interpret the informal description of the technique in different ways, the supply chain models built using SCOR will become ambiguous and potentially error-prone. This practice may result in syntactically incorrect models that cannot be used by any third party. Software vendors who provide SCOR modeling tools are in danger of implementing the technique incorrectly. Ultimately, the two objectives of reducing the design space and enabling cross-organizational information sharing cannot be met.

Incorrect supply chain models affect the managerial use of these models. We briefly discuss the problems resulting from incorrect models by referring to the three use case of the SCOR technique [36]:

- *Supply chain description* aims at providing an unambiguous description of an actual or planned supply chain for parties that are interested or involved in this supply chain. Incorrect design manifests in configuring the constructs of the SCOR technique falsely, for example, invalid linkage of constructs or missing constructs. If these deficits cannot be detected and repaired, the description is only understandable by the designer and the individuals that share the designer's interpretation. Hence, the model is limited to a small group and does not extend to all the supply chain stakeholders.
- *Supply chain measurement* is concerned with measuring the performance of connected activities and

the entire supply chain. For this purpose, the technique provides a standard set of metrics (e.g. cycle time, cost, flexibility) and standard formulae for analyses, which rely on correct models as outlined in the supply chain description. For incorrect models, the aggregation process would yield either incorrect or no results. Hence, the supply chain performance cannot be correctly measured.

– *Supply chain evaluation* is the task of assessing different designs and selecting the best configuration with regard to certain criteria. These criteria include metrics as defined by the SCOR technique. Evaluation is an iterative process of design (i.e., creating alternative models) and metrics-based measurement. If the measurement yields incorrect or no result for at least one model, then the evaluation will also become incorrect (by comparing configurations that differ due to the interpretation of the technique) or incomplete and not feasible (due to missing data).

Adding a formal specification to SCOR is non-trivial, unless we are able to get this information from SCOR's inventors or at least articulate their interpretation explicitly. However, SCOR was invented by a dynamic group of individuals who worked over a long period in a more or less informal organization. Hence, it is difficult to elicit this information from this group. What we need is a supply chain grammar that consists of constructs for supply chain design and rules that specify allowable combinations of these constructs. There are two basic approaches for defining this grammar: deduction and induction. Grammar deduction defines constructs and rules by analyzing relevant theories and axioms. Grammar induction learns constructs and rules from a set of observations – here, the SCOR supply chain models. The latter's precision, however, is negatively affected by the share of incorrect models in the set of observations.

Current solutions fall into the category of grammar deduction. However, no research endeavor has yet used the existing body of knowledge from SCM research for deduction. Instead, the main source of deduction is the informal description of SCOR, which is then interpreted by the respective researcher. The disadvantage of these approaches is that the deduction is not made explicit to allow for reproducibility.

We address the problem of the lack of explicit definition of SCOR by mapping SCOR supply chains onto directed graphs and formalizing the syntax and semantics. The mapping is a deduction process supported by the constructs from the SCM literature. These constructs enable us to enrich SCOR with additional constraints that have a strong theoretical underpinning. Thus, the objectives of this research are to: (1) develop the syntax and semantics of SCOR in the form of a supply chain grammar that allows for assessing the correctness of supply chain design, and (2) apply this artifact to a set of SCOR models to demonstrate its usefulness for model verification. The contributions of this research are the formal specification (grammar) of SCOR and analysis techniques for SCOR-based supply chain design.

The remainder of this paper is organized as follows. In section 2, we briefly introduce the SCOR technique and provide preliminary notions that will be used for enrichment by grammar deduction. In section 3, we discuss the approaches to the correctness of supply chain design and compare our work with the relevant literature. In section 4, we derive specific constraints on supply chain design from the SCM literature and provide the grammar. In section 5, we demonstrate the usefulness of our proposed grammar in verifying the correctness of SCOR models taken from industry sources. Section 6 concludes the paper and outlines some of our future work.

2 Preliminaries

2.1 SCOR Technique

SCOR consists of an intuitive graphical supply chain description language and a set of supply chain metrics that can be associated with supply chain activities. The graphical language is targeted for the business audience, who uses this language for effective communication of supply chain structures at different levels of abstraction. At the strategic level, SCOR provides a modeling technique for primary product flows; the resulting model is called a *SCOR thread diagram*. The designer can then add details to these diagrams by incorporating plan processes (information flow), secondary product flows (return of products to the supplier), and describing more fine-grained activities associated with the primary product flow, e.g., receiving orders, packaging, and routing shipments. These activities can be configured from a reference set of several hundred so called process elements. In the following, we consider only primary product flows, since this level represents the strategic configuration of supply chains.

A thread diagram shows the flow of products (including tangible goods and services) as a chain of linked activities. An example diagram is shown in Fig. 1. The technique provides the following elements:

– *Process* is an activity of either sourcing, manufacturing, or delivering a product (symbol: arrow-shaped rectangle). The symbols can have different colors to signify the type of activity; however, the color scheme is not precisely defined in the SCOR technique.
– *Product flow* represents the transfer of a product from one process to another (symbol: arrow).
– *Actor* is an organizational entity that executes one or more processes (symbol: label of process).
– *Tier* reflects the level of involvement of actors when considering the entire supply chain. Tiers arrange actors from left to right (symbol: vertical swim lane).

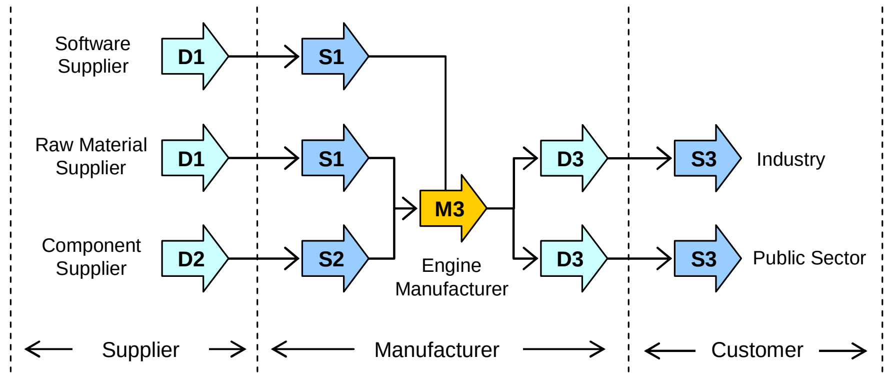

Fig. 1. Example SCOR thread diagram.

If a tier contains only one actor, then it is sufficient to add the actor label to only one process (instead of labeling all the processes). For example, in Fig. 1, the M3 process as well as all the other processes in this tier are executed by the “Engine Manufacturer” actor.

SCOR differentiates processes for primary product flows by the degree of customization (product specificity): (1) stocked products, (2) make-to-order products being manufactured for a specific customer order, and (3) engineer-to-order products being designed and manufactured to a specific customer requirement. This differentiation is then applied to all the processes of sourcing (S), manufacturing (M), and delivering (D) products. Each process is thus encoded by a 2-character code, which denotes the process category, e.g., D2 for delivery of make-to-order products. This differentiation is part of SCOR since 1997 (version 2.0) and was slightly modified in 2003 (version 6.0) by including the D4 process category for the delivery of retail products. This product specificity exists only for delivery processes.

2.2 Preliminary notions

We define the basic notions that formally capture the main elements of the SCOR technique for thread diagrams (as introduced in section 2.1). These notions will serve as the baseline for adding constraints on correct diagrams in the succeeding sections. They are minimal in the sense that we avoid making assumptions about the technique that may not be justified by relevant theory.

Definition 1a (SCOR thread diagram). A SCOR thread diagram is a directed graph $TD=(P, F, A, T, PC, PA, PT)$ where:

– P is a finite set of processes $p \in P$,
– F is a finite set of product flows $f \in F$ with $F \subseteq P \times P$,
– A is a finite set of actors $a \in A$,
– T is a finite set of tiers $t \in T$,
– PC is a function which maps each process onto a process category with $PC{:}P \rightarrow \{S1, S2, S3, M1, M2, M3, D1, D2, D3, D4\}$,
– PA is a function which maps each process onto an actor with $PA{:}P \rightarrow A$,
– PT is a function which maps each process onto a tier with $PT{:}P \rightarrow T$.

Using this definition, the thread diagram shown in Fig. 1 can be formally defined by all components of TD. For instance, the diagram contains eleven processes, which must be numbered, e.g., $P=\{p1, p2, .., p11\}$. Each arrow in the diagram denotes a product flow, e.g., $F=\{(p1, p2), (p3, p4), ..\}$. The processes $p1$ and $p3$ belong to the process category D1, thus $PC=\{(p1 \rightarrow D1), (p3 \rightarrow D1), ..\}$, and are contained in the Supplier tier, thus $PT=\{(p1 \rightarrow Supplier), (p3 \rightarrow Supplier), ..\}$.

