Introduction

When I was head of architecture at the newly global Dun and Bradstreet I needed to change my thinking from that of a software vendor, which I had recently been, to that of an information provider. To do that, I turned to value chains as a way of thinking about making information products. Michael Porter had created the concept of a value chain in the mid 80s (this was 1998). Porter's value chain supposed that the chain of supply and the chain of demand combined to form a single chain of value. His chain covered manufacturing and consisted of inbound logistics, manufacturing, outbound logistics, sales and support. Luckily I quickly found a Harvard Business Review article by Rayport and Sviokla on the virtual value chain. Having just worked on a data warehouse for Sun Microsystems, I recognised their value chain as being almost exactly the information pipeline we were using at Sun. A little more investigation showed that we could map almost all our data integration processes at D&B to the virtual value chain. In the twelve years since, this virtual value chain has worked without fail as the way to integrate information for an enterprise. This paper describes our view of this virtual value chain and how it can be used in the enterprise to model any process of interest to that enterprise. The relationship between the Porter value chain and the virtual value chain is shown in the diagram below.

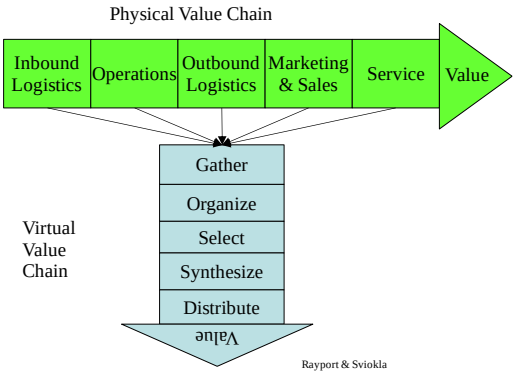


Figure 1 Real and Virtual Value Chain

Since the work at D&B we have seen many different virtual value chain implementations in many different industries. In all cases, these have been successful if three golden rules are followed. Where any one of the golden rules is broken, the implementation has been unsuccessful (which is why we think they are golden rules, not just guiding principles). To start with, we will go over the golden rules. Then we will go over the elements of the virtual value chain, including certain elements that Rayport and Sviokla may have overlooked.

Golden Rules

The first golden rule is to separate supply from demand. This is the rule that sets up the value chain. Separating the supply from demand means that you need to model supply separately from modelling demand. In other words, you need at least two databases for your value chain: one for supply; and one for demand. The supply database will need to be a record keeping database, that is have an entity relationship model and follow normalisation rules. This database will have a single data model that unifies all the sources that make up the value chain. The demand database will need to be a reporting database, that is have a multi-dimensional model and be de-normalised for reporting use. Immediately with have two parts of the virtual value chain: putting data into the supply side database we call 'Organising'; and putting data into the demand side database we call 'Making Selectable' following Rayport and

Sviokla. The difference between their view of these stages of the value chain and our view is that between the two stages we put a publish and subscribe queue into which all changes are placed. The reason for this will become clear later on.

There are two obvious ways in which you can break this golden rule. Firstly, you can only model supply. This 'supply-only' value chain is typical of initiative lead by IT rather than the business. It is often characterised as 'build it and they will come' but unfortunately, they never do. When I was consulting for Cendant I was told by the IT people that they had built a house holding database with 100 million house US holder records in it. This was built in their Hospitality division which owned several large US hotel chains. I was very impressed until I heard from their business people that they had never managed to get a report out of it. The second obvious way to break the rule is to build the demand side only. This is typical of business initiatives that do not involve IT. Very often the database is implemented in a spread sheet or an access database. Sometimes it is implemented in a more sophisticated tool such as Cognos TM1 but fed by spread sheets. In all cases, the problem is that the data is often inaccurate, often inappropriate and seldom complete.

