Dependencies Between Data Decisions

Frank G. Goethals, Wilfried Lemahieu, Monique Snoeck, and Jacques Vandenbulcke

F.E.T.E.W. – K.U.Leuven – Naamsestraat 69, B-3000 Leuven, Belgium
SAP-leerstoel Extended Enterprise Infrastructures
{Frank.Goethals, Wilfried.Lemahieu, Monique.Snoeck,
Jacques.Vandenbulcke}@econ.kuleuven.be

Abstract. In this paper we show that storing and transmitting data is a complex practice, especially in an inter-organizational setting. We found 18 data aspects on which heavy consideration and coordination is important during a software process. We present these data aspects and point out that these data aspects are dealt with at different levels within Extended Enterprises. A good software process embraces the idea that choices have to be made on these 18 data aspects, and it recognizes the dependencies between the aspects, and the dependencies between decisions made at different levels in the enterprise.

1 Introduction

Setting up an enterprise is a very complex matter. We distinguish between two views on an enterprise: *tasks* that change the state of an enterprise, and *data* that maintain the state of an enterprise (see Simon [1] and Hirscheim [2]). In this paper we draw attention to the data-side of the enterprise. The complexity of this side follows from the fact that many data-related decisions have to be made (18 'data aspects' are presented in this paper), and that many dependencies exist among these decisions. Decisions on these aspects should thus be aligned. This is, however, not the only complicating factor: the decisions reoccur along three dimensions (see Figure 1).

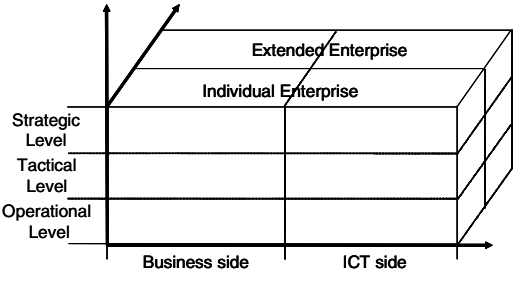

Fig. 1. Three dimensions along which the decisions re-occur (as discussed in [3])

First, decisions are made at the business-side and at the ICT-side of an enterprise. Secondly, decisions are made at strategic level (general principles and the like), operational level (decisions for a specific project), and tactical level (decisions valid

Q. Wang et al. (Eds.): SPW/ProSim 2006, LNCS 3966, pp. 48 – 55, 2006.

for all projects). Thirdly, we need to acknowledge that companies nowadays integrate their systems with those of other companies. Data exchanges within so-called Extended Enterprises (EEs, i.e., *collections* of partnering companies [4]) are even more difficult to realize than internal data exchanges. Therefore, a third dimension in the picture shows decisions are made at the level of an individual enterprise and at the level of the collection of collaborating enterprises.

The decisions made in the boxes of this figure should be aligned. Every software development process should therefore recognize 1) the dependencies across different boxes, and 2) the dependencies among the decisions made within each single box. That is, decisions makers are *dependent* upon each other. Therefore, coordination is needed. Importantly, this is not only true when entirely proprietary software is used, but also in the case of (1) proprietary software using standards, or (2) Commercial Off-the-Shelf (COTS) software. (1) Standards are *the* coordination instrument in a Business-to-Business setting. However, standards hardly deal with all 18 aspects in all boxes of Figure 1, let alone that they would deal with the links between the different boxes. (2) While COTS software packages may deal with all 18 aspects at operational level at the ICT-side; a fit is still needed with decisions on the 18 aspects made in other boxes as well.

There are thus a big number of dependencies that need to be managed during the software process. We note that this 'dependency-view' is – at least theoretically – also acknowledged in the Enterprise Architecture way of working. However, in practice, the attention there often goes entirely to the architectural descriptions rather than to the *usage* of these descriptions to manage dependencies. The dependency-driven way of working suggested here is to be imbedded in an Enterprise Architecture-driven one. By doing that, it is acknowledged that architectural descriptions should be 'Enterprise'-wide in the broadest sense of the word: the descriptions are meant to manage dependencies across projects, across business- and ICT people, and across the individual enterprises that form an Extended Enterprise.

In what follows, we first discuss the link between Enterprise Architecture and dependencies in some more detail. Then we present the 18 data aspects and we illustrate why coordination on these 18 aspects is important by showing the (sometimes infinite) range of possible values. Finally, it is acknowledged that making decisions on all 18 aspects at once is unrealistic, and that 'decision-components' need to be made that are placed in some sequence to get a process.

