Loose Coupling Based Reference Scheme for Shop Floor-Control System/Production-Equipment Integration

J. Acosta-Cano[*1], F. Sastrón-Báguena[2]

[1]División de Estudios de Posgrado
Instituto Tecnológico de Chihuahua
Chihuahua, México
*jacosta@itchihuahua.edu.mx
[2]División de Ingeniería de Sistemas y Automática (DISAM)
Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid
Madrid, España

ABSTRACT
Coupling shop floor software system (SFS) with the set of production equipment (SPE) becomes a complex task. It involves open and proprietary standards, information and communication technologies among other tools and techniques. Due to market turbulence, either custom solutions or standards based solutions eventually require a considerable effort of adaptation. Loose coupling concept has been identified in the organizational design community as a compensator for organization survival. Its presence reduces organization reaction to environment changes. In this paper the results obtained by the organizational design community are identified, translated and organized to support the SFS-SPE integration problem solution. A classical loose coupling model developed by organizational studies community is abstracted and translated to the area of interest. Key aspects are identified to be used as promoters of SFS-SPE loose coupling and presented in a form of a reference scheme. Furthermore, this reference scheme is proposed here as a basis for the design and implementation of a generic coupling solution or coupling framework, that is included as a loose coupling stage between SFS and SPE. A validation example with various sets of manufacturing equipment, using different physical communication media, controller commands, programming languages and wire protocols is presented, showing an acceptable level of autonomy gained by the SFS.

Keywords: Loose coupled systems, production equipment integration, reconfigurable manufacturing systems.

RESUMEN
Acoplamiento del sistema informático de control de piso de producción (SFS) con el conjunto de equipos de fabricación (SPE) es una tarea compleja. Tal acoplamiento involucra estándares abiertos y propietarios, tecnologías de información y comunicación, entre otras herramientas y técnicas. Debido a la turbulencia de mercados, ya sea soluciones personalizadas o soluciones basadas en estándares eventualmente requieren un esfuerzo considerable de adaptación. El concepto de acoplamiento débil ha sido identificado en la comunidad de diseño organizacional como soporte para la sobrevivencia de la organización. Su presencia reduce la resistencia de la organización a cambios en el ambiente. En este artículo los resultados obtenidos por la comunidad de diseño organizacional son identificados, traducidos y organizados para apoyar en la solución del problema de integración *SFS-SPE*. Un modelo clásico de acoplamiento débil, desarrollado por la comunidad de estudios de diseño organizacional, es resumido y trasladado al área de interés. Los aspectos claves son identificados para utilizarse como promotores del acoplamiento débil entre SFS-SPE, y presentados en forma de esquema de referencia. Así mismo, este esquema de referencia es presentado como base para el diseño e implementación de una solución genérica de acoplamiento o marco de trabajo (*framework*) de acoplamiento, a incluir como etapa de acoplamiento débil entre SFS y SPE. Un ejemplo de validación con varios conjuntos de equipos de fabricación, usando diferentes medios físicos de comunicación, comandos de controlador, lenguajes de programación de equipos y protocolos de comunicación es presentado, mostrando un nivel aceptable de autonomía del SFS.

1. Introduction

Manufacturing systems face highly demanding conditions due mainly to rapid evolution of science and technology. Such evolution facilitates manufacturing of higher quality products at lower prices and shorter delivery times. In order for manufacturing systems to keep track of these

technological advances, shop floor control software must be capable of integrating manufacturing equipment without considerable adaptation effort. Solutions have been proposed in the area of shop floor software system–set of Production Equipment integration (SFS-SPE) focused on meeting some of the existent requirements, but a definition of the minimum requirements is missing. Also, diversity of manufacturing equipment supports reconfigurable manufacturing system implementation, but at the same time it becomes a challenge for SFS developers. In the current manufacturing systems, a set of key characteristics are required as described by [1, 2, 3], including scalability (in terms of production volume) and integrability (ready integration and future introduction of new technologies). Such characteristics lead to consider, during the SFS-SPE integration process, several factors such as physical media of communication, programming languages, nature of equipment and coupling components. For such factors, [4, 5, 6], among others, state that it is practically impossible to achieve a standard or set of standards, that represent the best solution for all cases and aspects of integration widely known and accepted by all manufacturers. Following, a more detail discussion is given for each of these four factors.

Physical media of communication. Various standards concerning the physical environment supports the integration of equipment, some of them are open standards (RS-232, IEEE-485, USB, IEEE488, FireWire and Ethernet, etc.) and others are proprietary standards (DH-485, Modbus, SINEC, among others). Both standard types allow the use of the same physical medium of communication on different manufacturing equipment controllers. On the other hand, the diversity of standards implies the possibility that in a given application, the set of manufacturing equipment may involve different types of standardized media that need to use different code segments depending on the used standard. Some proposals allow homogenization of the instruction set for accessing different physical media of communication for a given application type, such as:1) VISA [7] for instrument control, 2) IEEE / NEMI PR 1533-1998 for open architecture controllers and 3) Equipment communication standards such as OPC, MAP and SEMI [8], for integration of industrial equipment. However, the need for different code segments when there are different types of applications still remains. Some proposals consider the use of a single standard to provide the same code to be used in various application types, solution which is considered impossible by some authors [9]. One of the objectives of this paper is to present a nonstandard dependent solution, easing incorporation of required code of a particular standard into the SFS at execution time.

Control programming languages. SFS-SPE integration involves SFS adaptation to the language used by the driver of each manufacturing equipment (e.g., languages for robots programming such as Inform II, Darl, Melfa and KRL among others) as well as adaptation to the set of commands instructing the driver from the SFS (example, load, abort, and reset). Each time new manufacturing equipment must be integrated (replaced or added) to the system, the SFS faces the problem to adapt to the control language of the new equipment. Therefore, the SFS must be capable of handling a wide variety of languages that depend on the type and manufacturer of the equipment controller. In this regard, some efforts have been reported to reduce this dependence, based on the idea of a universal language, as it is the case of Robot Script or Robot C, proclaimed as universal languages for robots programming [10, 11]. Other standards such as ISO 6983 or EIA RS 274 and ISO 14649 (STEP-NC) are aimed to programming CNC machines and some others for PLC programming such IEC 61131-3 standard. It is important to notice that each proposal is directed to a particular type or nature of equipment (robots, CNC and PLC's). Moreover, even for equipment of the same nature different language requirements may exist, [12]. As an example, for robot-like equipment there is a difference in instruction regarding their application (e.g., arc welding, material handling or painting). Furthermore, different robot programming techniques are available, including online programming, off line programming and robot programming using augmented reality, [13]. It is common, to find in the same manufacturing system, different equipment nature which lead the SFS to handle different control languages. Although standardization contributes to reduce the variety of proprietary solutions, a need to solve the problem of managing a set of control languages remains, because these languages are linked not only to the variety of manufacturers, but also to the variety of programmable equipment natures. Therefore, this

trend to language universality has not proved to be an effective way to solve the control language problem. In this paper we propose a different approach to solve this problem, without the need of a universal language.

Manufacturing equipment nature. Another relevant aspect, to manage SFS-SPE integration, arises from the diverse nature of equipment existing on the particular production floor. This is a first importance aspect, not only for the problem regarding control languages mentioned in the previous section, but also for the requirement of the SFS to track the status of the part, sub assembly or product during their manufacturing process (e.g., polished piece, welded sub-assemblies,...). Lichtveld and Van der Zon, [14], call this as the double dimension of the system (physical and software dimensions). This requirement implies that SFS records (e.g., in the database) the effect or transformation of the part operated by the manufacturing equipment. This effect depends upon the operation type, which in turn depends on the type or nature of the equipment used For example, the effect of a drill operation on a piece must be recorded as drilled part and the welder operation on an assembly must be recorded as a welded assembly. Moreover, diversity of manufacturing equipment to be controlled by the SFS is commonly managed with implementations dependent on equipment type, [15, 16]. The large amount of equipment types or nature reported in the literature, [17, 18, 19], makes the two-dimensional problem be virtually impossible to handle through code implemented specifically for each type of processes or equipment. Therefore, it is necessary to identify a level of abstraction of manufacturing equipment resulting in a manageable variety of equipment by the SFS. Some abstractions of manufacturing resources (materials, equipment, time, energy) reported in the literature are too generic for this purpose [20, 21, 22, 23, 24, 25 and 26], among others. Other abstractions focus at manufacturing equipment in a reduced domain, for example flexible manufacturing cells by [27, 28, 29 and 30], among others. An interesting classification for such purposes is the one proposed by [31], regarding the possible operations to be performed by manufacturing equipment, which is adopted in the ISO technical report 10314-1, [32], and used by other authors such as [33], although there are no proposed solutions based on it, reported for the problem of two dimensions. In this paper, we use a variant of this classification to achieve a manageable number of nature types of manufacturing equipment applicable to a broad class of manufacturing systems.