3 Related Work

3.1 Supply Chain Design

Two dimensions are constituent to the task of supply chain design. The *process dimension* relates to answering the question which activities must be performed by the designer in what order to produce the supply chain model. SCM research yields a plethora of analytical models and optimization methods [6, 37]. The *result dimension* relates to the constructs, i.e., the conceptual vocabulary of the problem domain, and the formalisms used for articulating these constructs. Correctness of supply chain design is the ultimate concern of the result dimension. The SCOR technique addresses the result dimension.

The SCOR technique has been documented in a handbook [36], which provides definitions of all the aforementioned elements. The handbook, however, falls short of providing a formal specification in the form of a grammar that is unambiguous and free of interpretation. It also provides very little information on how to link processes under consideration for process categories, actors, and tiers. The lack of a formal grammar has led to the conclusion that SCOR is less useful for analyzing supply chains through quantitative means [4, 19]. *Arns et al.* argue for reducing the role of SCOR to description, whereas all model analysis tasks would require a more capable language providing a well-defined execution semantics [4].

The literature yields several approaches for amending SCOR with a grammar. These approaches differ in the specification language used and the way they augment the SCOR technique.

Becker et al. [7] choose the Entity-Relationship-Model (ERM) for specifying a meta-model of SCOR. The rationale is that the ERM is adequate for capturing the SCOR constructs as well as the resulting meta-model can easily be transformed into a database schema for model storage and retrieval. An interesting aspect of the proposed meta-model is the *process category condition*, which enforces that processes of a certain category must be connected with processes of another category (e.g., "make-to-stock" is proceeded by "deliver make-to-stock"). These conditions may help in assessing the correctness of supply chain designs, though they have not been made explicit, but were illustrated by an example only. Unfortunately, this research provides little information on how the meta-model was constructed. The authors try to provide convincing arguments for the meta-model and report about a prototype implementation; however, the prototype is not concerned with correctness of supply chain designs.

Millet et al. [26] propose a set of possible relationships between process categories denoted as rules. They assume that designing a supply chain implicates a certain body of rules. The explication process for these rules is, however, not described. The rules are not formally specified. In addition, the rule set is at best incomplete, since it misses product flows between different *Make* processes.

SCOntology is an ontological approach to formalizing the SCOR technique [15]. The rationale for using the Web Ontology Language (OWL) [38] is that OWL provides more expressiveness for defining concepts and their interrelations than ERM. The scope of the proposed SCOR ontology is defined by the so called competency questions. Insofar as these competency questions are concerned, a justification for them is not provided by these authors. In addition, the ontology is described using graphical means only, and thus lacks axioms. A brief case study is supplied to demonstrate the validity of the proposal.

The ontological approach by *Sakka et al.* [33] does not interpret the textual descriptions contained in the SCOR handbook, but starts with the meta-model that is implemented by the software tool ARIS/SCOR. This meta-model, which is specified in ERM, is then mapped onto an OWL ontology. The advantage is that this approach preserves all constructs and rules contained in the baseline meta-model. However, *Sakka et al.* admit that the designers of ARIS/SCOR were interpreting the SCOR handbook and thus made assumptions, which are unknown to the tool user.

The most comprehensive ontological approach is SCOR-FULL [42], which goes beyond SCOR thread diagrams by including SCOR metrics and input/output information. This ontology is aimed at the semantic interoperability of supply chain designs, without paying attention to correctness. Similar to [15], the rationale of this ontology is limited to answering an initial set of competency questions.

Our approach differs from existing research as follows. First, by grounding the grammar deduction on constructs and rules from the SCM literature, we aim at reducing the risk of interpreting the SCOR handbook in a subjective way. This risk may lead to a grammar that contradicts the insights from SCM research. Second, our research is informed by the use of ontology languages, but the proposed grammar is independent from the usage of a particular ontology language. Third, we address the formal correctness of supply chain design and aim at providing specific means for assessing this property.

3.2 Business Process Management

Much progress has been made on developing methodologies for assessing and preserving the correctness of business process models. Since supply chain design also describes the business activities carried out (those for supplying products from suppliers to the final customers), we review contributions from the business process management (BPM) literature that may be beneficial for supply chain design.

Process verification determines whether a process model complies with a specified structure and behavior. Verification depends foremost on the existence of formal semantics of the process description language used. Many widely used languages for business process modeling lack formal semantics, e.g., Event-driven Process Chains (EPC) [1] and the Business Process Modeling Notation (BPMN) [12]. Therefore, BPM research has investigated their mapping to more powerful modeling techniques, which also supply analysis techniques for correctness properties. Of particular significance are the works that adopt analysis techniques using Petri-nets. *Mendling et al.* [24] propose a Petri-net approach for detecting errors in EPCs and apply it to a set of real-world EPCs taken from the SAP Reference Model. They show that these EPCs are error-prone, because the model designers did not conform to the EPC semantics.

Supply chain design shows similarities to business process modeling. However, we need to be aware of important differences between product flow and control flow. The SCOR technique does not make the semantics of the product flow construct explicit. For instance, let us consider the diagram given in Fig. 1. The process of category M3 has three ingoing arcs (sourcing) and two outgoing arcs (delivery). Does it mean that this process transforms all three ingoing products into the two outgoing products? Or can this process be executed if at least one ingoing arc is activated? The answer cannot be given, because the execution semantics is unclear. We need to keep in mind that supply chain design is concerned with defining the structure of supply chains, not their behavior. For instance, SCOR lacks logical connectors, which are common in control flow descriptions.

Arns et al. [4] combine the SCOR technique with a business process language as follows: They propose using a custom notation called ProC/B; the advantage is that ProC/B models can be translated into Petri-nets, which allow for analyzing behavioral properties to a great extent. Activities in such models are encoded as SCOR process categories. The only contribution of SCOR is the vocabulary for activities. This approach results in two modeling phases: First, a SCOR thread diagram is created. Then, its activities and flow relationships will be used for creating the ProC/B model. The disadvantage is that the second phase requires decisions to be made about the control flow, but this information is *not* supplied by the diagram from the first phase (no execution semantics of process categories).

Grammar was first used as a metaphor for describing business processes in organizational studies and has since then spread to BPM research. *Pentland* [30] proposed a systematic approach for developing models of organizational processes by adopting the grammar metaphor. This approach was then extended by *Lee et al.* [22] for using process grammar for constraining the design space of business processes. The objective of constraining also holds for the SCOR reference model, which should help creating supply chains by referring to valid supply chain structures that are supplied by only SCOR. Our research is influenced by the grammar metaphor. Unlike *Lee et al.*, who use production rules as constructs for context-free grammars, we employ graph algebra that allows asserting constraints on valid graphs.

Surprisingly, *Pentland* also proposes in one of his early works [29] the process grammar approach for supply chains. He argues that supply chains are well-suited for grammars because of their repetitive constituents, high degree of modularization, and centering on product flows, which all result in a rather limited set of supply chain constructs. He defines a supply chain grammar of seven constructs (activities) and nine “tentative” supply chain patterns. This grammar was motivated by experiences gained from three case studies. However, its expressiveness is severely limited, e.g., patterns are sequences of activities only, with no further constraints on valid linkages as well as no formalization of the grammar.

4 Grammar Deduction

In this section, we describe the process of deducing the grammar for SCOR thread diagrams from the SCM literature. For each element of the basic model (as defined in section 2.2), we add theoretical findings that further constrain the supply chain design.

4.1 Supply Chain Literature

Supply chain is the unit of analysis of SCM, which over the past 30 years has evolved from a field in operations management into a discipline of management research [10]. The recent past has seen an increasing debate about the state of SCM as a discipline [8] as well as a call for theory building [9, 20].

For our purpose of grammar deduction, it is important to state that the majority of SCM researchers use existing theories from other disciplines to explain different aspects of the supply chain [8]. SCM is inter-disciplinary, which encompasses logistics, purchasing, operations management, marketing, strategy, and others. Therefore, the grammar deduction will include both supply chain body of knowledge and its antecedents. The scope of the body of knowledge is constrained by supply chain design as defined in section 1, i.e., determining the long-term, basic structure of supply chain activities, which are connected by product flows. We are thus interested in constructs that are commonly used in SCM research to describe these structures. Due to the absence of a single “original” theory of supply chain, we extract relevant constructs from seminal SCM works in the related fields. These constructs are higher order abstractions that can be used in supply chain models, specifically in supply chain design (descriptive nature of constructs).

