Smart SOA

IT Foundations and Enterprise Relativity

Piet Jan Baarda

An approach for defining the IT foundation for an enterprise is investigated in this paper. The main premise is that the foundation of enterprise IT must have sufficient quality before anything can be built on top of it. Another premise is that such a foundation can consist of a portfolio of services supporting the implementation independent essence of the enterprise. Our proposed Smart SOA approach addresses the definition of such a service portfolio in a practical way, by only capturing those services for which a business case exists. In our Smart SOA method no organizational boundaries are used. Instead a service portfolio is defined based on an enterprise whose boundaries are determined by a service business case. Organizations do play a role as the legal entity owning and maintaining service portfolios. The details of Smart SOA still need to be filled in. As a first step the DEMO method, for developing a model of the implementation independent essence of an enterprise, is reviewed as a possible candidate for incorporation in our method. It is shown that 'enterprise' and 'essence' are relative concepts. Real world artifacts can only be classified in DEMO terms after a perspective has been selected. Examples are given where this is demonstrated. The most illuminating examples come from enterprises providing documental, informational or business services to their environment. Our Smart SOA method needs to encompass an analysis phase for defining the enterprise before the DEMO method can be applied.

Introduction

For a long time I have been wondering why organizations continue to spend huge amounts of money and effort on IT while the shortcomings of their IT foundation are ignored. They spend money on the latest IT hype while remaining unable to identify their customers, have great difficulty introducing new products and sales channels, are unable to present a consistent face to their environment, take a decade to incorporate an acquisition, and request the same information from their clients again and again.

- Want customer intimacy? Buy a CRM solution.
- Want agility? Buy an ESB.
- Want integration (finally)? Buy a best of breed EAI solution (sigh).
- Want scalability? Put it in the Cloud.
- …

Should the IT foundation not be put in order first before stacking more IT solutions on top of it and making things worse?

Yes, of course. Common sense dictates that anything built on a soggy foundation is bound to fail. And indeed it does. Still, for some reason, it is a general pattern.

Why it exists can possibly be explained, if you like conspiracy theory, by the presence of a Consultancy/Top management/IT industrial complex. Just like the military-industrial complex Eisenhower warned us about in his farewell address to the American population in 1961. However interesting that analysis may be, it is *not* part of this report.

This report focuses on getting the IT foundation right.

Let us assume that the decision makers of our organization have seen the light and want to get their IT foundation right once and for all. They have asked their enterprise architects to come up with an approach. The ideal foundation allows the organization to thrive in an ever changing environment. The only assumption the architects can make is that the *essence* of the organization will not change. In other words we do not have to prepare for a scenario where the organization changes, let us say, from a bank into a shoe factory. In that case it is acceptable to discard the foundation and start anew. That makes very much sense.

Getting the foundation right first

How should the architects go about defining and realizing this foundation?

As the essence of the organization is the only thing we can assume to be stable, our foundation can only be based on that essence. The challenge is to make it independent of any implementation of that essence. Otherwise its expected life time would be the same as the life time of that implementation. So we are looking for the *implementation independent* essence. Our task is also to enable the organization to *thrive* in its unpredictable environment. For our competitors the predictability is the same; zero. To thrive we need to be better prepared for the unknown future than the competition; we need to be more agile. Agility is defined as the ability of an enterprise to thrive in a continuously changing and unpredictable environment (what a coincidence!) [Conboy, 2004].

Based on these demands we will look into the following disciplines:

1. Enterprise Engineering. Enterprise Engineering is a discipline that studies the enterprise from an engineering perspective. We expect the DEMO method [Dietz, 2006] to be helpful in determining the implementation independent essence of an enterprise. Such a model exactly covers the stable core we want to support with our IT foundation.
2. Enterprise Architecture with the SOA architecture style. The main promise is enterprise agility and that is exactly what we want.

First we investigate the difference between an organization and an enterprise. The architects are employed by an *organization* and are supposed to serve *its* purpose. On the other hand we talk about *Enterprise* Engineering and *Enterprise* Architecture. Do we need to differentiate?

Organization versus enterprise

As said we talk about an *organization* because the enterprise architects in our scenario are employed by an organization. We simply define it as:

> *A legal entity that is not an individual person.*

The definition of enterprise we use is from Robbins [Robbins, 1990]:

> *A consciously coordinated social entity, with a relatively identifiable boundary, that functions on a relatively continuous basis to achieve a common goal, or set of goals.*

While an organization is an enterprise, an enterprise is not necessarily an organization.

Dietz' Enterprise Engineering Manifesto ads the engineering aspects to the definition of enterprise [Dietz, 2010a]:

> *In order to perform optimally and to implement changes successfully, enterprises must operate as a unified and integrated whole. Unity and integration can only be achieved through deliberate enterprise development (including design, engineering, and implementation) and governance.*

And

> *Enterprises are essentially social systems, of which the elements are human beings in their role of social individuals, bestowed with appropriate authority and bearing the corresponding responsibility. The operating principle of enterprises is that these human beings enter into and comply with commitments regarding the products (services) that they create (deliver). Commitments are the results of coordination acts, which occur in universal patterns, called transactions*[1].