Coupling components. The SFS-SPE integration process involves different operating environments. Such environments are coupled based on software elements. That is the case of solutions based on the concept of drivers [34, 35] or envelope elements [16]. In this way, each one of the manufacturing equipment is seen as a software element or object to be operated from the SFS environment. Such elements (intermediaries) perform a dual function consisting of presenting a homogeneous view of the manufacturing equipment to the SFS and handling internally the peculiarities of the equipment functionality. Under this reasoning, the object-oriented paradigm has been used for a long time as a basis for proposals to solve the integration problem, where the advantages of the paradigm have been proclaimed, [3, 29, 36, 37, 38, 39]. Much of the efforts have focused on meeting requirements imposed by the operating environment of distributed objects, [15], where there are software solutions based on component technology such as CORBA, DCOM, .NET, Java or message queues. These technologies have supported the development of effective environments of operation, so that much of the system could be built using one of these technologies, but the problem arises when one system requires more than one of these technologies (e.g., MSMQ and JAVA), given that this type of development requires considerable effort, [5, 40]. For those situations, web service-oriented solutions are a better approach, because they make use of the stack of Internet standards (3WC), widespread and accepted by various computer platforms, [41]. Other approaches to SFS-SPE integration solution are presented using discovery (detection) and automatic integration of equipment, employing service oriented architecture (SOA) such as universal plug and play architecture, [42, 43] and solutions reported in the area of pervasive or ubiquitous computing, [44].

The solutions described in the previous paragraph are given from the point of view that in [45] called

wire protocol. Such solutions take into account mainly the technological component of the coupling, without taking into account other coupling aspects. In addition to the technological aspect, in communication between objects, there are other coupling components to be identified and defined, such as the definition of types of processes and equipment, as well as description of the processes requested by the SFS. A solution that does not include these other components stiffens the coupling. Another objective of this paper is to present a solution that includes not only the definition of the coupling technological aspects but also aspects such as types of processes, equipment and description of the process to be performed by the manufacturing equipment.

In this paper a reference scheme is proposed as a way to analyze and solve problems that arise in the SFS-SPE coupling, such as adaptation for using a standard or a standard set of physical media of communication, equipment control languages or computer platforms, as well as dealing with diversity of equipment nature or manufacturing processes. The scheme presents fundamental loosely coupled concepts in an organized manner adapted to be used for designing, implementing and validating the SFS-SPE coupling subsystem. Such structure makes possible managing other components of the coupling, in addition to the technological component, without explicitly considering detail description of the equipment at design time, enabling the SFS to remain unchanged when the equipment set is modified. The approach is based on loose coupling (LC) concept as abstracted by [46], for application in the organizational design area. This Orton and Weick´s model is translated and enriched in order to make it useful for supporting the SFS-SPE coupling problem solution. The proposed reference scheme includes basic guidelines for requirement identification, design, implementation and validation of the system supporting coupling solution development and scheme incorporation of partial independent solutions. To validate such reference scheme, an SFS-SPE coupling framework was designed, implemented and applied to a set of experimentation manufacturing cells.

2. Concepts abstraction of SFS-SPE integration

Software system controls manufacturing equipment through parameterized commands sent to the corresponding equipment controller. An example is shown in Figure 1 that includes the relevant elements for communication between software system and manufacturing equipment.

Programming languages, names of functions, commands, parameters and syntax depend largely on manufacturer and type of both the equipment controller and the physical media of communication. A compact form of expressing this dependency is shown in (1), which expresses that sentence (S), for sending orders from SFS, depends on: a). Function (F1) of the physical media to be used and its manufacturer (Pm), b). Command set (C) to be executed by the system controller, depending on the type and manufacturer of equipment (Te and ME respectively) and c). The set of arguments (A) associated with the corresponding command (C).

S {F1 (Pm), C (Te, Me), A(C)} (1)

Where:

S = Communication sentence.

Pm= Physical media =| Ethernet | RS232 | RS485 | USB | DH485 | GPIB |…

F1 = Function of Physical media = | SendData | WritePort | DataArrival|…

Te= Type of equipment = | Robot | Lathe | Oven | Press | Milling | Drill |…

Me=Manufacturer of equipment =| Mitsubishi | Motoman | Bridgeport | Fagor | …

C = Command = | Load | Reset | Mov | …

A = Arguments of command =| Prog1 | P1 | …

Another aspect to be considered in SFS-SPE integration is registering the effect on the manufacturing physical system in the SFS, due to

command execution of the manufacturing equipment (software dimension of the system). This effect is mainly reflected in the state of the equipment and the state of the piece operated by the equipment. Equipment and piece states will depend on the executed command and the operation performed on the piece, respectively. An expression of it in a compact form is shown in (2).

Ef {ES (E, C), PS (P, O)} (2)

Where:

Ef = Effect of operation.

E = Equipment set = | Eq1 | Eq2 | Eq3 | …

C=Command set= | Run | Pause | Abort |…

ES = Set of equipment states = | Idle | Loaded | Operating | Maintenance | …

P = Pieces set = | p1 | p2 | p3 | …

O= Operations set = | Drill | Groove | Polish | Weld | Temper | Move | Load |…

PS= Set of piece states = | Drilled | Grooved | Polished | Welded | Tempered | Moved | Loaded | …

Expressions (1) and (2) abstract the main factors and its interrelation during the SFS-SPE integration process. The wide range of possible values, to be taken by these elements, makes impossible to include them explicitly in their entirety in the software system during design time. In this paper, supported by the LC concept, a set of coupling concepts is identified and proposed to be used during development of software system, to support the coupling / uncoupling of new equipment types using different languages, operating standards and computer platforms.

3. Loose coupling modeling

Coupling can be seen as the grouping of two systems, parts or devices, so that their combined operation will produce the desirable results, [47]. The concept of loose coupling, originally described in the area of organizational systems by [48], has been of application in different areas; although since its origin, it has been regarded as an underspecified concept, [40, 46, 48, 49]. Indeed,

Figure1. Shop floor software system (SFS) commanding a set of production equipment (SPE).

Orton and Weick, [46], encourage taking advantage of such already existent loose coupling, instead of moving to hard coupling. In this section, a translation to the SFS-SPE integration area is proposed for the LC model, given in [50] and its reconceptualization made in [46]. In addition, such a model is enriched here, introducing coupling components to support its interpretation and application to the solution of the SFS-SPE coupling problem.

Orton and Weick, [46], abstracted the LC concept from the point of view of the organizational design area. For the SFS-SPE integration problem, four relevant aspects of the model are described next, along with an interpretation of them from the point of view of this particular area. A summary view of this interpretation is shown in Table 1. A fifth aspect of the original model, named *direct effect* was left out of the following discussion due to space reasons.

3.1 Environmental elements

For LC existence in the organization, the model defines as environment conditions, the relative absence of regulations and also various media could lead to the same end, Table 1. These conditions exist in SFS-SPE integration environment of the software system, because there is an ample variety of possible values that integration elements can take, (e.g., some equipment with different characteristics is available in the equipment market to perform a given process) as well as absence of effective regulations that homogenize their integration to the software system. These environment conditions of the software system lead into the inability to identify a priori the characteristics of each one of the equipment to be integrated, which Orton and Weick defined as cause indeterminacy. Such indeterminacy is dealt with the concept of LC compensators.

3.2 Types of loose coupling

Coupling between certain types of organization elements are also analyzed and established by Orton and Weick as causes of LC existence. These are identified as coupling between elements in different hierarchical levels speaking different languages and coupling between intentions and actions, Table 1. In the case of SFS-SPE, there are levels with this type of coupling, because software system and equipment levels use different languages. In the second type of coupling (intentions-actions), the model states that LC is generated when actors have more room for self-determination, that is, the intention is achieving a particular purpose and the action will depend on available resources. In our case, it means that, in order for LC to exist in SFS-SPE, software system should release the task order using a high level of agnosticism about the nature, type and model of available manufacture equipment.

