Advances in Robotics & Automation

ISSN: 2168-9695

Asif and Webb, Adv Robot Autom 2016, 5:2
DOI: 10.4172/2168-9695.1000147

Research Article Open Access

Enhanced Cell Controller for Aerospace Manufacturing

Asif S* and Webb P

School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, MK43 0AL, UK

Abstract

Aerospace manufacturing industry is unique in that production typically focuses on high variety and quality but extremely low volume. Manufacturing processes are also sometimes unique and not repeatable and, hence, costly. Production is getting more expensive with the introduction of industrial robots and their cells. This paper describes the development of the Flexa Cell Coordinator (FCC), a system that is providing a solution to manage resources at assembly cell level. It can control, organise and coordinate between the resources and is capable of controlling remote cells and resources because of its distributed nature. It also gives insight of a system to the higher management via its rich reporting facility and connectivity with company systems e.g., Enterprise Resource Planner (ERP). It is able to control various kinds of cells and resources (network based) which are not limited to robots and machines. It is extendable and capable of adding multiple numbers of cells inside the system. It also provides the facility of scheduling the task to avoid the deadlocking in the process. In FCC resources (e.g., tracker) can also be shared between cells.

Keywords: Flexible manufacturing system; Distributed network cell controller; Cell controller; Software programmable logic controller; Aerospace cell controller; Web services

Introduction

There is little use of automation in aerospace manufacturing compared with other industry sectors such as automotive, pharmaceutical or white goods. This is because the product volumes tend be small but the product lifecycles may exceed 30 years. High quality parts must be used and all the associated processes must be traceable and verifiable. The variety of parts is also high and this low volume high variety mix means that a large number of conventional automated systems would be required that will have low utilisation rates and therefore be uneconomical. A further and significant complication is that in operations such as turbine blade repair the manufacturing requirements are not known until the process has started.

The Flexible Manufacturing System (FMS) [1] was developed to overcome some of these issues. FMS utilises the integration of automated component storage, tool delivery and CNC machines with an overall computer control unit to support and monitor the performance of the system. These FMS installations have been very successful; however in many cases the flexibility has been very low when trying to produce a wide variety of changing components within existing cells [2]. In the case of aero-structure assembly the problem becomes even more complex. Because of the large size of assemblies, the automation (robots) must be moved to the part rather than the conventional approach of moving the parts between individual cells or along a flow-line. This means that multiple processes are likely to be performed using the same processing equipment but in different locations. To enable this, a new approach of cell control and organisation is required. For example a robot may be used for a drilling task in one part of the factory and then used later for applying sealant in a completely different part of the factory.

A manufacturing robot cell can consist of 20 or more robots along with other machines. They are then controlled by Programmable Logic Controller (PLC). Flexa is the acronym of Flexible Automation Cell. The term flexibility is defined as the system's ability to adopt the change easily [3].

Flexa is based on the concept of FMS. There are four classes of manufacturing attributes that need to be considered while designing manufacturing system: (i) cost, (ii) quality, (iii) time and (iv) flexibility. FMS is the approach for flexible and cost-effective means of manufacturing. FMS can consists of two or more computer-managed workstations, material transport system and another computer that controls the transportation operations, tools and other related information. There are a number of components involved in forming FMS like material handling and storage system, programming language and network infrastructure, workstations and human labour. The reason behind using FMS is that businesses want the groups of machines and tools to form a system along with programming and network that can work continuously and with minimal intervention from Human Labour [4].

There are lots of operational tasks of FMS implementation like machine loading, part routing, grouping, tool management, scheduling, etc. Scheduling is an important element (when for about) of FMS operations. Holonic manufacturing scheduling has been used for the scheduling of cells [5] but it doesn't state any communication between the cells and its resources [6]. It is not much of an improvement over the Factory Coordinator Cell in the Holonic approach which can communicate as a liaison between cells. Moreover although it refers to the automation of cells but any modification in the cells' resources and the processes could take up to a year for planning and implementation. EAS (Evolvable Assembly Systems) has further improvement through Instantly Deployable Evolvable Assembly Systems which is a plug and produce system. However none of these approaches allow reconfiguring the system at runtime hence unable to make any change while system is running [7].