There are many other definitions of organization and enterprise in circulation; we stick to these because they serve our purpose well.

Smart SOA

So how can we use Enterprise Engineering and SOA for our IT foundation? If we use it to actually implement a solution it will no longer be an *implementation independent* IT foundation. An actual implementation may be short lived if circumstances change; that may be a change at the business end: changing product offering, changing regulations, changing business partners, changing channels or combinations of those. It may also be a change at the IT end: changing technology and changing solution paradigms. What we need are building blocks that can be used in all of these scenarios; essential building blocks. Not their implementation but their definition, the interfaces. To summarize: the only thing we can do is define a set of interfaces of building blocks that capture the essence of our organization. Let's call these building block interfaces 'business services'. This is illustrated in Figure 1. This portfolio is used for contract-first development for a specific implementation.

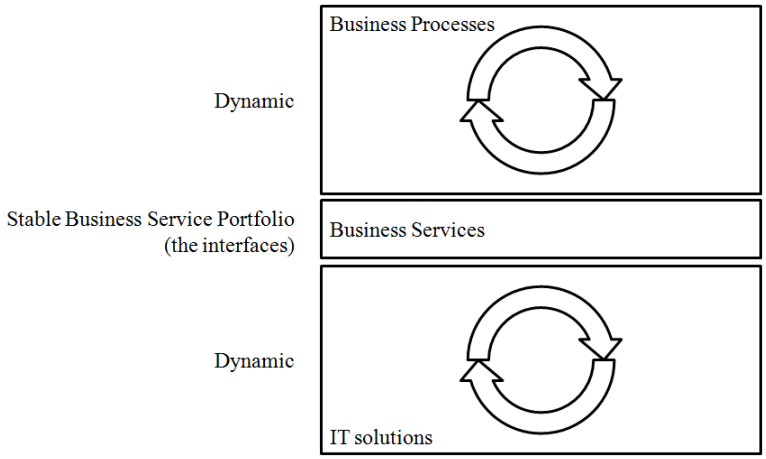

Figure 1 - Stable essential implementation independent business services.

To avoid losing the audience at this point a few examples of such business services are given. For a bank obvious essential implementation independent business services are 'open account', 'make payment' and 'get balance'. For a library 'start membership', 'loan library item' and 'return library item'. The service definition includes a description of its function and details of operations, messages, and a data model used with syntax and semantics.

So… is this the right approach we need?

Define a portfolio of services based on the implementation independent essence of our organization.

For many organizations this will present a big step forward towards optimal agility. But…to start a massive program to define this portfolio and realize a first implementation is not a good idea:

[1] This quote from the Enterprise Engineering Manifesto, in my mind, unnecessarily limits itself to the transaction paradigm, it closes the door for research into other approaches (for which I have no suggestions by the way). The focus on unification and integration for optimal performance and agility is more essential.

1. **Justification.** A lot of effort will be spent in areas where there is no urgency, making the effort impossible to justify. For example, for a global bank the creation of a service like 'get most profitable clients' may be very attractive but will take such a huge effort globally as to make it practically unfeasible. Let alone justifying the complete service portfolio forming their IT foundation.
2. **Changing organization.** Also in real world situations creating an IT foundation along organizational lines will not be ideal as organizations tend to do mergers and acquisitions, split off parts, work together in a chain or are suddenly owned by multiple parent organizations. It really means that, for our IT foundation approach, organizational boundaries do not mean very much. Organizations should simply be treated as the temporary legal boundaries which they are. From an information perspective there is a need for something more flexible and more stable at the same time.

We need a smarter approach that does justice to the above two arguments.

Let us take a step back. Why are we considering building a *complete organizational* IT foundation?

Why not do it only when there is a clear business case to do so? And also allow this business case to go beyond organizational boundaries or zoom in on certain parts of the organization only.

This means a rephrase of our approach:

> *First: find business cases for enterprise agility improvement through the use of services. This is not limited by the organizational boundaries in any way.*
>
> *Second: define a portfolio of services for the independent essence of each of these enterprises.*

The business case comes from specific change scenarios: changing demands for products and services, changing regulations, changing business partners, hosting IT services for changing parties, accommodating acquisitions or combinations of any or all of these. Such scenarios have been described by Baarda in "Your SOA needs a business case" [Baarda, 2008]. It is clear that there is no business case in a completely static situation. In that case we should put this initiative on ice and wait for changing circumstances to create a case.

We use the SOA paradigm only for defining these services. SOA in a very lean, selective fashion; Smart SOA! [2]

In the next figure (Figure 2) an overview is given of the structure of the envisioned service portfolio definition method. It is illustrated with an example from the financial industry as seen from the perspective of an organization initiating SOA initiatives, Bank(1). Like most banks it

---

[2] This is in contrast with SOA as positioned by several standards bodies like Open Group and OASIS. Both have produced a Reference Architecture (RA) for SOA that clearly positions SOA as an object that can be implemented. Open Group states in the introduction of their RA document:

> *This specification presents a SOA Reference Architecture (SOA RA), which provides guidelines for making architectural, design, and implementation decisions. The goal of the SOA Reference Architecture is to provide a blueprint for creating or evaluating an architecture. Additionally, it provides patterns and insights for integrating these fundamental elements of an SOA as exemplified in the layers of an SOA. (italic emphasis added)* [Open Group, 2009]