3.3 Loose coupling compensators

Orton and Weick suggest operating organization under LC conditions, instead of being considered them as unsatisfactory conditions that should be reversed. They propose to avoid strategies leading to a hard coupling that neutralize the dialectical sense (autonomy-communication) of the concept. Towards that end they propose, in their LC model, a series of so-called compensatory strategies named enhanced leadership, focus attention and shared values, Table 1. These strategies support organization's operation, keeping LC conditions as no obvious sources of order that administrators can use to influence dispersed organizations, [46]. A discussion of such compensatory strategies is given in the following.

3.3.1 Enhanced leadership compensator

Orton and Weick remark that leadership has an important role managing a LC organization. Although some theorists view management of LC systems as a problem and call for stronger leadership but loose coupling calls for subtle leadership, [46]. Enhanced leadership can simultaneously provide centralized direction and coordination, while recognizing the value of increased discretion.

Orton and Weick point of view could be perceived as LC management by enhanced leadership, implying sensitivity to diverse systems components (discretion) and ability to control organization through delegation. In SFS-SPE coupling, enhanced leadership can be translated into software abstraction of each one of the systems components as delimited entities and delegating implementation of function particularities to each one of them.

Orton and Weick Model		Translation to SFS-SPE.
Model voices	Coupling elements.	
Environment of a loose coupled system	Several media lead to the same end.	There are alternative pathways to achieve the same end; wide variety of equipment capable of performing the same process.
	Relative absence of regulations.	Existence and acceptance of standards from different points of view such as equipment nature, manufacturer or equipment model.
Types of loose coupling	Coupling between hierarchical levels.	Coupling between software system and equipment controller.
	Coupling between intentions and actions.	Coupling between the intention of performing certain process and both the equipment available for it and freedom to make decisions to accomplish it.
Compensators	Enhanced leadership.	Commanding discretely and delegating implementation details to the equipment.
	Focus attention.	Focusing on essentials for control of equipment operation.
	Shared values.	Preferably, coupling using general-purpose technologies.
Consequences of loose coupling	Persistence	Reduction in responsiveness of the system to changes in its environment.
	Buffering.	Blocking and preventing propagations of changes.
	Adaptability:	Assimilation and adaptation to changes.
	Experimentation	Adaptability supporting solution identification.
	Collective judgment	Using general purpose standards instead of standards of a particular domain. (Shared values).
	Discrepancy	Taking into account situations not considered at design time.
	Satisfaction.	Ease to fulfill a need (less connection between elements).
	Effectiveness.	Achievement of better solutions through experimentation and innovation.

Table 1. Relevant aspects of the Orton and Weick model translated into SFS-SPE coupling area.

3.3.2 Focus attention compensator

Orton and Weick observed that individuals can compensate for loose coupling by carefully selecting both targets and controlling resources. Managers can build ongoing behavior on subordinates focusing only on controllable and essential behaviors, providing the freedom for subordinates to adapt the behavior to local needs. In SFS-SPE coupling, focus attention compensator can be translated into only essentials for equipment operation control, enabling the software system to carry out its coordination role and letting each one of the equipment free to perform its particular function.

3.3.3 Shared values compensator

For Orton and Weick, this form of compensation is especially crucial because it often constitutes the sole remaining basis that holds together a loosely coupled system. Cultural preferences are highlighted through the shared values compensator as a means to achieve agreements in an environment of loose coupling. It is seen as the ultimate source of order remaining when there is uncertainty between means and ends. For SFS-SPE integration, cultural preferences or shared values compensator is equivalent to the recommendation made by other authors on using general-purpose technologies widely disseminated and accepted, [5, 49, 51], instead of using technologies designed to couple specific system elements defined by Schmid, [52], as solutions for a specific domain (e.g., semiconductor systems).

3.4 Consequences of LC in the organization

Orton and Weick identified several effects on organizational performance reported when operating under loose coupling. These effects could be translated into the SFS-SPE integration as follows.

*Persistenc*e. Loose coupling reduces responsiveness of organization to changes in its environment. LC systems are less conductive to system wide changes. Thus, the organization tends to remain invariant for a long time against changes in their environment, but still performing well. In this case SFS should remain invariable against changes in equipment types, languages, operating standards and computer platforms.

Buffering. Loose coupling blocks and prevents propagations of changes. In this case, modifications in the SPE should not affect the SFS.

Adaptability. Unlike persistence and buffering, that neutralize the impact of change, adaptability means assimilation and adaptation to change. A type of adaptability is useful to cope with the causal indeterminacy (unaware of scenarios to deal with), through actions to be taken for untangling causality. Orton and Weick report three types of adaptability. Next, an interpretation of each one of them is presented regarding the particular case of SFS-SPE coupling.

a) Experimentation or exploration. Having this kind of adaptability, the system favors the coupling of manufacturing equipment to explore better solutions. It is applicable to complex problems where identification of the solution equipment is not a trivial task.

b) Collective judgment. Where a disagreement exists between the means to achieve the ends, Orton and Weick state that the system still coheres if there is agreement on common preferences (shared values). For SFS-SPE loose coupling it is encouraged the use of general purpose standards instead of particular domain standards, therefore promoting adaptability of the application through multiple perspectives, taken from other environments collectively accepted.

c) Discrepancy (Preservation of discrepancy). Discrepancy favors adaptability of system to complex problems, because the minority influences help enhance cognitive effort of the majority. For SFS, the system is proposed to account for situations not considered at design time (which could be called discrepancy). The system takes into account this kind of adaptability considering the inability to include all possible situations.

Satisfaction or fulfillment of a need. Loose coupling reduces conflicts between elements of the SFS-SPE because of less connection between elements (less chance of conflict), offering ease of deviation in certain aspects without affecting other elements of the system. Setting of objectives is easier on localized levels.

Effectiveness. In the area of organizational design, some authors associates organizational effectiveness with return of investment and market share. Effectiveness, in computer control system integration, may be associated to the capability of the system to experiment and innovate, which brings the achievement of greater effectiveness through better solutions.

4. Development of the proposed reference scheme

Interpretation of the model, as shown in Table 1, may be twofold; it shows the possibility of LC existence in SFS-SPE coupling and also the fact that it may be seen as a set of concepts (scheme)

based on LC compensators, [46], that could help to operate the system under an autonomy-communication conditions. In order to apply such a model, a coupling component set with three main subsets is here introduced; role, task and technology. Role is defined here as an abstraction of the possible tasks to be performed. Task is defined as the coupling component that describes the details of the process to be made. Technology is the means to communicate the task to be executed by the equipment. These components are implicit in the integration elements of expressions (1) and (2). They could be explicitly used in a reference scheme where a combination with the loose coupling compensators of Table 1, supports the analysis and design of SFS-SPE integration elements as shown in Table 2, where the adapted model with the set of coupling components and their relationship with LC is presented. Following, a detailed description of such adapted model along with its relation with integration elements of (1) and (2) is given.

4.1 Role (Te, O and Ef)

Equipment type abstracts the set of tasks the equipment can perform. For instance, milling type equipment plays the role of milling, which means the equipment has the ability to perform task or operations such as grooving, drilling, etc. Therefore, equipment type (Te), operations (O) and operations effects (Ef) are associated with role component.

4.1.1 Role-shared values

Applying shared values compensator, to role component, implies identification of terms widely known and accepted for representing types of tasks that permit a classification according to their role playing in the shop floor. Classification in terms such as milling or turning that are widely known and accepted, allows representing some processes independently of particularities, but their representativeness is not effective for the software system, because such classification would require a very large amount of terms. Other proposals, such as using the functional entity and its derivations, proposed by AMICE (CIMOSA) have not being accepted because of its complexity, [53 and 54]. A manageable classification for manufacturing equipment, in terms widely representative, can be reached through the compensators focus attention and enhanced leadership, as describe in the following subsection.