2 Enterprise Architecture and Dependencies

Cook [5] states that architectural descriptions work like standards: they restrict people in choices they can make. Standards function as a coordination instrument (see Mintzberg, [6]). One key question is then 'what needs to be coordinated?', or stated differently: 'what dependencies need to be managed?' (see Malone and Crowston's definition of coordination [7]). We expected to find an answer to this question by assessing important Enterprise Architecture (EA)-frameworks, and the models they suggest to create (see e.g. [8, 9, 10, 11, 12, 13, 14]). Unfortunately, the answers we found were disappointing. The suggested models do not seem to be based on a thorough investigation of the dependencies that exist, and are as such far from complete.

The renowned Zachman-framework for example [8] is said to be comprehensive. We argue it is not. While other critique could be added, here we restrict ourselves to one line of thought. As we stated, there are basically two views on an Enterprise: *tasks* that change the state of an enterprise, and *data* that maintain the state of an enterprise. In Zachman's framework a 'data' model is something like an ER-diagram. For one thing, this neglects the fact that data is often not present in structured form, and often not even made explicit (i.e., implicit/tacit knowledge). More importantly, this neglects the fact that data *is not just there*: data is made by a system in a location at some moment and has to be transmitted using some medium at some moment to another location for use by another system and this has to happen in a timely, secure, … fashion. Data dependencies do thus range much further than just knowing which data exists in which database so it can be reused in other projects. We note that such dependencies become particularly visible in an EE setting where different companies are dependent upon each other with respect to the decisions made on data aspects.

EA-frameworks do thus not seem to give a good overview of the dependencies they should manage. Unfortunately, classic dependency-theory [15, 16, 17, 18, 19, 20, 21, 22] seems to be scarcely out of the egg as well. The main focus of such theories is that one resource put out by one task is needed as an input for another task, and that a number of dependencies between tasks can therefore be suggested. Unfortunately, similarly as what we mentioned for Zachman's framework, classic dependency-theory only looks at *what* data is needed for (or created by) what task. It does not assess when data should be transported using what means to what location, etc.

Having a complete image of choices that need to be made and respected, and links between those choices is important not only to realize an effective system, but also to confront enterprise architects with the wide range of options they actually have. Companies who try to get competitive advantages have to be creative. Creativity should show in creative enterprise architectures, rather than in creative programming. While programmers know the building blocks to play with and are creative in using them, enterprise architects have a hard time to oversee their building blocks and thus to use them creatively. If one truly manages the building blocks, one will see that the building blocks can be arranged differently for different companies the company is doing business with. For example, imagine the case of a supplier with relatively expensive high-quality products, and who is assessing the 'data format' (see below) to be used. A long-term partner may get the price list in an xml format so it can easily be entered into his system. Other companies may get a nice graphical brochure with the prices. The latter 1) makes it harder for them to automatically compare prices across companies and 2) immediately shows them other information on the product: contents on which the supplier wants to compete.

In order to get a more complete image of the dependencies that do exist at the data side, we decided to study literature on diverse Business-to-Business integration (B2Bi) standards and B2Bi case studies. From this, we derived 18 data aspects that need to be dealt with. As such, these aspects can be seen as an extension to Enterprise Architecture frameworks, and to dependency-theory. The aspects are presented in the following paragraph.

3 The 18 Data Aspects

Space limitations make it impossible to deal with the aspects in detail. The first three aspects will be discussed in some more detail, to show the relevance of the three dimensions shown in Figure 1. We primarily point at issues that are interesting in a B2B situation.

