business transaction information management

What CAM Is

The CAM specification provides an open XML based system for using business rules to define, validate and compose specific business documents from generalized schema elements and structures.

A CAM rule set and document assembly template defines the specific business context, content requirement, and transactional function of a document. A CAM template must be capable of consistently reproducing documents that can successfully carry out the specific transactional function that they were designed for. CAM also provides the foundation for creating industry libraries and dictionaries of schema elements and business document structures to support business process needs.

Solving the Inherent Problem of Automated Information Integration

Automated information integration has been the Holy Grail of e-Business systems since before XML was conceived. Early attempts centered on the use of industry standard transaction formats typified by EDI messages[1].

These quickly showed that three needs were paramount: the ability to design transactions consistently, the ability to document their usage in a clear way and then the ability to drive software that can apply rules and test information content to ensure correct compliance.

The CAM approach provides these three critical abilities: –

- documentation of business interchange transactions,
- design-time assembly support with verification, and
- runtime checking of information content.

Next we consider the limitations of current approaches and technologies and how the capabilities that CAM provides can address these shortcomings.

The Present Tools and Their Limitations for Addressing the Problem

The advent of XML has also meant that other new tools are available to advance the technology of automatic transaction content handling. These include formal structure definitions using XML schema (***XSD***) with content control at the element level using datatyping; then ***XSLT*** rules and ***XPath*** expressions and semantic tools such as ***RDF*** and ***OWL*** to provide machine understanding of the role and usage of the XML based information itself. These tools typify the approach of controlling the actual XML instance documents and their content (e.g. transactions) by creating formal expression language that provides external descriptions about the possible instance documents that

[1] For more information on Electronic Data Interchange (EDI) – see N. American standards secretariat group site: http://www.disa.org

are allowed. The W3C organization has centered its efforts on this strategy[2].

People usually start this process by defining an XML schema that contains the rules of the actual XML structure they wish to use. This is often sufficient for simple small transactions between closed communities of business partners. However for general communities across whole industries this can rapidly become problematic as more and more context driven factors are added to the schema, since schema does not have by itself a native context driven selection mechanism. Further more a large need is the handling of industry lists for standard code values and particularly sub-sets of codes where again schema has no direct built-in method for this.

Another approach is to use UML[3] modelling techniques to create models of the information use cases, classes and instances and then derive XML schema definitions from these UML structure diagrams. The ebXML core components work and the OASIS UBL technical work have both used this approach. UML itself however has no mechanisms for capturing complex cardinality relationships in the models. Therefore this approach is limited in the same way that XML structure models in XSD are as well. Also industry dictionaries of component definitions are not well integrated into UML tools through federated registry capabilities. However, CAM supports this capability through its direct support for ebXML Registry systems and the ability to derive semantics in XML from such registries.

The Issue of Context in Business Interchanges

In automating information integration, knowing and defining context of use is the single most pervasive and important factor.

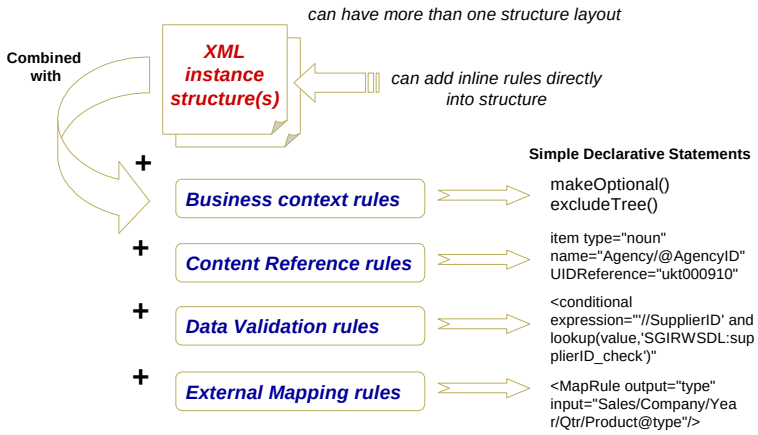
2 The World Wide Web consortium (W3C), were the creators of the XML syntax – for more information on XML. XSD, XPath, RDF and OWL see http://www.w3c.org

3 Universal Modelling Language (UML) as originally developed in Rational Rose and now part of the Object Management Group (OMG) specifications.

The older EDI systems however have no way to support dynamic context driven content assembly whereas CAM provides this better than any other XML syntax system today. In fact CAM templates work for both old EDI payloads and new XML based transaction structures.

What is CAM comprised of?

A CAM template has five sections: assembly structure(s), business context rules, content referencing, data validation and external mappings. They can be used



altogether or in combination. The first two sections are required; the remaining three are optional, so a CAM template can be as simple or sophisticated as the business needs dictate, see figure below.