	Shared values.	Focused attention.	Enhanced leadership.	
			Discretion	Delegation
Role (Te, O, E)	Encouragement of using widely available and accepted definitions of concepts and technology.	Identification and monitoring of equipment operation effect on the piece, based on equipment role.	Four types of equipment: Processor Manipulator. Transport. Storage/Retrieval	Set of delegated objects taking care of role aspects.
Task (C, A, PS,ES)		Focused on main stages of the process (i.e., start and end of operation). Part identification and equipment operating it.	Document encapsulation of task. Task specification at document name and path level.	Equipment task communication management by an envelope object. Proxy objects delegated to update piece and equipment state.
Technology (PM, F1)		Location of actors (objects) in both coupling ends (network or queue address). Location of task files.	Dynamic code incorporation of delegated object classes. (selection at execution time).	Specific physical media and syntax management of equipment controller through envelope and driver delegated objects.

Table 2. Summary of reference scheme proposed for integration of SFS-SPE based on coupling components and loose coupling compensators.

4.1.2 Role-focus attention

In the Orton and Weick model, focus attention compensator states that loose coupling is favored through reduction of coupling regulations and focusing on essential details, [46]. From the role component point of view, this can be translated as equipment management by software system using an abstracted view of the equipment. Following the focus attention establishment of using only essential details, the equipment should be abstracted focusing on the operation effect caused on the work piece, see Table 2. The number of effects to be managed by the software system, that is, the set Ef will depend on the size of the sets Te and O in expressions (1) and (2), associated to the role component as described in Subsection 4.1. A way to carry out set Ef reduction, based on the enhanced leadership compensator, is explained as follows.

4.1.3 Role-enhanced leadership (Discretion/ Delegation)

Types of equipment (Te) must be defined from the perspective of their use by the software system, [15, 16]. The type of processes (O) that the equipment operates can be used as a guide for equipment roles identification. However, the type of processes reported in literature is too ample, [17, 18, 19].

In order to obtain a manageable number of roles (types of equipment), here it is proposed a direct one-to-one association between generic operations and equipment types. It is here, where the discretion compensator takes place and suggests looking for a classification of manufacturing operations (O) without excess of details (discrete classification). The classifications made by some authors, [31, 33], and the technical report ISO TR-10314-1, [32], for manufacturing operations (O) were considered here as the basis for a tractable classification of four types of equipment (Te): transportation, manipulation, processing and storage/retrieving, see Table 2. In this way, each piece of equipment in shop floor could be represented in the software system by one of these four roles, reducing drastically the coupling details to be managed, Figure 2.

Moreover, role component analysis under Orton and Weick's delegation compensator leads to the conclusion that software system should avoid taking care of specific role details when commanding manufacturing equipment. Such details should be delegated to intermediate (external) coupling elements (e.g., CEquipment subclasses in Figure 2). Thus, the manufacturing equipment is to be abstracted as generic equipment (proxy) object which [14] describe as generic components.

The elements for Te, O and consequently Ef subsets could be deduced in a one to one correspondence to the four elements listed above, as shown next.

Te= | Processor | Manipulator | Transporter |

| Storage-retriever | (3)

O = |Processing | Manipulation | |Transportation | Storage-Retrieval | (4)

Ef =| Processed | Manipulated | Transported | | Stored-Retrieved | (5)

Therefore, control software system delegates updating properties of the software piece object to the equipment software object limiting its effect to one of the four types of equipment roles previously identified (storage/retriever, transport, process and manipulation). Hence, not only the software system focuses to the main (essential) coupling details, but also it keeps the one-to one-correspondence between elements of the physical system and elements of the software system, presented in the object-oriented paradigm, [36], as well as in which [14] calls the double dimension of the components (physical and software), Figure 3.

4.2 Task (C, A, ES, PS)

The details of the operation to be performed are communicated to the equipment controller through commands and command arguments. For instance, commanding a robot, to perform a movement to position P1, may be sending the MOV command and P1 as command argument to robot controller. Therefore, given that commands (C) and their arguments (A), in expressions (1) and (2), describe the details of operation, these elements may be associated with task component. Command execution causes an effect in both equipment and part states (ES and PS), in order that, these integration elements (states and effects) are also associated to task component.

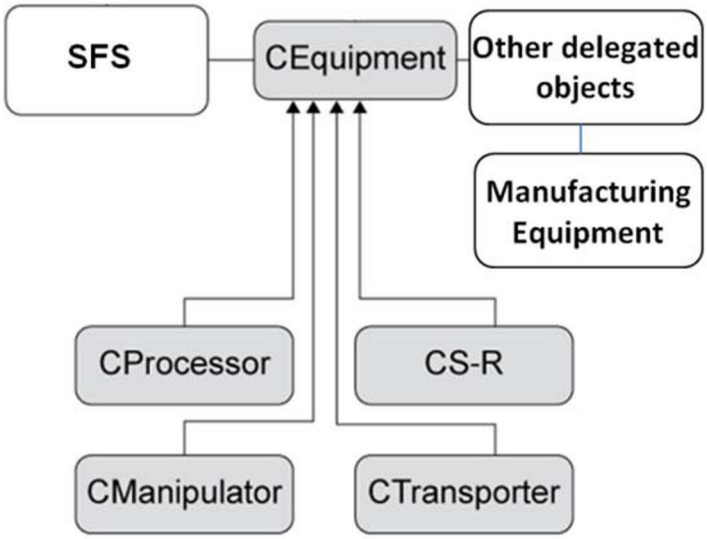

Figure 2. Role of manufacturing equipment based on delegation/discretion.

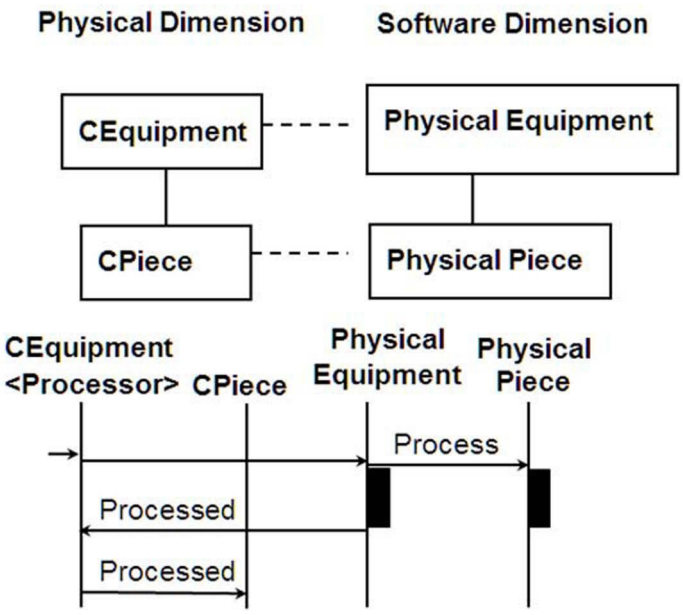

Figure 3. Management of the system double dimension.

4.2.1 Task-shared values

This compensator, applied to task component, implies task description in common terms, this means, a description in a widely known and accepted way. Although, several widely known standards (languages) allow homogenizing task description, such standards depend on task or process type, and sometimes, for some task types, a sufficiently dominant standard is not available. On the other hand, task description, with its particularities for a given case, is needed. Also, associated to task component, not only its description for dispatching order execution in given equipment is needed, but also for tracking its achievement and software elements updating. Next, a proposed solution is described, based on the LC compensators focus attention y enhanced leadership.

4.2.2 Task-focus attention

The relative absence of regulations, prescribed by focus attention, should also be considered for software system task management. The manner the software system manages the task component influences the coupling strength. Thus, an acceptable level of detail should be used by software system during the process of task component management. The focus attention compensator suggests focusing only on essential details. It can be translated to focusing on reducing the abstraction level used to establish the parameters of expressions (1) and (2), associated with the task element (C, A, PS, ES), describing the job to be done. It is proposed as essential details just those required for task identification, keeping software system from being aware of task implementation details. Nevertheless, software system has to manage enough details required for monitoring task execution process, see Table 2. A proposal with the adequate level of detail is described as follows based on the enhanced leadership compensator.