4.2 Deduction from SCM Frameworks

Descriptive constructs can be found in research that condenses the terminology used in SCM and frames the main issues into conceptual frameworks. These works represent the effort to consolidate the abundant but disparate literature [11, 14, 25]. An important contribution stems from *Lambert* and *Cooper* [20], whose framework consists of three main elements: *Supply chain network structure* includes the firms and the links between these firms. *Supply chain business processes* move the product from supplier to the customer, and *SCM components* are managerial variables that are used to integrate the business processes.

There are strong ties between these elements and supply chain design as discussed below. The *Network structure* answers the following question: who are the relevant supply chain members (in SCOR: actors) with whom to link the processes? Relevance is determined by examining whether an actor carries out value-adding activities to produce a specific output for a customer or market. Therefore, supporting actors, who for example only provide resources to other actors, are not the unit of analysis. We deduce that: (1) every actor's processes must be linked to at least one process, and (2) no actor exists without such a process. Using formal notion, we represent these two constraints C1 and C2 as follows:

ID	Description	Formal definition
C1	Each process *p* has at least one incoming or outgoing product flow *f*.	For each $p \in P$ there exists at least one $f \in F$ with $f=(m, p)$ or $f=(p, m)$, and $m \in P$, $m \neq p$.
C2	Each actor *a* carries out at least one process *p*.	For each $a \in A$ there exists at least one $p \in P$ with $PA(p)=a$.

The framework further defines structural dimensions. The *Horizontal structure* introduces the construct of *tier*, which is defined as the set of actors sharing the same horizontal position within the end points of the supply chain. Thus, all tiers can be arranged in graphical models with no overlaps. It has become common practice to place the final customer as the right most tier; this holds also true for the SCOR technique. When referring to a particular tier, all the tiers to its left are called upstream and those to the right are called downstream. For expressing the horizontal segmentation, we first introduce a numbering scheme for tiers by extending the definition of the thread diagram (definition 1b). The function *N* assigns an integer to each tier; the tier number ranges from 1 to $|T|$ for the total number of tiers.

Definition 1b (SCOR thread diagram). A SCOR thread diagram is a directed graph $TD=(P, F, A, T, PC, PA, PT, N)$ where *N* is a function that defines the order of tiers, $N(T):=\{1,..,|T|\}$.

Using this definition, the formal representation of the diagram shown in Fig. 1 can be enriched as follows. The diagram is made of three tiers $T=(\{Supplier, Manufacturer, Customer)\}$, which are arranged from left to right. Therefore, we add $N=(\{Supplier \rightarrow 3, Manufacturer \rightarrow 2, Customer \rightarrow 1)\}$. Then, we add the constraint C3, which prevents the existence of "empty" tiers.

ID	Description	Formal definition
C3	Each tier *t* contains at least one process *p*.	For each $t \in T$ there exists at least one $p \in P$ with $PT(p)=t$.

The *Vertical structure* refers to the number of actors within each tier. Depending on the number, a tier may be characterized as rather narrow or wide. The narrowest tier is a tier that contains only one actor and process; this requirement is already captured by C3 and C2.

The *Horizontal position* describes the actor's closeness to the point of origin and the distance from the point of consumption of the supply chain. The point of origin is the tier for which no further supplier exists (tier denoted by $N=|T|$). The point of consumption is the tier in which no further value is added, but the product is consumed (tier denoted by $N=1$). For SCOR, we deduce that every thread diagram has: (1) one origin tier that includes at least one process with no incoming product flow, and (2) one consumption tier that includes at least one *Source* or *Deliver* process with no outgoing product flow. To be able to state constraints on the number of incoming and outgoing flows, we first need to introduce the notion of predecessor and successor processes. For a given process *p*, we denote its predecessor processes by $\bullet p$ and its successor processes by $p\bullet$ (definition 2). For instance, in Fig. 1, the process of D1 (denoted by *p1*) in the Supplier tier has no incoming product flow, thus $\bullet p1=\{\emptyset\}$, and one outgoing flow to the process denoted by *p2*, thus $p1\bullet=\{p2\}$.

Definition 2 (predecessors, successors). For $p \in P$, $\bullet p=\{m|(m, p) \in F\}$ denotes the set of predecessors of *p*, with $m \in P$, and $p\bullet=\{m|(p, m) \in F\}$ denotes the set of successors of *p*, with $m \in P$.

Then we are able to define the constraints for the origin tier (C4) and the consumption tier (C5).

ID	**Description**	**Formal definition**
C4	The left most tier *t* contains at least one process *p* with no incoming product flow *f*.	For $t \in T$ with $N(t)=\|T\|$ there exists at least one $p \in P$ with $PT(p)=t \wedge \|{\bullet}p\|=0$.
C5	The right most tier *t* contains at least one process *p* of *Source* or *Deliver* with no outgoing product flow *f*.	For $t \in T$ with $N(t)=1$ there exists at least one $p \in P$ with $PT(p)=1 \wedge PC(p) \in \{S1, S2, S3, D1, D2, D3, D4\} \wedge \|{\bullet}p\|=0$.

The existence of origin and consumption tiers implies that another tier, which comprises the focal firm, lies between these tiers. The definition by *Mentzer et al.* makes this implication explicit by defining supply chain "as a set of three or more entities ... directly involved in the upstream and downstream flows" [25]. Therefore, the number of tiers, as well as the actors is at least three. Thus, C6 and C7 are the cardinality constraints on the tiers and actors.

ID	**Description**	**Formal definition**
C6	Each thread diagram *TD* consists of at least three tiers (*t*).	For any *TD*: $\|T\| \geq 3$.
C7	Each thread diagram *TD* consists of at least three actors (*a*).	For any *TD*: $\|A\| \geq 3$.

With respect to *Supply chain business processes*, the framework in [20] provides taxonomies of processes and process links. Both taxonomies are, however, more detailed and broader than SCOR. In particular, they consider also information flows. The SCOR categories of *Source*, *Make*, and *Deliver* map to those of procurement, manufacturing flow management, and demand management. Product specificity is not found in the framework.

The set of nine *SCM components* spans a wide range of managerial variables by which activities across the supply chain are integrated. They address physical/technical, as well as behavioral variables. Due to the framework's abstract nature, we can only deduce that supply chain design is one SCM component (under the term "product flow facility structure", which determines the "network structure for sourcing, manufacturing, and distributing across the supply chain" [20]).

Referring to the main elements of the SCOR technique as provided in definition 1a (section 2.2), we found corresponding descriptive constructs in these frameworks, which also define the terminology of SCM. We mapped the framework's constructs to SCOR and enriched the definition to some extent. To further underpin SCOR, we need to study SCM and its antecedents for constructs and findings about processes, product flows, and their interdependencies along the supply chain.

4.3 Processes

SCM research yields a variety of process classifications, which differ in the level of detail and coverage (i.e., flow of product, information, and resources). For instance, there are seven classifications provided in [29], eight in [20], and ten in [25]. When breaking these classifications down to the activities that directly modify the product with regard to its structure, location, or market, the resulting activities can be grouped into three basic activities of any firm: (1) buying resources from other firms, (2) combining and converting these resources into products, and (3) selling these products to customers. These activities are also constituent to the SCOR technique under the terminology *source*, *make*, and *deliver*, respectively.

They correspond to the decision areas that represent the operations management origin of SCM [6, 8].

Product specificity is the second determinant of SCOR processes. The rationale is that stocked, make-to-order, and engineer-to-order products each require different operational strategies [13]. This determinant can be traced back to manufacturing management, which uses these types of product specificity to describe *when* a particular product is linked to a particular customer order:

– *Make-to-stock (stocked product):* a particular product is *not* linked to a specific customer order, but the order can be fulfilled by any product instance from stock.
– *Make-to-order:* a particular product is linked to a specific order at the time of order.
– *Engineer-to-order, design-to-order:* a particular product is linked to a specific order at the time the collaborative engineering starts.

Operations management has emphasized that dedicated methods are required for these product specificities [16, 18]. Moreover, specificity is an important determinant for deciding about the decoupling point, i.e., the tier, where the linkage between a particular product and the order is established [28].

The dependencies between product specificity and supply chain tier must be considered in using the SCOR technique. Prior to adding further constraints, we must define the two determinants of the processes – activity type and product specificity – formally. We add these classifications by defining the SCOR technique in definition 3 and using SCOR's terminology (management process for activity type). This definition contains three sets for process categories, management processes, and product specificities and two functions that map process categories to management processes and product specificities.