While OASIS describes theirs as:

> *This Reference Architecture Foundation is not a complete blueprint for realizing SOA-based systems. Nor is it a technology map identifying all the technologies needed to realize SOA-based systems. It does identify many of the key aspects and components that will be present in any well designed SOA-based system. In order to actually use, construct and manage SOA-based systems, many additional design decisions and technology choices will need to be made. (italic emphasis added)* [OASIS, 2009]

You may argue that our 'Smart SOA' is not really SOA as it does not comply with the definitions of official bodies and you have a point there. In my mind that is only a matter of semantics. I still continue to call it 'Smart SOA' and anybody is free to call it something else.

plays a role in a global network of banks in support of international payments. The approach taken is to develop business cases for service portfolios. The areas concerned are indicated in gray ellipses (this is just an example).

One is a case for increasing agility in the printing department of Bank(1), another the improvement of the customer support function of Bank(1) and the third example is the improvement of the agility of the international payment function of a network of banks ranging from Bank(1) to Bank(n). In the method, we propose to define three separate 'enterprises' whose business covers the areas of interest. Case one concerns serving the internal Bank(1) customers with printing services. The other case concerns taking care of the customer support function of Bank(1). Note that in that case we are not interested in the bank as a whole, only the part that provides customer support. The last case takes the network of banks as an enterprise performing international payments for its customers, being the collection of customers of Bank(1)...Bank(n) for international payment handling. Some may have noticed that in real life a chain of banks has already created the service portfolio named E3 in the figure. The organization maintaining this portfolio and representing a chain of banks is SWIFT. The SWIFT organization was started in 1973, long before the term SOA was coined.

For each 'enterprise' a separate and independent DEMO model is created from which a collection of services is defined. All services of Bank(1) are governed by Bank(1), independently of how this is organized; maybe through a central service portfolio management function or possibly a better choice, by a several units governing the 'enterprises' as taken from the cases. The point is that enterprises can be defined based on a case for developing a set of services.

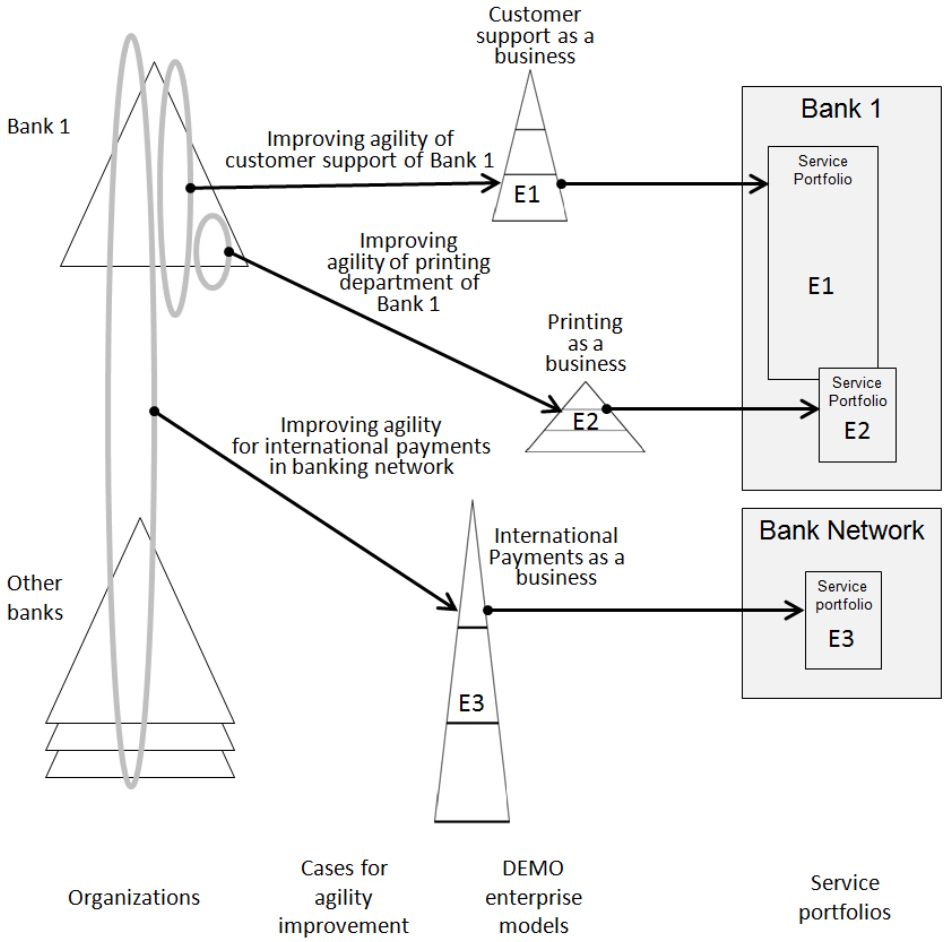

Figure 2 – Overview of Smart SOA method

This overview clearly demonstrates that before any raw material describing the organization and workings of an enterprise can be interpreted we must define our perspective first.