4.2.3 Task-enhanced leadership (Discretion / Delegation)

This compensator supports loose coupling through delegation and discretion for task description. Discretion concept in the sense of enhanced leadership can be understood as software system being discrete (moderate) in dispatching task execution order. Arguments (A) in expressions (1) and (2), represent details such as speed or point coordinates, to be used by equipment controller at execution time. This information needs not to be explicitly specified by software system, but it can be stored in a file (e.g., CNC program, robot program or operation instructions document for manual equipment operator). Therefore, it is proposed that software system communicates commands and arguments (implementation details) discretely transmitting the full name of this file, see Table 2. However, not all commands (C) and arguments (A), in expressions (1) and (2), regarding task component, may be included in such a file. Some equipment controller commands, highly depending of equipment model or manufacturer, such as program load, run or pause, still have to be explicitly managed by the software system. In order to alleviate software system of taking care of these controller particularities, this command management process can be delegated (delegation concept compensator) to a software object that would act as an equipment envelope object.

Parameters PS (piece state parameter set) and ES (equipment state parameter set), in expressions (1) and (2), are being related to task component, because piece or equipment states depend directly on task execution. In this sense, it is proposed that software system focuses on the state updating process that is directly associated to the starting and ending events of task execution. Regarding equipment states, a set can be discretely established as proposed by ISA in S88 standard (Idle, Running, Complete, Holding, Held, Restarting, Pausing, Paused, Stopping, Stopped, Aborting and Aborted). On the other hand, piece states set could be managed according to the four possible types of operation discussed in Subsection 4.1.3. Therefore, both the piece and the equipment states can be discretely managed in the software system. Furthermore, according to software system commands and equipment manufacturing feedback, software system can delegate updating both equipment state and piece state to software objects, representing the manufacturing equipment, proposed in Subsection 4.1.3, see Table 2. Thus, focusing on essential details supported by discretion and delegation concepts, software system would be independent of details in task component parameters A, C, PS and ES, being managed in design time using reduced sets, as follows

C = File name and location (6)

A = File name and location (7)

PS = | Processed | Loaded/Unloaded | |Transported | Supplied-Removed | (8)

ES = | Idle | Running | Complete | Holding | |Held| |Restarting | |Pausing | Paused | |Stopping | |Stopped | Aborting | |Aborted | (9)

4.3 Technology (PM, F1).

Communication tasks to equipment controller is a process that involves various technologies levels such as physical media and their respective functions, including software technology or web services protocols. The function of this component in the coupling, specifically in a plug and play SOA environment, is clearly described in [43] as an extra software layer to integrate plug and play devices. So that, integration elements in expressions (1) and (2), physical media (PM) and functions required for task communication (F1) are associated to technology component.

4.3.1 Technology-shared values

This compensator encourages the use of widely known and accepted media for communicating tasks to the manufacturing equipment for their execution and its tracking. An ample variety of computer platforms and communications protocol exists, either at network computer level or at controller equipment communication level, making impractical including all of them during application design time. On the other hand, limiting the application to one subset of them it is quite arbitrary. Object-oriented paradigm has been used to make transparent the technology used and supported in wire protocols, such as web services, SOA and plug and play, but such solutions remain hooked to the specified wire protocol. A solution allowing the use of different wire protocols is described next based on the compensators focus attention and enhanced leadership.

4.3.2 Technology-focus Attention

Software system must be freed from managing technology component details others than essential ones, as it is suggested by focus attention compensator. As a starting point, the minimum or essential detail level definition process could be the full location identification of the object to be communicated, (see Table 2). This is because in the communication process, at least the transmitter object, should know where the receiver is. Indeed, it does not require explicitly to know who the receptor object is, furthermore, technical protocols should be considered as a secondary level of details, and can be delegated using the enhanced leadership compensator described in the following subsection.

4.3.3 Technology-enhanced leadership (Delegation/Discretion)

Discretion (Enhanced leadership) compensator for the case of the technology coupling component could be interpreted as software system agnosticism about implementation details of communication protocols. These details should be managed with reserve and circumspection (discretely), that is, without excessive details. The delegation compensator induces the use of an external set of delegate elements (classes) between software system and equipment controller. A proposed structure is shown in Figure 4, where these considerations are taking into account. The software control system sends a generic command (e.g., operate), using its own computer platform, to a proxy object, which manages the mentioned wire protocol. Therefore, the wire protocol is completely transparent to the software control system. Receiving a command in the specified wire protocol from the proxy object, the envelope object takes care of a specific driver controller, making transparent the specific driver controller for the proxy object, which can be in a different computer platform. The driver object, which is in the same platform as the envelope object, manages the specific controller commands, liberating the envelope object from this task.

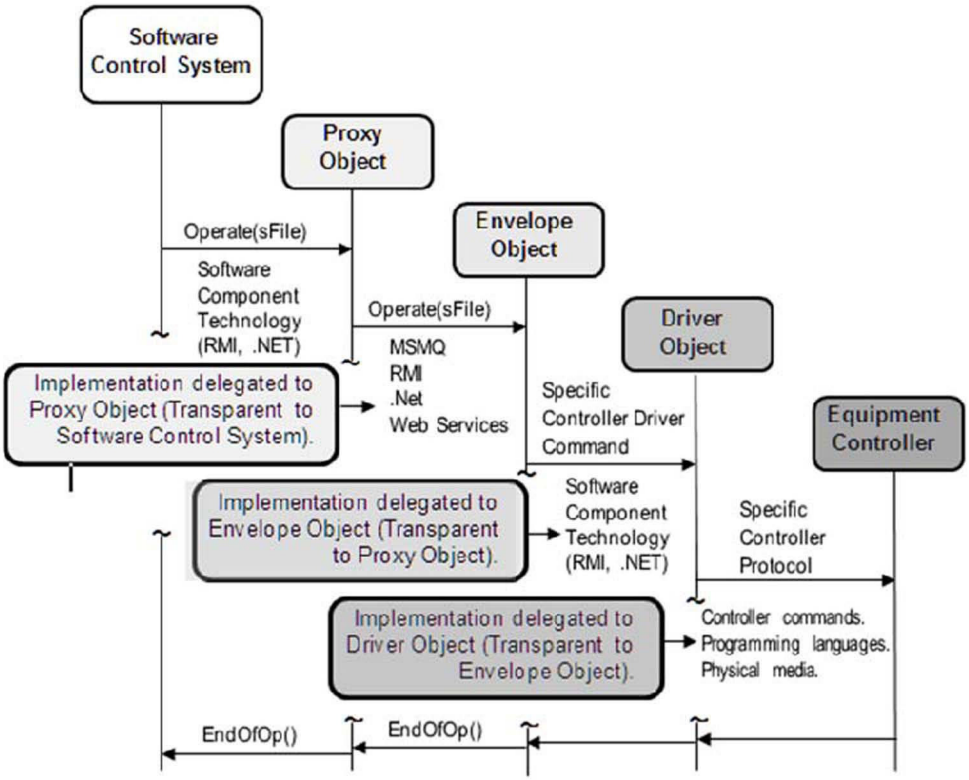

Figure 4. Technology delegation/discretion using a structure of external set of delegate elements.

Therefore, at the SFS end of the structure transmission functions are implemented at generic level, according to delegation compensator. These functions are gradually specialized by three basic levels throughout the objects of the structure, in order to achieve the specific communication features required by the equipment controller. The first level consists of the SFS using its own computer platform for task communication requirements. Second level supports communication between objects running on different computer platforms. At the third level, for each piece of the equipment, particular details are handled such as commands, programming languages and physical media, specific for equipment controller.

Proxy object, as introduced in the role-component analysis (see Subsection 4.1.3), may be used to support the first level of the structure. Thus, SFS communicates with proxy objects, through a common platform. These objects in turn take care (delegation compensator) of transmission details required to communicate the task to the next level of the structure. At a second level of the structure, the computer platform to be used should be as required by the equipment controller. So that, it is important to point out that the communication between first and second levels may involve two different computer platforms. In addition, the second level receives the task commanded by SFS and translates it to controller specific commands. The envelope type object was introduced in the second level of the structure of Figure 4, in order to perform remote communication and command specialization. Following, a brief description of the way the structure objects are incorporated to the system and how they implement their function, is given.

Given that the communication protocol, between SFS and proxy objects is independent of the equipment controller, they should operate on the same computer platform, in order to get a straightforward communication between them using software component technology of the particular platform,

resulting independent of physical media (Pm) and specific functions (F1) of equipment controller. This situation, together with the solution regarding role and task components management, let the communication process between first and second level be reduced to functions operate for sending the name of the file, where the task to be performed is given, and receiving end of operation (EndOfOp), by the proxy object, as shown in Figure 4.