Definition 3 (SCOR thread diagram technique). The SCOR thread diagram technique is a tuple $TDT=(C, M, CM, S, CS)$, where:

– C is the set of process categories $c\in C$, with $C=\{S1, S2, S3, M1, M2, M3, D1, D2, D3, D4\}$,
– M is the set of management processes $m\in M$, with $M=\{Source, Make, Deliver\}$,
– CM is a function which maps each process category $c\in C$ onto a management process $m\in M$, with $CM=\{S1\rightarrow Source, S2\rightarrow Source, S3\rightarrow Source, M1\rightarrow Make, M2\rightarrow Make, M3\rightarrow Make, D1\rightarrow Deliver, D2\rightarrow Deliver, D3\rightarrow Deliver, D4\rightarrow Deliver\}$,
– S is the set of product specificities $s\in S$, with $S=\{Stock, Order, Engineer, Retail\}$,
– CS is a function which maps each process category $c\in C$ onto a product specificity $s\in S$, with $CS=\{S1\rightarrow Stock, S2\rightarrow Order, S3\rightarrow Engineer, M1\rightarrow Stock, M2\rightarrow Order, M3\rightarrow Engineer, D1\rightarrow Stock, D2\rightarrow Order, D3\rightarrow Engineer, D4\rightarrow Retail\}$.

4.4 Product Flows

Product flows are of paramount importance to supply chain design, since they implement the linkages between actors and their processes. In this section, we clarify the semantics of product flow, which is regarded as a critical shortcoming of the SCOR technique. We define the semantics by asserting constraints on the product flow relation, i.e., on $F\subseteq P\times P$.

4.4.1 Product Flows Within Actors

The rationale for product flows is that each flow must indicate that the preceding process has added value to the product, i.e., each process adds value to the product up to the process of consumption by the final customer. SCM is also concerned with the value-adding activities that take place within an actor. Thus, the actor is not perceived as a “black box” of input/output relations, but regarded as a set of value-adding activities. For this reason, the SCM frameworks contain classifications of such activities [20, 25].

In SCOR, the three management processes of *Deliver*, *Make*, and *Source* span a set of nine potential product flows as shown in Fig. 2. However, only the downstream flows indicate added value; these flows are *Source* to *Deliver*, *Source* to *Make*, and *Make* to *Deliver*. In addition, manufacturing is often a complex activity that adds value in several steps. Therefore, processes of *Make* can be connected with other *Make* processes. The literature denotes these systems as multi-stage manufacturing systems [21].

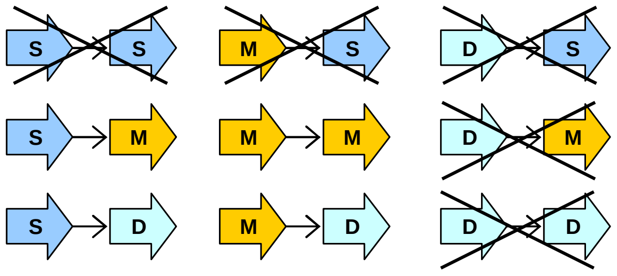

Fig. 2. Possible product flows inside an actor by management process.

Next, we formulate the constraint C8 for capturing the possible product flows inside an actor.

ID	Description	Formal definition
C8	Each flow inside an actor is between one of the following management processes: *Source* to *Deliver*, *Source* to *Make*, *Make* to *Make*, or *Make* to *Deliver*.	For each $f=(p_i, p_j)$ with $PA(p_i)=PA(p_j)$: $(CM(PC(p_i))=\{Source\} \wedge CM(PC(p_j))=\{Deliver\}) \vee$ $(CM(PC(p_i))=\{Source\} \wedge CM(PC(p_j))=\{Make\}) \vee$ $(CM(PC(p_i))=\{Make\} \wedge CM(PC(p_j))=\{Make\}) \vee$ $(CM(PC(p_i))=\{Make\} \wedge CM(PC(p_j))=\{Deliver\})$

We define the product flow as $f=(p_i, p_j)$. First, we require that the two processes p_i and p_j belong to the same actor, i.e., by referring to the function *PA*. Second, we enumerate the four allowed combinations by using the function *PC* to each process (which yields the process category, e.g., *D1*) and applying the function *CM* (which yields the respective management process, e.g., *Deliver*).

4.4.2 Product flows between actors

These flows materialize through the transfer of a product from the supplier’s *Deliver* process to the buyer’s *Source* process. The analysis of the dyadic buyer-supplier relationship is an important antecedent of SCM [2]. We must restrict product flows between actors to buyer-supplier relationships that add value downstream in the supply chain. Fig. 3 illustrates that these flows take place at the interface of two tiers (example on the left hand), but not inside the same tier (example on the right hand).

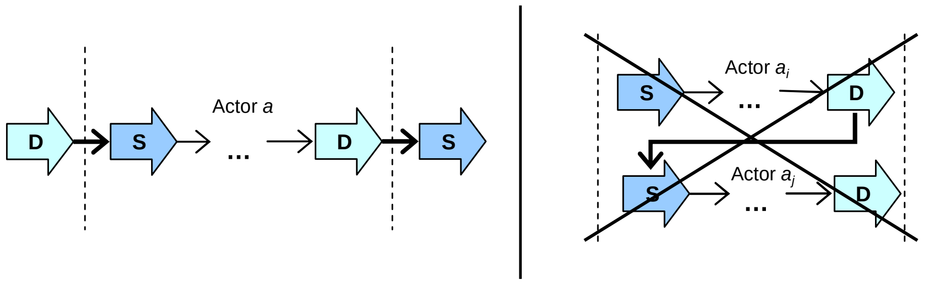

Fig. 3. Possible (left) and forbidden (right) product flows between actors.

In constraint C9, we consider a product flow $f=(p_i, p_j)$, which takes place between two different actors (we use the function *PA* to separate the actors). First, the actors must not only be different, but the actor of process p_i must be the supplier of the actor of process p_j; hence the tier number of p_i must be greater than that of p_j. Finally, we state the buyer-supplier relationship by using the functions *CM* and *PC*.

ID	Description	Formal definition
C9	Each flow between two actors connects a *Deliver* process with a *Source* process and the preceding actor's tier is left from the succeeding actor's tier.	For each $f=(p_i, p_j)$ with $PA(p_i) \neq PA(p_j)$: $N(PT(p_i)) > N(PT(p_j)) \wedge CM(PC(p_i))=\{Deliver\} \wedge CM(PC(p_j))=\{Source\}$

4.4.3 Dependence on Product Specificity

The specificity of a given product does not change along the supply chain. The reason is that specificity is defined (as described in section 4.3) by the time the respective product is linked to a particular customer order. Once a product is linked to an order, the linkage cannot be broken by downstream processes [28], unless the product is transformed through manufacturing into another product.

First, we look at product flows between two actors as restricted by constraint C9. The source process in the downstream tier is the activity of buying the product from the upstream tier, thus the specificity of both processes must be the same (e.g., buying a make-to-order product is only possible from the deliver process of make-to-order) except for retail products and its respective D4 process. We retrieve the specificity of both linked processes by using the function *CS* and define constraint C10.

ID	Description	Formal definition
C10	Each flow into a *Source* process of S2 or S3 starts at a preceding actor's *Deliver* process of the same product specificity.	For each $f=(p_i, p_j)$ with $PA(p_i) \neq PA(p_j) \wedge (PC(p_j)=\{S2\} \vee PC(p_j)=\{S3\})$: $N(PT(p_i))>N(PT(p_j)) \wedge CM(PC(p_j))=\{Source\} \wedge CS(CM(PC(p_i)))=CS(CM(PC(p_j)))$

It is worth noting that D4 was added to the SCOR technique as a variant of D1 (available since version 6.0). Retail products can be retrieved from either S1 or S2 processes and will then be sold at a retail store, which maintains the D4 process. We include this case into a specific constraint (C11).

Second, we analyze product flows within actors. Flows within actors describe the value-adding activities, which can be more complex in terms of number of processes and flows. In particular, we must pay attention to all four cases mentioned in section 4.4.1 and their interplay.

