How to Solve the Business Standards Dilemma

The Context Driven Business Exchange

Disclaimer & Liability Notice

How to Solve the Business Standards Dilemma – The Context Driven Business Exchange

Applies To:

SAP NetWeaver

Summary

Web Services are important, but are insufficient in solving integration requirements in collaborative business scenarios. This is because Web Services by themselves do not guarantee that the business information exchanged between different enterprises will be understood equally well by all systems. Web Services do not say anything about the semantics (meaning) of business information. Therefore Web Services are not enough for e-collaboration between business partners and do not ensure that systems can communicate effectively across company boundaries. A common understanding requires that the semantics of the business information is based on a grammar and library that is well known and understood.

This requisite common understanding is only reachable if the semantics of the business information is based on a grammar and library that is well known and understood. The CCTS (Core Components Technical Specification) standard from UN/CEFACT provides the answer. CCTS methodology enables a common understanding of semantics at a syntax-independent level. SAP is using this international standard for harmonizing the business information used by Enterprise Services – the so-called SAP Global Data Types (GDTs).

This is the first article in a series “How to Solve the Business Dilemma” that explains why the CCTS approach is so essential in developing a common understanding of business information. This article with the subtitle ”Context Driven Business Exchange” provides an overview in usage of CCTS. Future articles will describe the main parts of the CCTS methodology more deeply.

By: Gunther Stuhec

Company: SAP AG

Date: 20. October 2005

Table of Contents

Introduction

The eXtensible Markup Language (XML) provides a syntax for creating business vocabularies and exchanging business information. Many believe that creating XML vocabularies is sufficient to achieve data interoperability. However, this assumption is false. XML by itself does not guarantee that XML expressed business information relating to business processes exchanged across different enterprises will be understood equally well by all systems. This is because the XML syntax only provides for creating markup languages used as metadata, it does not address how the underlying business information must be modeled, named and structured.

To achieve truly interoperable e-collaboration between business partners, something more than XML must be employed. For example, XML allows the usage of synonyms for the same entity. To ensure that systems can understand eachother, even across company boundaries, there must be a common understanding of the underlying data. Such a common understanding is only reachable if the semantics of the business information is based on a standard grammar and library that is well known and understood by both humans and machines.

Fortunately, a standard exists that makes this a reality. The CCTS (Core Components Technical Specification) from UN/CEFACT and ISO provides a methodology for semantic data modeling that achieves a common understanding of data on a syntax independent level.

The Dilemma Today

The role of semantics is not limited to the development of XML-based interfaces. Semantics are an integral part of all aspects of the development and use of business data models. These include various kinds of interfaces within and between business applications. To achieve real interoperability, every data model must represent the specific requirements in a structured and technical way that both humans and machines can understand and process. The difficulty in this is that currently most modeling languages focus almost exclusively on the technical aspects of creating the model rather than the semantic aspects necessary for true interoperability. Compounding the problem is that none of the current tools or methodologies has developed a universal language to describe the semantics of its models in unambiguous terms. This complicates mapping and integration between interfaces, requiring careful

analysis on the part of the developer to ensure correct understanding and interpretation and thereby incurring great expense

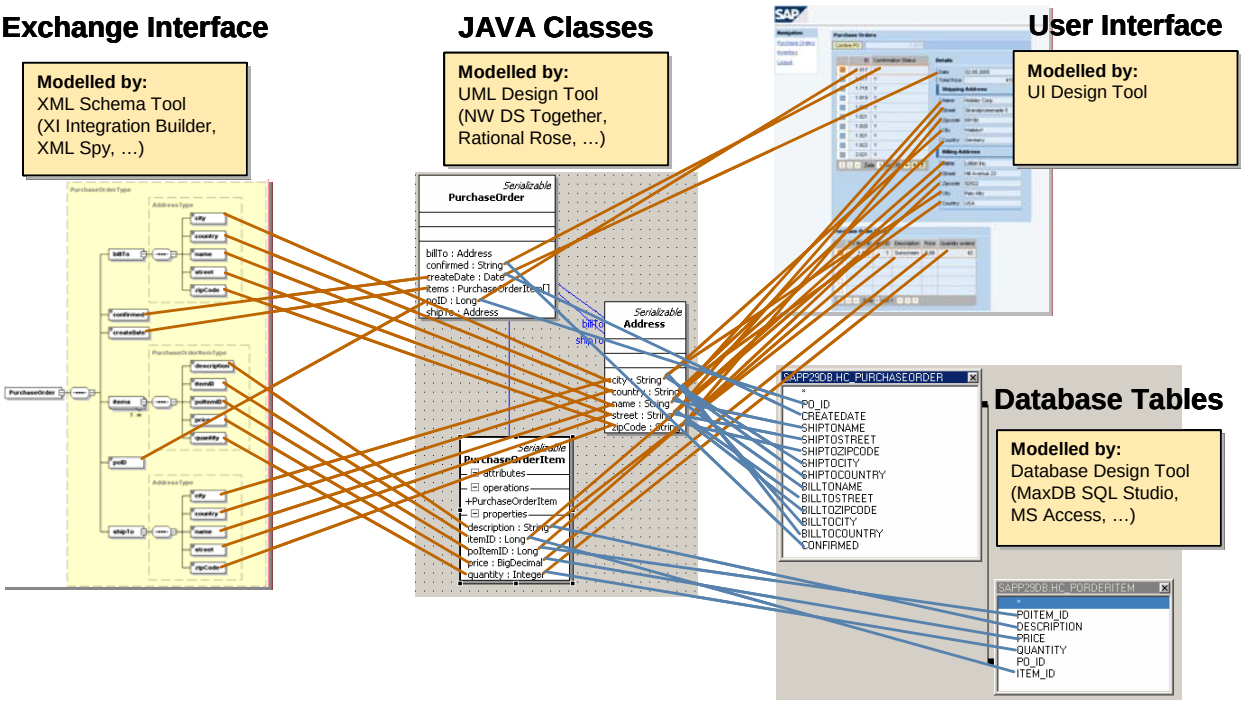
Within Business Applications

Semantics apply to, for example, the following key interfaces of business process models and business applications:

- Class diagrams of all object oriented language based applications (e.g. Java, Objects, C++),
- Structures and definitions of data base tables,
- Schema for the external data interchange interfaces and
- Layout and structure of the user interfaces

Figure 1 illustrates these data associations of an application example in which purchase order data is entered through a UI, persisted in a data base, developed in Java Classes, and exchanged by an XML-based interface. All these data structures are based on different structures and naming because all too often, independent users and developers use different tools, terms and meanings in each step of the process. Thus, very different data definitions, each developed independently at high expense within an application, are not easily interoperable. They must be integrated with one another to allow for internal reuse. This problem is compounded when dealing with external B2B interfaces.