Envelope object, in second level, supports transparency to proxy object regarding controller particularities. It also takes care of the communication protocol requirements imposed by the remote condition of equipment controller. It allows proxy objects focus on task transmission and register state changes of equipment and piece software objects as proposed in Subsection 4.2.3 (task component).

As mentioned before, sometimes controller characteristics may force proxy and envelope objects to be implemented in different computer platform. Communicating objects using software component technology implemented in different computer platform (i.e., CORBA, COM, Java) is a cumbersome task [5, 40], but web technology has been reported as a more effective solution in such cases, [41]. Although web services provides flexibility to the system, in cases where only one computer platform is used, software component technology may be taken into consideration for communicating proxy and envelope objects.

Figure 5. Summary of loose coupling reference scheme.

Finally, an ample variety of open and proprietary standards for communication physical media, controller commands sets and programming languages are found at controller equipment level. These are highly dependent of equipment model and manufacturer. The purpose of the third level of the structure is to take care of these controller particularities. As mentioned before, a driver object contained by envelope object may handle these particular characteristics, making them transparent to proxy-envelope objects communication. It is important to remark that envelope and driver objects operate in the same computer platform, making a straightforward process for integrating the specific driver object to the system even at runtime. Therefore, from the SFS point of view, the integration elements regarding technology component may be reduced as follows

$$Pm = \Phi \text{ (Empty set)} \quad (10)$$

$$F1 = | \text{Send} | \text{Receive} | \quad (11)$$

A summary of the propose reference scheme is shown in Figure 5, considering the three identified coupling components: role, task and technology.

5. A coupling framework design based on the proposed reference scheme

Considering the proposed reference scheme, the dialectical problem (coupling/decoupling) could be included into the SFS or removed out of it in a coupling framework. In Figure 6 a coupling framework is shown out of SFS.

Next, such solution, based on a coupling framework outside SFS is developed. A framework structure based on classes is shown in Figure 7. The four different communication stages are considered. The first coupling stage communicates commands from the SFS to CEquipment of the framework, which is in the same computer platform as the SFS. SFS uses software component technology for the instance creation of proxy objects (subclasses of CEquipment: CProcessor, CTransporter, CManipulator and CS-R), as shown in Figure 8. Such proxy object types may be specified at execution time. SFS-Proxy objects communication is simplified, because local area network protocols can be used.

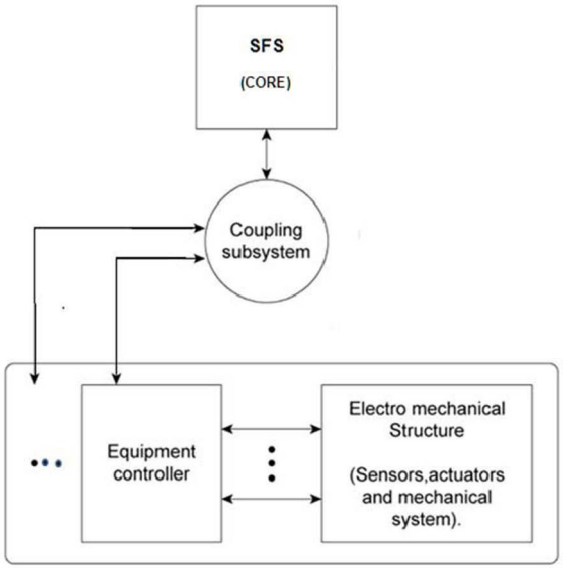

Figure 6. Coupling/Decoupling solution based on coupling framework outside the SFS.

In the second stage, Figure 9, the command is transmitted to CEnvelope which could be in a different computer platform. Therefore, communication between CEquipment and CEnvelope should match the wire protocol imposed by the equipment controller. Hence, CEquipment and CEnvelope communication protocol only changes if computer platform of a new equipment controller changes, but the rest of their attributes and methods remain.

The third stage is shown in Figure 10. Using the same computer platform, CEnvelope transmits commands to CDriver. As mentioned before, CEnvelope is to be modified if wire protocol changes. CDriver takes care of the specific equipment controller particularities. Therefore, a particular CDriver class implementation is required for commanding each specific equipment controller. The object implementation is identified by its type name, given at execution time, and software component technology is proposed for object instancing.

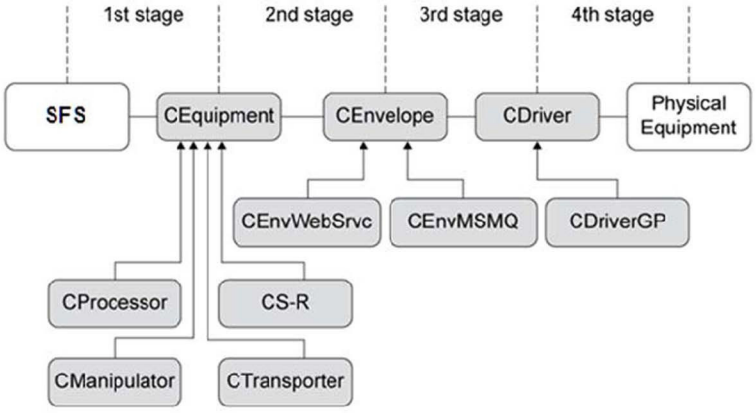

Figure 7. Class diagram of a coupling framework.

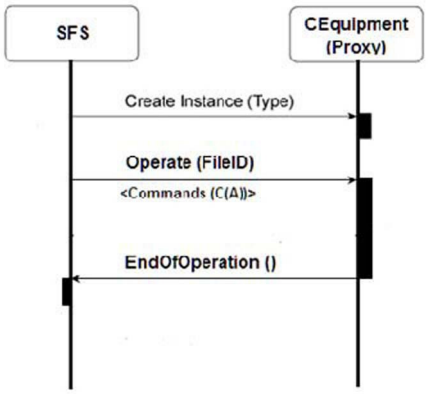

Figure 8. Sequence diagram of coupling first stage.

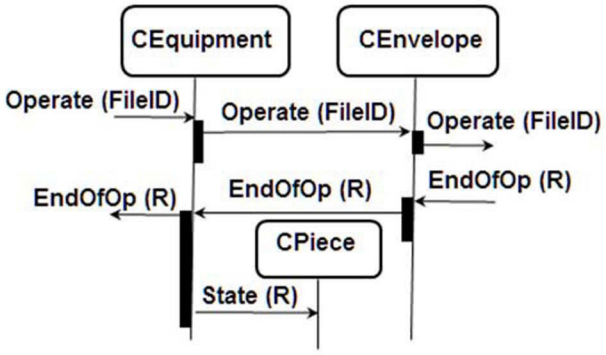

Figure 9. Sequence diagram of coupling second stage.

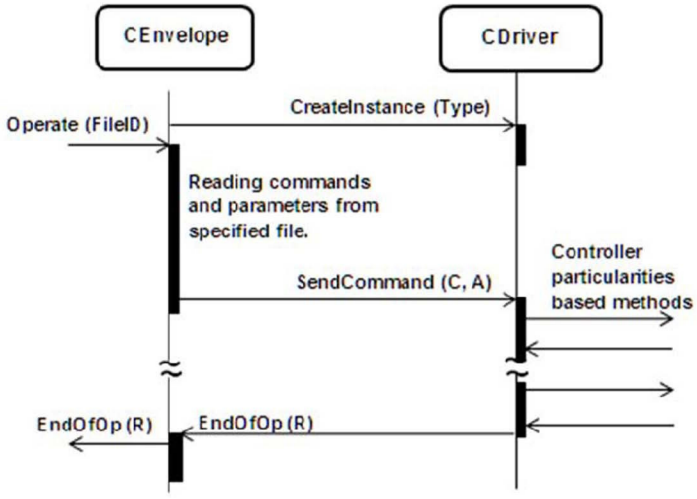

Figure 10. Sequence diagram of coupling third stage.

After this instantiation process, envelope and driver objects are connected and ready to communicate to each other through LAN protocol, independently of physical media (Pm) and physical media access function (F1). It is important to note that controller particularities, regarding controller and equipment programming commands, are managed inside commands file, making them transparent to envelope object.