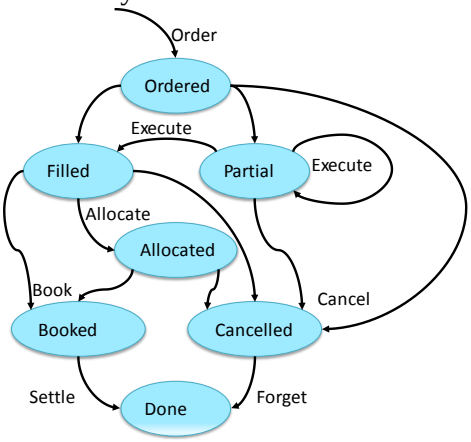


Figure 2 Trading Process

The process models the state of a trade and has seven states and seven events. If you are asked, for example, to provide post-trade reporting for clients of a broker you could use this model to help you. If all they are interested in completed trades, then you don't have to worry about partial fills and allocations, you just have to show the booked and cancelled trades. This means you only need to find the book and cancel events in the source systems. However, by modelling the whole process, you can meet the current demand and start the incremental building of the whole process. Because, as sure as day follows night, once you have met the demand for post-trade reporting, the business will start asking you to do the rest of the process.

The second golden rule is to drive the value chain from demand, not supply. That is, find out what the business user wants then construct the reporting model that satisfies their demand and then find out what record keeping systems can source that model. This approach stops you from 'boiling the ocean'. It also enables you to grow your virtual value chain, if necessary, report by report. What we have discovered over many reporting engagements is that you can almost always model the demand as a process. That is, as a state change model of some entity that changes as business events occur. For instance, if you are modelling trading in capital markets, you might model the trading process as the states of a trade, as shown below.

The third golden rule is that you must source the value chain with business events not data extracts (unless the database contains business events, in which case it is really just a log). This rule comes out of the approach outlined in the previous paragraph. The events that change the state of a trade are business events. It is certain that these events are recorded somewhere in the operational systems of the enterprise, but it is not certain that they are recorded as transactions. Often the record keeping systems update the information in the business events in their databases. For instance, they may record the new balance rather than the amount traded, in which case, if there is more than one transaction

during the day for that account, the second transaction will overwrite the first and information will be lost. This was the case when I was consulting at CIBC. They found that their data warehouse would not always reconcile with their General Ledger. Every so often there would be a discrepancy. When we analysed the two data gathering processes we discovered that the GL gathered transactions summarised into GL accounts, but the data warehouse extracted data from databases.

A second problem with using data extracts is that they are likely to be run in batch at the 'end of day'. This means that they can never supply reporting data, especially for dashboards, in near real time. If you are providing a dashboard of train arrival times, it is useless to extract the information overnight.

A third problem with using data extracts is the problem of 'changed data capture'. The data in the database does not, in general, show what has changed since the last time the extract was run. Many data warehouses get around this problem by shipping all the data and comparing it to what the data warehouse already holds. This is very expensive and gets more and more expensive as the size of the data warehouse and the record keeping data increase.

At the beginning of this paper, we showed the relationship between the real and virtual value chains. This was a little too schematic as we know, from the front middle back paper, that the record keeping systems are bound to follow a front middle back pattern no matter which part of the real value chain they are for. So a more general view of the relationship, in information system terms, of the two value chains is shown below.

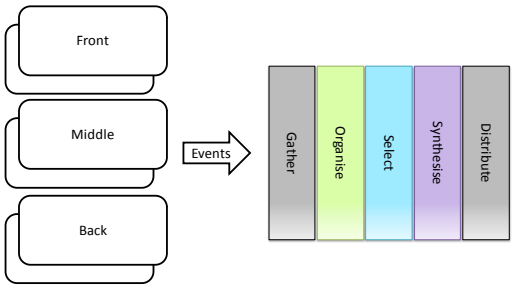


Figure 3 The two value chains

'Front middle back' is the information systems view of the enterprise's real value chain, no matter what kind of enterprise it is (not just manufacturing) and this provides the events that are gathered as the raw material of the virtual value chain. If there are events needed for the virtual value chain that are not normally recorded in the real value chain, these must be obtained as reference data from another source. An example of this is market data (the prices and volumes of instruments traded on exchanges or over the counter) that may be needed in capital markets reports.

