# Key factors for information dissemination on communicating products and fixed databases

Sylvain Kubler, William Derigent, André Thomas, Eric Rondeau

# Key Factors for Information Dissemination on Communicating Products and Fixed Databases

Sylvain KUBLER, William DERIGENT, André THOMAS, and Éric RONDEAU

Research Centre for Automatic Control of Nancy, Nancy-University, CNRS, Boulevard des Aiguillettes, F-54506 Vandœuvre-lès-Nancy, France
{firstname.lastname}@cran.uhp-nancy.fr

**Abstract.** Intelligent products carrying their own information are more and more present nowadays. In recent years, some authors argued the usage of such products for the Supply Chain Management Industry. Indeed, a multitude of informational vectors take place in such environments like fixed databases or manufactured products on which we are able to embed significant proportion of data. By considering distributed database systems, we can allocate specific data fragments to the product useful to manage its own evolution. The paper aims to analyze the Supply Chain performance according to different strategies of information distribution. Thus, different distribution patterns between informational vectors are studied. The purpose is to determine the key factors which lead to improve information distribution performance in term of time properties.

## 1 Introduction

Intelligent products or products carrying their own information are more and more present nowadays. [10] quotes the example of clothes able to carry their own information and thus enabling the washing machine to automatically adapt its washing program. In one of our previous works, [8] highlight several possible scenarios in different sectors: Supply Chain Management, healthcare [2], home automation. Such applications rely on ever more complex information systems using a multitude of information vectors, in order to allow product information to be available anywhere and at anytime. These vectors may be fixed (desktop computers) or mobile devices (PDA, laptops, sensors, RFID technologies...), short-lived or even invisibles (concept of disappearing computer, ubiquitous computing). More generally, the concept of the Internet of Things [5] based on the RFID usage enables to access to information disseminated on any kind of physical object and to develop new smart services and applications.

According to Meyer [12], in the context of supply chain management, few researches has been conducted on "intelligence at object", i.e products carrying their own information and intelligence. In fact, most of the time, products are only given an identifier (stored in a RFID tag) referring to a software agent or a database (approach used by [13]). This mode of information management is diametrically opposed to works initiated since 2003 by the PDMS (Product-Driven

Manufacturing Systems) community, which advocates a physical information distribution on the product. In that case, a product carries physically a part, or even the totality of the information needed for its manufacturing or to manage its evolution all along its life cycle. Our previous work [9] aimed at prototyping a new type of materials, in which it is possible to write a significant quantity of information by inserting thousands of micro RFID tags. This new type of material is then referred to "communicating material". We developed an industrial process to produce a communicating textile with up to $1500 tags/m^2$. Meyer concurs with the PDMS community by stressing the fact, in an increasingly interconnected and interdependent world involving many actors issued from different domains, supply chain information should not be stored in a single database but should be distributed all over the supply chain network. In fact, substantial information distribution improves data accessibility and availability, compared to centralized architectures. However, update mechanisms of the distributed information are needed in order to avoid problems related to data consistency and integrity. This type of architecture is thus more complex to design than centralized architectures. As a result, product information can be spread out on mobile or fixed devices or even directly on the product, via simple RFID tags or communicating materials. Centralized architectures or highly distributed architectures can be employed. One might then wonder what the optimal information distribution is. The present paper aims to study the different ways to distribute information over a network composed of centralized, distributed databases and "communicating products", which may store information fragments as well. This study will determine the key factors which lead to improve information distribution performance. The performance is analyzed regarding the time required for accessing to the information system during the product life cycle. Based on this influent factors determination, in a further work, we will be able to implement an experimental design leading us to control the best way to disseminate information on the informational vectors.

This question is addressed in several steps. First, the data distribution is introduced, and then an overview on conducted researches on distributed databases over fixed and mobile devices is presented in section 2. Then, a case study extracted from this overview and adapted to our context is detailed in section 3. It only considers two types of informational vectors: fixed computers and communicating products. This case study is then used as a basis of comparison and evaluation between two different architectures of information distribution (one forbids data allocation on products while the other allows it). The evaluation process relies on several specific tools and a methodology using jointly two discrete-even simulators, one using Petri Nets (CPN tools) and the other dedicated to network protocol development and measurement (OPNET). This simulator looks for evaluating the manufacturing lead-time of a given number of communicating products all along the supply chain, by taking into account manufacturing run times, network delays, times to read/write information for both distributed databases and communicating products. This piece of software is presented in the section 4. Finally, the section 5 presents the results obtained

with the case study and an analysis of the main factors impacting on the performance of the information distribution.