ID	Description	Formal definition
C11	Each flow into a *Source* process of S1 starts at a preceding actor's *Deliver* process of specificity *Stock* or *Retail*.	For each $f=(p_i, p_j)$ with $PA(p_i) \neq PA(p_j) \wedge PC(p_j)=\{S1\}$: $N(PT(p_i))>N(PT(p_j)) \wedge CM(PC(p_i))=\{Source\} \wedge CS(CM(PC(p_i))) \in \{Stock, Retail\}$

Let us consider the example shown in Fig. 4, which shows that the actor sells two products. The stocked product is bought via a S1 process and then sold via a corresponding D1 process. The make-to-order product results from two subsequent M2 processes, with the first transforming a make-to-order product into an intermediate product, and the second process, combining it with another stocked product into the final product. What we are missing so far is the semantics of the *Make* processes: Manufacturing transforms productive inputs into products of higher value, thus every *Make* process transforms the product. On the contrary, *Source* does not transform the product, but transfers the product to the next process of either *Make* or *Deliver*. Similarly, *Deliver* transfers the product to another tier.

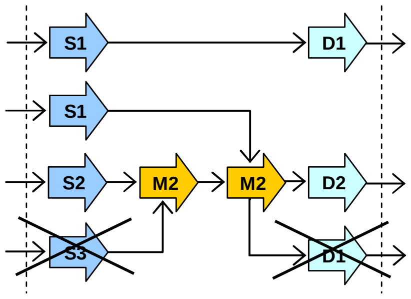

Fig. 4. Example of possible and forbidden product flows within an actor.

The issue of product transformation vs. transfer is closely related to product specificity. We summarize this dependency by analyzing the four cases of product flows within actors.

Source to *Deliver* transfers a product, which will be directly sold to the customer. The delivery process may link the product to a particular customer order, thus specificity can increase from S1 to D2. The S1 process can also transfer the product to D4. These requirements are captured by C12 and C13.

ID	Description	Formal definition
C12	Each flow from *Source* of S2 or S3 (pre) to *Deliver* (suc) connects to processes of the same specificity.	For each f with $f=(pre, suc) \wedge PC(pre) \in \{S2, S3\} \wedge CM(PC(suc))=\{Deliver\}$: $CS(CM(PC(pre)))=CS(CM(PC(suc)))$
C13	Each flow from *Source* of S1 (pre) to *Deliver* (suc) connects to processes of specificity *Stock*, *Order*, or *Retail*.	For each f with $f=(pre, suc) \wedge PC(pre)=\{S1\} \wedge CM(PC(suc))=\{Deliver\}$: $CS(CM(PC(suc))) \in \{Stock, Order, Retail\}$

Source to *Make* transfers a product that will be transformed into another product. The specificity may increase along the supply chain, but not decrease (C14). In the example in Fig. 4, it is forbidden to link the S3 process with the M2 process, because the manufacturing relies on a product specification, but this specification would not be available due to the engineer-to-order product of S3.

Make to *Make* is the product transfer in multi-stage manufacturing. Again, specificity may increase in succeeding stages, but not decrease (C15).

Make to *Deliver* is the transfer to the last process within the actor. The product flow must respect the correspondence of specificities (C16).

ID	**Description**	**Formal definition**
C14	Each flow from *Source* (pre) to *Make* (suc) connects processes of the same or higher specificity.	For each f with $f=(pre, suc)$ $\wedge\ CM(PC(pre))=\{Source\} \wedge CM(PC(suc))=\{Make\}$: $(CS(CM(PC(pre)))=\{Stock\}$ $\wedge\ CS(CM(PC(suc)))\in\{Stock, Order, Engineer\}) \vee$ $(CS(CM(PC(pre)))=\{Order\}$ $\wedge\ CS(CM(PC(suc)))\in\{Order, Engineer\}) \vee$ $(CS(CM(PC(pre)))=\{Engineer\}$ $\wedge\ CS(CM(PC(suc)))=\{Engineer\})$
C15	Each flow from *Make* (pre) to *Make* (suc) connects processes of the same or higher specificity.	For each f with $f=(pre, suc)$ $\wedge\ CM(PC(pre))=\{Make\} \wedge CM(PC(suc))=\{Make\}$: $(CS(CM(PC(\mathrm{pre})))=\{Stock\}$ $\wedge\ CS(CM(PC(suc)))\in\{Stock, Order, Engineer\}) \vee$ $(CS(CM(PC(pre)))=\{Order\}$ $\wedge\ CS(CM(PC(suc)))\in\{Order, Engineer\}) \vee$ $(CS(CM(PC(pre)))=\{Engineer\}$ $\wedge\ CS(CM(PC(suc)))=\{Engineer\})$
C16	Each flow from *Make* (pre) to *Deliver* (suc) connects processes of the same specificity.	For each f with $f=(pre, suc)$ $\wedge\ CM(PC(pre))=\{Make\} \wedge CM(PC(suc))=\{Deliver\}$: $CS(CM(PC(pre)))=CS(CM(suc))$

4.5 Correctness Properties

In this section, we summarize the grammar deduction process by describing the usefulness of each constraint for verifying the correctness of SCOR thread diagrams. Each constraint represents a particular correctness property. Table 1 shows the correctness properties for the constraints C1 through C16.

Table 1. Correctness properties for SCOR thread diagrams.

ID	Correctness Property	Usefulness
C1	Value-adding processes	Determines processes that do not add value to the product.
C2	Value-adding actors	Determines actors that do not add value to the product.
C3	Value-adding tiers	Determines tiers that do not add value to the product.
C4	Origin tier	Determines the existence of the origin tier.
C5	Consumption tier	Determines the existence of the consumption tier.
C6	Minimum tiers	Determines if the diagram lacks tiers.
C7	Minimum actors	Determines if the diagram lacks actors.
C8	Correct product flows inside actors	Determines false connections of processes inside an actor.
C9	Correct product flows between actors	Determines false connections of processes between actors.
C10	Correct flows into the *Source* of S2 and S3 from supplier	Determines false usage of product specificity between actors for S2 and S3.
C11	Correct flows into the *Source* of S1 from supplier	Determines false usage of product specificity between actors for S1.
C12	Correct flows from the *Source* of S2 and S3 to the *Deliver* processes inside actors	Determines false usage of product specificity inside actors for S2 and S3.
C13	Correct flows from the *Source* of S1 to the *Deliver* processes inside actors	Determines false usage of product specificity inside actors for S1.
C14	Correct flows from *Source* to *Make* processes	Determines false usage of product specificity inside actors for *Source* to *Make* processes.
C15	Correct flows from the *Make* to *Make* processes	Determines false usage of product specificity inside actors for *Make* to *Make* processes.
C16	Correct flows from *Make* to *Deliver* processes	Determines false usage of product specificity inside actors for *Make* to *Deliver* processes.

If a particular constraint is violated by a given diagram, then we are able to (1) identify the incorrect elements of the model and (2) interpret the reported problem by referring to the informal description of the respective constraint. For instance, constraint C9 requires that each flow between two actors connects the *Deliver* and the *Source* processes where the preceding actor belongs to the tier on the left of the succeeding actor's tier. If this constraint is breached, we characterize the problem as a false connection of processes between actors. The usefulness of the constraint is that it helps determine such problems.

5 Evaluation

From the perspective of the design science paradigm [17], the artifact that has been developed in this research is the SCOR grammar. The objective of this evaluation is to demonstrate the usefulness of this artifact for assessing the correctness of existing SCOR thread diagrams.

5.1 Evaluation Procedure

We obtained models from the SCOR website (http://supply-chain.org/filemanager/active), which provides both case studies and training material described in either reports or slides. We selected the models based on size (minimum of 10 processes to exclude "toy" models). The selection was documented and the models were stored in their original format. The evaluation set contained 8 models. Table 2 shows the characteristics of these models.

Table 2. Basic characteristics of the sample.

ID	**Source**	**Industry**	**Year**	**|T|**	**|A|**	**|P|**	**|F|**
TD1	Cheng et al.	Construction	2009	5	6	13	11
TD2	GE	Jet engines	2006	5	5	13	11
TD3	KLATencor	Service parts	2006	4	5	12	12
TD4	Schenker	Logistics	2006	5	7	18	17
TD5	Marine Corps	Military deployment	2006	3	7	28	27
TD6	Nortel	Telecommunication	1997	4	6	15	14
TD7	SCOR Primer		1997	5	6	11	9
TD8	SCOR Configuration		1997	4	7	20	20
			MIN:	3	5	11	9
			MAX:	5	7	28	27
			MEAN:	4.38	6.13	16.25	15.13

For representing the selected models in a prototype system, we chose a *Description Logic*-based approach [5]. Description logic (DL) is a family of formalisms for representing knowledge within a domain and is well-suited for reasoning about this knowledge. These formalisms have been adopted successfully for model verification in a number of related areas [3, 31]. For specifying the SCOR grammar, we employ the *Web Ontology Language* (OWL 1.0) [38] and its complementing *Semantic Web Rule Language* (SWRL) [39]. The latter is used for determining the correct and incorrect model elements (instances of the ontology) and adding them to specific classes for each constraint C1 through C16.