Figure 1 - Data Associations of Business Application Interfaces



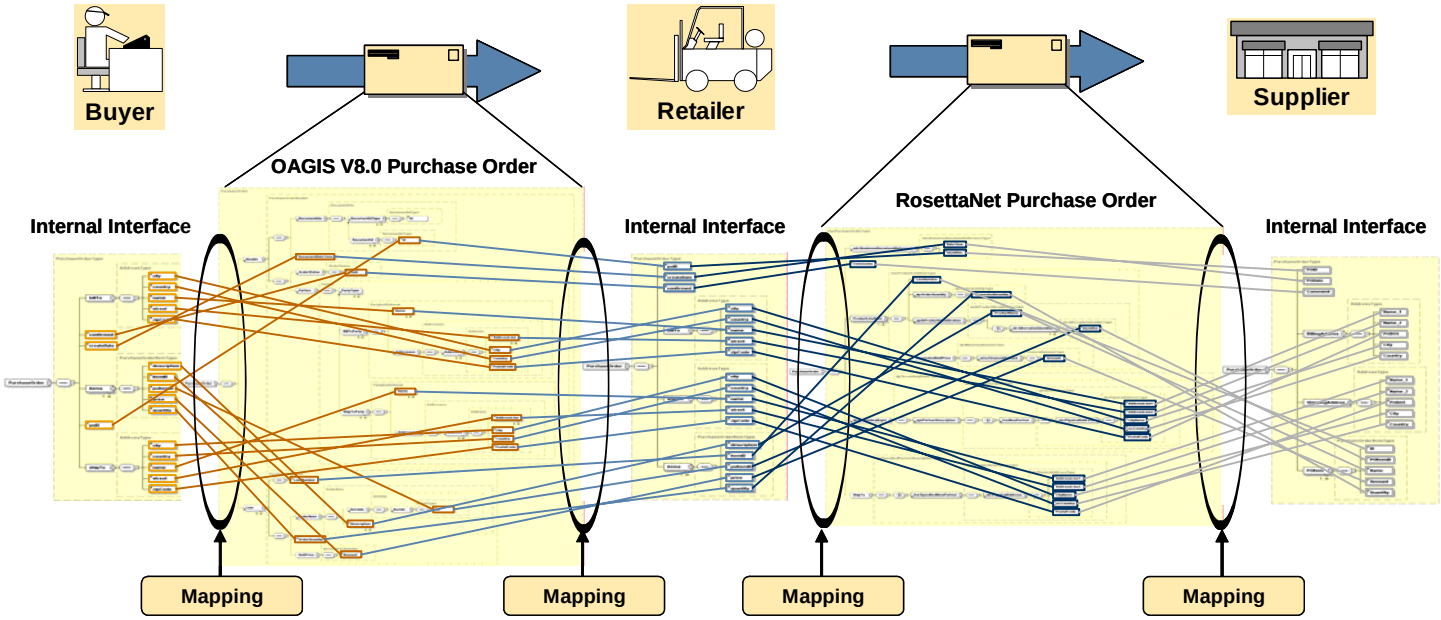
Between Business Applications

Web Services address interoperability on a technical level and are necessary for implementing collaborative business scenarios. But they are not sufficient since they do not define how business information like a "Purchase Order" needs to be defined in a certain context so that everyone can understand and process it in the same manner. Many industry - specific consortia, like CIDX, PIDX, OAGi,

and RosettaNet, try to solve th is gap and define industry - specific XML - based vocabularies. But these XML vocabularies are based on different methodologies for representing the semantics of business information , thus, similar entities like, for example, a purchase order, are designed quite differently . This leads to new problems in collaborative business, because a mapping between these vocabularies is required but often difficult and expensive. This problem can be called the "business standards dilemma", i.e. there are certainly standards that address particular requirements, but the fact that there are so many and that they are designed so differently represents a new challenge. The quintessence for solvjng this problem is a common methodology in order to get a semantical ly unambiguous representation and usage of business information. The UN/CEFACT CCTS considers especially th es aspects.

Figure 2 gives an example of how totally different XML based business data can be. The internal business data structures, which are based on business-oriented objects, are usually not the same structures that are used by an external business partner. Everyone uses their own interpretations and assumptions of the semantic aspects of the business data for their specific definitions and uses. For a business data exchange to be realized, the involved business partners must first agree on a standardized representation of the business data to be exchanged. They have to map their internal business structures to the agreed business data structure, which may be expressed as an agreed-upon syntax-specific format, and may be incorporated as a business standard.

Figure 2 - Current electronic business information flow in a value chain



But this kind of mapping is very inefficient, costly and time intensive. Why? Because the following questions must be solved before business information can be exchanged:

- There are many different business oriented dialects (CIDX, PIDX, RosettaNet, OAGi, UN/EDIFACT, ASC ANSI X.12, xCBL, UBL, etc.). Which business dialect should be used?
- The different organizations, since they work independently and have their own methodologies, create interfaces and standards that use different names and structures for the same data. For example, the supply chain exchange in Figure 2 mapped a single date element into the following disparate names: `createDate (Buyer)` → `DocumentDateTime (OAGIS)` → `createDate (Distributor)` → `BusinessDocumentReference/DateTime (RosettaNet)` → `PODate (Seller)`. Are these really entities with the same semantic meaning?

- The entities are of different complexity, with different cardinality and placed in different positions within the structure of an interface (see following schema example). How can these entities correctly be mapped between each other? What should be done, if the cardinality is different? What should be done if the required entity is positioned at a completely different level?

- The level and consistency of information is quite often different between the entities of the different interfaces. As shown in Figure 3, the element "Price" in the buyers proprietary interface does not have additional currency information at all, whereas the element "Amount" within the OAGIS V8.0 Purchase Order has an additional mandatory attribute for currency. How should this information should be created and inserted into the required exchange structures, and what are the costs associated with creating mandatory data for one structure that is different than the mandatory data for other interface structures?

Figure 3 - XML Schema Examples

Buyer Proprietary Application Interface Format

```
    <xs:complexType name="PurchaseOrderItemType">
        <xs:sequence>
            <xs:element name="description"
type="xs:string"/>
            <xs:element name="itemID" type="xs:long"/>
            <xs:element name="poItemID" type="xs:long"/>
            <xs:element name="price"
type="xs:decimal"/>
            <xs:element name="quantity" type="xs:integer"/>
        </xs:sequence>
    </xs:complexType>
```