The computer software applications can be utilised for assistance

***Corresponding author:** Seemal Asif, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, MK43 0AL, UK, Tel: +44 1234 750111; E-mail: s.asif@cranfield.ac.uk

Received March 11, 2016; **Accepted** April 27, 2016; **Published** April 30, 2016

Citation: Asif S, Webb P (2016) Enhanced Cell Controller for Aerospace Manufacturing. Adv Robot Autom 5: 147. doi:10.4172/2168-9695.1000147

in FMS. There are several software techniques that are used for FMS and application of service oriented architecture (SOA) in the aerospace industry. This was proposed by the Telematics Department of Hamburg University of Technology [8]. Their methodology proposed the collaboration between various enterprise systems in aerospace industry. They used web services interfaces for the integration and communication between existing systems i.e., ERP (Enterprise Resource Planner). The study shows how to make the system flexible by using the SOA approach.

They proposed a set of processes along with verification for the implementation of SOA in aerospace industry applications. Their system development steps were the following:

1. Requirements Capturing → Graphical Model: Analysing the data and information flows. Model the requirements and analyse the interaction between different actors i.e., use case modelling [9].

2. Requirements Validation: with process owners who are the current users of the system.

3. Identification of cooperation parameters → Selection of Web Service Protocols (WS-Protocols): derive the cooperation specific parameters i.e., security, safety, etc. The derived parameters then helped to select the WS-Protocols/Simple Object Access Protocol (SOAP) which satisfied the requirements [10].

4. WS Protocols Verification: Verification using Temporal Logic of Action for concurrent reactive system. It will then give feedback to previous step if necessary [10].

5. Implementation.

6. System Validation Testing.

Their approach gave the idea of implementing SOA for an aero-structure assembly FMS. However their concept does not give any detail how it can interface with the cell and its resources.

RAPOLAC (Rapid Production of Large Aerospace Components) is another example of automated cell [11]. It is referring to the automation of a cell and the resources using the feedback data from a laser tracker. It is also talking about the automation of a single cell and not multiple cells. It does not detail the underlying architecture for the automation. The Automated Assembly of Wing Panel for A340-600 has the similar footprints [12,13].

The problem with the current automated cells is that they are not fully automated and there is need to do the manual intervention like loading programs on controllers, loading programs on PLC, changing programs on PLC etc. A little advancement towards minimising the human intervention is the implementation of SOA. The work for the aerospace industry using SOA was analysed [8] it is giving useful approach for coordinating between the aerospace industry applications but that is not giving any information about cell itself and how to communicate between resources e.g., robots.

Automotive industry may not need rapid changes in the cells and processes but the aerospace industry is unique in the sense of every part and its manufacturing. The need to reposition the entire robot around the part is understandable in aerospace industry as work can be done on different models in the same cell. And there may be need to change the entire cell structure to assemble different parts of the wing. Hence there is a need for a completely automated and flexible system which can cater the needs of aerospace manufacturing industry.

Flexa Cell Coordinator

Flexa Cell Coordinator (FCC) is a fully automated system which receives programs and transfers it on the required resources in a controlled manner. It is the part of Flexa Cell [14] in which it is coordinating between cells and its components. It is responsible for execution of received programs in a conflict free way on the required resources. It is also important to run and manage multiple cell coordinators at the same point in time. For this purpose the use of Software Programmable Logic Controller (SoftPLC) was introduced which means there is no need of hardwired binding of resources with the cell.

A recipe is composed of program(s) and information about the resource on which its program need to run. A program is the set of instructions which need to run on specific resource. Program has to be in the language which resource can process for example comau c3g controller can run PDL2 (programming language) files and instructions [15].

FCC receives the program outside of the system using web services in a specific format called recipe. It un-marshals the recipes and schedules it according to the availability of resources. It then activates or creates the sub coordinator which is having SoftPLC and the controller of the programs and resources. The controller called program manager downloads the program onto the resources and gives control to SoftPLC program to control the resources and program execution. It is also responsible of getting data back from resources (if required) and sends it back to main program of FCC called application manager. And from here the data can send back to the resource of the recipe. FCC is having two way communications as it accepts the data in form of recipe and sends data back.