Each model was manually mapped to the ontology and the assertions stored in an OWL knowledge base using the *Protégé 3.5* editor and framework. *Protégé* provides full support for both OWL and SWRL, and has become a reference tool for developing ontology-based applications. The mapping procedure was required because the original models are not machine-readable (due to the absence of a data exchange format). Each model was analyzed for correctness by using the built-in reasoner in *Protégé* and checking for each constraint separately. The reasoner populated the specific classes with the correct and incorrect instances.

5.2 Results

The model verification found that seven out of the eight models in the sample contained errors, i.e., violated one or more constraints. Collecting this data required performing the verification for each model separately and then handing the results over to a data analysis tool. The current prototype system is not designed for presenting the verification results directly in the graphical model. Fig. 5 presents a visualization of the verification results for model TD2 by highlighting the erroneous processes and product flows, and identifying the constraints that have been violated (shown in square brackets).

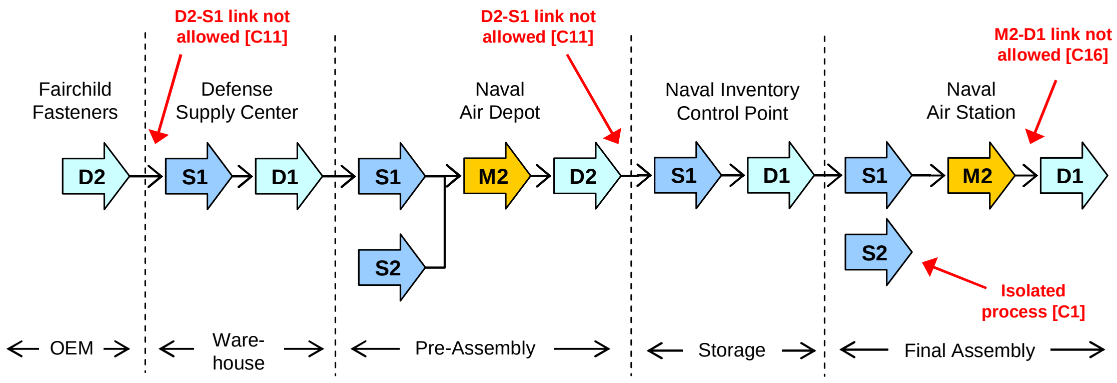

Fig. 5. Model verification results for model TD2 (four errors identified).

We observed several cases in which the designer apparently used a customized version of the SCOR technique. While most of these changes are extensions that introduced new constructs, the model TD3 contains modifications of the original constructs. Let us consider the model TD3 shown in Fig. 6.

First, the diagram arranges tiers vertically and uses a specific symbol for vertical processes between actors of the same tier. This modification, however, does not add information to the underlying formal representation, but changes the visual presentation. Whether this presentation is more intuitive or better suited for communicating supply chain design, remains to be assessed. The second modification introduces compositions of D1 and D4 processes denoted by D1/4. Again, this modification can be mapped to the formal model without losing information. Specifically, each composite process can be replaced by two elementary processes (e.g., D1 and D4), which must then be connected to the respective preceding and succeeding processes of the composite process.

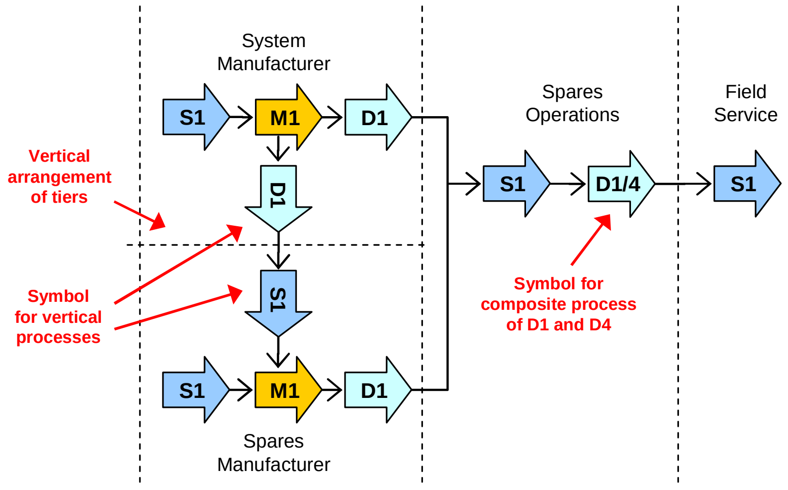
System Manufacturer
S1
M1
D1
D1
S1
S1
M1
D1
Spares Manufacturer
Spares Operations
S1
D1/4
Field Service
S1
Vertical arrangement of tiers
Symbol for vertical processes
Symbol for composite process of D1 and D4

Fig. 6. Customized SCOR technique used in model TD3.

We summarize the verification results in Table 3. The table shows for each model (TD1 through TD8) and constraint (C1 through C16) the findings as follows: ‘Ok’ indicates that the constraint is fulfilled and ‘Error/x’ indicates that x number of model elements breach the respective constraint. For instance, ‘Error/1’ for C1 means that one process is not linked to any other process. If the model contains no element relevant to the constraint, then the constraint is not applicable and we indicate this case in the table by a hyphen. For instance, C15 is concerned with links between *Make* processes; however, no model in the sample contained any link between such processes.

We calculate two metrics for each model. The *process error rate* (PER) represents the relative number of erroneous processes compared to the total number of processes in the model. This metric is affected by constraint C1, which is the only constraint dealing with processes. For instance, TD1 contains one such process out of 13 processes, hence the PER is 7.7%. Similarly, the *flow error rate* (FER) measures the proportion of erroneous product flows and is dependent on C8 through C16. In addition, we define the *constraint error rate* (CER), which signifies the relative number of error-prone models compared to all models for which the constraint is meaningful. For instance, C12 was breached by two out of four models that contain relevant product flows and thus CER is 50.0%.

Table 3. Results of model verification.

	TD1	**TD2**	**TD3**	**TD4**	**TD5**	**TD6**	**TD7**	**TD8**	**CER**
C1	Error/1	Error/1	Ok	Ok	Ok	Ok	Error/1	Ok	37.5%
C2	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C3	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C4	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C5	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C6	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C7	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C8	Ok	Ok	Ok	Ok	Error/1	Ok	Ok	Error/2	25.0%
C9	Ok	Ok	Error/1	Ok	Ok	Ok	Ok	Ok	12.5%
C10	Ok	-	-	Ok	Error/2	Error/1	-	-	50.0%
C11	Ok	Error/2	Error/1	-	Error/2	Ok	Error/2	Error/1	71.4%
C12	Error/1	Ok	-	Ok	-	Error/1	-	-	50.0%
C13	-	Ok	Ok	-	-	-	Ok	-	0.0%
C14	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	0.0%
C15	-	-	-	-	-	-	-	-	-
C16	Ok	Error/1	Ok	Ok	Ok	Ok	Ok	Error/4	25.0%
PER	7.7%	7.7%	0.0%	0.0%	0.0%	0.0%	9.1%	0.0%	
FER	9.1%	27.3%	16.7%	0.0%	18.5%	14.3%	22.2%	35.0%	

For further analysis, we divide the sixteen constraints into three larger groups as follows. C1 through C7 enforce the correct usage of the main constructs of process, tier, actor, and product flow, without specifically considering their interrelation. C8 and C9 check the correct usage of the construct of management process, thus *Source*, *Make*, and *Deliver*. Finally, C10 through C16 assess the correct usage of the construct of product specificity. We aggregate the verification results accordingly in Table 4.

Table 4. Results of model verification per constraint group.

Constraint Group	TD1	TD2	TD3	TD4	TD5	TD6	TD7	TD8	CER
Usage of the constructs of process, tier, actor, and product flow	Error	Error	Ok	Ok	Ok	Ok	Error	Ok	37.5%
Usage of the construct of management process	Ok	Ok	Error	Ok	Error	Ok	Ok	Error	37.5%
Usage of the construct of product specificity	Error	Error	Error	Ok	Error	Error	Error	Error	87.5%

5.3 Discussion

The results presented in Table 3 and Table 4 lead to a number of observations. First, we found errors in all but one model. However, the error-free model TD4 stems from the transport logistics in which all processes are of make-to-order specificity and no manufacturing takes place. TD4 thus contains only S2 and D2 processes, which describe a product flow with no branches. This could limit the appearance of syntactic errors. On the other hand, each error-prone model violates two or three constraints.