OAGIS V8.0 Purchase Order

```
<xs:element name="UnitPrice" type="AmountPerQuantity"
minOccurs="0"/>
…
<xs:complexType name="AmountPerQuantity">
    <xs:sequence>
        <xs:element name="Amount" type="Amount"/>
    </xs:sequence>
</xs:complexType>
…
<xs:element name="Amount" type="Amount"/>
<xs:complexType name="Amount">
    <xs:simpleContent>
        <xs:extension base="xs:decimal">
            <xs:attribute name="currency"
            type="Currency"  use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

Synopsis of Current Situation

Despite the dawn of the XML and Web Services eras, data interoperability issues remain unsolved. Internal and external mappings of different structures are still required. The time, analysis, and development efforts necessary to achieve these mappings remain as onerous as ever. Mapping experts and data consultants who completely understand the semantic meaning of each entity in a specific interface or standardized structure remain an integral – and costly – part of the equation.

As an example, development of the mapping of a complete purchase order from an internal format to an XML-based purchase order such as RosettaNet or OAGIS for exchange with a trading partner, requires up to 10 person days. This level of effort is virtually unchanged from the cost of a mapping between non XML-based interfaces of business applications and EDI standards such as UN/EDIFACT. This is because most of the effort is the time required for analysis, comparison, interpretation and representation of the semantics in the different business data structures. This situation is exacerbated when different business information exchanges across business functional areas (e.g. order-to-invoice; construction; transportation; customs; social services) occur within a system or corporation.

As long as the semantic discourse of data and standards exists, a holistic approach in B2B is not possible. In today's automated Web service environment, electronic commerce should normally take the form of automated processes between the business applications of trading partners. But as we have seen, today's level of automation is insufficient if it is still necessary to spend so much time in integration and mapping. In other words, it is still too expensive to establish a B2B framework for exchanging

business information with all involved business partners from many different industries and countries. This is especially true for SMEs (Small and Medium Sized Enterprises) who are left out of the B2B framework because it is too expensive for their limited resources.

Solving this Problem

So how do we solve this problem? Technical implementations can only solve packet exchange connection issues. They can not solve semantic understanding issues. Fortunately, elimination of the high cost of data integration and reduction of the TCO for business level interoperability is achievable - if all involved parties and modelers use the same language, semantic, syntactic, lexical, and uniqueness rules for representation of semantics at the business information level.

The solution requires a methodology, meta-model and framework in which everyone involved defines, harmonizes and uses the same semantic definitions; but allows these definitions to be applied in different contexts. The UN/CEFACT CCTS (Core Component Technical Specification) is the first modeling methodology that addresses all these aspects.

The CCTS methodology is comparable with the syntax (grammar) rules of a human language. Language rules typically govern the way the words in a sentence are arranged. CCTS provides this for data names, but goes much further. CCTS not only provides the grammar rules, it also defines a methodology for creating unambiguous building blocks that can be understood and interpreted by humans and machines in the same way. The Core Component Technology achieves this by using the following key concepts for the development of business data:

- **Logical modeling of objects**, which is comparable to the logical composition of paragraphs, business letters, articles or even books. This is based on that approaches like OO-approach, Codd's rules for relational databases, normalization forms and structuring according to semantical logic.
- **Naming convention**, which is a set of rules of grammar, based on ISO 11179 part 5, that govern the formulation of the names of business data elements. It maps pertinent aspects of the business context onto the names of the elements.
- **The context** for giving the correct meaning of the business data, which is in relationship to the other parts and perceptions in which the communication of the data occurs. This so-called Context Driver Principle establishes predefined context categories, like geopolitical region, industry, business process etc., that identify and distinguish the usage of an element.
- **Syntax neutral** semantic information that can seamlessly be transformed into various kinds of technical representations without any information loss. A natural language can be also used by different media: speech and writing.

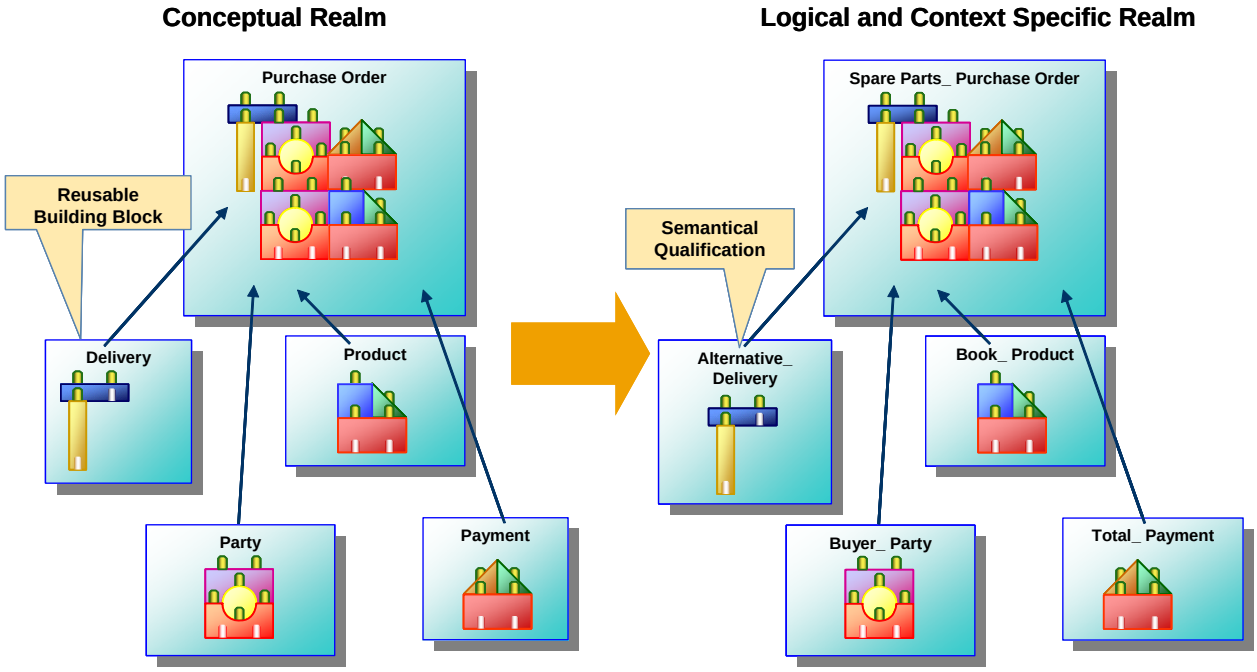
Beyond these key concepts, the heart of CCTS is its building block system for creating data aggregations. This building block system is an approach to decomposing complex business data structures and business information into modular, unambiguous and reusable building blocks. The CCTS approach follows well known Codd's Rules for data base management systems[1]. Codd's rules are considered the