Gathering

Gathering for the virtual value chain involves arranging for business events to be sent to a certain place from the systems that recorded them. In general, these events then have to be checked both that they are syntactically correct and semantically correct. If an event has a syntax error it must be rejected. If it has a semantic error (for example, it refers to a product we don't know about) it should be pended and retried. Eventually, it should be sent to an error queue for a human workflow to deal with it. Correct events are sent on to the organise stage. Gathering can be done in real time or batch. Events can be sent to organising using FTP, HTTP or MQ. Although ETL is acceptable as a transport mechanism for collection (effectively just the load bit) it is unlikely to be effective at the syntactic and semantic checking. Many data warehouse systems try to load the transactions into a data store first and then check for validity. This is a poor approach

as it creates a data store of bad data. It also prevents the system ever being real time. It is not really possible to combine collecting and organising, they are separate steps.

The logical architecture of gathering is shown in the diagram below.

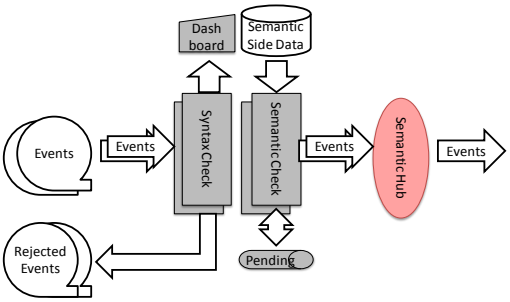


Figure 4 Architecture of Gathering

Organising

Organising applies to two kinds of business event: a transactional business event such as an order for a product; or a master business event such us updating a person's name and address. Organising master business events is often called master data manager and it may be possible in many cases to combine the role of MDM with that of organising. Organising master events involves matching the event to see if it is a new one, in which case a global identifier must be allocated, or an existing one, in which case it may be necessary to link a new local identifier to the global one. This happens when two different source systems both insert the same new master element, for instance when both HR and Payroll add an employee. If probabilistic matching is used, it may also be necessary to send the match and its possible candidates to a human to resolve. When a master entity changes the change must be recorded. It is probably not acceptable to update the record in the database. Instead a new record should be inserted for the changed values showing the time from which they are valid. This is sometimes called 'time domain record keeping'.

A special form of master business events is the set of reference data needed by reporting. This is data received from outside the organisation.

The bulk of the events are transactional. These are classified for their parties using the global identifiers. Parties include the buyer, seller, agent, product, place, time and so on. Transactions for the same business event will normally be inserted into a table for that event. As these tables quickly grow large it is normal to spill them to archive every month or so.

In both cases, domains (that is, possible values for the attributes of the recorded entities) are standardised, for instance all use the convention 1 for Female and 0 for Male in the Gender domain.

The result of accepting an event into organising is that it emits a change, either master or transactional, to a 'daily' change queue. This queue of changes is the interface between supply and demand. The selecting stage is one subscriber of this queue, each real time synthesiser is also a subscriber. The logical architecture of organising is shown in the diagram below.

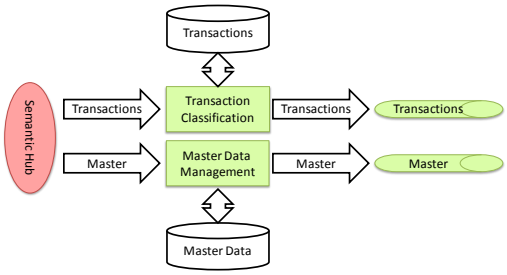


Figure 5 Architecture of Organising

Selecting

The third stage of the virtual value chain is selecting. This reads the queue of daily changes (if it reads them more often than daily then it is probably better just to call them queued changes) and does two things with them. The master events are used to update the dimensions. The issue of dimension change can be handled in several ways. These are the type 1, 2, 3 and 4 dimensions. Using type 2 dimensions, where each change is a new record for the entity, is the default approach. The transaction events are used to insert transitions in the fact tables as a

new time series.

This creates an ‘atomic star’ data store. At this stage there is no aggregation or summarisation of data.