1. Data content. Companies of course have to determine the content they want to share. While this may seem straightforward, it is not. For example, data content alignment is needed between different companies at a high level and at a low level. Alignment at a high level is for instance illustrated by Hansen, Nohria and Tierney [23]. They talk about two strategies for content management within companies. One strategy is to make information on the business itself explicit (to 'codify' information) so that it can be reused. Another strategy (the 'personalization' strategy) is to make information explicit about who knows what. They found that the content management strategies have to fit the business proposition of the companies. For example, companies like McKinsey and Bain primarily use the personalization strategy because they are strategy consulting firms. They are expected not to deal with standard solutions for standard problems, and thus not to store standard solutions. The market expects such practices, however, from Ernst & Young for example, which deals with the same problems over and over again. Once companies know what *type* of information they need to share, they can investigate what concrete information is needed (i.e. low-level alignment). Please note that in an Extended Enterprise setting, a collection of companies may want to appear to the outside world as one entity, and that the content they share with the outside world should reflect this. Also, in an Extended Enterprise it may be possible to create new content. For example, if an airline company, a car rental company and a hotel chain together offer trips they generally only have information on their own sales. By keeping the information together at the level of the collection of collaborating companies, data is available on how many customers booked an airplane seat as well as a hotel and a car. This data may then be linked to data on (individual/grouped) marketing campaigns and the like to do data mining.

2. Data format. Data has to be transmitted in some format. This first involves choosing between textual format or graphical format for example. If a textual format is chosen, it has to be decided whether a proprietary format or a standard format will be used. If a standard format is to be used, a concrete standard has to be chosen. For example, UBL, CBL, and cXML all offer standardized business documents. UBL (Universal Business Language) for example defines seven documents such as 'order', and 'invoice', and gives accompanying XML-schema definitions. Interestingly, specifications exist for automatically rendering a classic visual of the content of the XML documents, for example as a .pdf document, meant for human usage. This visual can serve as a boundary object between the business people of the different companies, while the XML files serve as a boundary object between their computer systems.

3. Roles. Different systems play different roles in a data exchange. In our research we have identified seven primitive roles. The Needy wants to process some data. The Needy may differ from the Initiation Event Originator. The latter is a node where an event originates (e.g., a 'request') that initiates the message transmission towards the

Needy. An Initiation Event Emitter (e.g., a 'requestor') is a party that transmits such an initiation event. This event can be sensed by an Initiation Event Sensor (e.g., an intermediary that groups requests from many parties). This party receives an initiation event from outside. The data that is needed by the Needy originates at the node of the Response Data Originator (e.g., the creator of a requested price list). A party that sends the data towards the Needy is called a Response Sender. A party that receives the data is called a Response Receiver. The roles are illustrated in the figure below.

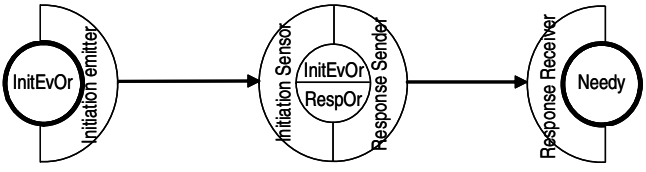

Fig. 2. The seven basic roles in an end-to-end transmission

In practice, one system may play several roles and one role may be played by several systems (and e.g., only vis-à-vis specific other systems). Roles may be discussed at the level of entire enterprises, departments within enterprises, specific people or computer systems within departments, etc.

4. Data distribution. Data may only be stored in one location, or in several locations. For example, in the health care industry the idea has arisen to share information on patients among authorized institutions if a patient enters one of these organizations for help. Because so many different institutions may have information on the patient, an institution needing information would need to contact all other institutions. Therefore, a central point has been entered in the network (an additional Provider role) where a Needy can request information. In the Netherlands, the central point itself does not have a copy of the patient's data. However, it has information on where information on some patient can be found. In the English set-up, however, the central point *does* contain information on the patients.

5. Exact physical system location. For each system that is involved, a specific physical location has to be determined. Data may for example be replicated on the premises of a close partner, or on the premises of a trusted third party. Also, it may all be stored together ('centrally'), it may be stored close to users, in a big city or not, etc.

6. Storage medium. Two angles can be considered per the storage medium: 1) the availability, reliability, capacity, security, transportability etc. of the medium, and 2) distinguishing between ICT-systems (ranging from Database-systems to CD-Roms and USB-keys), people (with knowledge in their minds), paper, etc.

7. Transmission network. Which nodes will be connected directly? For example, one could connect every node to every other node or connect every node only to one other node, or connect all nodes to a central node.

8. Transmission area. Through which geographical areas will connections pass? One may have to pay attention to 'hostile territories'.