FCC Architecture

The methodology for control and organisation developed makes a number of assumptions about the process and the production resources being used. These are as follows:

- Production is assumed to be chaotic due to the number of processes and the likelihood of concessions needing to cleared and in the case of repair the process sequence is not known until the part has been inspected.

- The resources (robots, machine tools etc) can be used in different sequences for different operations either by physically relocating them or changing the root of a part through the resources.

The issues noted above mean that the use of a conventional control methodology using a Programmable Logic Controller (PLC) and a number of machines (resources) physically coupled together. The approach of running preloaded programmes on resources is impractical as in the cases above the individual cells would need to be physically reconfigured for each operation. A new way of approaching this problem is to use a central cell controller. It is capable of producing any number of virtual controllers which can take control of local groups of machines to form virtual cells which then behave like conventional physical cells. These sub coordinators are software applications which are tied to their resources using a common interface. The overall cell controller is responsible for decoding recipes, allocating and scheduling resources and launching and destroying instances of the sub-controllers when required. It also contains a database which is used to store status information to allow recovery of system status in the case of an equipment or process failure.

A diagram showing the structure of the cell coordinator is shown in Figure 1.

In summary:

- The use of a SoftPLC means that there is no longer any physical hardware associated with the cell.
- All the production systems for example machine tools, robots or measuring systems are classed as networked resources which have a common interface which allows them to be interrogated and identified automatically.
- When a particular production sequence is identified a 'recipe' is generated from which the required resources can be identified, allocated and programming information loaded.
- The resources are co-ordinated using SoftPLC which is connected to the resources over a network.
- The overall control is provided by a cell co-ordinator which allocates a virtual sub co-ordinator for each cell
- Once the task is finished then the virtual cell sub-coordinator is closed and all the resources are freed up and made available for other tasks that may be waiting for resources.

The individual elements have the following functions:

Application manager

The Application Manager is responsible for the control of the FCC. The Application Manager has responsibility to activate the FCC and communicate between the FCC's components. The Application Manager also has responsibility to communicate with the rest of the world via the web services layer.

FLEXA scheduler

This is needed to schedule the task among resources. The Scheduler will be able to identify the available resources and will allocate the task to them according to the recipe - sent from Flexa Database (FDB). The Scheduler will also be able to resolve and avoid conflicts (Deadlocks, etc).

Status database and monitor

The Cell coordinator status database is used to record the status of the resources available and the availability of resources (if they are available for handing over to a task). The status database monitor has active two way connection with the status database monitor which monitors all the activities of the resources and records the status of all the current recipes.

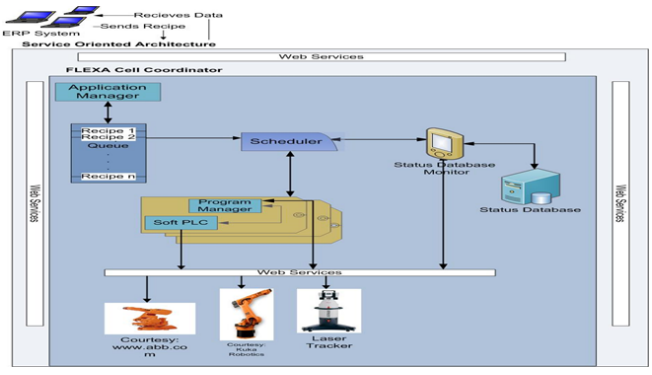

Figure 1: Flex cell coordinator structure.

Recipe queue

The Recipe Queue accepts the request from the Application Manager to execute a particular recipe, if there is one present. The recipes will be sent out for execution by the scheduler on first come first serve basis.

Cell sub-coordinator

The Cell Sub-Coordinator is comprised of a Program Manager and the SoftPLC. The program manager receives the recipe from Scheduler and delivers programs to the resources(s). The SoftPLC also takes its program from Program manager and controls the resource(s) accordingly. There can be multiple FLEXA Cell Sub-Coordinators all of which will be controlled by the Application Manager. One sub-coordinator will work with one set of resources and each other one will use a different set of resources.

FCC Design

The biggest challenge in designing a flexible control system with varying cell resources is to design a loosely-coupled structure without losing efficiency and yet the system should also be easy to reconfigure and extend. Since the cell resources usually come from different manufacturers and use different platforms and communication protocols the traditional distributed computing technologies such as COM (Component Object Model), could lead to very tightly a coupled relationship between cell resources. Any changes to the system such as a newly installed metrology system may need some alterations within the original software which significantly reduces flexibility and re-configurability [16].