[1] Codd, E. (1985). "Is Your DBMS Really Relational?" and "Does Your DBMS Run By the Rules?" *ComputerWorld*, October 14 and October 21.

authoritative doctrine for traditional RDBMS data modeling as well as the current OO-modeling approach as typified by UML[2] from OMG.

CCTS differentiates its building blocks into generic building blocks with meaningful business information which can be used in every context; and specific building blocks with a unique business semantic definition in a specific business context. Put another way, CCTS creates context-neutral building blocks that constitute the conceptual data models and physical/logical data models that make up traditional data modeling. The following figure gives a flavor of the CCTS key concepts, which will be described in more detail in future articles.

Figure 4 - Concept, Structure and Semantics of Component Building Blocks



As shown in Figure 4, "*Purchase Order. Delivery. Address*" is a conceptual component that is used whenever orders must be delivered to an order recipient at a given address. If "*Purchase Order. Delivery. Address"* is used in the context of a purchase order for representing the alternative delivery address, the component moves from the conceptual realm to the physical/logical realm by extending the generic name through appending semantically unambiguous qualifier terms.

In our example, we append the semantic qualifier "*Spare Parts*" to the term "*Purchase Order*", and the semantic qualifier "*Alternative*" to the term "*Delivery*" to give the data concept of "*Spare Parts_ Purchase Order. Alternative_ Delivery. Address*". This arrangement means that basic components can be used to define different, but generally applicable templates (as with a Lego building kit). These templates can also be adapted and modified to suit the context of different industries, sectors, business processes, products and countries. The following figures shows the key principles of the CCTS methodology, especially how different building blocks can be classified and used in different contexts.

[2] ISO, Unified Modeling Language Specification, ISO/IEC 19501:2005; January 2005

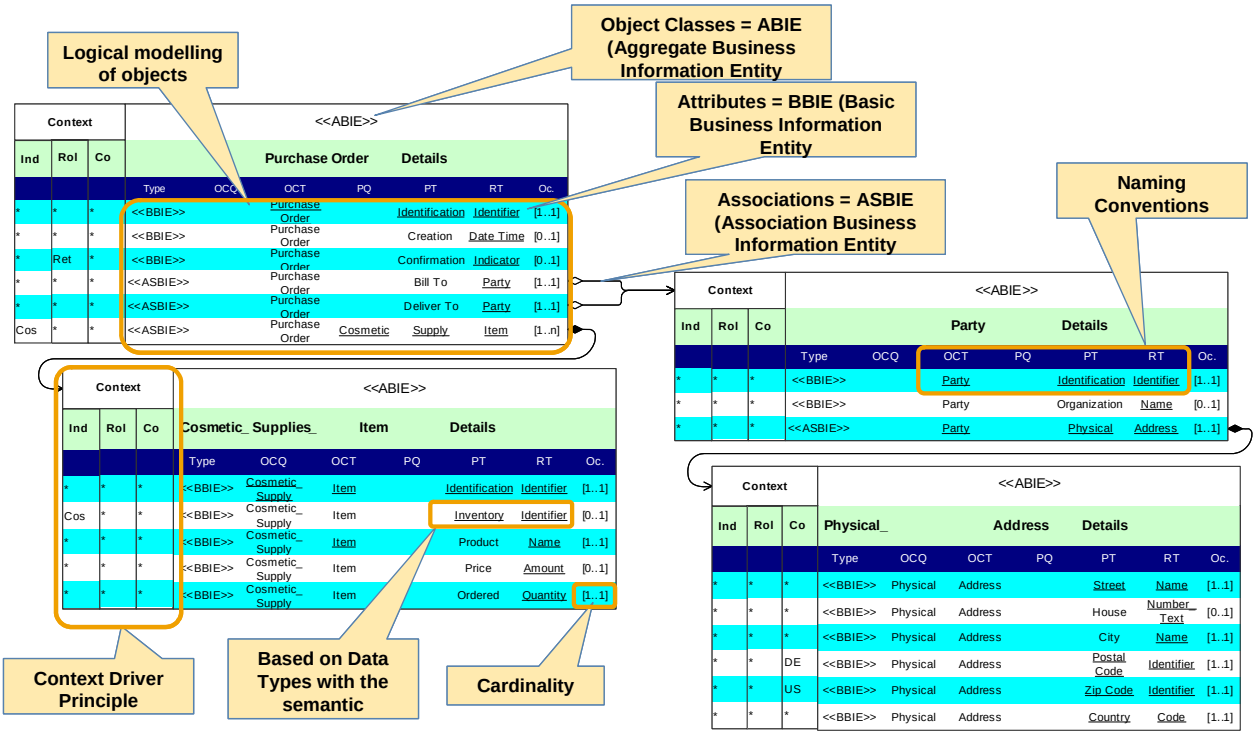
Semantic-Driven Object Modeling

CCTS is the first methodology to provide syntax-independent libraries of semantically unambiguous building blocks that you can combine and customize to suit your requirements. The CCTS methodology enables you to construct your business information and documents dynamically at design time through the rich semantics of its building blocks. These building blocks contain features that make them ideal for much more than just electronic business information sharing. CCTS-designed building blocks are ideal for physical and logical data models for data bases as well.

A special visualization of all the features and building blocks that CCTS enables can be based on UML with some additional CCTS-specific aspects. This is because the initial focus was on business information exchanges. However, as CCTS implementers have begun to apply the concepts, they have quickly seen the value of expanding the role of CCTS to include every aspect of data definition, storage, and use.

The following figures provide a suggestion of how CCTS-based business data models could be represented visually. These visual representations are based on the primary CCTS features: Logical Modeling, Naming Conventions, Context Driven Categorization and Technical Syntax independence.

Figure 5 - Visual Representation of CCTS based Building Blocks



Object classes[3] are a key construct in the CCTS model. An object class is a collection of related pieces of information in a logical order that together convey a distinct meaning in a specific context. CCTS has object classes for both conceptual and physical data constructs. To help differentiate, CCTS names its object classes for the conceptual realm “Core Components” and for the logical realm “Aggregate Business Information Entities (ABIE).” Just as object classes have properties, an ABIE may have one or more

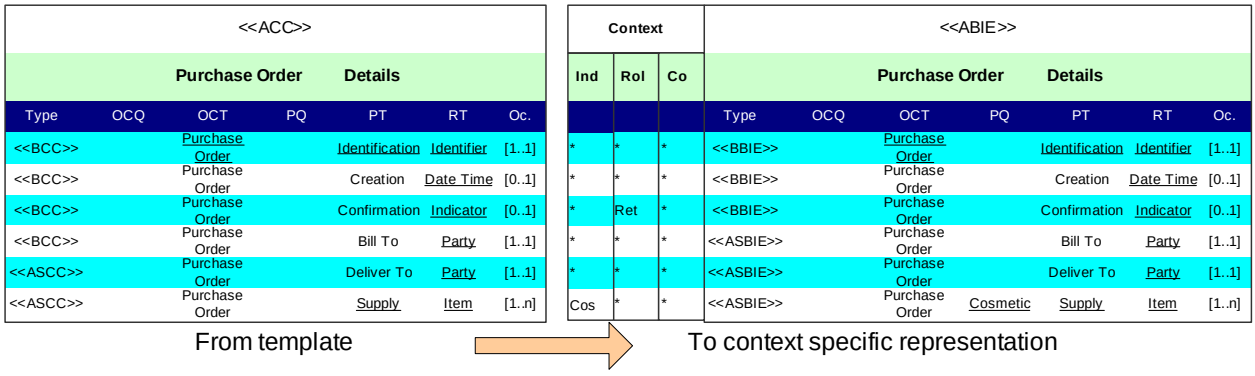
[3] See definition of Object Class in Information Technology - Metadata registries: Naming and Identification Principles for Data Elements, International Standardization Organization, ISO 11179-5

singular business characteristics for identification and/or description of the specific object class in a given business context. These characteristics are expressed by so-called Basic Business Information Entities (BBIEs). These BBIEs are the equivalent of generic data elements – in that they have both a property "Property Term" (PT) and a data type "Representation Term" (RT). An ABIE may also have one or more associations to other ABIEs. These associations represent complex business characteristics that are required for a full description of that ABIE in a given context. These associations are called Association Business Information Entities and are the semantic expressions of class associations in UML models.