Movement of events from the daily change queue to the data store can be done using ETL middleware. Most middleware will support a ‘publish and subscribe’ queue as an input. The logical architecture of Selecting is shown in the diagram below.

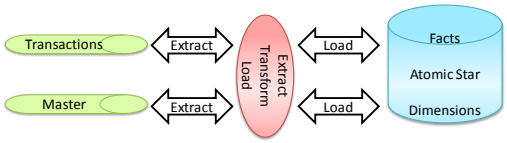


Figure 6 Architecture for Selecting

Synthesising

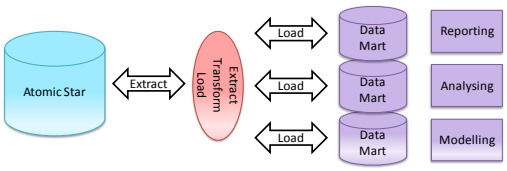
As soon as Selecting completes, synthesising starts. There are two parts to synthesising. The first creates the data stores to be used for reporting. The second generates the reports themselves. This can be done by users running their reports or by agents generating reports.

Creation of the databases is best done using ETL. It extracts the new time series and the dimensions (it is probably best to refresh the dimensions completely) and pushes them into the reporting databases. The new time series is an insert, the refreshed dimensions is a replace.

Whereas the data storage technique for organising and selecting is likely to be relational, things are not so clear synthesising. The accepted approach is to generate cubes which can be relational, but in many cases can be much faster using more directly multi-dimensional database management systems.

Synthesising can be based entirely on integrated data, in which case time series are always complete and the only change is when the new time series and refreshed dimensions are received. Or synthesising can be entirely real time, by subscribing to the daily change queue to add transactions and change entities as they arrive. This is a good approach for real time dashboards. Or synthesising can be both, in which case at any time there will be an incomplete time series building up with transactions that have not yet been fully integrated. When the new time series arrives, the incomplete one can be deleted and replaced with the complete new one.

The logical architecture of synthesising is shown in the diagram below.



Distributing

The final stage of the virtual value chain is the distribution of the synthesised reports to the interested parties. This can be offered as a push of content from synthesising to operational systems that need integrated data. For instance, the collecting systems need a push of data from synthesising so that they can check the next batch of data for semantic validity. Similarly, the semantic hub needs a push of data so that it can convert entity references from source end points to entity references in target end points. It may also be necessary to distribute synthesised content outside the domain that creates it, for outside use. It is unlikely that another domain will use synthesised information products (they should rather be getting a push of the business events causing change for their own synthesis) but people will want to get synthesised content for their own use. Typically the agent used is some kind of report writing or formatting tool, most typically a spreadsheet. Integrating GET for synthesised content into such tools is a major part of distributing. The logical architecture for distributing is shown below.

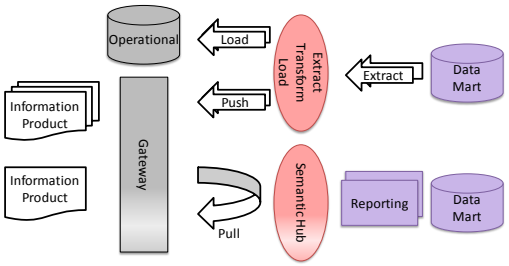


Figure 7 Architecture for Distributing

The diagram makes clear that in general what is being distributed is an information product. This is true for distribution to internal staff and to external customers, suppliers, partners and regulators. There can be a push of a set of information products (for example, pushing credit reports) or a pull of a single information product (requesting a credit report).

Metadata Management

The virtual value chain can be thought of as an engine that takes raw materials, in the form of business events, in at one end and produces information products at the other end. The engine needs oil to run smoothly and continually. In the virtual value chain the role of oil is played by the metadata that is needed at each stage of the value chain. Without good management of metadata it is very hard to sustain the value chain in the face of change. Source events change and the reporting databases change outside of the control of the person running the value chain. In addition, we want to be able to grow the value chain incrementally, which itself implies continual change.