Finally, in the fourth stage, CDriver instanced by CEnvelope using software component technology, communicates the command in detail to the physical equipment, Figure 11. Therefore, CDriver changes whenever is required by a new physical equipment but CEnvelope and CEquipment may remain the same, changing only when the operating computer platform of the new equipment changes. Using a file for command and argument description not only is useful to free SFS, proxy and envelope objects from handling controller commands and arguments syntax particularities, but also to make it transparent at driver object level, because commands and arguments are sent directly to the equipment controller as they are received from the envelope object.

6. Validation

The proposed coupling framework was applied to different manufacturing cells such as a robotized automatic test system (ATE), two different flexible manufacturing cells and a batch production subsystem of chemical products.

An SFS dispatcher was implemented for sending orders to shop floor equipment as specified in Section 5. The relevant characteristics of the equipment used in the mentioned manufacturing systems are listed in Table 3.

The instance diagram of Figure 12, shows the needed object types used to implement the proposed coupling structure for the mentioned manufacturing cells and the subsystem of chemical products. The SFS dispatcher creates instances of proxy types (CEquipment) that are used to represent each one of the actual equipment. This is a straight forward process using software component technology. Only two implementations of each one of the CEquipment subclasses (CProcessor, CManipulator, CTransporter and CASRS) were required to instantiate all the twelve required proxy objects. Therefore, such classifications, together with the generic commands, allow SFS dispatcher being independent of equipment particularities. Implementation of CEnvelope1, CProcessor1, CManipulator1, CTransporter1 and CASRS1 were done using queue messaging system (MSMQ). On the other hand, CManipulator2, CTransporter2 and CEnvelope2, used web services protocols. Messaging queue was proved to be a robust implementation solution when the SFS and the controller equipment are in the same computer platform, taking care on its own of many details of distributed object system operation.

Figure 11. Sequence diagram of coupling fourth stage.

Also, given that CEquipment and CEnvelope protocol must match, only two implementation of CEnvelope class were required to instantiate all the twelve envelope objects. On the other hand, due to CDriver carries the particularities of each one of the equipment controllers, an implementation of CDriver class may be required for each one of the different equipment. CDriver implementation faced different particularities. CDyna and CUP20 require special implementations due to the communication based on parallel Inputs/Outputs signals specifically designed for the equipment controller. CNC-Motion manages the cnc machine command console through Ethernet. CG-P type driver objects used the VISA standard to access equipment with different physical media such as RS-232 (Seiko, CSR, PLC, and Vision Unit) and GPIB (ATE). Finally, using OPC standard, the driver object of type C-OPC manages controllers based on PLC from different manufacturers. It is important to notice that factors C and A in expression (1), regarding controller particular commands in programming equipment operation, are transparent to CDriver coding. Also, it was shown that incorporation of a particular driver object to the respective envelope object is eased by using software component technology.

Figure 12. Instances types required during validation process.

7. Conclusions

It was shown that adaptation of the loose coupling concepts, proposed in the reference scheme, provides a holistic solution for the problem of integration SFS-SPE. In previous reports, regarding this type of integration problem, where LC based solutions are claimed, the design effort focuses primarily on the coupling technological component, but the conceptual support is missed, such as identification of requirements in order for LC to exist, compensators elements as well as behaviour to be expected from a LC system. It was demonstrated that the proposed scheme, based on LC compensators and the definition of coupling components, including the technological component, represents a suitable solution for integrating SFS-SPE, preserving the dialectical sense autonomy/communication, feature strongly associated with LC concept and missing in other solutions. Thus, the SFS required no change for coupling different sets of manufacturing equipment which involved various different communication protocols, programming languages and equipment types, not explicitly considered during design time. Other reported solutions exhibited some limitations due to design time decisions such as computer platform and standards of protocols to be used for manufacturing equipment integration. In the validation exercise, the coupling framework showed an acceptable level of persistence by requiring only two types of sets of proxy objects and two types of envelope objects. The abstraction of the manufacturing equipment in only four types of proxy objects as part of the coupling structure, leads to a low variety of software objects types used to represent each of the manufacturing equipment and simplifies commanding them from the SFS. In other proposals, these objects have a closer association with a more specific nature of the equipment, such as robot or PLC proxy objects, implying reusability for equipment of that nature, but requiring much more new proxy object implementations to integrate new equipment of different nature. The higher level of abstraction of the equipment nature, proposed in the reference scheme, allows the use of the same type of object proxy in integration of a wider variety of equipment types. However, in the proposed structure, when protocol changes arise from requirements of a different computer platform in remote communication with the equipment (for example, MSMQ to web services); changes are still needed in implementation of proxy object. Downstream, in driver objects, changes may be required sometimes when new equipment is added. On the other hand, the suggested abstraction level to be used for task component management, throughout the system (SFS, proxy, envelope and driver), loosens the coupling between drivers and equipment commands (controller and languages commands) required for task execution. This expands the application domain of a typical driver, because it is freed from the process of translating the received high-level commands. Also, access to the physical media must be supported by driver object; therefore, using physical media standards reduces the need for driver changes. However, the use of widely accepted standards or standards that support different physical media (for example, Visa), does not assure that the physical media requirements for integration of new equipment will not cause changes of driver object implementation. Hence, it is important to highlight the use of a common computer platform, as it is suggested between SFS and proxy object as well as between envelope and driver objects, because it supports the dynamic selection (run time) of the proxy or driver objects. The changes made in the coupling framework, for adapting each one of the manufacturing equipment to the SFS, were bounded to the specific coupling stage involved, thereby avoiding change propagation upstream (buffering). The SFS also showed ability to assimilate change in situations unforeseen at design time (adaptability). Also, using widely accepted standards, the coupling framework conforms to the type of adaptability defined as collective judgment (shared values). In addition, using MSMQ and web services, not considered at design time in the validation example, demonstrated that the proposed scheme supports incorporation of new partial solutions. Such as those generated by the emergence of new technologies, which had not been considered in other reported solutions.

Name	Manufacturer	Communication protocol	Nature
Robot UP20	Motoman	Hardware especial	Manipulator, Processor
Robot	Seiko	RS-232	Manipulator
Robot	CRS	RS232	Manipulator
Lathe CNC	DYNA	(Hardware especial).	Processor
Router CNC	CNT Motion	Closed application; no protocol available. (PC based controller).	Processor
PLC based conveyor	PLC SLC 500	RS 232	Transporter
Test Workstation (artificial vision)	PC-Camera based system	Ethernet	Processor
Test workstation (measurement instruments)	Fluke, Textronix, Agilent	GPIB, RS 232, Ethernet	Processor
ASRS	PLC SLC 500	RS 232	Storing and retrieving

Table 3. Characteristics of manufacturing equipment set used for validation.

References

[1] M. Mehrabi, et al., Reconfigurable Manufacturing Systems: Key to future manufacturing, J. Intell. Manuf. n° 11, pp.403-419, 2000.

[2] R. M. Setchi and N. Lagos, Reconfigurability and Reconfigurable Manufacturing Systems State-of-the-art Review, in 2004 IEEE International Conference on Industrial Informatics (INDIN´04), Berlin, June, 2004, pp. 529-535.

[3] L. Aiping and L. Chao, Reconfigurable Manufacturing System Modelling Method based on object-oriented technology, in 2009 IEEE International Conference on Mechatronics and Automation, Ghangchun, 2009, pp.3420-3425.

[4] R. Zurawsky, Scanning the Issue. Special Issue on Industrial Communication Systems, Proc. of the IEEE, June 2005, pp. 1067-1072.

[5] S. X. Ye and R. G. Qiu, An Architecture of Configurable Equipment Connectivity in a Future Manufacturing Information System, in IEEE International Simposium on Computational Intelligence in Robotics and Automation, Kobe, Japan, 2003, pp. 1144-1149.

[6] J. R. Martinez Lastra and I. M. Delamer, Semantic Web Services in Factory Automation: Fundamental Insights and Research Roadmap, IEEE Trans Ind. Inf.,vol. 2, nº 1, pp. 1-11, February 2006.

[7] National Instrument, NI-VISA User Manual. Part Number 370423A-01, National Instrument, 1997.

[8] A. J. R. Aendenroomer et al., "Message Discovery in SEMI Communication for Rapid Equipment Integration", in IEEE International Conference on Industrial Informatics, Singapore, 2006, pp. 525-530.