For example, a document produced by the printing department of Bank1 is a *product* (a B-item[3]) delivered to their customers. The *same* document may contain information needed for the business of Bank1, making it a D-item. And again the same document is probably meaningless for the chain of banks and nothing more than paper and think they could not care less about.

This also shows the relative nature of the concepts of *enterprise* and *essence*.

We call our method Smart SOA, because we apply the SOA paradigm very selectively, only in areas where it is beneficial and not because we have embraced the SOA paradigm and want to apply it dogmatically.

We are not making any assumptions on the software architecture of the realization of these services.

An idea: for defining the software architecture of a service we can apply the same Smart SOA approach; a service as a little enterprise in itself. Only when there is a case to realize the service as a composite service, in other words: when the agility demands justify the existence of lower level services will these be defined and realized. It looks like stretching the definition of enterprise, but there is nothing preventing us from viewing a service as something that conceivably can be performed by human beings.

The remainder of this report we will investigate the use of DEMO for modeling the implementation independent essence of an enterprise and Smart SOA. At the same time the *relativity* of the concept enterprise is shown. That seems like a complicating factor at first, and in a way it is, but allows us to create foundations not in a dogmatic way but just enough, just in time.

For those not familiar with the DEMO method the basics are presented in the next section. Others can safely skip this section.

DEMO basics

DEMO is an acronym for 'Design and Engineering Methodology for Organizations'. It is a method for creating a model of the implementation independent essence of an enterprise. It captures the essential *coordination* aspects, not the specifics of the *actual production*. The actual production is not the subject of Enterprise Engineering and is by its nature product specific. While the baking of pizzas is very different from weather forecasting, for DEMO it is just 'production'. DEMO is based on the PSI-theory (Performance in Social Interaction) or Ψ-theory for short, which in turn is based on the following five axioms [Dietz, 2006]. The DEMO basics form an important basis for the reasoning in this paper.

1. **Operation axiom.** The operation of an enterprise constitutes of the activities of actor roles, which are elementary chunks of authority and responsibility, fulfilled by subjects. In doing so, these subjects perform two kinds of acts: production acts and coordination acts. By performing production acts the subjects contribute to bringing about the material or non-material products delivered to customers in the environment of the enterprise. By performing coordination acts the subjects enter into and comply with commitments regarding production acts.
2. **Transaction axiom.** Coordination acts are always performed as steps in a universal pattern. This pattern is called a transaction. A transaction always involves two actor roles, the initiator and the executor and is aimed at achieving a well defined result. The basic transaction pattern consists of:
    - Request by the initiator.
    - Promise by the executor.
    - Production of the requested result.
    - State by the executor.

[3] For those not familiar with DEMO please check section 'DEMO basics' on page 6.

- Accept by the initiator.

Between promise and state the actual production takes place, the production of the promised result. Note: except for a single symbol the actual production is not modeled. See Figure 3.

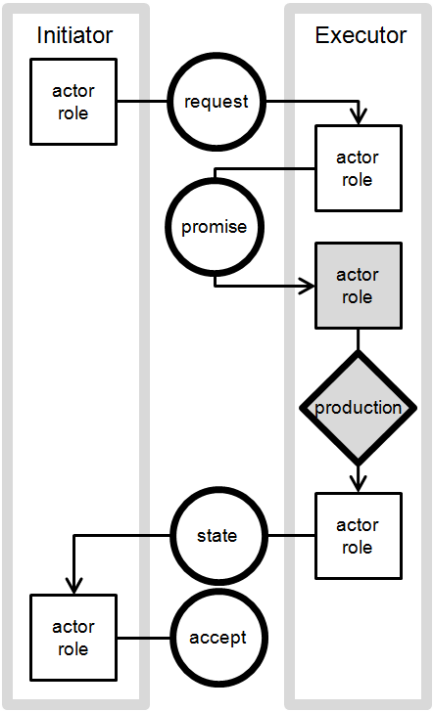

Figure 3 - Basic transaction pattern.

There is also a standard pattern that includes decline, quit, reject and stop scenario's. Also a cancellation pattern exists. For this paper the basic pattern suffices. The DEMO notation uses a single symbol for the standard universal transaction pattern. See Figure 4. This is one of the reasons DEMO models are very compact when compared to traditional business models. Another, and even more important reason, is the capture of the implementation independent essence of an enterprise only.

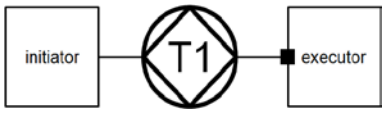

Figure 4 - Single symbol for basic transaction pattern.

3. **Composition axiom.** This axiom states that every transaction can contain a hierarchy of sub-transactions. A transaction is either a customer transaction, is enclosed in some other transaction or is self-activated. The top level transactions are those requested by customers. All other transactions occur in support of those.
4. **Distinction axiom.** There are three distinct human abilities that play a role in the operation of actors, called performa, informa and forma. The performa ability, "through the form", is the ability to create new, original things, directly or indirectly through communication. It includes creating material or non-material products. This ability is at the core of doing business. Transactions performed at the business level are called essential or ontological transactions. The informa ability is "what is in the form" and concerns the content aspects of data or documents. The informa ability is responsible for the infological transactions; reproducing or deriving the knowledge that is needed by the performa actors. The forma ability, in turn, concerns datalogical transactions; the ability to deal with recorded information items, called data or documents. The function of the forma ability is to care for the storage and retrieval, the copying and the transmission of documents containing the information needed by the informa actors. In the real world these abilities are often performed by a single person, for example a sales person selling a product (ontological) checks first if the product is in stock (infological) and updates the stock

amount in a database (datalogical) when the product is accepted by the customer (ontological) without other persons involved.