## 2 Distributed Database Systems

### 2.1 General distribution framework

During the product lifecycle, users could have to access to product information for diverse reasons, either in the design phase, the usage phase or still the recycle phase. As exposed before, information can be stored both on the product and-or on fixed databases. Information are therefore bind to one or more relational data models, which have to be fragmented and distributed by the best way on these informational vectors. One example retracing briefly a bobbin lifecycle is presented in the figure 1. We can see 5 data fragments $[F1..F5]$ distributed between the product and the database ($F1$, $F4$, $F5$ allocated to the database system and $F2$, $F3$ to the product). By reconsidering the example given by [10], the washing machine could access to data fragments located both on the product and on the database according to its queries. In our researches, we are looking for assessing different distribution patterns between both informational vectors (manufactured products and fixed databases) by taking into account the access times for reaching information. Work on distributed databases considering fixed and mobile environments are introduced in the next section.

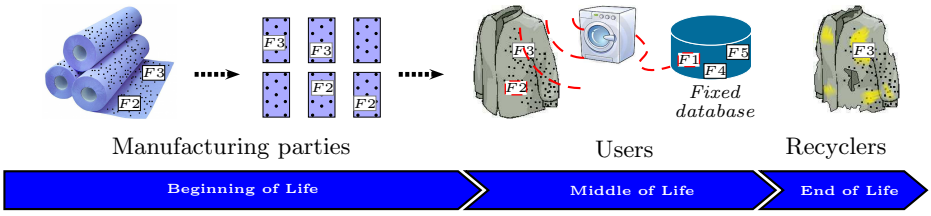


**Fig. 1.** Information distribution on products and fixed databases

### 2.2 Distributed databases through literature

The main objective of the data dissemination in an information system is to make the dissemination process transparent for users: location, partitioning and replication transparency. Indeed, no matter why, where and how the data repartition is achieved from a user's point of view. The structured data distribution regarding relational data models (within distributed database systems framework) is carried out in two steps: The partitioning of the data model follows up by the allocation phase of the resulting fragments. Many approaches and mechanisms exist for ensuring the best partitioning and allocation of the relational model regarding to the environment and some applicative constraints.

Basically, the partitioning aim is to subdivide the concerned relational data model. Thus, the resulting fragments will be allocated at specific informational

vectors in order to improve system performance. Three types of fragmentation exist: vertical [14], horizontal [1], mixed/hybrid [15]. The vertical fragmentation aims to break up a relation into a set of relations. It consists in dividing the attributes of a relation (i.e the columns of a relational table). The horizontal fragmentation aims to break the large number of object instances into disjoint subsets. It consists in partitioning the tuples of a relation (i.e the rows of a relational table). The hybrid fragmentation first divides the relation horizontally, and then splits each of the obtained fragments vertically or vice versa.

As stated previously, the allocation phase takes place subsequently to the fragmentation phase and the aim is to establish the optimal fragment assignation on the databases. Usually, methods tend to assign fragments to the clients requesting them mostly via objective functions that we attempt to minimize or maximize [7,6]. Let us note that it is possible to perform data replications, or in other words, to replicate a same fragment on several databases. This has the dual benefit of maintaining the system reliability and of increasing performance (e.g reduction of the overload traffic) [16]. However, replication mechanisms are necessary for handling both the modification broadcast (updates) on replica and also the information access rights (to authorize one site or one group of sites to modify replica). The applicative expectations have an influence on the mechanism to implement and actually two parameters have to be characterized: *When* and *Where*? *When* do the updates have to be propagated? Two modes are available: Synchronous (S) and Asynchronous (As). The As mode makes it possible to carry out local modification without needing to inform its peers (contrarily to the S mode). *Where* do the updates have to be performed ? Two principles exist: Update everywhere (Ue) and Primary copy (Pc). The Pc principle allows one site to perform modifications on a data fragment contrarily to the Ue mode which allows one group of sites. Finally, four types of replication may be considered: Ue-S, Ue-As, Pc-S and PC-As. Also note that the memory storage limitation of mobile devices is a problem frequently encountered in the literature. Accordingly, some authors focus on the data summarization [3,11] (subclass of the data mining) whose primary aim is to reduce the information somehow. [4] list the summarization methods used for distributed database systems and mention the fragmentation/allocation method (used in our study).