[9] G. D. P. Z. Putnik, "A semiotic framework for manufacturing system integration - Part 1: Generative Integration Model", Int. J. Computer Integr. Manuf. Vol. 23, N. 8-9, pp. 691-709, 2010.

[10] J. Lapha, "RobotScript™: the introduction of a universal robot programming language", Ind. Robot: Int. J., Vol. 16, Num. 1, pp. 17-25, 1999.

[11] G. B. M. Biggs, "A Survey of Robot Programming Systems", in Proceedings of the Australaian Conference on Robotics and Automation, Brisbane, Australia, 2003, pp. 1-10.

[12]C. Duncheon., "Universal robot language - does it support agile automation?", Ind. Robot: An Int. J., pp. -, Issue 1, 1999.

[13] Z. Pan, et al. "Recent progress on programming methods for industrial robots", in Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), 2010, pp. 1-8.

[14] M. Lichtveld and B. Van der Zon, "A New Architecture for Equipment and Clusters for Backend Processes", in SEMI Technology Symposium: International Electronics Manufacturing Technology (IEMT) Symposium, 2002, pp. 398-402.

[15] A. Lobov et al. "Service Oriented Architecture in Developing of Loosely-coupled Manufacturing Systems", in IEEE International Conference on Industrial Informatics (INDIN 2008), Daejeon, Korea, 2008, pp. 791-796.

[16] A. Ferrolho and M. Crisóstomo, "Intelligent Control and Integration Software for Flexible Manufacturing Cells", IEEE Trans. Ind. Inf. Vol. 3, Num.1, pp. 3-11, 2007.

[17] OMG, CORBA Based Machine Control, Object Management Group, 1998.

[18] G. Chryssolouris. "Manufacturing Systems. Theory and Practice", New York, Spring. Ver., 2a. ed. , 2005.

[19] M. P Groover., "Fundamentals of Modern Manufacturing, Materials, Processes and Systems", México, McGraw Hill, 2007, pp. 194-884.

[20] S. Melnyk. and P. Carter, "Production Activity Control", Homewood Ill, Irwin/APICS Ser. in Prod. Mngmnt, 1987, pp. 30-32.

[21] AMICE, CIMOSA: Open System Architecture for CIM, Brussels, Spring. Ver. 2nd ed., 1993.

[22] F. Vernadat, "Enterprise Modelling and Integration Principles and Integration", London, Chap. & Hall, London,1996, PP.45.

[23] G. Plossl, "Orlicky`s Material Requirement Planning", New York, McGraw Hill, 1994, pp. 11.

[24] T. E. Vollman. et al., "Manufacturing Planning and Control Systems", New York, McGrawHill,1997, pp. 803.

[25] T. Lunn and S. A. Neff, "MRP Integrated Material Requirements Planning and Modern Business", Irwin, 1992, pp. 7.

[26] F. Cheng et al., "Development of a system framework for the computer integrated manufacturing execution system: a distributed object oriented approach", Int. J. Computer Integr. Manuf., pp. 384-402, 1999.

[27] S. Owen et al., "A Modular Reconfigurable Approach to the Creation of Flexible Manufacturing Cells for Educational Purposes", in Intelliigent Robots and Systems Proceedings. of the IEEE Conference on, 1997, pp.1-13.

[28] F. Gonzalez and W. Davis, "A Simulation Based Controller for Distributed Discrete Event Systems with Application to Flexible Manufacturing", in Proceedings of the 1996 Winter Simulation Conference, 1996, pp. 845-852.

[29] F. Chan and J. Zhang, "Object Oriented Architecture of Control System for Agile Manufacturing Cells", Proceedings of the International Conference on Management of Innovation and Technology, 2000, pp. 863-868.

[30] G. Berio and F. Vernadat, "Enterprise modelling with CIMOSA: Functional and Organizational aspects, Production Planning and Control", Prod. Plann. Contr., 2001, pp. 128-136.

[31] H. Engelke et al. "Integrated Manufacturing Modelling System", IBM J. Res. Dev. Journal of Research and Development, vol. 29, nº 9, pp. 343-55, 1985.

[32] J.G. Nell and N.B. Christopher, "Standards Road Map Project. Standards Classification Strategy and Methodology. A task of the manufacturing-enterprise integration project", National Institute of Standards and Technology, Gaithersburg MD, 1999.

[33] F. Chan and J. Zhang, "Modelling of agile manufacturing systems", Int. J. Prod. Res.,vol. 39, nº 1, pp. 2323-32, 2001.

[34] H .Freund and H.-J. Buxmaum, "Universal Work Cell Controller-Application Experiences in Flexible Manufacturing", Proceedings of the IEEE Conference on Intelligent Robots and Systems, 1994, pp. 56-63.

[35] J. White and E. Cone, "Using Measurement Studio GPIB to Accelerate Development with Visual Basic", National Instrument, Application Note, Austin Tx, 2001.

[36] S. Adiga, "Object-Oriented Software for Manufacturing Systems", London, Chap.Hall, London, 1993, pp. 26.

[37] G. Berio et al. "The M* OBJECT Methodology for Information System Design in CIM Environment", IEEE Trans. Syst. Man Cybern., vol. 25, no. 1, pp. 68-85, 1995. (MObjMethod.pdf)

[38] J. Barry et al., "NIIP-SMART: An investigation of Distributed Object Approaches to Support MES Deployment in a Virtual Enterprise", Proceedings of Enterprise Distributed Object Computing Workshop EDCO'98, 1998, pp. 366-377.

[39] R. Fûrich et al., "A Component Based Application Framework for Manufacturing Execution System in C# and .NET", in 40th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific 2002), 2002, pp. -.

[40] C. Pautass and E. Wilde, "Why is the Web Loosely Coupled? A Mult-Faceted Metric for Service Design",World Wide Web Conference-WWW2009, Madrid, 2009, pp. 911-920.

[41] M. P. Papazoglou et al. "Service-Oriented Computing: State of the Art and Research Challenges", Computer, 2007, pp. 38-45.

[42] Sang Chul Ahn et al. “uPnP Robot Middleware for Ubiquitous Robot Control”, in The 3rd International Conference on Ubiquitous Robot Control and Ambient Intelligence (URAI 2006), 2006, pp. 53-58.

[43] S. Helal, “Standards for Service Discovery and Delivery”, IEEE Pervasive Comput, vol. 1, no. 3, pp. 95-100, 2002.

[44] M. Kwiatkowska, “Introduction”, Philos Transact A Math Phys Eng Sci, pp. 3365-3668, 2008.

[45] S. Resnick et al., “Essential Windows Communication Foundation For .Net Framework 3.5”, Boston, MA, Pearson Education Inc., 2008, pp. 2.

[46] J. D. Orton and K. Weick, "Loosely Coupled Systems: A reconceptualization", Acad Manage Rev, pp. 203-223, 1990.

[47] Real Academia de la Lengua, "Diccionario de la Real Academia Española de la Lengua", [Online], Available: http://www.rae.es. [Accessed 5th January 2012].

[48] J. D. Thompson, “Organizations in Action: Social Sciences Bases of Administrative Theory”, New York, McGraw Hill, 1967, pp. 4-10.

[49] A. Cucinotta et al. “A Real-Time Service-Oriented Architecture for Industrial Automation”, IEEE Trans Industry Inform,vol. 5, no. 3, pp. 267-277, 2009.

[50] K. Weick, “Educational organizations as loosely coupled systems”, Adm Sci Q, no. 21, pp. 1-9, 1976.

[51] R. Schrenker and T. Cooper, “Building the Foundation for Medical Device Plug-and-Play Interoperability”, Medical Electronics Manufacturing, pp. 10-21, 2001.

[52] H. A. Schmid, “Framework Reuse over Different CIM Subdomains”, in M. E. Fayad and R.E. Johnson, Domain-Specific Application Frameworks, New York, John Wiley and Sons Inc., 1999, pp. 67-84,

[53]C. Schlotz and M. Röck, “Reorganization of a production department according to the CIMOSA concepts”, Comput. Ind. Vol. 27, pp. 179-189, 1995,

[54] P. Zuesongdham, “Combined Approach for Enterprise Modelling: CIMOSA and SOA as dynamic Architecture maritime logistics”, in W. Kersten et al. (eds.), Innovativ Logistics Management. Competitive Advantages through new Processes and Services, Berlin, Erich Schmidt Verlag, 2007, pp. 134.