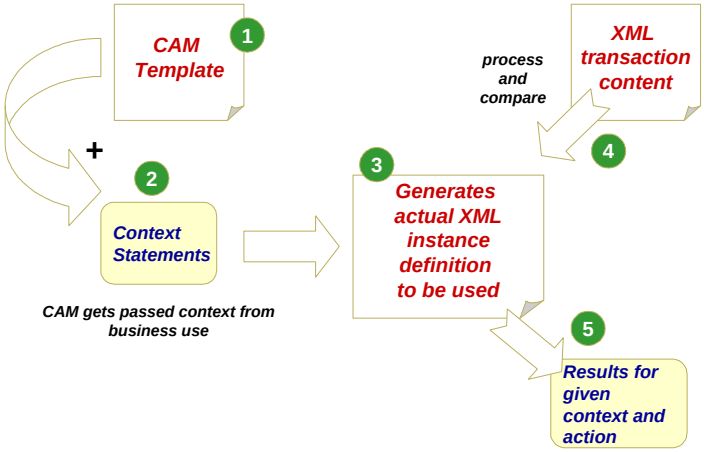
What does the addition of context to CAM do?

In addition the CAM system provides a global context passing mechanism, and an XML rendering for declaring context variables. This context mechanism is at the heart of why CAM is so effective. By allowing users to quantify what their context factors are precisely, this removes the guess work from business transaction exchanges between partners and allows them to formulize their collaboration agreements exactly. It also makes re-use and identifying potential candidate CAM template models much easier. Simply knowing what the external parameters are can instantly qualify if a template makes

sense for a user and allow them to therefore select it.

How Does CAM Execute?

A CAM implementation requires that a reference CAM template be available along with the CAM processor engine itself. The available jCAM processor[4] is implemented in Java and so can be used in a variety of environments. Along with the CAM processor will typically be an XML instance to be manipulated by the processor. Also CAM maybe used as a web service and sample WSDL binding definitions are available to facilitate this.



A practical example of what CAM does

Next the following figure shows how the CAM template details are used to create a specific transaction instance. Consider for example a scenario from the current automotive industry work on aligning inventory control systems using the Open Applications Group (OAGi) business transaction formats called BODs (Business Object Documents). In the case of such OAGi BODs the CAM template provides the context driven rules and structure instances derived from the master BOPD models that given a particular car manufacturers requirements will create the actual resolved structure and rules for the OAG BODs inventory management information exchange required.

Each BOD also has associated with it a complete list of each information element and basic details of its content. The CAM technology has been designed fundamentally to leverage such dictionaries of noun definitions and so the available OAGi BOD semantics provides the perfect match for this. The content reference section of the CAM templates provides the means to facilitate this.

The next section provides and overview of how to use such specific CAM features and techniques.

How Do You Create CAM Templates and Rule Sets?

There are several steps required in building a CAM template depending on how complete you would like your template to be. To fully utilize CAM you may want to fill out all six sections of the CAM template. Most initial templates however only use two or three sections for structure, business context and content reference sections.

The first thing you require when constructing a CAM template is an actual XML model of the information structure you wish to manipulate. You can either enter this directly from scratch into the CAM template, in XML, or you can cut and paste an example from some XML that you are already using. So in the example we are considering – you would create a sample instance for a car manufacturers system of a typical document transaction.

Next you go through that XML structure and do four things:

- mark each field where information will occur with the “%%” substitution placeholder
- physically enter static information exactly as and where it should occur in the XML
- duplicate parts of the structure that may be choices, and label those parts with selection names (e.g.

[4] The jCAM processor is open source can the resource site is http://jcam.org.uk for the project.

as:choiceID="USA-address", and as:choiceID="Canada-address")
- assign inline rules using the CAM functions to denote local formatting and other conditions right into parts of the XML structure as needed (e.g. as:setMask="dYYYY-MM-DD").

Once this is completed then you can move to the Business Context section of the CAM template. CAM assumes all elements and attributes are required in a structure unless you denote them otherwise. So the second step is to indicate the CAM elements that are optional and also elements that are repeatable. You can also use XPath conditions to express business rules choose when things should be used. So you may make an element optional only if a certain condition applies. You can also omit parts of structure, or include others depending on conditions, and link structure components together so if one is selected another is then required as well (conditional cardinality).

As you create these business rules you may find some conditions relate to values in the XML structure associated with the CAM template itself, or others may come from global parameters that you need to pass to the CAM processor. These global parameters can be defined in the header section of the CAM template and then referenced in the business rules or the XML structure itself using the $name syntax[5].

After creating business rules for the use of the structure, you then go to the next step in the template by entering the content reference section. In this section you can provide standard definitions for elements. You also need to use this section if you want to take advantage of the extended features CAM has for validating attribute use and their content.