A multitude of interesting approaches are proposed in the literature, we therefore feel it is necessary to confront our proposition with them in order to compare and assess our distribution models. In this sense, works reported by Hababeh [6] seem interesting as basis of comparison. Indeed, a fragment distribution method is developed and then applied on a case study, which can be easily extended to our application. In what follows, two distribution architectures will be defined, the first one does not consider the presence of communicating products able to store data fragments, i.e all information is located on databases. In fact, we rely on the distribution defined by Hababeh. The second one considers communicating products able to store data fragments, thus, diverse distribution patterns of fragments between the product and databases will be possible. The next section introduces this case study and then the adaptation realized in this paper.

# 3 Case study presentation

## 3.1 Reference distribution pattern

Hababeh proposes a fragment distribution approach based on a two step process: first, the sites (clients and databases) are clustered according to communication costs, and then data fragments are allocated to the different clusters via an optimization function. This approach is applied on a specific case study, including 3 databases, 3 clients which perform read and write accesses on a set of fragments (8 in total: $[F1..F8]$). The resulting optimal allocation [6] is depicted on the figure 2, the access pattern to the 8 fragments performed by each client is specified, too (number in brackets indicates the number of bytes). The next section formulates the adaptation of this case study to our logistic scenario. In fact, we match parameters and data defined by Hababeh with the supply chain tasks, actors: number of databases and clients, query patterns, data fragments...

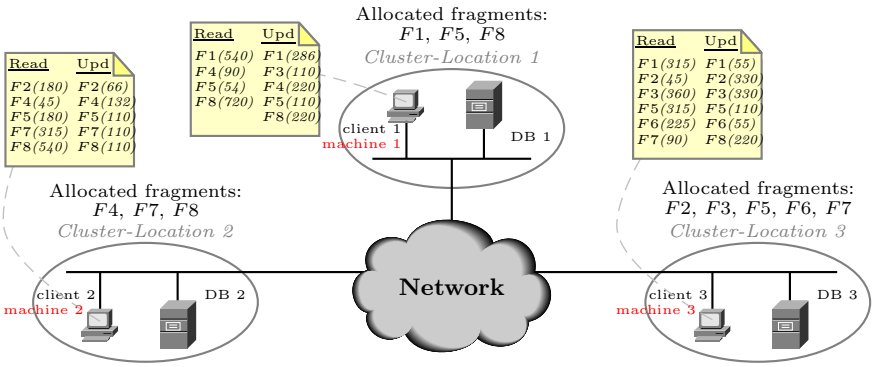


**Fig. 2.** Optimal distribution architecture established by [6]

## 3.2 Adaptation of the logistic process

A supply chain process consists of a set of tasks in a planned pattern or sequence (rout sheet). These tasks may correspond to manufacture operations, transport phases... and can be performed on different physical locations (e.g supplier). These locations may dispose of local databases and can implement their own information systems (related to their tasks), but they can also access to remote databases if a collaboration between actors exists. As a matter of fact, databases are distributed (or federated) through one or more relational data models. Inspired by our current researches, the applicative framework considered is related to a supply chain management process dedicated to the textile industry.

In order to adapt the case study described previously to our logistic scenario, we consider three remote locations corresponding to the clusters 1, 2 and 3 introduced on the figure 2. One physical transformation is carried out on each location: the operation 1 and the operation 2 (performed on the location 1 and 2 respectively) consist of cutting a set of *bobbins* 1 and *bobbins* 2 respectively, the

operation 3 (performed on the location 3) consists of sewing the textile pieces resulting from the two previous operations. Each location has one machine to achieve its own operation, this machine requires information after the arrival of products (range of product, production order...) and updates some of this information (notifications...). Therefore, the applicative characteristics defined for each client in Hababeh will be matched to each machine (located in each cluster). In other words, the machine 2 will have the same fragment access pattern (read and write accesses) than the client 2 defined in Hababeh and so on. Likewise, each location disposes of a local database and shares a relational data model, this data model has to be distributed on the three databases. Taking into account of the input parameters defined in Hababeh (query pattern, architecture...), the optimal distribution considered in our paper may be defined as shown in the figure 2: $F1$,$F5$,$F8$ allocated to DB1, $F4$,$F7$,$F8$ to DB2...