5. **Organization axiom.** The organization of an enterprise is a heterogeneous system that is constituted as the layered integration of three homogeneous systems: the B-organization (from business), the I-organization (from in-intellect) and the D-organization (from document). The relationship between them is that the D-organization *supports* the I-organization, and the I-organization *supports* the B-organization. See Figure 5

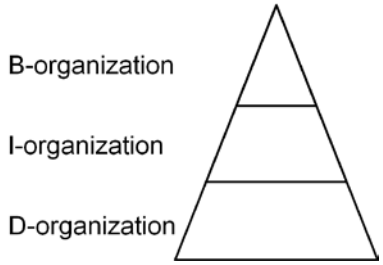

Figure 5 - An enterprise as a layered integration of B-, I- and D-systems.

The method consists of six steps, through which the modeler freely iterates until the model is complete:

1. The Performa, Informa, Forma Analysis. All relevant existing pieces of enterprise knowledge are divided in three sets, according to the distinction axiom.
2. The Coordination-Actors-Production Analysis. The performa items are divided in coordination acts and results, production acts and results and actor roles according to the operation axiom.
3. The Transaction Pattern Synthesis. The results of the previous steps are compressed using the transaction axiom into transaction types.
4. The Result Structure Analysis. Transaction types are structured using the composition axiom with transaction types interaction with the enterprise environment at the top.
5. The Construction Synthesis. For all transaction types the actor roles initiating and executing actor roles are determined.
6. The Organization Synthesis. The selection of the boundaries of the enterprise modeled is reconsidered.

From raw material to essential model

The material used in the Performa, Informa, Forma analysis may include *implementation dependent* and *non-essential* content, while the resulting model contains only the *implementation independent essential* aspects of the enterprise in question.

You may wonder how this comes about.

It is the result of filtering out only the coordination aspects of transactions where the enterprise delivers some product or service to its environment on the request of some party from that environment – the customer. This includes the informa and forma items supporting this, but nothing else.

This means the enterprise in the role of customer of another enterprise is not modeled, unless in support of delivering a product or service itself. For example the employee restaurant is not included because it is not directly linked to any production although descriptions may have been included in the raw source material.

The starting point is an enterprise. The selection of its boundaries is not part of the method and implicitly a single organization is assumed. Only the last step, the organization synthesis, addresses briefly the enterprise boundaries and is presented as a minor step.

In our opinion the determination of the boundaries of the enterprise to be modeled is a crucial step. In current DEMO practice it is assumed that the selection of the boundaries is self evident, it is along organizational lines or a very logical subdivision.

This paper may help in making this a more important step in the process by showing its relativity. For Smart SOA we base our selection on the existence of a (business) case for agility improvement in terms of social system boundaries and products and services to include in the model.

In the next chapter DEMO is investigated in terms of how it can support our need for these relative enterprise definitions and more in general if it can be used for Smart SOA.

Can DEMO be used for Smart SOA?

The fitness of DEMO for Smart SOA is presented through the use of some representative examples. A main worry was that often in DEMO modeling there is a lot of discussion when an enterprise does not produce physical products but D-, I- or B-items to its customers. Often I- and D-items are treated as absolute items conflicting with the relative position we need them to have. The examples are using the DEMO method rigorously with strict separation between production and supporting items, especially when it concerns non-material production results.

Examples of DEMO and the production of B-, I- or D-items

The examples are shown in terms of set theory as used by the Performa-Informa-Forma analysis:

B, the set of all performa items. Performa items are the actor roles, coordination acts and results and production acts and results contributing to bringing about the goods and or services that are delivered to the environment of an enterprise. B contains all performa items of all enterprises.

I, the set of all informa items. Information in general.

D, the set of all forma items. Recorded information in general.

Note that these sets are not the items of one specific enterprise but encompass those of all enterprises. See Figure 6.

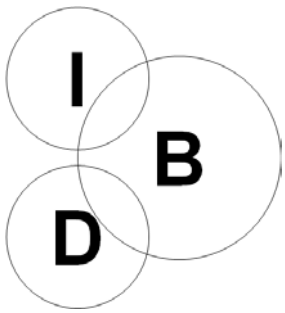

Figure 6 - Ψ-theory: B-, I- and D-sets

The following statements are true:

$$I \cap D = \emptyset \quad (1)$$

$$I \cap B \neq \emptyset \quad (2)$$

$$D \cap B \neq \emptyset \quad (3)$$

Equation 1 captures the fundamentally different character of forma and informa items. Equations 2 and 3 capture the fact that the set of performa items includes informa and forma items: *information and recorded information as production results*. An organization can provide information and recorded information to its customers as part of its set of products and services, aside from the information and recorded information it needs for its own business. This is shown in Figure 6.

