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 between manufactured products and fixed databases.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. [9] 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 [7], we highlight several possible scenarios for intelligent products 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...) or even invisibles (concept of disappearing computer, ubiquitous computing). More generally, the concept of 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 [11], 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 [12]). 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 [8] 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: *CPN tools* and *OPNET Modeler*. This piece of software is presented in the section 4 and assesses 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. 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 may access to product information for diverse reasons, either during the design phase, the usage phase or still the recycling 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 Fig. 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 [9], 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 of the data fragments 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.

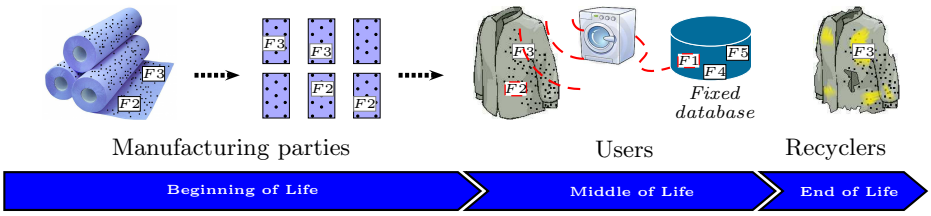

Fig. 1. Information distribution on products and fixed databases

2.2 Distributed databases through literature

The main constraint 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. Data distribution can generally be accomplished by two consecutive 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 the environment and some applicative constraints.

Basically, the partitioning aims at subdividing the relational data model. Thus, the resulting fragments will be allocated to specific informational vectors in order to improve system performance. Three types of fragmentation exist: vertical [13], horizontal [1], mixed/hybrid [14]. 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 its 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 to minimize or maximize [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, saving time for users) [15]. 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 [3,10] (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 data fragments (8 in total: [$F1..F8$]). The resulting optimal allocation [6] is depicted on the Fig. 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...

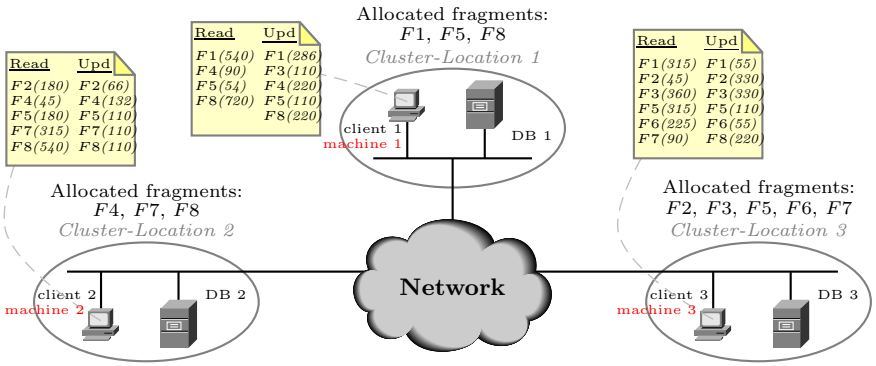

Fig. 2. Optimal distribution architecture established by [6]

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 manufacturing operations, transport phases... and can be performed by diverse suppliers. These suppliers may dispose of local databases where their own information system is implemented (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. Specifically, the scenario is related to the manufacturing of a simplified headrest composed of two textile parts. Each part is cut out of a different textile bobbin and then

sewn together. As a result, this scenario is divided into three tasks carried out by three suppliers respectively. The operation *cutting 1* and *cutting 2* are performed in parallel by the supplier 1 and 2, the resulting textile pieces are sewn thereafter by the supplier 3. Each supplier disposes of 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...). In order to adapt the case study of Hababeh to our logistic scenario, we assume that each supplier's location corresponds to the clusters 1, 2 and 3 introduced in the Fig. 2 and by this fact, we match applicative characteristics defined for each client in Hababeh to each supplier's machine. In other words, the machine 2 has the same read/write access pattern on the set of data fragments than the client 2 defined in Hababeh and so on. Likewise, each location disposes of a local database and shares the same relational data model, 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 is defined as shown in the Fig. 2: $F1$,$F5$,$F8$ allocated to DB1, $F4$,$F7$,$F8$ to DB2...

The Fig. 3 illustrates in form of Petri Nets the synoptic of our logistic process. Each operation is defined by a transition, and let us note that we design the Petri Net model by working on hierarchical views. Consequently, the distribution aspect, in other words, the optimal distribution of the data fragments described previously will be detailed in the lower views (i.e in the section 4.3). This first distribution does not take into account the possibility to allocate data fragments on products. However, we dispose of communicating products on which data fragments can be stored. Therefore, two types of products through the logistic process are implemented and thereby, two types of architectures are feasible:

- *DiPA* (**Di**screte **P**roduct **A**rchitecture): no possibility to allocate data fragments on products, we consider only discrete products[1],
- *CoPA* (**Co**mmunicating **P**roduct **A**rchitecture): data fragments can be allocated to communicating products.

