ABSTRACT. There is an increasing demand from firms to evolve their applications to be more co-operative. The problem is to realise this migration and to maintain the quality of existing applications which are reliable due to previously encountered difficulties. We will propose a method based on an organisation of dynamic workgroups (their composition can evolve with time) composed of components. Thus method is entirely derivable (after a formal verification) into rules integrated into a co-operative architecture allowing the co-operation.

In order to propose an operational method, the last stage will propose to obtain rules derived from a specification language associated to the method. Our approach allows the provision of co-operation thanks to the application re-engineering without modifying any existing component.

KEY-WORDS. Component, co-operation, engineering, method, re-engineering, re-use.

1. Introduction

More and more applications are composed of distributed components (a component corresponds to a production cell, an automata, a software component, etc.) linked across a network. Even if these components work with a common goal, we cannot tell they co-operate. This is a too weak co-operation.

A growing need to provide a migration of these non-co-operative distributed applications (often old) towards co-operation has been expressed by firms [14]. Components of such applications cannot synchronise, they don’t know how to exchange information and they are not able to constitute themselves into workgroups (set of components with a common goal and representing one entity). Generally, we can establish some communications with the help of external operators (human or dedicated machines – e.g. Figure 1), but it is impossible to have or workgroup organisation able to evolve as the time running with an integration/exclusion of components. How many times we do not encounter such organisations where the task of workstations are just to do the re-input of information which have been already entered, or to get another time one information to do a synthesis whereas it should have been more efficient if we were able to get it as it was produced?

The problem is that computer science responsible hesitate, and sometime refuse heavy modifications of these applications. Most of the time they are running for a long time, as long as they are reliable. Sometimes, source code is not easily adaptable : abilities are not present info the enterprise, the source code is lost or is no longer maintainable, etc.

Most of the time the only thing granted is the “transplantation” of a network remaining under-exploited.
Our proposition is to remove all or part of external operators expressed on the schema above (i.e. Figure 1). We want to automatically manage the circulating of information. We also want to manage the organisation into dynamic workgroups [3]. After the re-engineering of the application (i.e. Figure 2), information which were previously dispatched with external operators are now managed automatically across the network. They are directly communicated from the producer to the consumer.

To provide an automatic co-operation we need a complete re-organisation of the application. In the method we propose, we consider two abstraction levels: the (dynamic) workgroup and the operative component. We base the co-operation on the constitution of workgroups and on the communication with elements of co-operation (events, messages and data).

1.1 From the Method…

The method dissociates the co-operative aspects on which we base on our contribution, from the operative aspects done by components (the constructive unit).

A component represents an operative device (hardware or software), eventually complex, pre-existing and not updatable. Its is seen as a black box [5] needing and producing information. It can achieve one task and is able to communicate with (only) its environment. The set of information needed/produced by a component are only from and to its environment (i.e. Figure 3). Its environment is composed of the set of entities with which is can directly communicate or interact.
We have a top-down approach. We start from the highest level (the constitution of workgroups – virtual entities) to the lowest (components composing workgroups). We bring to the fore the need to have and to manage dynamic workgroups. Next, we will show the key elements to have the dynamic and the information (persistent or not) which have to be exchanged and shared between and into these workgroups.

These preliminary steps realised, we will see how to link some elements of co-operation and how to manage the missing ones (provided before with the external co-operation). Because we wish to have a global approach, the last step of the methodology will be to derive it into a set of rules [17]. They will manage workgroups and their dynamic, they will also manage the circulating of the various identified elements of co-operation.

1.2 …to the Implementation

The solution we propose consists in providing a co-operative platform managing automatically the distribution of elements of co-operation, the creation of synthetic information, and the constitution and the management of workgroups. This co-operative layer will consequently be able to identify and to dispatch elements of co-operation. Therefore, the proposed method have to identify key elements and have to use them to provide the co-operation. The goal is to obtain a more coupled co-operation.

To avoid any misunderstanding, we have to precise an important point. We do not conceive from scratch co-operative applications. So we are only interested in constitution and evolution aspects of workgroups, and on communication aspects between components and groups. We do not want to intervene on their functionalities, neither on the information system as it should be using semantic inter-operability techniques (these techniques allows the co-operation between different modelling languages and between heterogeneous databases [1]). In particularly, we suppose that operative components already exist, and we suppose that the information system of the application has already been defined. This suppositions are natural with a re-engineering approach. So, our contribution only deals with the technical inter-operability.