The architecture described in the previous paragraph corresponds to the Optimal Distributed Architecture (ODA), without any possibility to allocate data fragments[1] on communicating products. In the second stage, we allow data fragment allocation on the products (according to their memory capacity and characteristics). We denote this architecture: ODAP (Optimal Distributed Architecture considering communicating Products). The idea is to highlight the potential benefits that could be achieved regarding a supply chain via such an ODAP architecture, where manufactured products act as mobile databases as opposed to a classic architecture (ODA). The figure 3 illustrates in form of Petri Nets the synoptic of the part of the supply chain peculiar to our application. It consists of the ODA architecture implementing classic product and of the ODAP architecture implementing communicating product. Let us note that we design the Petri Net model by working on hierarchical views. Consequently, the distribution aspect will be detailed in the lower views (in the section 4.3).

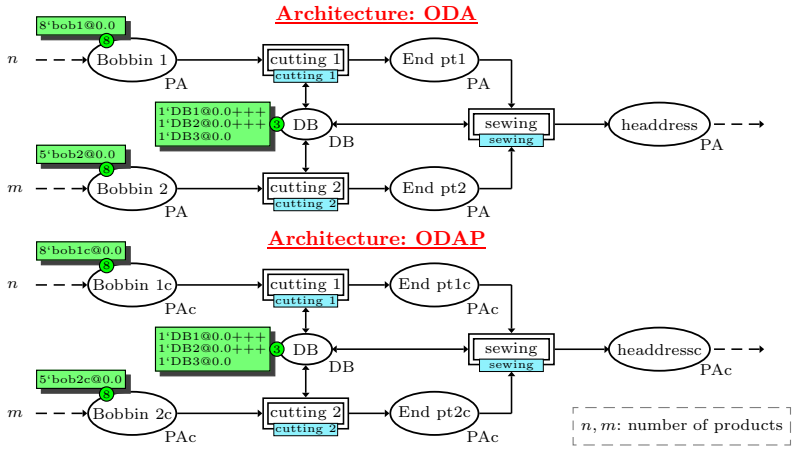


**Fig. 3.** Global view of the supply chain process

[1] Products are only given an identifier (stored in a RFID tag) referring to a database.

# 4 ODA and ODAP architecture modeling

## 4.1 Architecture

A description of how the assessment and the comparison are undertaken of both architectures (ODA and ODAP) is proposed in this section. The evaluation architecture relies on two discrete events simulators and its usage process is depicted on the figure 4. This architecture is composed by two sub-systems. The first one is a tool for editing, simulating, and analyzing Colored Petri Nets (CPN tools). As shown on the figure 4, the supply chain behavior is simulated via this tool, or at least the considered part which has been exposed in the section 3.2 (operation 1, 2 and 3). It allows to deal with sharing of physical resources taking place into the system (databases, machines, manufactured products...), operation times, queuing tasks, times for reading/writing information on databases or still on manufactured products (considering the ODAP architecture) and so on. The ODA and ODAP distribution patterns are specified in this tool. Let us note that for the ODAP architecture, all the possible combination of distribution between the product and the distributed fixed databases are realized (i.e $2^k$ possibilities with $k$ the total number of fragments). The second tool is the OPNET network simulator which is primarily aimed at developing and validating network protocols. However, it allows estimating various parameters on specific case studies, such as network times, overload traffic, equipment processing times, battery life, etc. In our study, the OPNET tool is used for assessing the *round trip time*[2] to achieve read/write queries on databases, considering the ODA architecture (we do not take into account the possibility to read and write the product). To do this, the physical architecture and the distribution adopted in the section 3 have to be specified in OPNET. The resulting times are then injected into CPN Tools. The next sections introduce the methodology employed for assessing the ODA and ODAP architectures.