Without the help of a proven set of rules it is quite hard to define a complete and logically correct assembly of all these ABIEs. But some rules help to get a similar logical and semantic structure. For example, some of the twelve Codd's rules and the process of normalization are very helpful to organize all these Business Information Entities (BIEs). Furthermore, CCTS defines the semantic aspects that describe how BIEs are to be grouped together, e.g. grouping through semantic differences and equivalencies.

To preclude two independent developers ending up with very different BIE structures for the same semantic concept of a complete assembly, all BIEs are based on context-neutral conceptual data models. As mentioned above, these conceptual data models are called Core Components (CCs).They are the default for the creation of semantically precise and meaningful business information that always adheres to the same structure. Specifically, the Aggregate Core Component (ACC) is the context-free template for building a context-specific ABIE. These ACCs always include one or more Basic Core Components (BCCs) or Association Core Components (ASCC). These BCCs and ACCs are the properties of the Core Component class, just as their context-specific counterpart BBIEs and ASBIEs are the properties of the BIE class.

Figure 6 - From template to business context specific entities

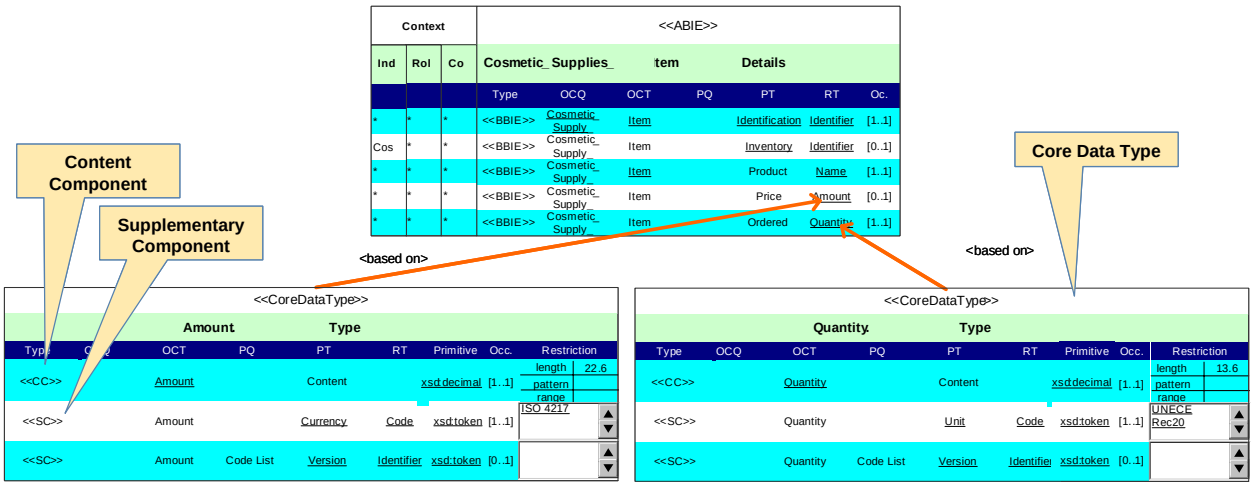


Data Typing

BCCs and their equivalent BBIEs are always based on a fixed list of "Core Data Types." These core data types are those typically used as generic data types in data structures and information exchanges – such as *Amount*, *Code, Identifier, Indicator, Measure, Name*, or *Quantity*. These Core Data Types[4] represent the smallest piece of information in a business data model and its relevant characteristics. Core Data Types have one content component for the business information value and one or more supplementary components that provide additional information necessary for the understanding of this primary business information.

[4] Core Data Types is a term, which will be released in the next version of CCTS (Version 2.1). These are currently referred to in CCTS as Core Component Types.

Figure 7 - Usage of Core Data Types



For example, the "*Amount. Type*" in Figure 7 is a core data type that defines an amount with a corresponding currency unit. The content component carries the decimal representation of the amount value. This value has no meaning by itself as it is just a number. The additional supplementary component "*Amount. Currency. Code*" provides the additionally required information – currency code – that is necessary to understand what the amount is expressing in complete terms. In CCTS, the "*Amount. Currency. Code*" supplementary component is based on an international code list for currency codes – ISO 4217 – to ensure that the value of amount will be unambiguously interpreted by users and machines.[5]

Naming Convention

The tremendous flexibility of a natural language leads to extreme difficulties in semantic interoperability of words. A big problem in today's B2B environment is that everyone is using their own, usually undocumented, naming conventions for the contents of their data models. Anyone can define their own names for objects, attributes and associations. This leads to tremendous confusion when attempting to create data interoperability since each dataset will most likely have the same semantic meaning but be expressed by synonyms (e.g. Surname, Last Name, and Family Name) and/or different meaning of terms (e.g. bank as a financial institution and bank as the side of a body of water). Syntactic, semantic and lexical rules mostly vary by organizations such as corporations or standards-setting bodies for business sectors; each can establish rules for term formation within its context.

The CCTS naming convention is simplistic in its approach, yet powerful in its application. Naming conventions recommended in ISO 11179 are implemented for both the conceptual context-neutral CC (Core Components) constructs and for the real world context-specific BIE (Business Information Entity) constructs.[6]. The naming rules for the BIE constructs build on those for their underlying CCs. The name for each BIE additionally captures its context, and as such is a label for a certain business information requirement. This business requirement could be an object or a complex concept

[5] ISO 4217:2001 Codes for the representation of currencies and funds

[6] The UN/CEFACT CCTS Specification (also published as ISO 15000-5 under the auspices of ISO TC 154), provides the actual implementation rules necessary to make 11179 a reality.

The common naming convention for both constructs unambiguously captures the semantics behind the construct. This is accomplished through individual rules for:

- Semantic rules that enable meaning to be conveyed.
- Syntactic rules that relate items in a consistent, specified order.
- Lexical (word form and vocabulary) rules that reduce redundancy and increase precision.
- Uniqueness rules that document how to prevent synonyms and homonyms occurring within the scope of the naming convention.

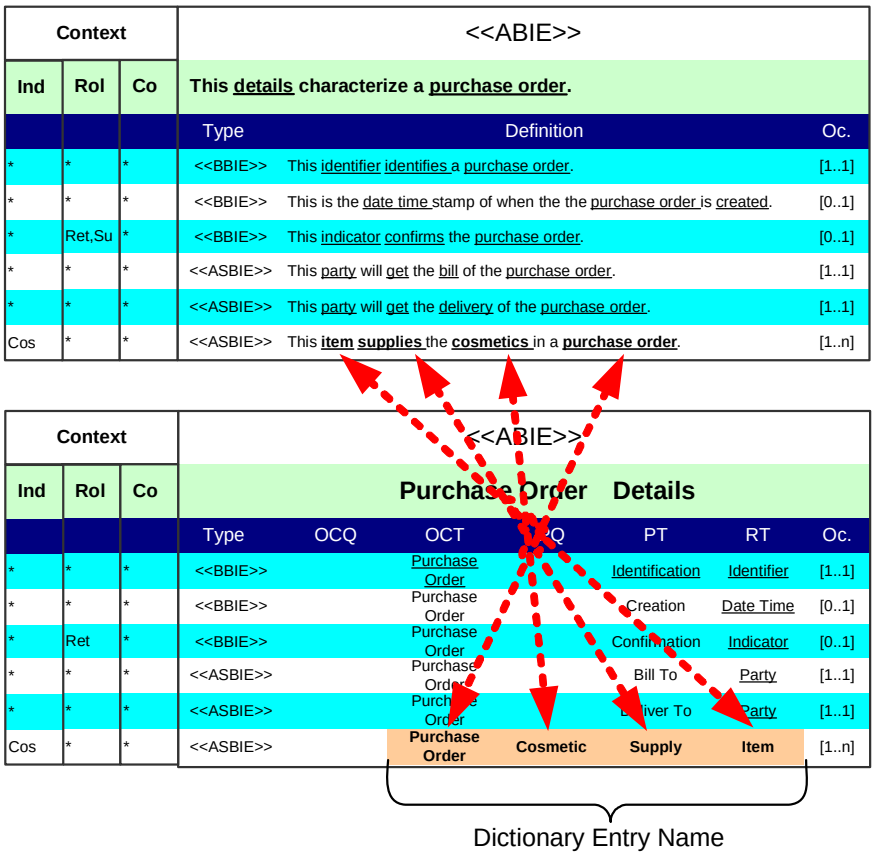