9. Transmission medium. As for the storage medium, two viewpoints can be taken: 1) availability, reliability, capacity, security, etc. of the medium, and 2) distinguishing between specific media such as telephone, Internet, postal mail, etc. As an illustration,

note that business people and ICT people may have a different perception of the medium (e.g., Internet telephony).
10. System availability. A data transmission can only happen during the 'operational time' of nodes and connections and if capacity is available.
11. Initiation events. Different (combinations of) events can initiate and inhibit a message transmission.
12. Initiator party. Initiating events may originate in different nodes. For instance, the sending-system (Response Sender) may initiate transmissions itself (e.g., when sending purchase orders), or initiations may happen by a Needy-system.
13. Immediately or postponed. A transmission may (have to) be started immediately or the transmission may be postponed for some time (e.g., because messages are not permanently being processed within the node).
14. Transmission relationships. Messages may be related to each other in different ways. In short, it has to be assessed whether a message sent to one system can/cannot/has to be sent to another system as well at the same moment or at a later moment (i.e., simultaneous start or arrival or not). Also, it should be investigated whether the transmission of message A can/cannot/has to be accompanied or followed by the transmission of a message B. In classic database systems development cardinalities get much attention. In a B2B context, putting cardinality requirements upon data transmissions rather than on the data itself seems more realistic. For example, if a supplier *receives* an order, this order has to be *forwarded* to his supplier (not knowing whether he will store the data persistently or not).
15. Unit/Batch. Data can be transmitted in units or in batches. We note this distinction is different from the one between sending data immediately and postponing transmissions (see 13. above), although both aspects are often grouped under the name 'real-time vs. batch'.
16. Coarse/Fine-grained. The data that is stored and transmitted may be fine-grained or coarse-grained. Different parties may want to use the data for different purposes and may desire different levels of granularity for those purposes.
17. Meta-data/Production-data. Data may be meta-data or production data. For example, an intermediary (e.g., playing the Initiation Event Sensor and the Initiation Event Emitter roles) may only have meta-data about where requested data is stored.
18. Authorizations. Authorizations may be related to the content that is transmitted, the party to who it is transmitted, the format, the timing, the location, etc. (i.e., authorizations are related to all issues mentioned above).

Now we know the 18 aspects on which coordination is needed, let us have a short look at their interdependencies.

4 A Dependency-Driven Software Process?

The fact that there are 18 data aspects makes it impossible for decision makers to deal with all aspects at once. Some order has to be taken, and only a small number of data aspects can be dealt with during every step. Interestingly, there are interdependencies between different decisions. Examples of interdependencies are the following:

(9→5) *If a fast transmission medium is available a big distance is acceptable.*
(1→6) *If data content is highly confidential use a very secure storage medium.*
(2→1) *If you use some standard format (e.g. RosettaNet) then you may restrict yourself to transmitting only the content defined there.*
(6→11) *If the storage medium is human it is not desirable to fire a request for updates every minute, but rather to subscribe.*

Given the fact that there are dependencies between the different data aspects presented above, one would expect that some order could be given to the data aspects, or at least that some (highly interdependent) data aspects should be dealt with simultaneously, while others can be treated apart.

In our research we have tried to group data aspects on which decisions are highly interdependent. While looking for the interdependencies between the aspects we, unfortunately, found that most aspects are dependent upon most other aspects. Moreover, the degree of dependence is likely to differ from case to case. From this, it is clear that it is inappropriate to suggest the existence of components of data aspects that should be dealt with together in general. Therefore, we have created a 'tool' (actually an Excel-sheet) that shows the interdependencies between different data aspects. That is, for each of the aspects it is investigated how the choice of this aspect depends upon choices made for each of the other aspects. We have also suggested a value for each dependency (from 0 to 9). Remarkably, this value is likely to fluctuate from case to case. Assumed that values are given, an algorithm could evaluate all possible combinations of components of data aspects that should be dealt with together. Although forming components on such a basis is not academically correct, one needs to be pragmatic in this matter: companies cannot deal with all interdependencies at once, and need to make abstractions.

5 Conclusions

The contribution of this paper is that it identifies 18 data aspects that can be used creatively, and on which stakeholders throughout the Extended Enterprise need to reach agreement. While we cannot claim the 18 aspects are all the data aspects that actually exist, it seems very unlikely that any important ones would be missing. It is important to consider the 18 aspects and their interdependencies in every software process. Moreover, this paper acknowledges that the software process is not just taking place at the operational ICT level but is an integrated part of the entire Enterprise Architecture effort. Alignment is needed between business and ICT decisions; between strategic, tactical and operational decisions; and between decisions made for individual enterprises and those made for the collection of collaborating companies. Any software process that is not embedded in this philosophy is likely to result in software that is not aligned with the business, with other internal projects, or with other parts of the Extended Enterprise project that are being implemented by partnering companies.