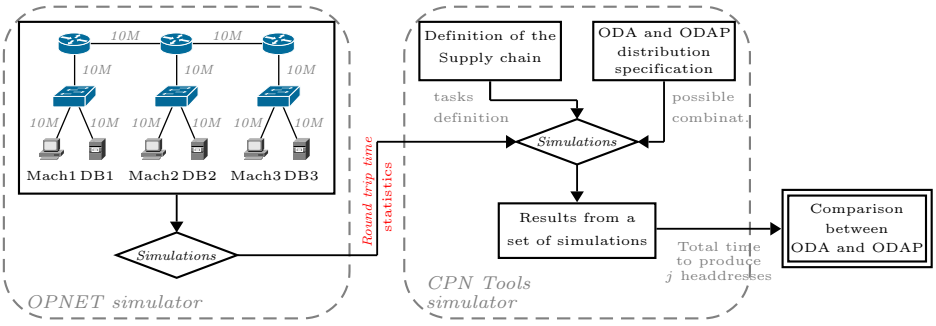


**Fig. 4.** Usage process of evaluation architecture

[2] Time between sending the first packet of the request and receiving the last packet of the response

### 4.2 Estimated "round trip times" of ODA via OPNET

First, the network interconnecting the client machines and the fixed databases is defined in OPNET (see figure 4). Thereafter, it is necessary to create system partitions on each server in order to allocate the data fragments (i.e the ODA distribution specified in the figure 2). Therefore, a replicating protocol has to be implemented owing to the replication of $F5$, $F7$ and $F8$. In our application, we implement Synchronous and Primary copy mechanisms described in the section 2. Subsequently, it is necessary to specify the applicative exchanges between equipments, i.e the query pattern (read/write) performed by each client machine on the databases. To do this, three models are implemented in OPNET: *Task*, *Application* and *Profile* models. Finally, it is possible to estimate the *round trip time* for a specific query sent from a client to a database.

Statistical tools are available in OPNET for computing averages, variances or still confidence intervals based on a set of simulations. In our study, 50 simulations have been running for a same scenario and then, both the *round trip time* average and the statistical variance have been extracted for each query. For instant, the table 1 gives the *round trip time* induced by a read (R) or write (W) query on $F1$ (fragment allocated to DB1). The machine 1 requires $3.6ms$ on average with a variance of $9\mu s$ to access to this fragment and spends $7.6ms$ and $7\mu s$ respectively to write it.

**Table 1.** Evaluated times regarding access query patterns: S-Ue

| | | Machine 1 | | | Machine 2 | | | Machine 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | DB1 | DB2 | DB3 | DB1 | DB2 | DB3 | DB1 | DB2 | DB3 |
| F1 | R | $3.6ms, 0.009\mu s$ | × | × | × | × | × | $7.8ms, 0.012\mu s$ | × | × |
| | W | $7.6ms, 0.007\mu s$ | × | × | × | × | × | $11.3ms, 0.015\mu s$ | × | × |

### 4.3 Petri Nets: ODA and ODAP architectures

As illustrated on the evaluation architecture (figure 4), the estimated *round trip times* are injected into CPN Tools and more exactly, they shall be set on timed transitions which reflect the read/write actions on databases. Second views describe the considered operations. For instance, the second view of the cutting 1 operation (introduced section 3.2) is presented in the figure 5. Both views of this operation, related to the ODA and ODAP architectures, are shown in the same figure because they are almost similar except for the read and write phases which will be discussed in more detail below. Three places from these two Petri Nets are bound to the first view (see figure 3), namely the place $Bobbin1(c)$ (port: $In$), the place $DB$ in which the three databases are defined (port: $I/O$ since they are shared resources between the three operations) and the place $pt1(c)$ in which cut pieces are stored (port: $Out$).