It was therefore proposed that a SOA would be more efficient. An SOA uses a web service as the basic element. This was originally designed to support interoperable machine-to-machine interaction over a network [17]. A web service is platform-independent as it uses the standardized SOAP as its communication protocol [18] and the XML format as its message exchange format. As a result, any device that supports TCP/IP communication can be programmed to provide a web service which greatly reduces the complexity of communicating with different platforms and environments.

Web services perform functions and actions which can be anything from simple function requests to complicated processes requests. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service which makes SOA naturally support plug-play. As web services are loosely coupled and have a generic communication protocol and data format, they can be easily deployed within a distributed system.

Therefore a SOA has been used to realise the FCC.

FCC Working

Steps: Following are the main steps for the execution of recipe inside FCC:

1. FCC receives recipe from Flexa System
2. Recipe placed in the Recipe Queue
3. Scheduler picks the recipe and schedule it according to the availability of resources

4. Scheduler activates/(creates instance) cell sub-coordinator

5. Program manager (cell sub-coordinator) picks the scheduled recipe

6. Program manager downloads the program on resources

7. SoftPLC controls resources with programs

8. Program manager sends result of SoftPLC back to FCC Application manager

9. Application manager stores the result into the database and sends results back to the system which send the recipe (source)

Cell sub-coordinator plays from step 5-7. It is possible to send the data back from resources to sub-coordinator and from sub coordinator back to FCC Application manager which can then send it back to the source of recipe if needed to.

FCC interface

FFC interface displays the summary of the FCC as showed in Figure 2. It presents numbers about the recipes, sub-coordinator, and resource status. The details of recipe and its programs can also be viewed. This is the main interface for the operator of FCC who can use to manage FCC. It enables the user to upload recipe, check the recipe and resource status, manage the resources, etc.

Recipe structure

Recipe is the XML file which contains the configuration about the recipe files i.e., resource programs. There are a number of ways to set up the recipe. It can point programs from other recipes as well. A single recipe can run in one cell sub-coordinator and collaborate with other cell sub-coordinators. Figure 3 is an example of recipe which is pointing a program from other recipe to execute after the execution of its program "Sample A". It can also contain other relevant information which may be useful for the execution of programs. In Figure 3 the part number information is added which is helping program to have information about the part number.

Scheduler

It is the integral part of FCC which is helping to schedule the recipe's programs without any conflict on the resources. Scheduling helps to save the time and effort of resources. It also helps to share the resources among cells based upon the recipe structure. Scheduling can be done using many available algorithms for example First In First Out (FIFO), Last In First Out (LIFO) or Round Robin. Here FIFO technique is used inside the FCC scheduler [19,20].

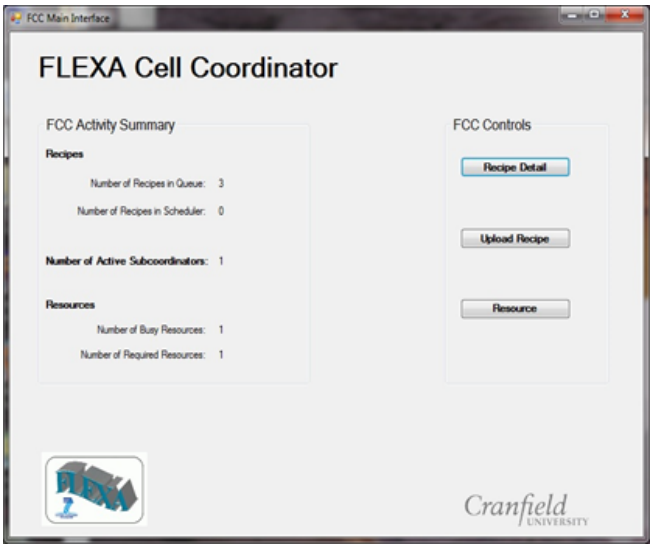

Figure 2: FCC main interface.

```
<program name="Sample A">
  <programname> Sample A </programname>
  <filename>A.cod</filename>
  <programtype>cod</programtype>
  <nextprogram recipe id=" 12345"></nextprogram>
  <previousprogram recipe id=" 12346">B.Cod</previousprogram>
  <resource>Comau</resource>
  <additionalinfo serial="1">
    <infodata name="measuring" type="partno" accesslevel ="resource">4711</infodata>
  </additionalinfo>
</program>
```

Figure 3: Recipe sample structure.