To understand the need for metadata better, it is useful to view the value chain as a sequence of schemas (descriptions of data) and transforms (moving the data from one description to another) as shown below.

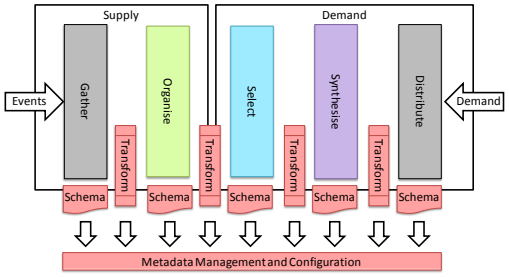


Figure 8 The metadata view of the virtual value chain

As you can see, the value chain is completely described by the schemas and transforms. If we can put all this metadata into a single metadata warehouse in a coherent meta-base, then we can use that meta-base to provide three things: a description of each data element in each stage of the value chain; the impact of changing any data element; and the lineage of any data element. In particular, data lineage is needed to provide the users of the information products with the business meaning of the data in their reports. This is illustrated below.

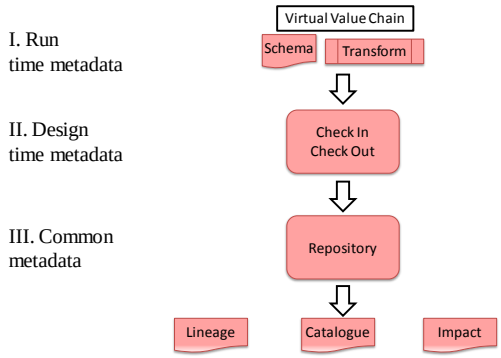


Figure 9 Metadata Value Chain

This is the metadata warehouse, sometimes called the knowledge value chain. It clearly distinguishes the design time metadata store, which is often supplied with the ETL tools, from the repository. It is quite difficult in today's environment to find a product for the repository. The repository is in effect a multi-dimensional view of metadata where version plays the role of time and place in the virtual value chain plays the role of geography.

Advanced virtual value chains have a business description of all the things that

can be in the value chain. This is known as an 'Ontology'. The holy grail of enterprise architecture is to have a complete and stable ontology of a domain and to be able to map each incoming data element to the ontology. This eventually makes it possible to generate all the transforms in the pipeline by mapping an incoming business event to the ontology.

Configuration

In addition to metadata that keeps the value chain running, you also need configuration information. This is used to set up new or changed event sources, new or changed data stores and new or changed demand. Ideally, most of this should be easy enough for the user to manage, though the reality is usually that IT does the configuration.

Workflow and Dashboards

Finally, workflows are needed at all stages of the virtual value chain. Crucially, a workflow is needed for managing feeds. There may be rejected or pended transactions. If a transaction is pended, a human workflow may need to be run to decide how to solve the semantic problem.

If organising fails to match an incoming entity, a workflow is needed to select the matching entity.

For the remaining stages, the ETL tool typically provides the needed workflows. Though that clearly depends on the choice of tool.

About John Schlesinger

John Schlesinger is a Principal at Atos Consulting where he leads its Enterprise Architecture practice. John is an advisor to enterprises specialising in middleware and integration architecture. He has lead integration architecture development in retail banks, investment banks, retailers and manufacturing, both for integrating applications and for integrating information.

John has worked both as a consultant and also as a developer with software companies. He has taken over two dozen program products to market at IBM, Information Builders, One Meaning, SeeBeyond and iWay Software. These products included the world's most successful commercial software (CICS) and the world's most successful data middleware (EDA/SQL). John also led the Architecture department at Dun and Bradstreet when its IT department went global.

A member of the ACM and the IEEE, John has an MA in Physics and Philosophy from Oxford University and a Post Graduate Diploma in Software Engineering from Oxford University.

John has spoken at numerous conferences including the CIO Cruises run out of New York, during one of which he was the first speaker on after the collapse of the World Trade Towers in 2001.

John can be contacted at john.schlesinger@atosorigin.com