Let us focus now on the Petri Net structure of the cutting 1 operation (see figure 5): when a bobbin1/$c$ (i.e bob1/$c$) arrives into the queue for being cut

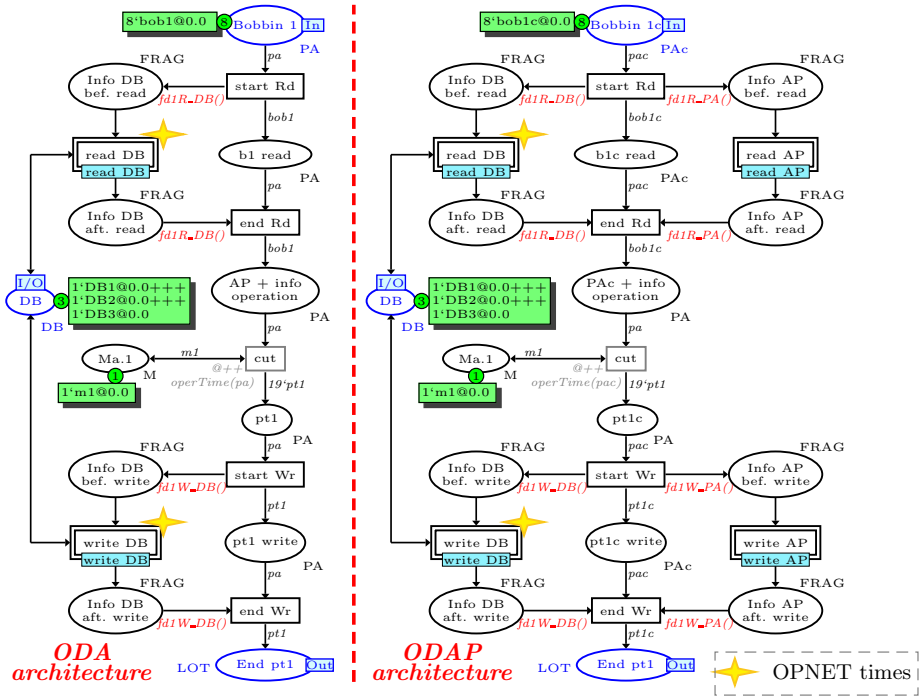


**Fig. 5.** Petri Net structure of the operation: cutting 1

(i.e places *Bobbin 1/Bobbin 1c*), we generate straight away the data fragments needed by the machine 1 for starting the operation (fabrication orders...). Let us remind the machine 1 needs to read the following fragments: $F1$, $F4$, $F5$ and $F8$. Considering the ODA architecture, these fragments are all allocated on the fixed databases, i.e into the place *Info DB bef. read*. Considering now the ODAP architecture, one part of these fragments can be allocated on the product and another part on databases. To do this, we add to the ODAP view the place *Info AP bef. read* to indicate that fragments should be read on the product rather than on databases. Thus, several distribution patterns between product and fixed databases can be defined. One possible combination might be to allocate $F1$, $F5$ to the product and $F4$, $F8$ to the databases. After having read the fragments, the cutting task (denoted "cut" transition) can start. The write phase is similar to the read phase. This Petri Net structure is used for the other operations: cutting 2 and sewing.

Views of the third level deal with the read and write phases of fragments, either on databases and on products, i.e the transitions denoted *read/write DB* and AP: figure 5). With regard to the *read/write DB* transitions, we added the statistical *round trip times* extracted from OPNET taking into account the client machines, the databases and the fragments (see table 1). With regard to the *read/write PA* transitions, we define several throughputs in the next section.

## 5 Results and Analysis

Considering the ODA architecture as our reference model, the purpose of our experimentation is then to determine the factors impacting positively or negatively on the ODAP architecture performance, and to identify configurations where ODAP should be benefit. In this article, we aim to study the influences of two parameters which are the communicating product throughput and the fragment distribution pattern. In fact, communicating products can exchange data with their environment at a given throughput. For our experiments, we consider 4 levels of throughput: $100Mbps$,$54Mbps$, $11Mbps$ and $1Mbps$. A fragment distribution pattern indicates the simulator how to place the different fragments, either on the distributed database or on the product as explained in the previous section. It is composed of eight boolean values [$F1$, $F2 \ldots F8$]; $\overline{Fi}$ meaning that the fragment $i$ is located on the database and $Fi$ meaning the fragment $i$ is located on the product. For example [$\overline{F1}$ $\overline{F2}$ $\overline{F3}$ $\overline{F4}$ $\overline{F5}$ $F6$ $\overline{F7}$ $F8$] informs the simulator that only fragments $F8$ and $F6$ should be placed on products and the others let on the database. In practice, all the different possible dissemination patterns are tested for a given throughput, which leads to 256 ($2^8$) experiments per throughput. Each experiment is simulated 10 times and the mean times needed to produce 85 headdresses using ODA and ODAP architectures are recorded. This number has been defined arbitrarily.