2. The co-operation into the application

An efficient co-operation needs a good communication as well as a good constitution into workgroups. This point is as important as the work achieved by components. A workgroup is a set of eventually distant components with a common goal. More generally, we use the notion of dynamic workgroups to answer as well as possible to the needs of the moment. They represent an additional abstraction level. They are added to the application with the objective to realise a new structure of it. On the contrary with reverse-engineering, re-engineering allows to modify semantic [9].

For example, when an urgent command is demanded, it is necessary to quickly adapt the production. Production cells may be totally or partially taken away from a less imperative activity to be allocated to another more important production task. Components will only receive information they need in their current work, respecting the famous adage “too much information kills information”. Beyond this caricature of the previous tense, this workgroup organisation, totally transparent to components, allows to regulate the circulating of information. It also allows to better target at potential addressees. An over-abundance of information implies sorts and so it slows down reactions when there is pertinent information. A good organisation of workgroups also allows an optimal management of components. It avoid some loss of time issued of the amount of information to be processed. It is particularly suitable with a real-time context

Over the gain provided with the filtering of information, we have to notice that the role of a component may be modified according to its workgroup belonging. For example, a component has to pilot a physical device. If we place it into a workgroup not containing this physical device, outputs it provides will not be communicated to the device but may be captured and sent to another component. Thus, this component will do a simulation or can do a control beside the component which actually pilot the physical device (very useful for example for fault tolerance management)

Constituted workgroups can exchange information between each of them. The co-operative platform has permitted the constitution and the distribution of workgroup information. They have to be constituted from those available from the components constituting the workgroup level. They represent synthetic information reflecting the whole workgroup activity. They also allow to link workgroups and so to enrich the information system of the application with a higher semantic, more suitable with a co-operative context. These information should be exploited to manage traceability knowledge and know-how capitalisation problems of the firm.
We can now represent our system with the following schema. Each component is included into one or several workgroup (i.e. C_i at Figure 4):

As we constitute workgroups, we need to establish precise rules. The enter or the exit of a component to/from a workgroup can only be done respecting some well defined rules. These rules are established according to the concerned entities, but also according to the global state of the application. This state is only known by the co-operative plat-form.

For example, a robot can be extracted from its workgroup only once it has achieved its current task. Another example with a workgroup composed of graphical interfaces associated to their corresponding workstation. If three interfaces (and so, workstations) are enough to the work, it is not necessary to include another one. It may even be harmful in case of lack of resources for example.

We need to ensure the coherency with the dynamic workgroup composition in order to avoid under/over work, inter-blockages, etc.).

The input/output management of each workgroup have to take into account the local and the global level. Each component individually produce information into its environment. The co-operative plat-form captures these information, analyse them to know the activity and the state of the workgroup. Thus, it will be able to decide for example to exclude a component from a workgroup if its activity do not justify its belonging.

**Presentation of the example**

To experience our approach, we decided to test it on a concrete example. So we contacted a road transport specialist in parcel distribution (the LAUSSUY Group). We applied our method to provide co-operation between existing components. According to the constraints, the objective was to improve the flexibility of critical parts (good supply management at the hub of storage/warehousing) and the dispatching of information (real time state of stocks). We will now briefly describe the functioning of this firm.

**Parcels**

A parcel may be carried by a customer himself to the firm. He deposits the parcel specifying the destination. Then, he receives an invoice. As soon as that parcel has been deposited, it will be registered (information such as weight, volume, and destination will be entered) and placed on a conveyor belt. Each parcel is labelled with an unique identifier until final delivery. This identifier allows real time follow-up of the parcel (in the way UPS - United Parcel Service - http://www.ups.com - do on Internet).