The idea of the content reference section is to build up a library and dictionary for your commonly used elements across CAM templates, but you can also use it to just locally define field details as well. To complete the content references you simple provide statements for each of the elements and attributes in your XML structure that require additional information processing. Each statement contains the XPath of the item in the XML structure cross referenced by a UID code value that uniquely identifies it. You can assign UID codes simply and easily using a suitable prefix that you select followed by a 6 digit number (e.g. "USPS000100", "USPS000101", and so on could reference fields in a US postal address structure).

Each content reference statement can also optionally contain data typing, field length and other semantics about your fields.

You can use include statements in CAM templates to optionally retrieve pre-built structure sections and associated content reference sections into your current template. This allows you to develop libraries of commonly used components, such as address layouts, product details, and customer information and then simply insert them as pre-built assemblies directly in where you need them. Also, because CAM can resolve syntax use through the content reference section you do not need to use cumbersome namespace references to achieve this. Therefore re-use of structure components is made dramatically easier.

The next step is to provide specific data validation rules that you may require. Typically these are rules that occur between and across fields in the XML structure, such as comparing the value of one date field to another date field. Again these use XPath expressions to construct the rules you require. For really complex conditions you may also choose to invoke an external module, such as a web service, to do the testing and return the result back to the CAM processor.

Often you will have completed your CAM template at this point. However you may need to relate content using the external mapping section. This allows you to manipulate information and integrate it with an external information source or format such as to a SQL

[5] e.g. $manufacturer="GM", $language="US-ENG"), or <description lang="$language">%%</description>

database table, or to a HTML form layout, or to another XML structure format.

This is an advanced feature set of CAM however, and depending on what information integration, (either inbound or outbound), that you need will then affect what rules and statements you put in this section.

You may also need to define an additional structure section definition, for instance for HTML or XML targets that you will be using and referencing. This is also the most powerful section of the CAM processor and you can provide complete content assembly and manipulation of business transactions and information using it.

This completes this overview of creating a CAM template. Next we consider utilizing these capabilities of CAM alongside traditional components of an information infrastructure to deliver enhanced system interoperability and agility.

How Does CAM Integrate with Other Application Integration/B2B Products?

The most obvious use for a CAM processor is in validating information content passing through a messaging system, and particularly a messaging hub. Here the CAM templates can contain business rules to ensure that the information received is compatible with the business systems supported by the messaging system. To complete out the picture a CAM processor may be used to re-structure the information presented to it, after first validating that it is correct.

In addition a business process engine may use a CAM processor to direct the creation or processing of business transactions for it. In this case a step in the business process will have a CAM template associated with it that will be invoked. The CAM processor can then produce the desired information processing for the business process engine.

Next a CAM processor can be deployed as a web service to allow business partners to pre-validate XML instances before using them in message exchanges. By presenting an XML instance to the web service and selecting a CAM template the partner can have returned a detailed report about the results of the tests and their outcomes.

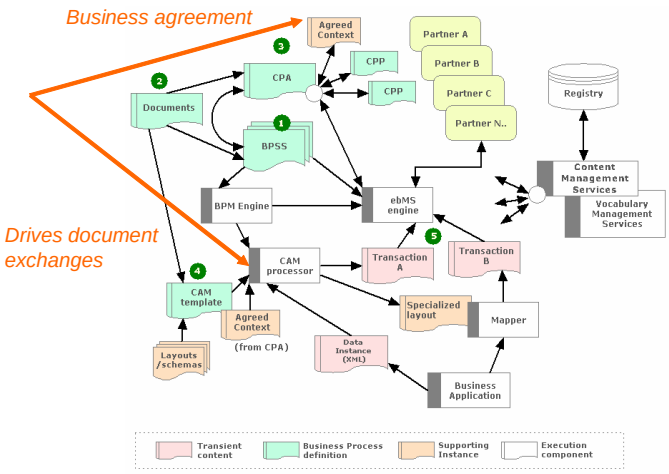
Within industry standardization efforts CAM templates can be used to capture all the design rules that apply to an industries use of XML structures. The CAM processor can then provide a validation and conformance service. Industry groups can also create CAM templates to document older legacy transaction formats and provide XML based semantics and usage rules for those.

When integrating with E-Forms systems the CAM processor can be used in a variety of modes. It could create forms given some input business transaction, or it could take the output from a form entry process and then validate it and re-structure the content and deliver it as a business transaction.

In short the CAM processor can be used in a variety of roles wherever manipulation or validation of information content structures is required.

The figure shown on the next page shows how CAM can be integrated into an eBusiness stack providing the orchestration of business transaction handling along with typical components of the eBusiness stack such as message handling services, business process management engines, application integration mapping services and industry registry of dictionary definitions and vocabularies.