A few representative examples with increasing complexity are presented. The examples are selected to represent the essential differences between production results and the information and recorded information that is needed for transactions with the enterprise customers. Production results are completely separate also when production results entail recorded information, information or services. DEMO is applied rigorously.

Let us also define:

B_E, the performa set of enterprise E.

I_E, the informa set of enterprise E.

D_E, the forma set of enterprise E.

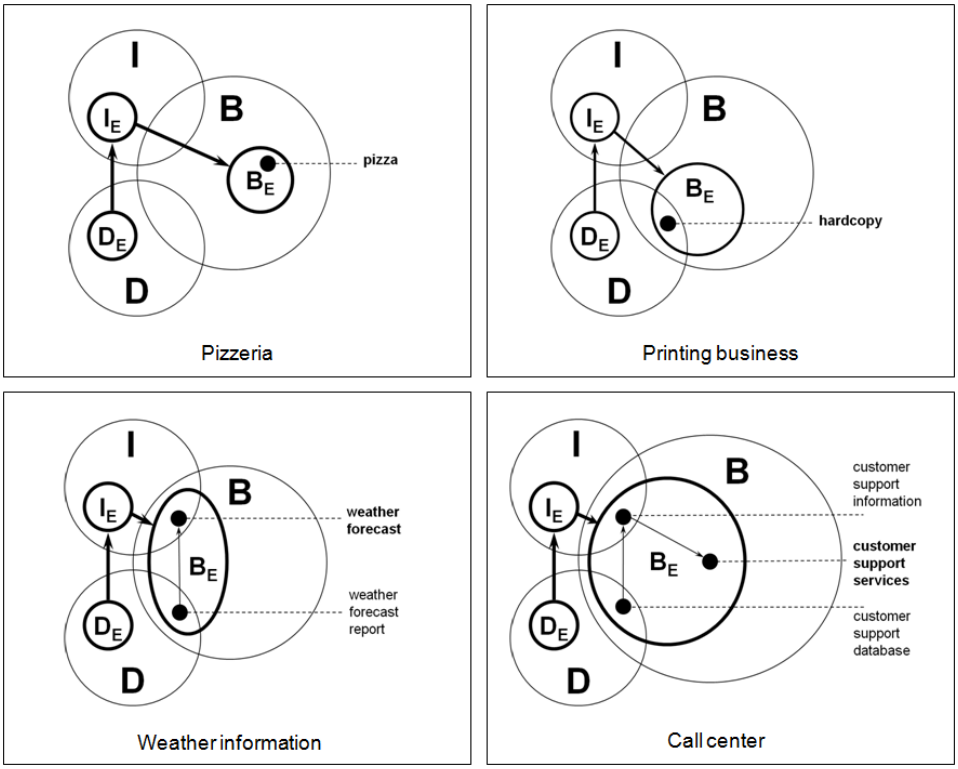

Figure 7 - Examples

- **Pizzeria.** The classic DEMO example [Dietz, 2006]. A simple case of an enterprise supplying physical goods to customers. The B-organization delivers pizzas to its customers. The pizza itself is a production result and part of the B-set, outside of the I- and D-sets. How the pizza is actually produced is not part of the model. To model more complex examples like those presented below a rule of thumb is to ask the question ‘where is the pizza?’. Meaning, what is the product offered to the customers of the enterprise? This is not always self-evident, especially in cases where documents, information or even business functions are the product. By looking for these products (‘where is the pizza?’) the distinction between production and true business transactions with their supporting information and documents can be recognized. DEMO models do not cover the actual production process, not for pizzas and not for any other product.
- **Printing business.** Enterprise delivering D-items to a customer. Note that although its products are in the D-domain they are outside the D_E domain and are treated by E as 'pizzas'; enterprise production results. I_E is concerned with information on making hard copies for customers, including outstanding commitments, paper in stock, machine capacity, prices, and delivery times. But I_E is not concerned with the content of the hard copies. That is something probably provided by the customers and is treated the same as 'pizza-ingredients' and is part of the actual production process that is not part of a DEMO model.

- **Weather Information.** Weather forecasting as a business. Enterprise delivering I-items to a customer. These I-items always have to use some medium to arrive at the customer, so always a D-item is involved also. The service delivered is a weather forecast using some medium. Again these I- and D-items are not part of the I_E and D_E sets and are 'just' products and conceptually the same as pizzas.
- **Call center.** This call center performs the 1st line customer support function for their clients, who have outsourced this function to the call center. The customers of the call center are *not* the persons calling for support services. Those persons are the customers of *the organizations that outsourced their customer support function* to the call center. Let's call these organizations XYZ. These persons are not even aware they are interacting with another organization than XYZ. In their perception they have a dialog with the organization from which they bought a product they need support for. So the true customers of the call center are XYZ themselves, not the customers of XYZ! The "pizza" is the delegated customer support function. There is possibly an essential customer support function for XYZ that is completely outside the production process.

  Some of you may need a pause here to let this sink in. In discussions with DEMO practitioners this is always a critical point where most stick with an absolute classification of documents, information and business (thereby mixing production with coordination).

  Essential transactions are about contracts for performing customer support *for* XYZ. If a DEMO model is made correctly of this call center the Performa-Informa-Forma result should be as illustrated in Figure 7.