**Routing progress**

According to the destination (regional or national), a parcel is dispatched on a conveyor belt which brings it to the corresponding aisle of storage. In these zones, parcels are grouped on European pallets with specific dimensions (0.80 * 1.20 meters). Then, operators carry out a loading plan to fill a truck to the maximum as well as possible (for example, for a semi-trailer, a loading plan will be equal to 33 European pallets). Computerised managing enables real-time stock states to be obtained and to know the processing stage of each parcel.
3. Presentation of the Method

3.1 Step 1 : Workgroups Level

The first abstraction level we consider is the workgroup level. We identify different workgroups able to be present as the application is running. This is an organisational level. The identification of these workgroups is realised according to preponderant constraints (geographical and hours constraints, method of work, etc.). These workgroups often corresponds to the structural divisions already present in the firm. Each workgroup represents an own entity. It has a special role, needs some information and produce other ones. To sum up, a workgroup has a sense into the organisational level.

The objective of the method is to provide co-operation thanks to an efficient communication. So, for each workgroup, we will identify elements of co-operation : events, messages and data exchanged. We will do a distinction between information need and those they produce. Each information will be precisely described with the natural language into a dictionary.

3.1.1 Workgroups Identification

![Figure 5: workgroups identification](image)

Reception workgroup has to receive parcels since they are left on the reception wharf: it is here that parcels are registered. According to their destination, parcels are sent to aisles and sent to consignees in the course of the national or regional round.

Parallel to parcel forwarding, the Market Service workgroup is kept informed of the developing state of parcels to be able to issue the corresponding memorandums and invoices.

Finally, the Truck Managing and Upkeeping group is the one concerned with maintenance and supervision of the fleet. It can also detect scarcity and indicate the type of truck which is lacking (Market Service may react and may possibly call congruent corresponding to ensure momentarily forwarding).

3.1.2 Elements of co-operation

![Figure 6: events needed/produced](image)

We can distinguish the three types of elements of co-operation previously described. For each of them, we have to add its enter into the dictionary (name, type, and semantic description as precise as possible).

Because we do not want to overload schemas, we will only present parts of schemas and we will only show events. Of course, each step have to be done for messages and data.

This chart presents for each workgroup, the events required and events produced. Thus, the National Shipment workgroup can (it can does not mean it will) react to the $E_{ACN}$ event (parcel available) and can trigger the event $E_{PN}$ (pallet ready available).
Once this work has been done at the workgroup level, we can go into the details of each one to study the component level.

### 3.2 Step 2: Components Level

We do the inventory of the set of existing components. They may be software (supply management, managing, interfaces etc) or hardware (production cells, programmable automaton etc). We identify workgroups where they can be members. To do that, we use our knowledge (described into the dictionary) of the role played by each component. Then, we will inventory events, messages and data produced and needed by each of them as we did at the previous step.

#### 3.2.1 Potential components membership to workgroups

As workgroups have a dynamic structure, components itemised are not all present at the same time. Moreover, a component can be a member of several workgroups (for example, a particularly expensive hardware component may be a member of several workgroups and so it is shared by all of them). In this case, one objective of co-operation is to organise this sharing according to application constraints.

For each workgroup, all potential component members for any one moment are described. We show three components HCI1, HCI2 and HCI3 possibly members of the National Shipment.

![Figure 7: potential membership within workgroups](image)

Components HCI1, HCI2 and HCI3 correspond to working stations of operators in charge of preparing a loading plan. They allow a real time state of loading of storage aisles. When a parcel is selected to be sent, it must be immediately removed from other interfaces to not be selected twice (thus, we avoid situations like a packer looking for a parcel already dispatched. It is an important loss of time source in this type of handling). We have now to study element of co-operations as we did it at the first step.

#### 3.2.2 Elements of Co-operation

We will illustrate our method by showing events for National Shipment workgroup that can be raised by components HCI1, HCI2, and HCI3 and thus, we reveal the required synchronisations:

![Figure 8: events needed/produced](image)

We will not detail all the events, but here are what could possibly be an entry into the dictionary.

- $E_C$: Signals the arrival of a new parcel to manage.
Step 1 and 2 allow the co-operative elements to be identified. They show what each component and workgroup require and produce. But, for the moment, we are not able to tell which components or workgroups are the addressees of events, messages and data produced. So, we focused on revealing co-operative elements but not the actual co-operation itself.