Table 2 summarizes the results obtained during the experimentation. Each line of the table corresponds to a given throughput value and each column to a specific configuration of the data distribution pattern: ODA (all fragments are allocated to the databases), full ODAP (all fragments are located on the product), and best hybrid ODAP configuration (some fragments are on the database, others on the product). Time values obtained for a given throughput and configuration are then reported in the table. As can be seen, for our scenario, full ODAP and ODA are quite similar in terms of performance, when considering $100Mbps$, $54Mps$ and $11Mbps$ throughputs. Therefore, it might appear that disseminating information all over the different informational vectors has no influence on the manufacturing time if the product throughput is high enough. With a correct throughput, it is then possible to imagine an information system completely distributed on a product network. When decreasing, the throughput yet acts as a very important constraint and full ODAP is clearly a bad solution. However, the best hybrid configuration always gives good results, which means some data can be stored on the product no matter what the throughput is.

In the figure 6 are plotted two curves representing the ODA and ODAP times ($y$-axis) for producing 85 headdresses, with a throughput of $1Mb/s$. On the $x$-axis are represented the $2^8$ possible combinations of distribution. Clearly, the distribution pattern has a very important effect in that case. In fact, the time needed to complete the production varies from $2'25''$ to $17'26''$. Based on these observations, it might be interesting to determine if each fragment has the same impact on the manufacturing time and if not, to identify the important fragments and evaluate their influence. To do so, a statistical analysis of the experiments done with a $1Mbps$ throughput has been initiated and showed that

all fragments are important, but some of their interactions as well. An interaction between 2 fragments means that the impact of these fragments all together on the product is different from the sum of the impacts of each fragment. In our case, interactions up to level 3 (interactions of three fragments) have to be taken into account. In the following, we use the term "factors" to denote significant fragments and interactions. For our scenario, there are up to 51 factors, as reported on the histogram $x$-axis in the figure 7. The impact of each factor is then estimated thanks to a multiple regression analysis that determines for each factor a coefficient, related to the linear regression equation. The value of a factor coefficient could be roughly considered as the effect of this factor on the manufacturing time. The higher the coefficient value, the more the manufacturing time increases. The figure 7 clearly shows that some fragments have a very important effect ($F1$, $F3$, $F5$), and others have a very moderate one, sometimes equal to interactions of level 2 (e.g the effect of $F4$ is almost the same as $F7 * F8$). As a result, our study demonstrates there are lots of factors with different effects to take into account when setting up an ODAP.

**Table 2.** Times obtained for 3 fragment distributions according to 4 throughputs

| Product throughput | ODA distribution | Full ODAP distribution | Best hybrid distribution |
|---|---|---|---|
| $100Mbps$ | $2'27''$ | $2'28''$ | $2'27''$ ($F5$,$F7$) |
| $54Mbps$ | $2'27''$ | $2'28''$ | $2'27''$ ($F5$,$F6$,$F7$,$F8$) |
| $11Mbps$ | $2'27''$ | $2'52''$ | $2'27''$ ($F7$) |
| $1Mbps$ | $2'27''$ | $17'26''$ | $2'28''$ ($F7$) |

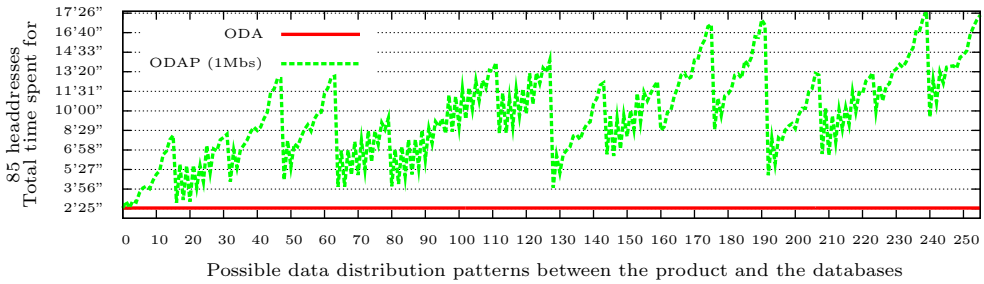


**Fig. 6.** Comparison between ODA and ODAP (product throughput= $1Mbps$)

## 6 conclusion

A multitude of informational vectors take place in Supply Chain environments as fixed databases or manufactured products on which we are able to embed significant proportion of data. By considering distributed database systems, specific data fragments can be embedded/allocated on these products (for example, data useful for their life cycle). The paper analyzes three specific distribution patterns