FCC scheduler has a meaningful and understandable interface which helps the user to understand about the scheduler activity. In Figure 4 interface shows that scheduler is working on two recipes which can be viewed in top left pane of Recipe Queue. The history of scheduled recipes can be viewed as well in scheduled/processing grid.

FCC salient features:

1. Supporting multiple cells concurrently
2. Resource sharing
3. Runtime resource management
4. Adding resource into the cells runtime
5. Adding cell into the FCC system on runtime
6. Complete logging and tracking of the system using database management system
7. Storage of recipes and scheduling
8. Capability of restarting the system after recovering itself from error
9. Simple and friendly user interface for operator
10. Distributable control

FCC Testing

Testing the FCC system proves its strengths and flexibility. It has been tested in various ways such as (i) support of multiple resources, (ii) adding/configuring resource while FCC is running, (iii) support of multiple cells concurrently, etc. SoftPLC was managed to recognize the system and operate all of the resources inside cell. FCC was able to send and receive recipes for all of the cells and was able to create instances of cell sub-coordinators for all of them. Figure 5 shows how SoftPLC manages multiple cell coordinators concurrently. This is the test which we run in the real cell Figure 6 shows the original cell.

Test Case 1

Cell Coordinators 2, Resources 4 (R1: KR200 with clamping end-

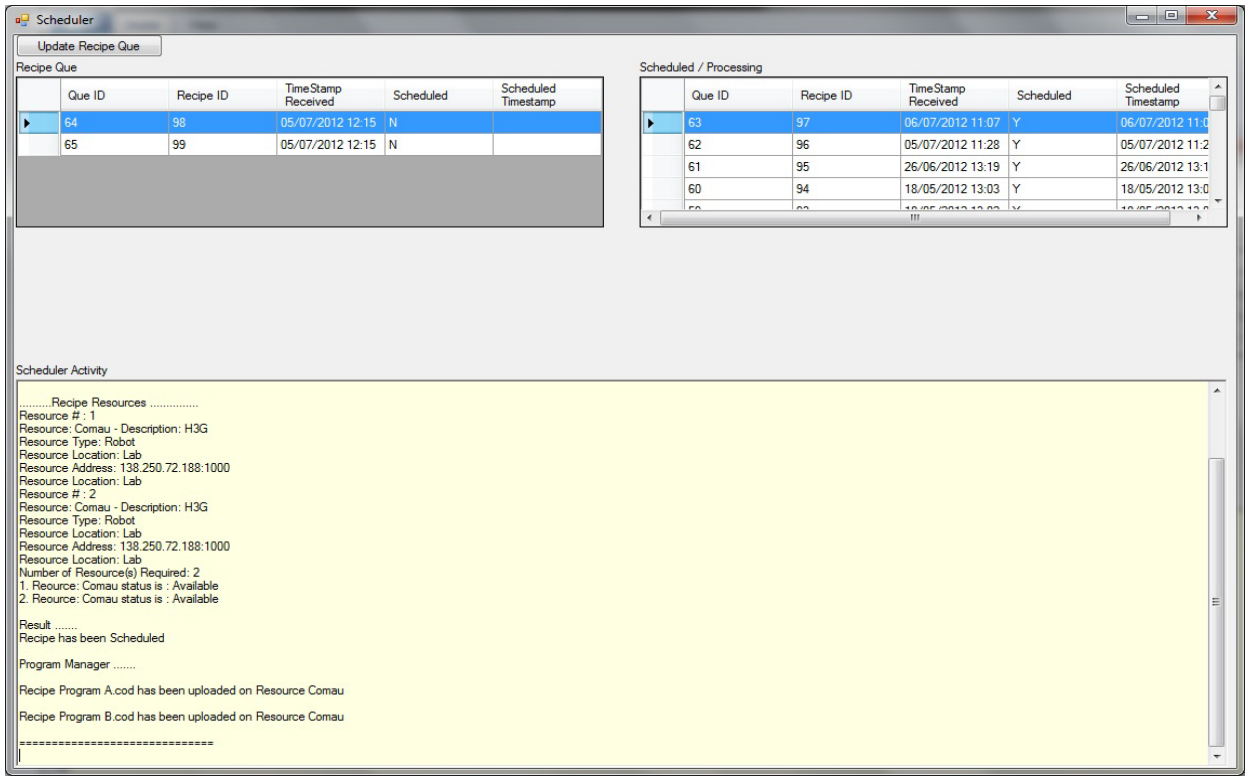

Figure 4: Scheduler interface.

Figure 5: The original cell [20].

effector, R2: NM45, R3: LEICA Laser Tracker, R4: H4 with Drilling end-effector),

R1, R2, R4 belongs to Cell Sub-coordinator 1,

R3 belongs to cell sub-coordinator 2,

Communication ontology: Cell 1 ↔ Cell 2,