3.3 Step 3 : Dynamic Groups Management

We need to know the constraints for each component to do its enter/exit to/from workgroups. Some rules will manage this dynamic (dynamic rules are ECA rules - when an event E occurs, if the condition C is verified, the action A is executed). Each component will have a dynamic rule associated to it. It will have as many rules as workgroups with which it can be associated. These rules will allow us to know when the corresponding component has do enter or exit of the associated workgroup [3]. So, when a certain event occurs (to identify here), it will imply the enter/exit of the corresponding workgroup. The condition evaluated by the rule will take into account the global state of the application to know if the action is reasonable or not. The action of the triggered ECA rule is a modification of the state of the co-operation and will imply a new distributions of information.

3.4 Step 4 : Links and creation of missing elements of co-operation

Elements of co-operation needed to manage the dynamic of workgroups have been defined, itemised and detailed into the dictionary established at the previous steps. We will know focus on linking these elements. This work will be realised at the two previous abstraction levels (workgroup and component levels), but it will also be realised for dynamic rules.

Practically, such a link will sometime suppose to format the information to be compatible with its addressee. Detective rules (described into the next section) of the co-operative plat-form will do this work when necessary. So the addressees will receive the information as they can process it.

At the end of this step, some elements (events, messages and data) necessary to do the co-operation will not be linked. If they are entries, that means that they are no produced by any entity of the application. On the contrary, if they are outputs, that means they are not used to provide the co-operation into the application. This stage is the most important. With the help of the dictionary containing the semantics of each co-operative element, it allows to link elements to ensure the co-operation. If some information are not produced but are needed, we will see how we will create them thanks to detective rules.

3.4.1 Links between elements of co-operation.

Firstly, we establish all links to elements corresponding to inter group co-operation. For this work we rely on the previously established dictionary. For example, we have in the Reception group, $E_{CN}$ « parcel to national destination is ready » and the National Shipment workgroup, $E_{ACN}$ « parcel to national destination is in storage aisle ». These two elements do not have the same name but are linked because they are a synchronisation point. When $E_{CN}$ is raised, the parcel is registered (destination, weight, volume, price, etc.). So, it can be added to the list of to national storage aisle parcels, and these can then be updated to the corresponding interfaces (those present within the National Shipment workgroup). Even if these two events have different semantics, they are a synchronisation point.

![Figure 9 : links between elements of co-operation](image)

Links can be realised between workgroups, but also between components. It have also to be done for each element of co-operation.

3.4.2 Creation of missing elements of co-operation

At the previous step, we have linked elements of co-operation when this link was direct. Sometimes, some needed information are not produced anywhere (the link has not be established). Using once more time the dictionary, we
can see that these information can be produced composing other ones. When this is impossible as the current state of the analyse, that means that some elements of co-operation have been forgotten. So we can come back into the analysis to add them. In some cases, even with the rollback, the needed information is not available. Perhaps, this information is too theoretical, too precise and practically not producible. Nevertheless, a partial come back into the analyse should allows us to identify another one or to construct another information less precise but enough sufficient.

We have to keep in mind that we intervene on the re-engineering domain and thus, we cannot always have all information we wishes. That is one of the main limit and constraint of this domain. As the application we are working on already exists, these information are effectively produced. So we can ensure we will effectively be able to produce all needed information.

Elements of co-operation are directly available do not contain enough semantic to react to complex situations. The composition of information will allows us to obtain more semantic and richer ones. They allow us to react only to specific and eventually complex situations and so to improve efficiency of the application.

We have to propose a powerful mechanism to detect a precise occurrence schema. This schema may be for example a repetition of some distinct events according a period. In fact, it can be any situation which can be algorithmically expressed. On previous works [16], we proposed a mechanism, called detective rules, derived from ECA rules. These rules raise events (composed events) after a composition. The principle is the following : a detective rule has an algorithm which is entirely programmable and which detect some event occurrences.

Most mechanisms have been developed with BDA domain where conditions are always boolean (and sometimes time based [8][10]). The detective rule principle however can perform compositions as complex as required. This manageability is very suitable for co-operative activities with many different parameters (temporal, value, activity state etc). When detected events validate the algorithm, the rule raises a composed event, eventually broadcasted on a network [15]. We extended this principle and generalised it to compose elements of co-operation [12] [13] [2].