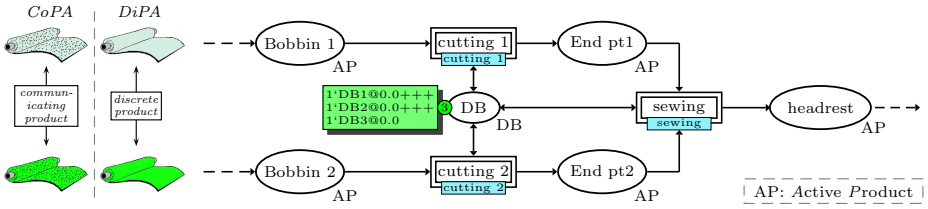

Fig. 3. Global view of the logistic process

The Fig. 3 illustrates this point in which discrete products (discrete bobbins) are implemented on the one hand, and communicating products (communicating

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

bobbins) are implemented on the other hand through the logistic process. The idea is to highlight the benefits that could be achieved regarding one or the other of these architectures, bearing in mind that manufactured products act as mobile databases in the *CoPA* architecture as opposed to a classic one (*DiPA*).

4 *DiPA* and *CoPA* architecture modeling

4.1 Architecture

A description of how the assessment and the comparison are undertaken of both architectures (*DiPA* and *CoPA*) is proposed in this section. The evaluation architecture relies on two discrete events simulators and its usage process is depicted on the Fig. 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). The logistic process sequence described in the section 3.2 is simulated via this tool as shown in the Fig. 4. 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 *CoPA* architecture) and so on. The *DiPA* and *CoPA* distribution patterns are specified in this tool. Let us note that for the *CoPA* 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 fixed databases.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 following sections describe respectively each tool.

4.2 Estimated "round trip times" via OPNET

First, the network interconnecting the client machines and the fixed databases is defined in OPNET (see Fig. 4). Thereafter, it is necessary to create system partitions on each server in order to allocate the data fragments as specified in the Fig. 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 from OPNET are used: *Task*, *Application* and *Profile* models.

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

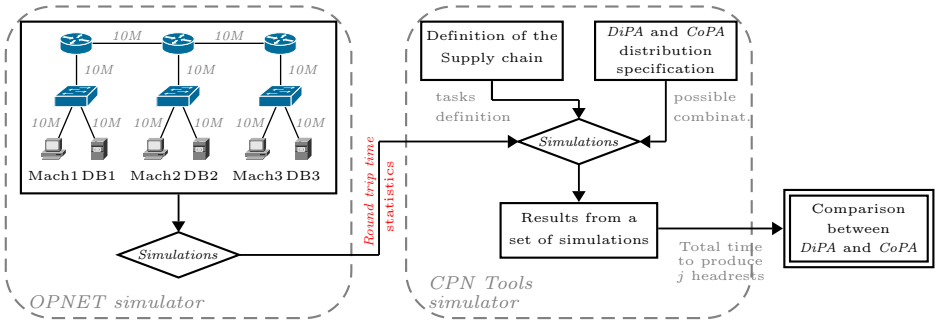

Fig. 4. Usage process of evaluation architecture

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, both the *round trip time* average and the statistical variance have been extracted for each query and 50 simulations have been running for a same scenario. For instance, the table 1 gives the *round trip time* induced by a read (R) or write (W) query on $F1$ (fragment allocated to DB1) and $F6$ (fragment allocated to DB3). 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. Likewise, the machine 3 requires $4.8ms$ on average with a variance of $23.24\mu s$ to access to $F6$ and spends $10.02ms$ and $29.9\mu s$ 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	$3.6ms, 9\mu s$	×	×	×	×	×	$7.8ms, 12\mu s$	×	×
	W	$7.6ms, 7\mu s$	×	×	×	×	×	$11.3ms, 15\mu s$	×	×
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
F6	R	×	×	×	×	×	×	×	×	$4.8ms, 23.24\mu s$
	U	×	×	×	×	×	×	×	×	$10.02ms, 29.9\mu s$

4.3 Petri Nets: *DiPA* and *CoPA* architectures

As illustrated on the evaluation architecture (Fig. 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. The second views of the Petri Net describe each operation. For instance, the Fig. 5 shows the

second view of the operation *cutting 1*. Both views of this operation, related to the *DiPA* and *CoPA* architectures are shown in the same figure. The only difference between both architectures lies in reading and writing data fragments which will be discussed in more details below. Three places from these two Petri Nets are bound to the first view (see Fig. 3), namely the place *Bobbin 1* (port: *In*), the place *DB* in which the three databases are defined (port: *I/O*) and the place *End pt1* in which cut pieces are stored (port: *Out*).