Second, as shown in Table 3, the highest CER is reported for C11 (71.4%), C10, and C12 (both at 50.0%). All three constraints relate to the usage of product specificity as a determinant of linked processes from *Source* to *Deliver* or *Deliver* to *Source*. In general, product specificity appears to be the most ambiguous construct of SCOR, since Table 4 reveals that its aggregated CER is as high as 87.5%. In the other two constraint groups, errors are found in three out of the eight models. The management process construct was used falsely as well: Two models contain a sequence of *Make-Deliver-Make* as shown in Fig. 7, whereas only *Make-Make* would be correct. One model arranged two actors that exchange products within the same tier instead of separating them into two different tiers (TD3, shown in Fig. 6).

When interpreting the results, in particular the relatively high error rates, we must be aware of the sample size. Since our sample is too small for further statistical analysis, it did not allow us to provide evidence in terms of factors that affect the error rate of a particular model. However, we believe that our initial findings suggest some ambiguity in the existing body of knowledge that is available to supply chain designers. This knowledge ranges from the SCOR documentation and supplementing training material to other references and software tools. We also found errors in the two models that originated directly from the SCOR organization, which were presented at the SCOR Fall Conference 1997 (TD7, TD8).

The proposed methodology and the prototype successfully detect errors in SCOR models. These errors must be corrected by the supply chain designer. Next, we illustrate how our approach can assist the designer in correcting incorrect models. We have to consider that often several alternatives exist for fixing the errors, from which the designer has to choose the alternative that matches the desired structure. For instance, the model in Fig. 5 shows for the OEM tier an invalid link between the D2 process and the S1 process of the Warehouse tier. There exist at least two alternatives for the designer, i.e., either modifying D2 to D1 or modifying S1 to S2. We can automatically generate these alternatives and present them to the designer by deriving a rule from the formal definition of C11 as follows: For each flow $f=(p_i, p_j)$ in error class C11, modify the model as follows: $CS(CM(PC(p_i))):=\{Stock\} \vee CS(CM(PC(p_j))):=\{Retail\}$.

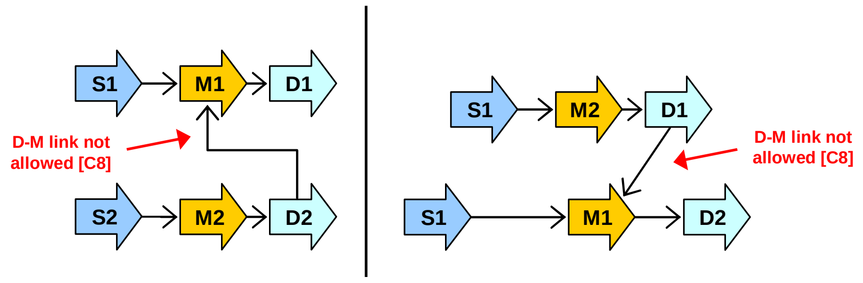

Fig. 7. Incorrect use of the management process construct in model TD5 (left) and TD8 (right).

In this way, we are able to construct rules for every constraint (by enumerating the alternatives that are offered by the constraint) and present the alternatives to the designer to choose from. A practical limitation of this repair approach is that fixing a particular error may cause subsequent errors. For instance, if the designer changes the S1 process of the Warehouse tier (Fig. 5) into S2, the link to its succeeding D1 process becomes invalid (violation of constraint C12). These cascading errors could propagate both upstream and downstream the supply chain and thus arriving at a correct model maybe somewhat cumbersome. Therefore, assisting the designer in developing correct models from the start is another direction to take by proactively providing suggestions for linking processes to prevent errors. Next, we outline how the deduced constraints can be used for assisting the designer in this task.

We consider an upstream design in which the designer defines all the tiers, adds processes to the right most tier, and then places linked processes in the upstream tiers. For instance, the right most tier contains a single S2 process denoted by *p1*, with *PC*(*p1*)={*S2*} and *N*(*PT*(*p1*))=1. Two other tiers exist. The question is what kind of processes *p2* may link to *p1* by the flow *f1*=(*p2*, *p1*)? We can generate all valid answers by analyzing those constraints that are concerned with flows into S2 processes:

– Constraint C8 is not relevant, because it only relates to flows inside an actor. Constraint C9 allows flows into *Source* processes from *Deliver* processes of preceding tiers. Therefore, we use the definition of this constraint and add an axiom to the design space: $(PC(p2){:=}\{D1\} \vee PC(p2){:=}\{D2\} \vee PC(p2){:=}\{D3\} \vee PC(p2){:=}\{D4\}) \wedge (N(PT(p2)){:=}2 \vee N(PT(p2)){:=}3)$.

– Constraint C10 helps us in suggesting the right product specificity of *p2*, i.e., specificity of *p2* must be the same as that of *p1*. We add to the design space: $CS(CM(PC(p2))){:=}CS(CM(PC(p1)))$.

– Next, we can assess the design space, which consists of two axioms. Since $CS(CM(PC(p1)))$ is given in the model as *Order*, we replace the second axiom by $CS(CM(PC(p2))){:=}\{Order\}$. Therefore, the design space is further reduced to: $PC(p2){:=}\{D2\} \wedge (N(PT(p2)){:=}2 \vee N(PT(p2)){:=}3)$.

Finally, the designer can select from these two suggestions by placing a D2 process as an input for the S2 process, in either tier left of the customer tier. In this way, we are able to generate candidate incoming flows for all the process categories and suggest them to the designer to choose from. We have illustrated the required analytical steps for the upstream design only. In case of the downstream design, the analysis would be quite similar, by taking from the constraint definitions those parts that are concerned with the succeeding process and converting them into axioms.

5.4 Implications

Our research has implications for both the users and developers of SCOR. Users should carefully revisit the existing SCOR documentation and base their supply chain design on a formal grammar instead of being inspired by only the brief descriptions and example models. Otherwise, they are in danger of developing models that may contain subjective interpretations of the technique and thus cannot be exchanged, formally analyzed, and shared with supply chain stakeholders. This risk would also hamper benefiting from the two posited advantages of reference models for supply chain design. The evaluation results suggest that the product specificity construct leads to a number of design alternatives that must be carefully evaluated by the user to select a correct design. At least, its original definition lacks clarity to enable its correct usage. However, the SCM literature provides sufficient underpinning, as discussed in section 4.4.3, for clarifying the syntax and semantics of processes and product flows under consideration with respect to product specificity.

Whereas the proposed methodology successfully detects incorrect models, our research also has implications for the use of validated models and the development of decision support systems (DSS) within SCM. These implications concern the third use case of the SCOR technique (discussed in section 1), namely, supply chain evaluation. The DSS must assist the designer in creating alternative designs and assessing their performance through appropriate metrics. For this type of DSS, the proposed artifact provides not only the formalization for model representation and storage, but also automatic generation of alternative designs. The difficulty in generating alternative designs is the sheer number of alternatives that may result from adding, removing, and modifying each single model element (processes, flows, actors, and tiers). The proposed artifact could be used for answering questions such as what downstream and upstream alternatives exist for a particular process. For instance, the M3 process in Fig. 1 has three upstream processes of category S1 and S2. We can generate alternatives for either process by utilizing the constraint that is concerned with these processes, namely, C14. The formal definition of C14 spans the design space, which consists of two alternatives for each upstream process (S2 and S3 for the S1 process, and S1 and S3 for the S2 process). In this way, the proposed constraints could be used by the DSS for generating alternative designs and presenting them to the designer. Similarly, while correcting invalid models, each alternative for a particular process may result in subsequent errors, both downstream and upstream. In this case, the DSS could apply the “repair” methodology as described in the preceding section to avoid cascading errors.

6 Conclusion and Future Work

This research presents a supply chain grammar and analysis techniques for SCOR-based supply chain design. The grammar adapts SCM constructs and rules to avoid making assumptions about supply chain design that are not justified by the literature.

There are three main results of this work. First, this research demonstrates that the SCM literature provides the rules to be able to effectively restrict the design space that is spanned by the SCOR technique. This research could be extended to support reverse product flows (return processes), information flows (plan processes) as well as other parts of the SCOR Model. However, any endeavor must first clarify the main constructs for primary product flows, which is the main focus of our research.

Second, our work represents an effort to improve the understanding of supply chain designs across organizational boundaries. Specifically, it highlights the need to clearly define the formal semantics of primary product flows in SCOR, which is missing so far. The initial evaluation suggests that the current adoption of the SCOR technique is negatively impacted by its informal description, which leads to error-prone supply chain designs.