Integrating CAM into an eBusiness stack



A Real World CAM Scenario

The original implementation of CAM by British Telecomm (BT) in the UK is providing transaction validation against hundreds of trouble ticket message formats that BT receives daily. These trouble ticket formats used in the telecommunications industry are XML messages using the xCBL syntax approach and are highly complex XML structures. The information carried in them is highly variable and differs according to the originators own telecommunications switches and configuration details. The message details change frequently every three to four months as new services and products are introduced and provisioned in response to demands from highly dynamic and technology driven markets within the industry.

Managing these structures and formats takes significant manpower and especially to detect when solution partners have made changes to their transactions and details without first notifying the systems connected to them. This may also happen inadvertently when new devices are installed in the field and they start producing new trouble ticket reports automatically.

Being able to direct and manage this implementation space using the XML scripts in the CAM templates significantly reduces this maintenance burden and also allows changes to be implemented rapidly and easily.

Partners can also begin to become involved in the process themselves by managing and maintaining the CAM template definitions that relate to their own production systems and configurations. They can then share those templates with their partners when changes are about to be placed in production.

How Does CAM Work With or Compliment Other Semantic Definition Initiatives (UDEF, UBL, OAG, RosettaNet, HL7, STEP, eprXML)

The CAM technology has been designed fundamentally to leverage dictionaries of noun definitions and libraries of transaction formats built from industry initiatives. By providing the content reference section in the CAM template the processor is able to automatically retrieve semantics from industry registries about individual elements in a XML transaction. This allows consistent definitions to be deployed across a set of industry business transactions. It also includes the ability to version and sub-version definitions. This is especially important for the processing of code lists. Code lists provide the basis for up to 50% of information flowing in legacy EDI transactions. Industry groups define extensive code lists for the products and services used by their members. Yet the W3C XML schema system does not support code list definitions directly. Therefore CAM provides a key capability with its built-in *lookup ()* functionality.

Industry initiatives today have created dictionaries of their nouns (elements) and verbs and labelled them with reference codes. They have also provided classifications and taxonomies for these as in the case of UDEF particularly. By allowing these industry groups to load their dictionaries into XML instances in a registry the CAM approach unlocks the potential of these dictionaries to effect significant improvements in interoperability of information in those industries.

Providing the means to assemble consistent transactions from pre-defined libraries of structure components and also to create

libraries of business process definitions that can be context driven are key functionality that CAM delivers.

By linking a CAM processor service to the registry these industry groups can then allow members to validate XML transactions against the standard definitions in their registries or download CAM templates and associated business process definitions that they can use directly in their own systems.

By combining CAM technology with a registry in this way provides the foundation for even more sophisticated semantics in the future. Taxonomies and ontologies are emerging as ways for machines to understand more about knowledge in a way that today only humans are able to. Registries can provide ontology and taxonomy driven search results. This allows agent software and human researchers to more effectively locate business transaction processes, such as CAM templates, and determine if they are suitable for the task they need.

Enhancing the work of industry groups is a significant benefit that CAM technology delivers today.

CAM can also be used in tandem with UML. UML does have a production rules system that can be used to output XML from UML diagrams and it is anticipated that this can allow UML models to output partially complete CAM templates that can then be hand-edited in XML to complete a full information model.

Perhaps one of the most exciting potential uses for CAM is with the emerging OASIS work on EPR (Electronic Process) specifications for service oriented applications[6]. Since these are required to be fully model driven, having the CAM templates able to resolve dynamically through context parameters the information exchange formats allows designers of eprAPL (EPR Application) solutions a rapid and consistent way of implementing these.

[6] For more information on the EPR work and the eprXML and eprAPL components – please see the OASIS BCM TC area at http://oasis-open.org

Summary

Reviewing what CAM represents and the opportunities it opens up we find a range of use cases; from sophisticated discreet tasks as part of specific architectures such as ebXML business processes or Service Oriented Architectures environments, to simple local document verification as a standalone business tool.

The value of the CAM approach is in managing context and business rules directly and tying these to consistent information semantics. The lesson learned is that consistent information exchanges must include context management. The result from this will be to transform the ability of industry to deploy successful and interoperable eBusiness information exchanges simply and quickly.

Contacts and Additional Information

The CAM technology is a product of the OASIS Business Content Assembly Mechanism Technical Committee. More details can be found from the specifications and documents that the committee has produced. These include the latest CAM specification, presentations introducing CAM, and examples of CAM templates. These and other documents can be obtained through the OASIS website at: www.oasis-open.org.

For more information on how to participate in CAM activities, please see the membership links from the main OASIS web site or the CAM TC area directly.

Content Assembly Mechanism Technical Committee

Chair: David Webber – drrwebber@acm.org

CAM Content Assembly Mechanism