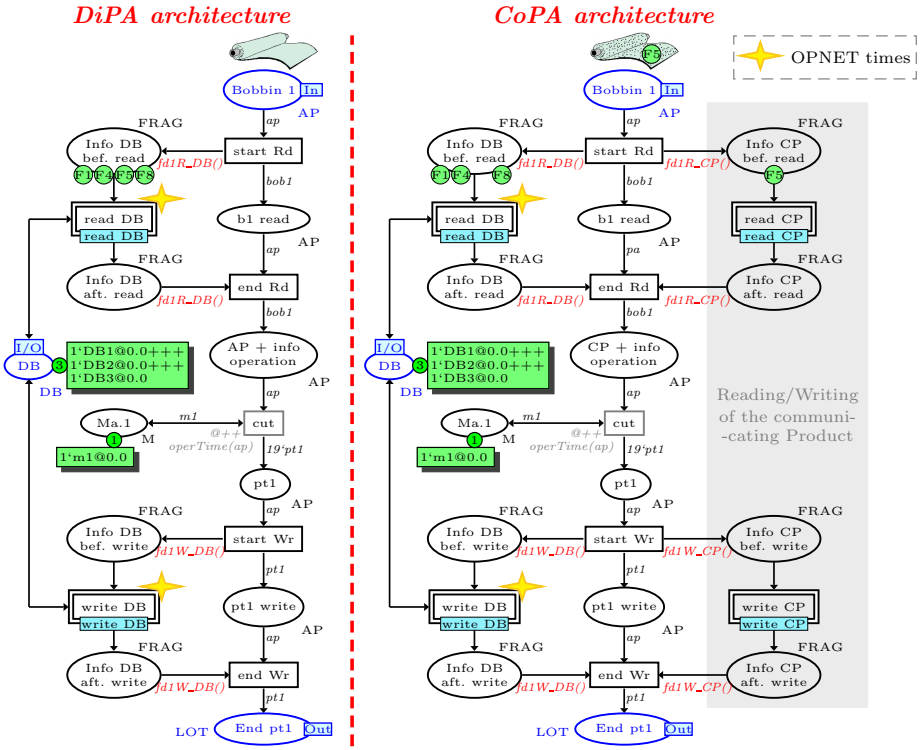

Fig. 5. Petri Net structure of the operation: *cutting 1*

Let us focus now on the Petri Net structure of the *cutting 1* operation (see Fig. 5). When a bobbin 1 arrives into the queue for being cut (i.e in the places denoted *Bobbin 1*), 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 *DiPA* architecture, these fragments are initially all allocated to the fixed databases, i.e into the place *Info DB bef. read*. Considering now the *CoPA* architecture, one part of these data fragments can be allocated on the product and another part on databases. To do this, we add to the *CoPA* view the right part (highlighted in gray in the Fig. 5) to indicate that fragments should

be read/write 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 $F5$ to the product and $F1$, $F4$, $F8$ to the databases, as shown in the Fig. 5. After having read the fragments, the cutting task (denoted "cut" transition) can start and then each resulting piece of textile are writing by a similar method than the read phase. This principle is reproduced for the other operations: *cutting 2* and *sewing*.

The time to read/write data fragments on fixed database is equal to the statistical *round trip times* extracted from the OPNET simulator 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 product throughputs, in other words, the time needed to read and write the fragments allocated to the communicating product are computed based on a specific throughput.

5 Results and Analysis

5.1 Simulation and Results

Considering the *DiPA* architecture as our reference model, the purpose of our experimentation is then to determine the factors impacting positively or negatively on the *CoPA* architecture performance, and to identify configurations where *CoPA* 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 data 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 data fragment i is located on the database and Fi meaning the data 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 $F8$ 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 headrests using *DiPA* and *CoPA* architectures are recorded. This number has been defined arbitrarily.

The 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 data distribution pattern: *DiPA* (all fragments are allocated to the databases), full *CoPA* (all fragments are located on the product), and best hybrid *CoPA* configuration (some fragments are on the database, others on the product and we note down the solution given the smallest time). Time values obtained for a given throughput and configuration are then reported in the table. As can be seen, for our scenario, full *CoPA* and *DiPA* are quite similar in terms of performance when considering $100Mbps$, $54Mps$ and $11Mbps$ throughputs.