These are similar rules to those used to define the grammar for the formation of sentences in a natural language. In a natural language, each sentence is regarded as having a subject and a predicate. Predicates usually consist of "a verb with or without objects, complements, or adverbial modifiers."[7] In order to get the unambiguous semantic meaning of the name of a data element, you need only to develop the name in a manner similar to determining the terms in the phrases of a sentence. Following these semantic rules exactly enables the meaning to be conveyed in CCTS for definition of the so-called "Dictionary Entry Names" of BIEs or CCs.

As mentioned above, CCTS rules are completely based on the ISO 11179-5 standard[8]. The key parts of a Dictionary Entry Name are "Object Class" for the representation of the subject (object), plus the predicate: the "Property Term" for the representation of the verb (property) component of the predicate and a "Representation Term" for the modifier (data type). In other words, the "Dictionary Entry Names" represent the business requirements that are normally defined in sentences. Figure 8 shows how Dictionary Entry Names can be derived from these definitions.

[7] http://www.webster.com/dictionary/predicate

[8] The UN/CEFACT CCTS Specification (also published as ISO 15000-5 under the auspices of ISO TC 154), provides the actual implementation rules necessary to make 11179 a reality.

Figure 8 - Tight association between sentence and Dictionary Entry Names



A forward/backward reading guideline[9] applied to the definition of a CC or BIE construct enables creation of the exact representation of its dictionary entry name. The prerequisite to applying this guideline is that the sentence is based on English grammar and includes all three parts: subject, verb and modifier. These "Dictionary Entry Names" can be interpreted and expressed by machines more effectively and can be understood by humans more easily. This approach enables representing all business requirements in a common way, and is extremely beneficial in discovering what constructs are similar and what constructs should only be used in specific context(s).

Context Driver Principle

Context defines the environment in which a business information entity is used. Context mechanisms should be sufficiently discrete so as to enable semantically unambiguous precision. In other words, a semantic meaning of a business information entity is always unambiguous, when considered in a specific context.

For example, the business information entity "bank" is not very precise if this entity is defined without context. The business information entity "bank" describes totally different objects in the industry area of "Finance" and "Marine". Therefore, it is not possible to define only one business information entity "bank", which can be used everywhere. Rather the context in which bank is used adds further semantic meaning.

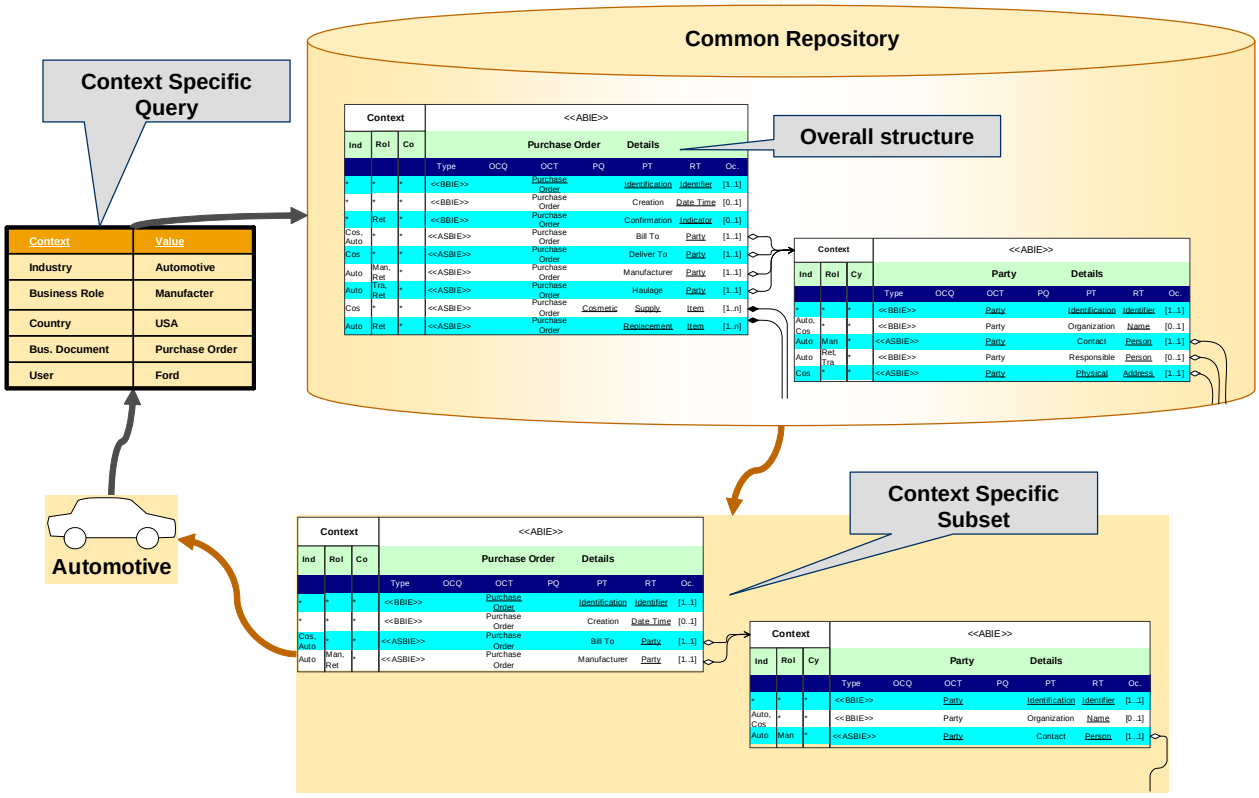
[9] See UN/CEFACT – Core Components User's Guide, 13 March 2004; Chapter 3.8.3.2, Page 34

This additional context is usually part of the BIE dictionary entry name through the use of object and property term qualifiers as discussed previously.

By following a discrete semantic data modeling process, the assignment of context and hence meaning follows a top-down approach. This requires modeling of the business process first where the overall business context will be set.

The context of the modeled business process will determine all subsequent context information. A number of predefined context categories (e.g. Business Process, Industry, Geopolitical, etc.) and predefined context values and meanings help to specialize the context in a standard way. CCTS provides this very context mechanism. Applying the CCTS context mechanism leads to the high-level context being specified and the context narrowed until the specific, unambiguous meaning can be derived. The context-specific representation allows the development of more specific business information for a specific business transaction, which fits exactly to the business requirements in which these business transactions will be used. This kind of context-driven specialization of data models is shown in Figure 9.

Figure 9 - Context driven specialization of data models



The ABIEs are stored in a common repository as discrete objects. Each stored BIE includes all entities and a classification in which the context of these entities should be used. Remember, the business information entity could be used in multiple different contexts. For example, the BIE "*Purchase Order. Creation. Date Time*" is used in every context. As shown in Figure 9, this is expressed by the wild card value of "*" being placed in each context category. However, the business information entity "*Purchase. Haulage. Party*" is used in specific contexts. This is reflected by discrete values being entered for applicable context drivers. As seen in Figure 9, this is expressed by the value of "Auto" (Automotive) in the Industry context driver column and the values of "Man" (manufacturer) or "Tra" (transport) in the Business Roles context driver column. Following our top down concept, the determination of the business role context "Manufacturer" or "Transport" restricts the overall model into the specialized data model, with the context-specific business information entities.

Usage of CCTS

If all key CCTS principles are well-known, adopted, and supported by tools to facilitate their application, significant improvements in data interoperability can be achieved through:

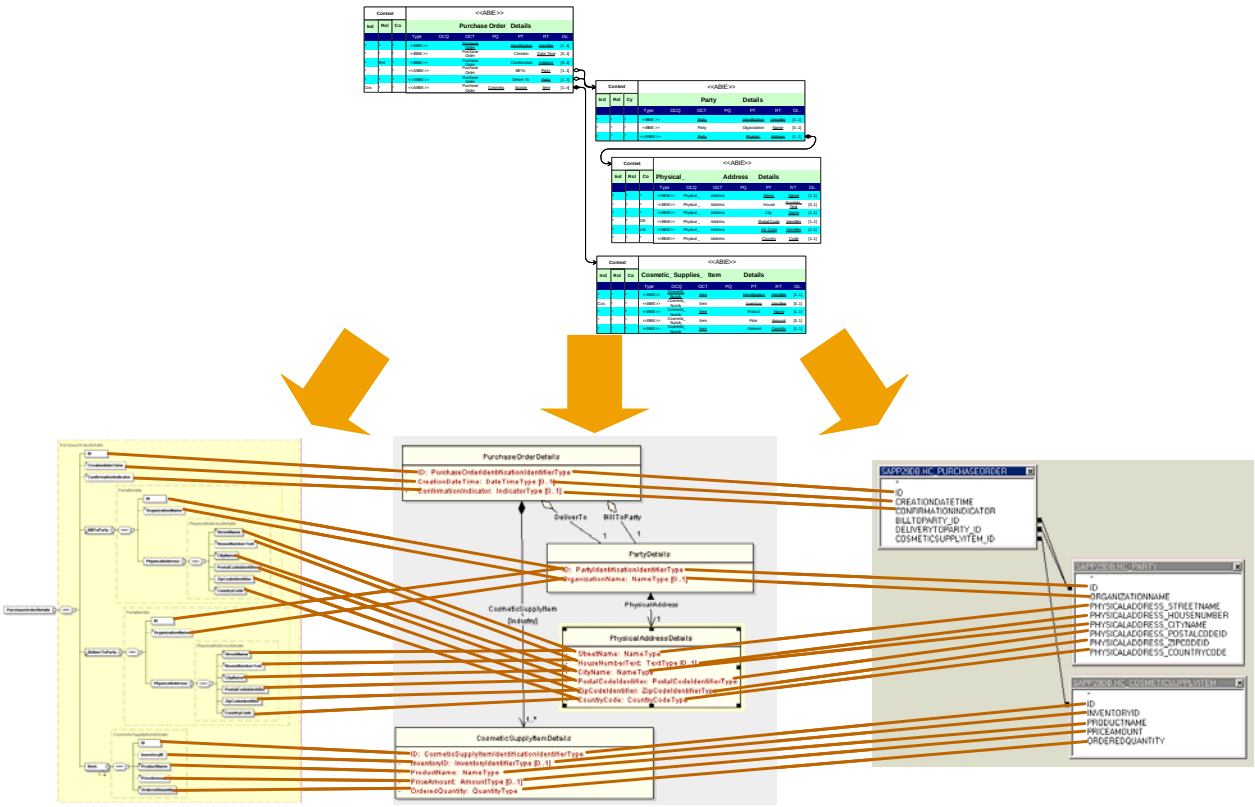
- Common representation of the data models of business application interfaces
- Distributed modeling of business data models
- Evolutionary adoption and classification of building blocks according to context-driven requirements
- Consistent exchange of business information in a value chain without resource-intensive mapping efforts.

Common Representation in all Implementations

The main objective of CCTS is the unique semantic representation of a business data model on a syntax-independent level that can be transformed in any kind of technical syntax (e.g. XML Schema, UML class diagrams, Java classes, relational data base tables) etc.

The transformation into other syntaxes must be expressed in transformation rules sufficiently rigid enough to ensure that the original semantic meanings will always be the same, and so that no semantic information in the element name is lost. Figure 10 outlines the overview of this concept.

Figure 10 - Transformation into diverse implementation syntaxes



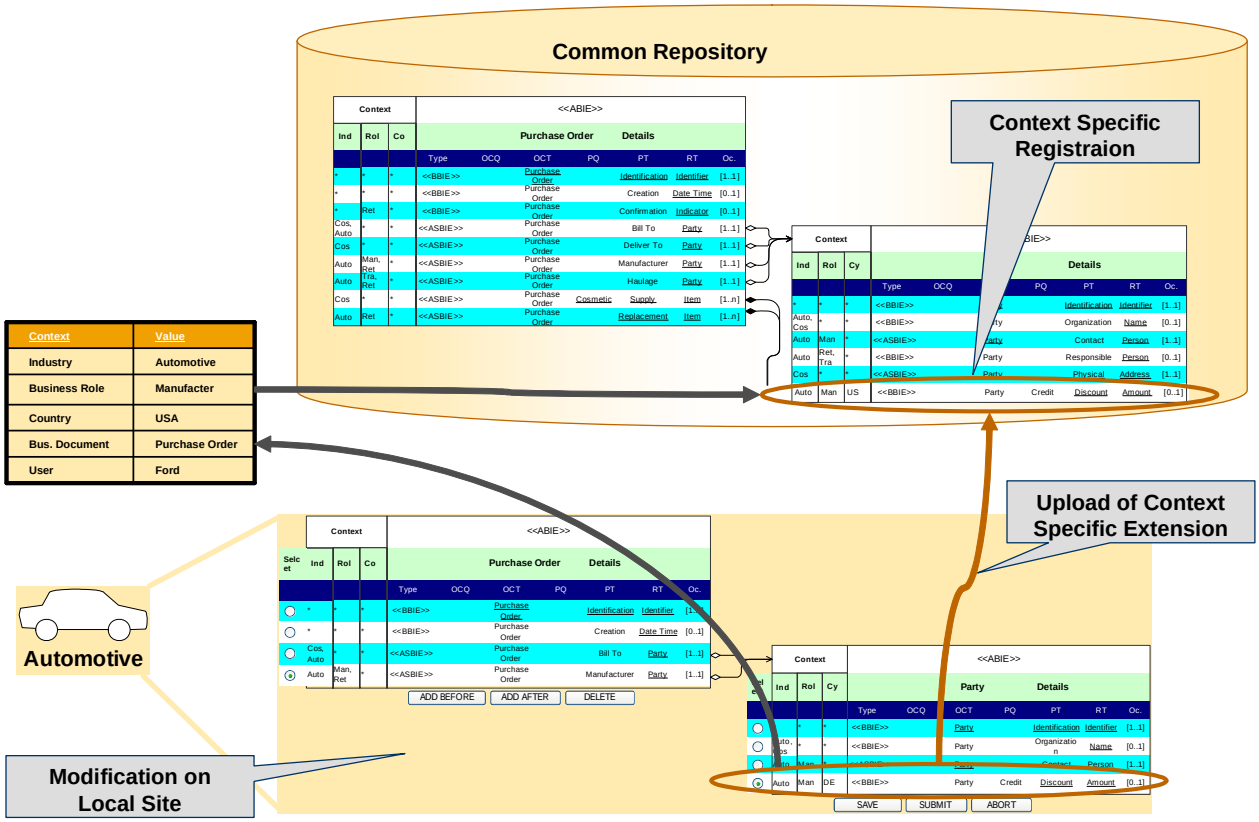
Collaborative Modeling

All key CCTS components can be stored in a common repository in order to establish a framework for collaborative business modeling where the business information can be shared and classified in a given business community. A very innovative and helpful result of implementing CCTS will be the evolutionary adjustment of business data according to real business requirements without regard to syntax restrictions.

As mentioned previously, BIE constructs are to be stored in a repository. These BIEs are additionally classified by the values assigned to their context categories, where the specific context value specifies in which context this business information entity will be relevant. These BIEs can be accessed through a registry function which makes them known and visible. In a shared registry/repository environment, any user can query and download the appropriate specialization of business data models that are relevant to the specific business context they are working in. For example, if a North American car manufacturer asks for a business transaction "Purchase Order Request", they will get a specialization of the data models according to the given context values they provided, like the user from the automotive industry in Figure 9.

This North American car manufacturer can use this specialization as provided, or can further specialize this business data model as required by their additional context requirements. As shown in Figure 11, they can also upload their context-specific modification back into the repository so that their specialization is publicly available for business partners (automotive industry) or other users, e.g. a cosmetics supplier in France who may have exactly the same or similar requirement.

Figure 11 - Uploading of context specific modifications

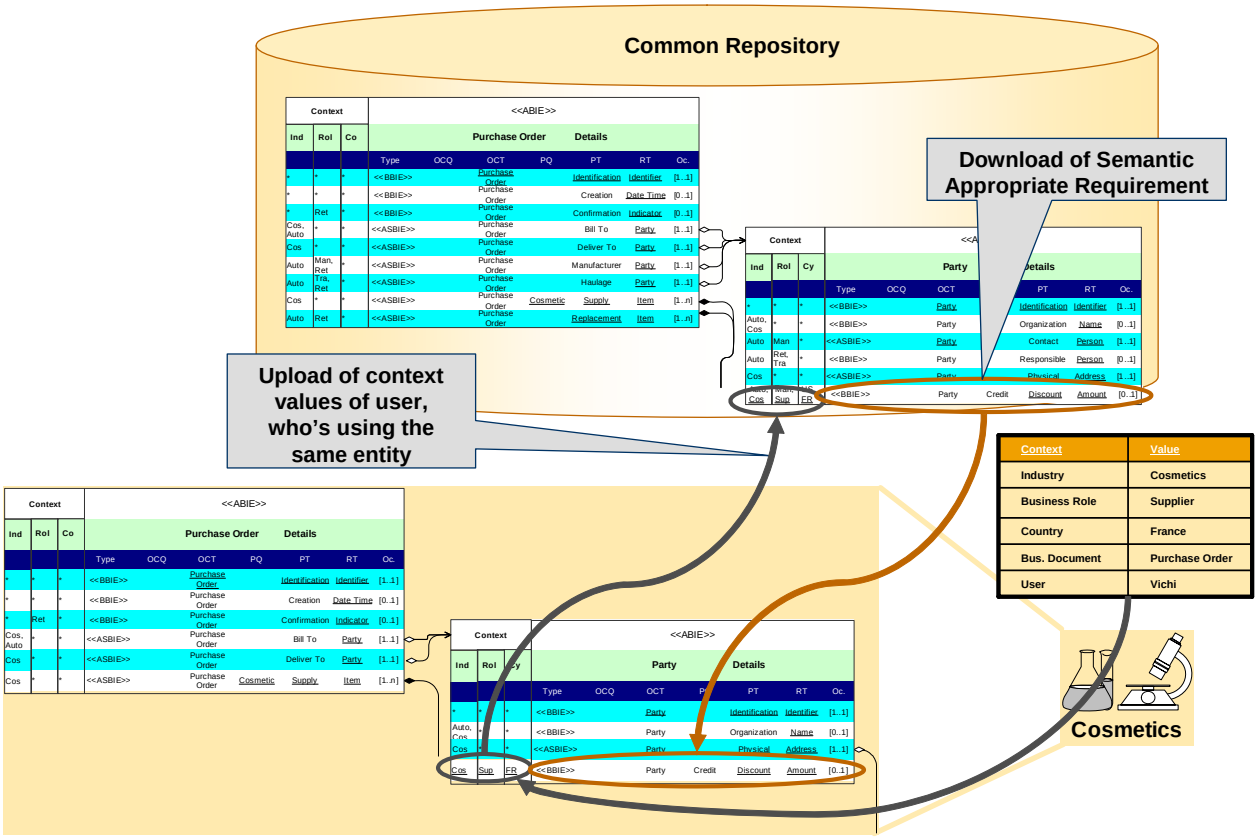


In this way, if another user such as the cosmetics supplier in France needs exactly the same extension, they do not have to develop this extension again. They can use the already defined contextualized and

stored BIE. Additionally, the cosmetics supplier context values will be added into the appropriate context categories so that other users will know that this extension is used in more than one context.

The simple example in Figure 12 shows how pre-defined business entities can be shared by a broad community, how the business information can be categorized and harmonized by analysis of real business requirements, and how the community can inform others of what is needed. It is possible that more than one car manufacturer in North America or more than one cosmetics supplier in France needs this extension. All users in a community can determine what they need, by the usage of the context specific classification. This principle is also very helpful for harmonization reasons, because only data that is used by many users in diverse context areas must be harmonized.

Figure 12 - Context specific selection of already registered extensions



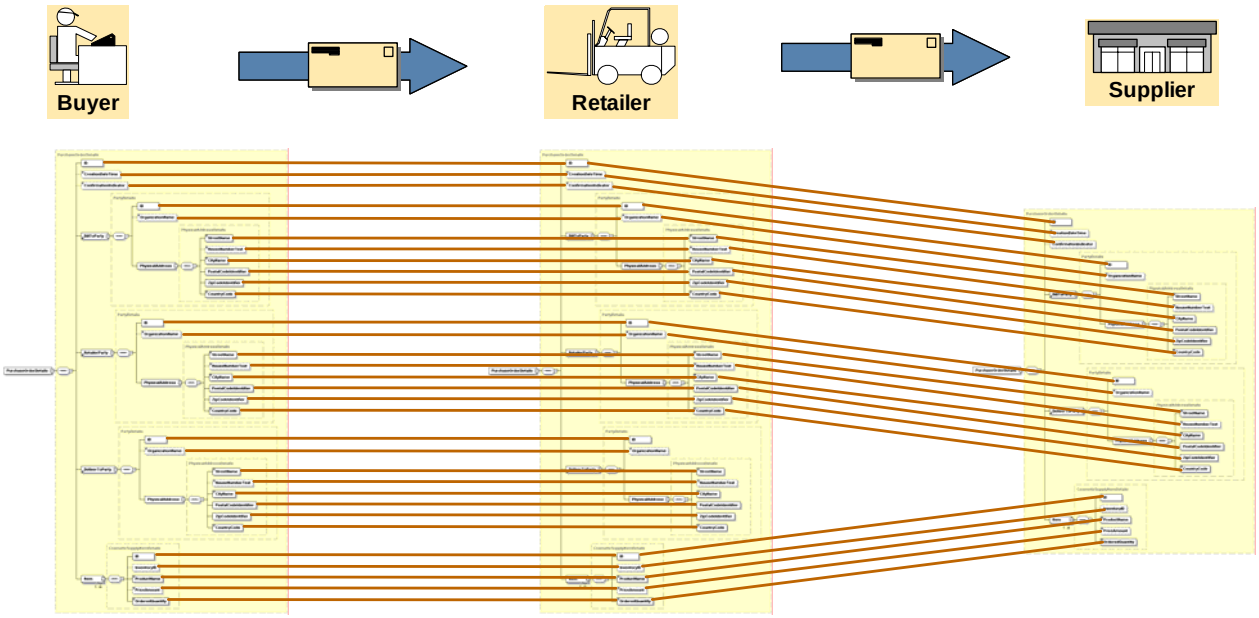
A clear defined governance model with predefined roles and authorizations and a fixed set of context values for each context category will enable users to define consistent and dependable business information entities. It will also avoid the development of duplicate and hard to use business data. UN/CEFACT provides this governance model through its Core Components harmonization group (TBG17)[10], UN/CEFACT CCTS provides many sources for specific context driver values, and the UN/CEFACT Techniques and Methodologies Group (TMG) is further defining the CCTS context mechanisms.

[10] See web page: http://www.disa.org/cefact-groups/tbg/wg/tbg17_main.cfm

The Context Driven Business Exchange Approach

Once trading partners adopt CCTS for the contextualized expression of their business information, they can achieve at their desire the same efficient implementation of interfaces in their internal applications that was described in the *Common Representation in all Implementations* section. Additionally, they will align their XML-based formats with CCTS constructs in an optimized XML schema definition (XSD) expression through the application of XML naming and design rules specifically designed for this purpose.[11] Such a solution will enable a direct exchange of the required business information without expending resources on proprietary mappings for each type of business exchange or data use.