Third, the grammar is an initial step in understanding the many factors that affect the correctness of models created, the perceptions of model users, and the performance of individuals who use these models for solving problems in a particular domain. Specifically, it is still unknown how a particular error rate is correlated with factors such as the design environment (e.g., grammar used, software tool support) and domain (e.g., industry, size and complexity of the diagram). These effects could be studied by incorporating the general findings from conceptual modeling research [40].

Our future work will use the deduced supply chain grammar to study the factors that affect the extent to which model users understand the domain semantics that is conveyed in a thread diagram. Our current study suggests that the original version of the SCOR grammar, which is provided in the SCOR handbook, causes difficulties for effectively defining mappings between real-world phenomena and their representations. For instance, the original grammar lacks rules that guide designers how to connect processes under consideration of the supply chain context. These rules were added by deduction from the existing SCM literature. Positing that deficiencies in the grammar exist, we plan to empirically validate in a laboratory experiment, the effects of such deficiencies on the user's problem solving performance when using models generated from the grammar.

Acknowledgements

The work of the first author has been partly supported by the German Ministry of Education and Research under the project InterLogGrid (BMBF 01IG09010E). The work of the second author has been partly supported by Sogang Business School's World Class University Program (R31–20002) funded by Korea Research Foundation, and Sogang University Research Grant of 2011.

References

[1] W.M.P. van der Aalst, Formalization and verification of event-driven process chains, Information and Software Technology 41 (1999) 639–650.

[2] J.C. Anderson, H. Håkansson, J. Johanson, Dyadic business relationships within a business network context. Journal of Marketing 58 (1994) 1–15.

[3] A. Ankolekar, M. Paolucci, K. Sycara, Towards a formal verification of OWL-S process models, in: Y. Gil, E., Motta, V. Benjamins, V., M. Musen (Eds.), The Semantic Web – ISWC, Springer, Berlin, 2005, pp. 37–51.

[4] M. Arns, M. Fischer, P. Kemper, C. Tepper, Supply chain modelling and its analytical evaluation, Journal of the Operational Research Society 53 (2002) 885–894.

[5] F. Baader, I. Horrocks, U. Sattler, Description logic, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, Springer, Berlin, 2007, 135–179.

[6] B.M. Beamon, Supply chain design and analysis: models and methods, International Journal of Production Economics 55 (1998) 281–294.

[7] J. Becker, R. Knackstedt, A. Stein, Extending the supply chain operations reference model: potentials and their tool support, in: ECIS 2007 Proceedings, Paper 123, 2007.

[8] K. Burgess, P.J. Singh, R. Koroglu, Supply chain management: a structured literature review and implications for future research, International Journal of Operations & Production Management 26 (2006) 703–729.

[9] I.J. Chen, A. Paulraj, Towards a theory of supply chain management: the constructs and measurements. Journal of Operations Management 22 (2004), 119–150.

[10] P.D. Cousins, B. Lawson, B. Squire, Supply chain management: theory and practice – the emergence of an academic discipline?, International Journal of Operations & Production Management 26 (2006) 697–702.

[11] S. Croom, P. Romano, M. Giannakis, Supply chain management: an analytical framework for critical literature review, European Journal of Purchasing & Supply Management 6 (2000) 67–83.

[12] R.M. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in BPMN, Information and Software Techology 50 (2008) 1281–1294.

[13] M.L. Fisher, What is the right supply chain for your product? Harvard Business Review 75 (1997) 105–116.

[14] B.J. Gibson, J.T. Mentzer, R.L. Cook, Supply chain management: the pursuit of a consensus definition, Journal of Business Logistics (2005) 17–25.

[15] S. Gonnet, M. Vegetti, SCOntology: a formal approach toward a unified and integrated view of the supply chain, in: M.M. Cunha, B.C. Cortes, G.D. Putnik (Eds.), Adaptive technologies and business integration: social, managerial and organizational dimensions, IGI Global, Hershey, 2007, 137–158.

[16] A. Gunasekaran, E.W.T. Ngai, Build-to-order supply chain management: a literature review and framework for development, Journal of Operations Management 23 (2005) 423–451.

[17] A. Hevner, S. March, J. Park, S. Ram, Design science in information systems research, MIS Quarterly 28 (2004) 75–105.

[18] C. Hicks, T. McGovern, C.F. Earl, Supply chain management: A strategic issue in engineer to order manufacturing, International Journal of Production Economics 65 (2000) 179–190.

[19] S.H. Huan, S.K. Sheoran, G. Wang, A review and analysis of supply chain operations reference (SCOR) model, Supply Chain Management 9 (2004) 23–29.

[20] D.M. Lambert, M.C. Cooper, Issues in supply chain management, Industrial Marketing Management 29 (2000) 65–83.

[21] G. Liberopoulos, Y. Dallery, A unified framework for pull control mechanisms in multi-stage manufacturing systems, Annals of Operations Research 93 (2000) 325–355.

[22] J. Lee, G.M. Wyner, B.T. Pentland, Process grammar as a tool for business process design, MIS Quarterly 32 (2008) 757–778.

[23] A. Lockamy, K. McCormack, Linking SCOR planning practices to supply chain performance: an exploratory study, International Journal of Operations & Production Management 24 (2004) 1192–1218.

[24] J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, G. Neumann, Detection and prediction of errors in EPCs of the SAP reference model, Data and Knowledge Engineering 64 (2008) 312–329.

[25] J.T. Mentzer, W. DeWitt, J.S. Keebler, S. Min, N.W. Nix, C.D. Smith, Z.G. Zacharia, Defining supply chain management, Journal of Business Logistics 22 (2001) 1–25.

[26] P.-A. Millet, P. Schmitt, V. Botta-Genoulaz, The SCOR model for the alignment of business processes and information systems, Enterprise Information Systems 3 (2009) 393–407.

[27] T. Moyaux, P. McBurney, M. Wooldridge, A supply chain as a network of auctions, Decision Support Systems 50 (2010) 176–190.

[28] J. Olhager, Strategic positioning of the order penetration point, International Journal of Production Economics 85 (2003) 319–329.

[29] B.T. Pentland, Process grammars: A generative approach to process redesign, Working Paper Series 178, MIT Center for Coordination Science, Cambridge, MA, 1994.

[30] B.T. Pentland, Grammatical models of organizational processes, Organization Science 6 (1995) 541–556.

[31] D. Rodríguez, E. García, S. Sánchez, Defining software process model constraints with rules using OWL and SWRL, International Journal of Software and Knowledge Engineering 20 (2010) 533–548.

[32] A. Röder, B. Tibken, A methodology for modeling inter-company supply chains and for evaluating a method of integrated product and process documentation, European Journal of Operational Research 169 (2006) 1010–1029.

[33] O. Sakka, P.-A. Millet, V. Botta-Genoulaz, An ontological approach for strategic alignment: a supply chain operations reference case study, International Journal of Computer Integrated Manufacturing 24 (2011) 1022–1037.

[34] S. Stephens, Supply chain operations reference model 5.0: a new tool to improve supply chain efficiency and achieve best practice, Information Systems Frontiers 3 (2001) 471–476.

[35] G. Steward, Supply-chain operations reference model (SCOR): the first cross-industry framework for integrated supply chain management, Logistics Information Management 10 (1997) 62–67.

[36] Supply-Chain Council, Supply chain operations reference model (SCOR®), Version 9.0, Washington DC, 2009.

[37] C.J. Vidal, M. Goetschalckx, Strategic production-distribution models: A critical review with emphasis on global supply chain models, European Journal of Operational Research 98 (1997) 1–18.

[38] W3C, OWL web ontology language, W3C recommendation 10 February 2004. URL: http://www.w3.org/TR/owl-ref/ (accessed on April 12, 2012)

[39] W3C, SWRL: a semantic web rule language combining OWL and RuleML, W3C Member Submission 21 May 2004. URL: http://www.w3.org/Submission/SWRL/

[40] Y. Wand, R. Weber, Information systems and conceptual modeling – a research agenda, Information Systems Research 13 (2002) 363–376.

[41] W.Y.C. Wang, H.K. Chan, D.J. Pauleen, Aligning business process reengineering in implementing global supply chain systems by the SCOR model, International Journal of Product Research 48 (2010) 5647–5669.

[42] M. Zdravković, H. Panetto, M. Trajanović, A. Aubrey, An approach for formalising the supply chain operations, Enterprise Information Systems 5 (2011) 401–421.