Purpose: To test the capability of Flexa Cell Coordinator between multiple resources and cells.

Scenario: The task was to do drilling on a nominal part. R2 was doing the pre and post inspection of the part. R1 was picking the part from loading bay, putting it on the jigs & fixture and then putting it on delivery bay. R4 was responsible for drilling on the part and R3 (virtually in separate cell) was checking the position and feeding its result to the R4 for getting the right position. Figure 7 shows the FCC cell described in this scenario.

Process: Recipes were sent for these cells. Recipes were received by Application Manager and Queued in Recipe Queue. After checking the resource availability Recipes were picked up by FCC Scheduler (on at a time) un-marshalled and scheduled. Two respective cell sub-coordinators were created which includes the Program Manager and SoftPLC. Programs were transferred to respective cell sub-coordinators. Program manager downloaded recipe's programs (set of instruction to run on resources) on the resources. All of the programs were then waiting signal from SoftPLC to get it executed. Program which was dealing with R4 for drilling was scheduled after the completion of pre-inspection by R2. The data was sent back to the program manager by the resource's (R2) program which was then transferred to cell coordinator and then drilling task was carried out by R4. Cell #2 was also activated at that point along with drilling. R3 was constantly checking the position of R4 and giving its feedback to Cell #1 program manager which is then passing information to R4 for getting the accurate position. R4 (Cell #1) and R3 (Cell #2) are communicating constantly via their program managers. R1 lifted the part from loading bay and put it on delivery bay after the completion of drilling process. At the end R2 did the post inspection of the part. Every cell's resources were controlled by their