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

Product throughput	*DiPA* distribution	Full *CoPA* distribution	Best hybrid distribution
$100Mbps$	$2'27''$	$2'28''$	$2'27''$ ($F5$,$F7$)
$54Mbps$	$2'27''$	$2'28''$	$2'27''$ ($F5$,$F6$,$F7$,$F8$)
$11Mbps$	$2'27''$	$2'52''$	$2'27''$ ($F7$)
$1Mbps$	$2'27''$	$17'26''$	$2'28''$ ($F7$)

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 *CoPA* 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.

In the Fig. 6 are plotted two curves representing the *DiPA* and *CoPA* times (y-axis) for producing 85 headrests, with a product 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 $17'26''$. As a result, when weak throughputs are considered, it is then really important to know which pattern to use in order to prevent performance loss.

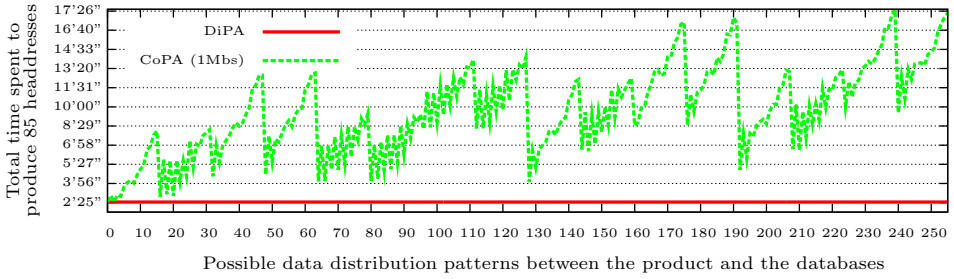

Fig. 6. Comparison between *DiPA* and *CoPA* (product throughput= $1Mbps$)

5.2 Key factor identification

Based on these observations, it might be interesting to determine whether each data fragment has the same impact on the manufacturing time in order to identify the critical data fragments (which impact negatively on the manufacturing

time) and then, to identify the reasons. To do so, the impact of each fragment and of their interactions on the manufacturing time is first studied, based on a statistical analysis of the experiments done with a product throughput equal to $1Mbps$. The key factors are identifying in a second step.

A statistical analysis shows that all data fragments have a significant impact on the manufacturing time, but some of their interactions as well. An interaction between 2 fragments (Histogram denoted Level 2) means that the impact of these fragments all together on the product is different from the sum of the impacts of each fragment. For our scenario, there are up to 35 non-negligible interactions, as reported on the histogram x-axis in the Fig. 7. The influence of each fragment and fragment interaction is then estimated thanks to a multiple regression analysis, represented by a coefficient value (y-axis) related to the linear regression equation. The value of this coefficient could be roughly considered as the effect of the data fragments on the manufacturing time when there are allocated to the communicating product. The higher the coefficient value, the more the manufacturing time increases. For instance, we can observe in the Fig. 7 that when the fragment $F5$ is allocated to the product, it impacts more on the manufacturing time than $F8$. The Fig. 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 smaller than $F7 * F8$). One can then wonder why some data fragments impact more on the manufacturing time than other ones? In what follows, a study is carried out in order to identify the reasons of this behavior and then the key factors.

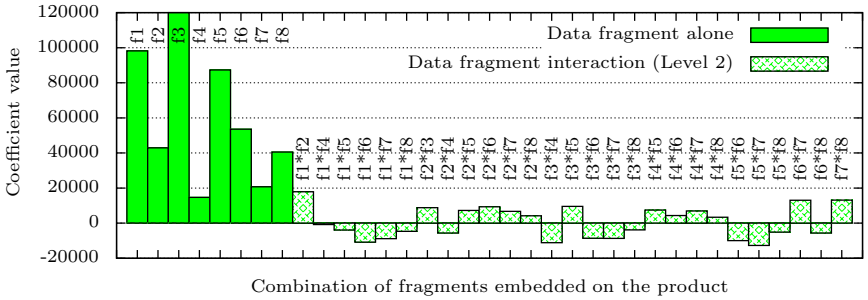

Fig. 7. Significant factors and their respective coefficient values

The aim is now to analyze and to identify the reasons why the coefficient values are more or less important. Each fragment size requested by each operation (*cutting 1*, *cutting 2* and *sewing*) is represented through the *Global process* histogram in the Fig. 8. Let us consider $F2$, the machine 1 does not read $F2$, the machine 2 reads and writes respectively 180 and 66 bytes of $F2$, and the machine 3 reads and writes respectively 45 and 330 bytes (see the access pattern in the Fig. 2. In total, 621 bytes of $F2$ are requested. Then, we normalize the fragment size by dividing it by the higher fragment size, i.e $F8$ with regard to