  If that is not the result we want, rethinking the perspective may help. When for example we want to model the essence of the customer support function we should zoom in on one of the organizations XYZ that outsourced to the call center and define an enterprise that does nothing but perform that customer support. That is the part B_E in 'Call center' of Figure 7. That may be entirely appropriate but is not the same as a model of the call center enterprise.

  Some may think this is an unnecessary distinction but the difference is crucial. The business of a call center is making money by in-sourcing. The business of a customer support function is providing…well…customer support, most likely as a cost center for a larger enterprise.

  The relativity also surfaces in terms of *essence*. For the organizations outsourcing to the call center it is just an *implementation choice* to not perform it themselves. Their DEMO model will not make any mention of the call center. But for the call center is it essential.

  This is comparable to the pizzeria example. For the pizzeria selling pizzas is essential business, for their customers most likely not (unless they resell the pizzas they just bought). And it does not matter at all to the pizzeria, they sell pizzas and what their customers do with it is none of their concern.

The I-organization as an enterprise.

The reasoning can also be applied to the three organizations as distinguished by the Ψ-theory itself. Why not treat the I-organization as an enterprise? With 'all information for the B-organization' as 'the pizza' (excluding information in the heads of the members of the B-organization). Then it would have its own I-organization.

From the outside perspective that I-organization would be responsible for the information of the information-organization. Regardless of whether or not it is being modeled, it is important to understand that in a real world organization this 'enterprise' does exist. A department responsible for providing information to the business of the organization needs information itself to be able to perform that function. For example, it needs to maintain the status of business information requests. A reason for modeling such lower level enterprises may be to find out what *their* information needs are. See Figure 8. This is of course also happening in the real world. For example, there really are individuals working on the 'payroll administration' of the 'payroll administration' (☺). There may be a case to define

that as an enterprise boundary, although I am not sure if I want to be involved in this (yawn). It also follows that this step can continue ad infinitum or more correctly until the enterprise is a single individual person. This is illustrated in Figure 9. This again shows the relative character of the concept of enterprise.

The world really is an infinite set of interconnected and overlapping networks and hierarchies of enterprises. Which of these 'enterprises' is selected for modeling depends entirely on the purpose of the model.

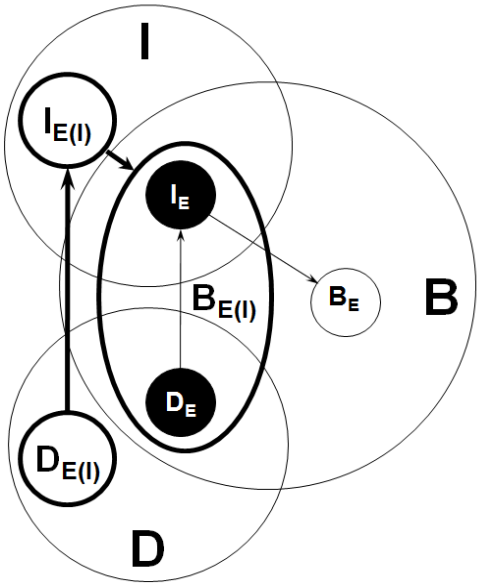

Figure 8 - The I-organization of E as an 'enterprise' E(I) delivering information services to E. Black dots are production results of E(I).

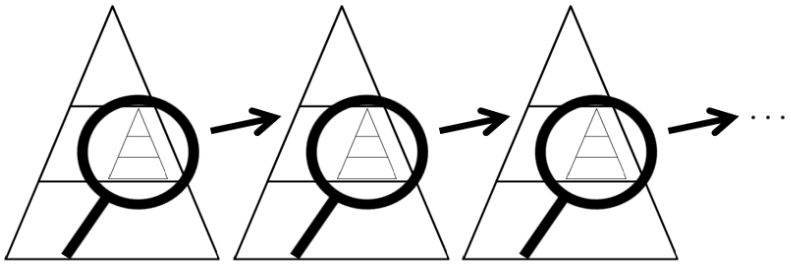

Figure 9 - Example of zooming in 'ad infinitum' (until individual person).

One reason for not using DEMO for this is a definition matter. The definition of enterprise mentioned 'social entities' and the Enterprise Engineering Manifesto mentions 'human beings'. Applying DEMO for the I-organization stretches this a bit. DEMO distinguishes performa, informa and forma *abilities* that can be performed by a *single* human being. With the I-organization as an enterprise this is modeled *as if these abilities are performed by separate human beings*. Still we can conceive of such an enterprise. One article has been written that touches the subject from an enterprise *realization* perspective [de Jong, 2010].

This is not the phenomenon we are describing here. We are talking about enterprises from the essential modeling perspective, independent of its realization; enterprises that exist in real life. Whether or not they are worth modeling only depends on the purpose of the model. For Smart SOA this depends on the business case for a service portfolio. This service portfolio in turn is an instrument for realizing an enterprise.

Conclusions on using DEMO

So far we have not found a reason for disqualifying DEMO for our Smart SOA method.

The consequences of the relativity of enterprise and essence are under-illuminated in DEMO where performa, informa and forma items are generally seen as absolute concepts.