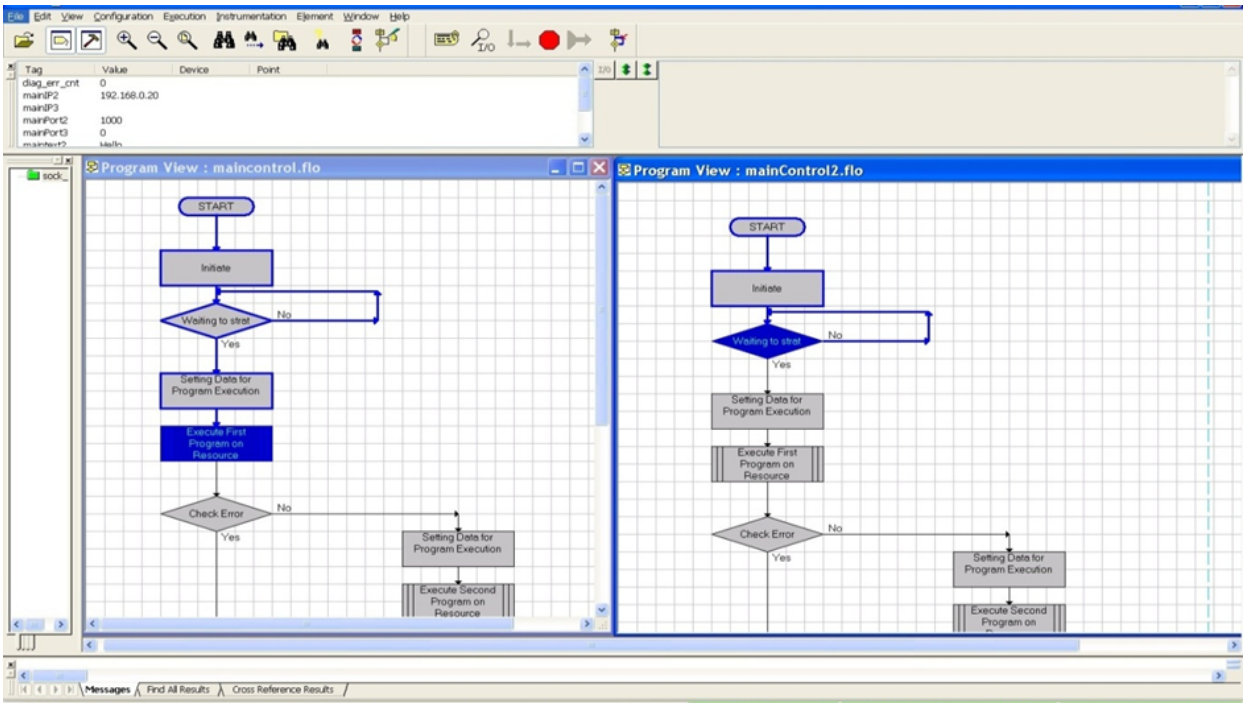

Figure 6: Soft PL Ccontrols resources in multiple cells.

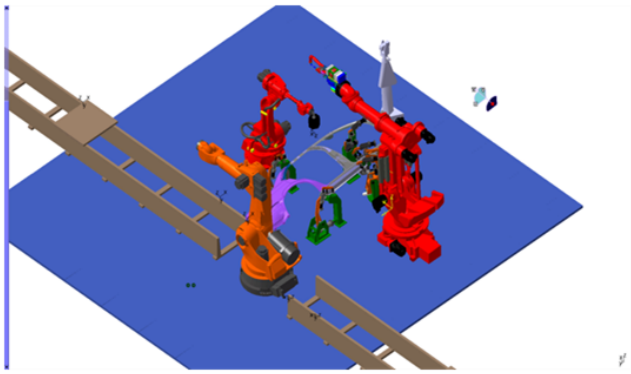

Figure 7: FCC test cell.

respective SoftPLC and the program managers. After the completion of process data was sent back to Application Manager of FCC. The Application Manager then forwarded the data to the source of recipes i.e., ERP System. Complete process was logged at every step.

In this case flexibility of the system was tested by adding new cell (and its resources) while system was up and running. This is mentioned in the process.

Test Case 2

Purpose: To test the capability of Felxa Cell Coordinator across multiple cells on multiple locations.

Cell Coordinator 3 (includes 2 cells from test case 1), Cell #3 Digital I/O (acting as four different resources for testing purposes) R5, R6, R7, R8,

Communication ontology: Cell 1 ↔ Cell 2, Cell 3,

	Number of Resources	Number of Cells	Time to complete the operation*	Number of errors	Total Time**
Test Case 1	4	2	12 µsec	0	20 sec
Test Case 2	8	3	168 µsec	0	75 sec

*Operation means the execution of recipe on the resource under the control of SoftPLC.

**Total time means time taken for all of the processes of FCC i.e., receiving of recipe, un-marshalling, scheduling, executing it on resources and sending result back.

Table 1: Test results.

Scenario: The test case 1 along with an additional cell which was physically distributed and placed on different location. R5, R6, R7, R8 were the lights attached to Digital I/O which flashed one by one.

Process: Recipes for all the cells received by Application Manager and Queued in the recipe Queue. After checking the resource availability Recipes were picked up by FCC Scheduler (on at a time) un-marshalled and scheduled. Three respective cell sub-coordinators were created which includes the Program Manager and SoftPLC. Programs were transferred to respective cell sub-coordinators. Program manager downloaded recipe's programs on the resources. All of the programs were then waiting signal from SoftPLC to get it executed. The same process was followed by Cell Sub-Coordinator 1 & 2 which is detailed in the process of Test Case 1 above. The Cell Sub-Coordinator 3 gives respective signal to SoftPLC to control R4, R5, R6 and R7. All of these resources light up when activated by corresponding physical switch. Cell sub-coordinator 3 worked independently as it did not need to communicate with any other cell. Every cell's resources were controlled by their respective SoftPLC and the program managers. After the completion of process data was sent back to Application Manager of FCC. Application Manager then forwarded the data to the source of recipes i.e., ERP System. Complete process was logged at every step.