Figure 13 - Real Interoperability in B2B



As shown in Figure 13, if the buyer, retailer, and supplier, as shown in Figure 2[12], adopt CCTS, they can exchange the same business data constructs (BIEs) with the same semantic meanings with multiple trading partners using XML based on the same sets of NDR's. Figure 13 also shows how context further refines the uses of the data. The "*Retailer_ Party. Details*" BIE will only be exchanged between the "*Buyer*" and "*Retailer*" business partners. The "Retailer" will use the exact same overall business information structure for business exchange of "Purchase Orders" with the supplier, but without "Retailer_ Party. Details" since it doesn't apply in that context.

Summary

For the last thirty years, the holy grail of business information exchanges has been a "mapping free" world. But no one has been able to achieve this because everyone is focused on their own requirements and defining their specific, inflexible business data models according their own assumptions. Compounding the problem, many solutions put all information for all business requirements into a single

[11] The UN/CEFACT XML Naming and Design Rules describe the XML syntax specific representation of CCTS artefacts in just such a fashion.

[12] Figure 2 shows a non-CCTS environment

data model. This has resulted in very complex, often incomprehensible and sometimes hardly implementable data models.

In this article we have described the optimal solution for business data exchanges in a B2B environment. As we have seen, a business data model is really no different than a sentence in a natural language. It gives information for specific business requirements in a given context in a more restricted way so that both humans and computers can understand and process it. Therefore, it makes sense to define a methodology for business data modeling that will have a similar grammar and vocabulary solution to achieve a common semantic understanding of business information in a given context for both humans and machines.

The UN/CEFACT CCTS international standard is the first methodology that contains these key features. As we have described, this methodology is not only advantageous for e-Business, it is also an integral part of modeling interfaces within applications using the same representation of the semantics. A complete implementation of CCTS within applications and between applications is an excellent contribution to the goal of reducing the TCO.

A CCTS-based registry/repository provides the requisite common knowledge for context driven business data models. It is a very effective and efficient way of discovering what is common in all contexts and what will be used in specific contexts. This is comparable with a general dictionary, like OED (Oxford English Dictionary), which can be used by everyone, and context-specific dictionaries for electronics, chemicals or aviation. These context-specific dictionaries includ all words in this specific context. But if you compare these dictionaries in detail, you'll often find the same words, which means that these words can be used in more than one context.

SAP has recognized the advantages of UN/CEFACT CCTS standard. Therefore, the new primary and reusable building blocks for business information in SAP NetWeaver™ are based on the CCTS approach. The so-called SAP GDTs (Global Data Types) are provided by the SAP NetWeaver XI™ and can be used equally and without loss of information internally in ABAP or Java applications. They can be used for business information exchange by the use of XML and other syntaxes. All new SAP applications which are based on SAP NetWeaver™ use the SAP GDTs exclusively. These GDTs are also expressed in XML using the UN/CEFACT XML Naming and Design Rules for Core Components. With the SAP GDTs now following this international standard, The SAP NetWeaver™ product has taken a huge step forward into the real, achievable, interoperable B2B world. More importantly, we are significantly reducing the large investments our customers have had to make in achieving business data integration.

Author Bio

Since his master's degree (MSC, 1993) Gunther Stuhec has worked with communications and EDI technologies. As a consultant in a software house for middleware and EDI systems he developed strategic concepts for customers and was responsible for various EDI projects. He joined SAP SI as a consultant in 1999, where he was responsible for implementing XML/EDI projects in conjunction with SAP systems. Since 2001 Mr. Stuhec works for SAP AG as a "Standards Architect" and has been involved in standardizing business standards on semantical and syntax level. He is chair of the UN/CEFACT Techniques and Methodologies Group (TMG) that is responsible for the development and maintenance of the UN/CEFACT CCTS standard. He is also a member of various international and national standardization bodies like UN/CEFACT, ISO, and DIN. He is actively involved in developing standards and serves as an interface between these bodies and SAP, introducing SAP's requirements into their work and incorporating their latest findings into SAP's development activities.