Acknowledgements. This paper has been written as part of the 'SAP-leerstoel'-project on 'Extended Enterprise Infrastructures' sponsored by SAP Belgium.

References

1. Simon H.A. (1994). The sciences of the artificial (2nd ed). The MIT Press, Cambridge, Massachusetts, p 247.
2. Hirschheim, R., H. Klein and K. Lyytinen, "Control, Sense-Making and Argumentation: Articulating and Exploring the Intellectual Structures of Information Systems", Proceedings of the Fifth Australasian Conference on Information Systems, G. Shanks and D. Arnott (eds.), Melbourne, Australia, September 27-29, 1994, pp.1-25.
3. Goethals F., Vandenbulcke J., Lemahieu W., Snoeck M., Structuring the development of inter-organizational systems: Web Information Systems Engineering conference - Brisbane November 22-24, 2004. Springer LNCS-series, Volume 3306, pp. 454-465.
4. Goethals F., Vandenbulcke J., Lemahieu W., Snoeck M., Cumps B. (2005), Two Basic Types of Business-to-Business integration, International Journal of E-Business Research, 1(1), 1-15, Available at http://www.idea-group.com/downloads/samples/IJEBR.pdf.
5. Cook, M. (1996). Building Enterprise Information Architectures. Prentice-Hall, 179.
6. Mintzberg, H. Structure in Fives, Designing effective organizations. Prentice-Hall, Englewood Cliffs, New Jersey, 1993, p. 305.
7. Malone T.W., Crowston K. (1994). Towards an Interdisciplinary Theory of Coordination, Computing Surveys, 26(1), 1994.
8. Zachman J. (1987), A framework for information systems architecture, IBM Systems Journal, Vol. 26, No.3, pp. 276-292.
9. Kruchten P. (November 1995), The 4+1 View Model of Architecture, IEEE Software, pp. 42-50.
10. Soni, D., R.L. Nord & C. Hofmeister, 'Software architecture in industrial applications', in: R. Jeffrey, D. Notkin (eds.), Proceedings of the 17th International Conference on Software Engineering, ACM Press, 1995, pp. 196-207.
11. Tapscott D., Caston A. (1993), The New Promise of Information Technology, McGraw-Hill, pp. 313.
12. The Chief Information Officers Council (September 1999), Federal Enterprise Architecture Framework Version 1.1, pp. 41.
13. Department of Defense - C4ISR Architectures Working Group, (December 1997), C4ISR Architecture Framework Version 2.0, pp. 239. Retrieved from http://www.c3i.osd.mil/
14. Department of the Treasury, Treasury Enterprise Architecture Framework, Version 1, pp. 164. Retrieved from http://ustreasury.mondosearch.com/
15. Van de Ven, A.H., Delbecq, A.L., Koenig, R. Jr. 1976). Determinants of coordination modes within organizations. American sociological review, 41 (April), 322-338.
16. Tillquist J., King J.L., Woo C. (2002), A representational scheme for analyzing information technology and organizational dependency. MISQuarterly, Vol. 26 No.2, pp. 91-118.
17. Thompson, J.D. (1967). Organizations in Action: Social Science Bases of Administrative Theory. New York: McGraw-Hill.
18. Alexander E.R. (1995). How organizations act together. Gordon and Breach, p 384.
19. Chisholm Donald (1992). Coordination without hierarchy, informal structures in multiorganizational systems. University of California Press, p 273.
20. O'Toole L.J., and Montjoy R.S., 1984. Interorganizationl policy implements: a theoretical perspective, Public Administration Review 44(6): 491-503.
21. Pfeffer J., Salancik G.R., 2003 (1978), The external control of organizations. A Resource Dependence perspective. Stanford University Press, California.
22. Crowston, K. 2003. A taxonomy of organizational dependencies and coordination mechanisms. In T. W. Malone & K. Crowston & G. Herman (Eds.), The Process Handbook: 85–108. Cambridge, MA: MIT Press.
23. Hansen M.T., Nohria N., Tierney T. (1999). What's Your strategy for managing knowledge?, *Harvard Business Review*, March-April 1999, p.106-116.