In this case flexibility of the system was tested by running two kinds of communication one is to run an independent cell and two is to deal with the communication between the cells. This is mentioned in the process.

The time of the completed operation calculated carefully by using internal time calculation routine built in the FCC (by using C# programming functions) which is double checked by using system stop watch. The test case results show that system was able to execute the recipes successfully on the FCC without any errors (Table 1).

Conclusion

As proved in the section above FCC can control multiple cells and collaborate in between them. Multiple cells can work independently as well and they can also be controlled by FCC as proved in Test Case 2. There are multiple scenarios which are tested with FCC but because of the space issue all of them cannot be illustrated here. In short FCC is fully capable of managing, coordinating and controlling cells. It becomes more cost effective with resource sharing and scheduling system.

Acknowledgments

This work was funded as part of FLEXA within the European Seventh Framework Programme (FP7). Figure 7 the CAD model for the tests is kindly developed by John Thrower. Particular thanks go to John Thrower, whose collaboration and input into this work has been crucial to its success.

References

1. Greenwood NR (1988) Implementing flexible manufacturing systems. John Wiley & Sons.
2. Shivanand HK (2006) Flexible Manufacturing System. New Age International Pvt Ltd Publishers.
3. Fricke E, Schulz AP (2005) Design for changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. Systems Engineering 8: 342-359.
4. Chryssolouris G (2006) Manufacturing Systems: Theory and Practice. New York: Springer Science + Business Media Inc.
5. Stecke K (1985) Design, Planning, Scheduling and control of Flexible Manufacturing Systems. Annals of Operations Research.
6. Gou L, Luh PB, Kyoya Y (1998) Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation. Elsevier: Computers in Industry pp: 213-231.
7. Onori M, Lohse N, Barata J, Hanisch C (2012) The IDEAS project: plug & produce at shop-floor level. Assembly Automation 32: 124 -134.
8. Kazlauskaite J, Minhas A, Vogt FH (2005) Applying Service Oriented Architecture in the Aerospace Industry. IMCM'05: International Mass Customization Meeting, Klagenfurt/Austria.
9. Pressman RS (2000) Software Engineering: A Practitioner's Approach. (5thedn) McGraw-Hill Higher Education.
10. Johnson JE, Langworthy DE, Lamport L, Vogt FH (2004) Formal Specification of a Web Services Protocol. Electronic Notes in Theoretical Computer Science pp: 147-158.
11. Escobar GP, Gault R, Ridgway K (2010) Robotic manufacturing by shaped metal deposition: state of the art. Industrial Robot: An International Journal 38: 622-628.
12. Hartmann J, Meeker C, Minshull A, Smith A (2000) Automated Wing Panel Assembly for the A340-600. SAE Technical Paper.
13. Holden R, Haworth P, Kendrick I, Smith A (2007) Automated Riveting Cell for A320 Wing Panels with Improved Throughput and Reliability (SA2). SAE Technical Paper.
14. Webb P, Asif S, Hogger S, Kosche T, Kiernan P (2015) Advanced Flexible Automation Cell Control for Aerospace Manufacturing. Aircraft Engineering and Aerospace Technology 87: 156-164.
15. Siciliano B, Villani L (2000) Robot Force Control. USA: Kluwer Academic Publishers.
16. Pires JN, Costa JSD (2000) Object-oriented and distributed approach for programming robotic manufacturing cells. Robotics and Computer-Integrated Manufacturing 16: 29-42.
17. Aphrodite T, Thomi P (2002) An overview of standards and related technology in Web Services. Distributed and Parallel Databases 12: 135-162.
18. Newcomer E, Lomow G (2005) Understanding SOA with Web services. Addison-Wesley.
19. Pinedo M (1994) Scheduling: Theory, Algorithms, and Systems. Springer.
20. Webb P, Asif S (2012) Advanced Flexible Automation Cell. Innovation for Sustainable Aviation in a Global Environment, Madrid.