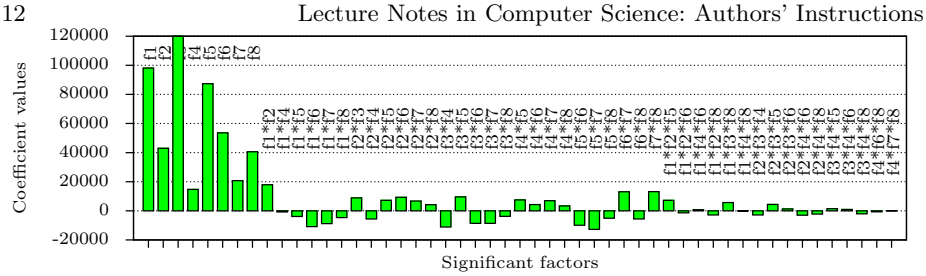


**Fig. 7.** Significant factors and their respective coefficient values

to determine the different parameters impacting on the manufacturing system performance and shows that choosing a good pattern is complex. In a further work, we are willing to implement an experimental design leading us to control the best way to disseminate information on the informational vectors.

## References


1. Apers, P.: Data allocation in distributed database systems. ACM Transactions on Database Systems (TODS) 13(3), 263–304 (1988)
2. Ausen, D.: Fobis: Foresight biomedical sensors. In: FOBIS-NICE meeting (2006)
3. Chan, D., Roddick, J.: Context-sensitive mobile database summarisation. In: 26th Australasian computer science conference. vol. 16, pp. 139–149 (2003)
4. Chan, D., Roddick, J.: Summarisation for Mobile Databases. Journal of Research and Practice in Information Technology 37(3), 267 (2005)
5. Gershenfeld, N., Krikorian, R., Cohen, D.: The internet of things. Scientific American 291(4), 76–81 (2004)
6. Hababeh, I., Bowring, N., Ramachandran, M.: A method for fragment allocation design in the distributed database systems. In: The Sixth Annual UAE University Research Conference (2005)
7. Huang, Y., Chen, J.: Fragment allocation in distributed database design. Journal of Information Science and Engineering 17(3), 491–506 (2001)
8. Kubler, S., Derigent, W., Thomas, A., Rondeau, É.: Problem definition methodology for the "Communicating Material" paradigm. In: IFAC Workshop on Intelligent Manufacturing Systems (07 2010)
9. Kubler, S., Derigent, W., Thomas, A., Rondeau, É.: Prototyping of a communicating textile. In: Industrial Engineering and Systems Management (2011)
10. Ley, D.: Becta. Ubiquitous Computing, emerging technologie 2, 64–79 (2007)
11. Lubinski, A.: Small database answers for small mobile resources. In: Intelligent Interactive Assistance and Mobile Multimedia Computing. pp. 9–10 (2000)
12. Meyer, G., Främling, K., Holmström, J.: Intelligent products: A survey. Computers in Industry 60(3), 137 – 148 (2009)
13. Morel, G., Grabot, B.: Special issue on intelligent manufacturing. IFAC Journal of Engineering Applications of Artificial Intelligence 16(4), 271–393 (2003)
14. Navathe, S., Ceri, S., Wiederhold, G., Dou, J.: Vertical partitioning algorithms for database design. ACM Transactions on Database Systems 9(4), 680–710 (1984)
15. Navathe, S., Karlapalem, K., Ra, M.: A mixed fragmentation methodology for initial distributed database design. Journal of Computer and Software Engineering 3(4), 395–426 (1995)

16. Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A survey of data replication techniques for mobile ad hoc network databases. The VLDB Journal 17(5), 1143–1164 (2008)