the *Global process* size (equal to 1810 bytes). On the second histogram, only the size of each fragment requested during the *sewing* operation (i.e by the machine 3) is represented, since it is the bottleneck of the logistic process. Regarding $F2$, the number of bytes requested by the machine 3 is equal to 375 bytes $(45 + 330)$ as shown on the *sewing* operation histogram, which is dividing by the higher fragment size, i.e $F3 = 690$ bytes. By focusing on the *sewing* operation, we can observe that $F1$, $F3$ and $F5$ are the biggest, which may partly explain the coefficient values of these fragments on the Fig. 7. However, it is not sufficient to explain it because $F2$ has almost the same size than $F1$ but the coefficient value of $F2$ is really smaller than $F1$. But, if we look at the *Global process* histogram, we note that $F1$ is bigger than $F2$, which may explain the significant difference about the coefficient values of $F1$ and $F2$ in the Fig. 7 Conversely, we can see that the coefficient value of $F8$ is not so high in spite of the size of $F8$, significantly bigger than the other data fragments (see the *Global process* histogram), but if we take a look at the *sewing* operation, we can remark that $F8$ is not so big. In conclusion, the *sewing* operation which is the bottleneck of the logistic process seems to impact significantly more the manufacturing time than the other operations. Thereby, it may be sensible to focus on bottleneck operations, tasks on the supply chain for the data dissemination issue, without forgetting to overlook the global process (as illustrated with $F1$ and $F2$). Thus, the data fragment size and the operation characteristics influence strongly the manufacturing time. Consequently, it may be necessary to reconsider sometimes the fragmentation method, which generates the set of data fragments.

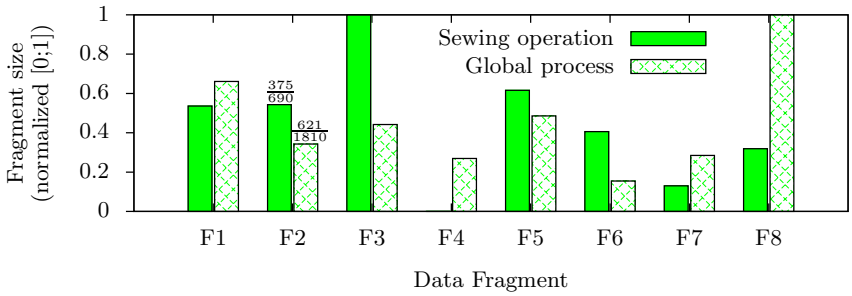

Fig. 8. Fragment size of each fragment with regard to specific operations

6 Conclusion

A multitude of informational vectors take place all along the 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 various distribution patterns between manufactured products and databases in order to identify the

parameters impacting the most on the manufacturing system performance and especially on the manufacturing time. This study shows that choosing a good pattern is not quite so simple. In a further work, we are willing to implement an experimental design leading us to control the best way to disseminate information on both vectors of information: manufactured products (e.g communicating products) and fixed databases.

Acknowledgments We thank OPNET Technology Inc. for providing the software license to carry out the simulations of this research and, the financial support of the CPER 2007-2013 "Structuration du Pôle de Compétitivité Fibres Grand'Est", through local (Conseil Général des Vosges), regional (Région Lorraine), national (DRRT and FNADT) and European (FEDER) funds.

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. Kubler, S., Derigent, W., Thomas, A., Rondeau, É.: Problem definition methodology for the "Communicating Material" paradigm. In: IFAC Workshop on Intelligent Manufacturing Systems (2010)
8. Kubler, S., Derigent, W., Thomas, A., Rondeau, É.: Prototyping of a communicating textile. In: Industrial Engineering and Systems Management (2011)
9. Ley, D.: Becta. Ubiquitous Computing, emerging technologie 2, 64–79 (2007)
10. Lubinski, A.: Small database answers for small mobile resources. In: Intelligent Interactive Assistance and Mobile Multimedia Computing. pp. 9–10 (2000)
11. Meyer, G., Främling, K., Holmström, J.: Intelligent products: A survey. Computers in Industry 60(3), 137 – 148 (2009)
12. Morel, G., Valckenaers, P., Faure, J.M., Pereira, C.E., Diedrich, C.: Manufacturing plant control challenges and issues. Control Engineering Practice 15(11), 1321 – 1331 (2007)
13. Navathe, S., Ceri, S., Wiederhold, G., Dou, J.: Vertical partitioning algorithms for database design. ACM Transactions on Database Systems 9(4), 680–710 (1984)
14. 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)
15. 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)