We found that real world items can only be classified if we define the *perspective* we are doing the classification from first.

As said: a paper document may seem like a clear case of a D-item in DEMO terms. But that is only the case for the enterprise for which the document contains information it needs to do its business. For an enterprise that produced the document it is a B-item. For any other enterprise it is just meaningless paper and ink.

This has consequences too for the gnome metaphor often used with DEMO [Dietz, 2010b]. The color of the caps the gnomes are wearing is not absolute! It depends on the perspective taken. In de previous example when seen from perspective of the enterprise that uses a paper document because it contains information they need for their business, the document is provided by **a gnome with a blue cap**. When seen from the enterprise that produced the document as their business **that same gnome is wearing a red hat**. For yet another enterprise that did not produce it and is not interested in the content of the document **the gnome is even hatless!** As said, for them it is just a piece of paper with ink on it. As meaningless for their enterprise as the socks the gnomes are wearing. Note that it is the same document and the same gnome at the same point in time, only the perspective taken determines the color of the cap. Before starting the Performa-Informa-Forma analysis this should be completely clear.

Next steps

The ultimate goal of this research is a formal method with tooling to derive a service portfolio in terms of a set of WSDL's straight from a model of the implementation independent essence of an enterprise. DEMO or a DEMO-like approach seems very promising as a starting point. The primary focus will be on the practical aspects first. That will provide a good basis for empirical research. Formal theoretical support for the reasoning above takes second stage at the moment as it will hinder progress for now. This needs to be addressed sooner or later.

Our Smart SOA approach needs to introduce an analysis phase 'selecting perspective' to define the boundaries and the services it provides to its environment before DEMO can be used. This together with a step 'business case'.

Aside from these steps of our Smart SOA method the derivation of service definitions itself is not clear yet. A direct mapping of DEMO transactions to service-operations seems obvious. Alternatively we take the perspective of the I-organization of the enterprise for which a service portfolio business case exists.

Wider implications

The insights into the relativity of the concept of enterprise raise all kinds of interesting other questions, whose implications go beyond the focus of our Smart SOA method. For example:

- To what extent can arguments used for the benefits of enterprise architecture be applied to the smallest enterprise of all; an individual person? Or the other way around? For example, are persons working according to principles more successful? (Maybe Covey [Covey, 1989]). I think they are! Covey lists 7 habits that conceivably can be used by larger enterprises also (larger than an individual person); be proactive, put first things first, think win-win, seek first to understand, then to be understood, synergize, make a habit of self renewal, and start with the end in mind. The latter may be a challenge as enterprises are generally not designed with mortality in mind.
- How can we ensure that nested enterprises all contribute to the success of the highest level enterprise? How do principles defined at the highest level translate all the way down to the lowest, personal, level? What about enterprises that *partly* overlap?
- It is clear that DEMO is fit for analyzing a single enterprise. How can it incorporate the points raised in the previous bullets?
- It is sometimes suggested to extend DEMO to capture *interaction* between enterprises. This is not made any easier by the relativity of the enterprise concept as demonstrated

in this paper. The danger I see is that DEMO will depart from its main attraction, the capture of the implementation independent essence of an enterprise, and is reduced to just another process modeling method. DEMO as the victim of the Peter Principle [Peter, 1969], in this case generalized and applied to enterprise engineering methods. The generalized Peter's principle says that "anything that works will be used in progressively more challenging applications until it fails" [4]. You can also simply call it method creep.

And probably many more.

References

[Baarda, 2008] P.J. Baarda: *Your SOA needs a business case*, Via-Nova-Architectura, November 10, 2008, http://www.via-nova-architectura.org/files/magazine/Baarda.pdf (February 24, 2011).

[Conboy, 2004] K. Conboy and B. Fitzgerald: *Toward a conceptual framework of agile methods: a study of agility in different disciplines*, Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research Newport Beach, CA, USA: ACM,2004.

[Covey, 1989] S. R. Covey: *7 habits of highly effective people*, New York: Simon & Schuster, 1989.

[Dietz, 2006] J. L. G. Dietz: *Enterprise Ontology: Theory and Methodology.* Springer-Verlag New York, Inc., 2006.

[Dietz, 2010a] J. L. G. Dietz: *Enterprise Engineering Manifesto*, Final version, June 2010.

[Dietz, 2010b] J. L. G. Dietz: *Rode tuinkabouters bestaan niet*, Sapio BV, 2010.

[de Jong, 2010] Joop de Jong, J. L. G. Dietz: *Understanding the realization of organizations,* Advances in Enterprise Engineering IV, Lecture Notes in Business Information Processing, 2010.

[OASIS, 2009] OASIS: *OASIS Reference Architecture Foundation for Service Oriented Architecture*, Version 1.0, 14 October 2009.

[Open Group, 2009] Open Group: *SOA Reference Architecture*, April 2009.

[Peter, 1969] L. J. Peter, R. Hull: *The Peter Principle: why things always go wrong*. William Morrow and Company, New York, 1969

[Robbins, 1990] S. P. Robbins: *Organization Theory*. Englewood Cliffs, Prentice-Hall, 1990.

[4] No particular publication, but known to be coined by a Dr. William R. Corcoran.