Let us illustrate this with the example according to the condition of the event $E_{SC}$ (an input of the Market Service workgroup). This event is needed to trigger the printing of customer invoices. To optimise this printing, this event will not be produced at each new parcel but when enough national or regional addressed parcels need to be invoiced. In this way, it can be produced by a rule (DR$_1$) from events $E_{CN}$ and $E_{CR}$ produced by the Reception workgroup :

![Figure 10: Elements of co-operation junctions](image)

3.5 Step 5: The formal check

The definition, links and the creation of elements of co-operation are described and enriched at each previous step of the method thanks to a specification language. Before implementing the co-operation, we have to formally validate the results obtained. We have to ensure that we have all we need. The specification language will help us to do that. It will verify that all elements to produce can effectively be produced [16].

3.6 Step 6: Derivation into rules

We can now derive the specification language into ECA and detective rules. This step will provide the implementation. Rules will be integrated into the plat-form. They will get, dispatch and produce information to the concerned entities. They will also manage the dynamic of workgroups.
If we want a good elements of co-operation circulating, we need that the specification language provides us some indications about each information. These indications will help us to fully qualify an element of co-operation: the workgroup or the rule providing the information, workgroup(s), component(s) or rule(s) addressees of the information, the type of information (event, message or data), the name of the information.

3.7 The co-operative plat-form

Our overall project is more than just the proposition of a method to model co-operation into distributed applications. We are primarily concerned with providing an operational method. For this, we use an architecture based on a network extension of the active rules concept [12].

During previous studies [18], we were interested in realising, with task and rendezvous mechanisms, an active layer for a relational ADB. Then, we extended this layer to include network aspects, then added co-operative aspects to allow managing distributed components, these components being organised in workgroups. Once this work was done, we had the basic mechanisms of a network based co-operative architecture [15]. This plat-form has entirely been developed with Java/RMI.

The whole activity is managed locally on each workstation by a set of mechanisms, as is done with CHOOE [13][4]. All information goes across the active system which broadcasts it wherever it has to go. For this, we have to provide mechanisms for local and/or distant management according to the ECA rules principle described previously.

This is the chart of our proposition:

```
Site 1
  CM

Site 2
  CM

Site 3
  CM

Site n
  CM
```

Each site on the network has a communication manager (CM) allowing it to communicate with others sites, and enabling rendezvous to be made with distributed rules. The CM manages all the synchronous (events) and asynchronous (messages) communication. As well, each site has an event manager (EM) and a rule manager (RM).

When an event occurs on a site, it is sent to the EM of this site. If it is recognised (i.e. is on the list of known local events), it will be transmitted to the RM which will make a rendezvous with the relevant rules. If this event has to be broadcasted, the CM receives it and broadcasts it toward the relevant sites. The CM receives distant events and transmits them to the EM which will do the same analysis as previously. Once the rendezvous is taken with a rule (a task), that rule will evaluate the condition, and if it is validated, will execute the action. It should be noticed that each of the components and each rule can raise new events. The CM also receives messages (the management of messages is done with the message passing [17] technique) from the network, and broadcasts them toward the correct components or rules.

Because of the topic of our paper, we will not describe any more the plat-form. A fully detailed description can be found in [2].

4. Conclusion

Problems on which is our topic are frequently encountered with firms, but few works have been done with a such global approach. The method we propose is not very light but the results obtained are interesting as they have been underlined by two important messaging companies.

More generally, this method seeks to solve problems encountered within the larger context of distributed co-operative applications which are likely to emerge from the new current work methods.
Our method approach is also original because it deals with a complete reuse of existing components and proposes to develop a layer “above” them. With this context, elements of co-operation may be seen as an interface between reused components and the co-operative layer insuring the co-operation as we described it.

Figure 12 : The ELKAR Global Approach

The proposed top-down approach enables backtracks to proceed by successive improvement. Furthermore, during the life cycle of the application, it is possible to add components, workgroups, returning to non identified elements and more generally to the whole work done, without erasing all the work carried out to date.

We are currently working on two types of continuations. The first one deals with the first part of the method but focuses more on reverse-engineering aspects [6] [7] [11] and with a more conceptual formalisation with its integration with UML. This formalisation have to identify, solve and manage co-ordination problems. The second issue concerns more the platform. It will be used as a base for works on distributed heterogeneous problems on which we are working on.

5. Bibliographie


