TAMPEREEN TEKNILLINEN YLIOPISTO

BO ZHOU

A GENERIC METHOD FOR MODELLING SYSTEMS OF SYSTEMS AND ITS APPLICABILITY TO THE FIELD OF FACTORY AUTOMATION

Master of Science Thesis

The topic was approved in Faculty of Automation, Mechanical and Materials Engineering Council Meeting on 17th August 2011.

Academic supervisors:
Prof. Jose L. Martinez Lastra
Dr. Aleksandra Dvoryanchikova

PREFACE

The process of my thesis writing was a journey for knowledge which started with the work in Special Assignment in Factory Automation. The purposes of the thesis are to introduce a generic SoS (System of Systems) modelling method and to evaluate its applicability to manufacturing systems in the domain of factory automation with two use cases. This work was carried out at FAST-Lab of the Department of Production Engineering of Tampere University of Technology.

The state-of-the-art of the thesis offers basic theory and paves the way for the generic SoS modelling method, which is the main result obtained. It was decided to organize the original content of the thesis in 3 chapters: 1) the generic SoS modelling method; 2) the first use case (testbed); and 3) the second use case (Pultrusion Process). This structure was found most appropriate to describe the developed methodology and findings than the classical methodology-results division of materials. Two use cases (testbed and Pultrusion Process) with different complex levels were analyzed and described according to the generic method. Results from these two use cases helped to evaluate the applicability of the method to manufacturing systems.

During my work over the thesis, many people have supported and encouraged me. Here I would like to express my thanks to Professor Jose L. Martinez Lastra for his support and for the opportunity to complete my thesis and to get the unique working experience.

To Aleksandra and Andrei for supervising my work, for their endless patience, enlightening help and encouragement. I am deeply grateful for everything they have done for my thesis, and for me.

I also want to express my thanks to my colleagues for their help: Johannes (my co-author for IECON11), Gorge, Axel; and to Bin, Jaacan, Yulia, Dazhuang, Xiaoyun, Yi and all my other friends for their friendship.

Most especially to my parents and Liu Junjiang, thank you very much for your love, support and encouragement. You are always the main source of my happiness.

Tampere, August 23th, 2011

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Machine Automation
BO, ZHOU: A Generic Method for Modelling Systems of Systems and its Applicability to the Field of Factory Automation
Master of Science Thesis, 70 pages
August 2011
Major: Factory Automation
Examiners: Prof. Jose L. Martinez Lastra, Dr. Aleksandra Dvoryanchikova
Keywords: factory automation, modelling method, system of systems

To solve the problems brought by new challenges such as customized production and shortened production life cycle, numerous new technologies have been introduced to the area of factory automation. The integration of the new technologies to manufacturing systems has enhanced the complexity of systems and lead to a new type of system – system of systems (SoS), whose properties are significantly different from traditional complex systems.

System modelling is an effective approach for analysis, design and maintenance of a system. It is essential for system engineering for reducing cost to a noticeable extent compared to facilitating construction. In order to capture the different properties possessed by SoS, a generic SoS modelling method is introduced based on computation of SoS characteristics and description of the interface activities between component systems.

For evaluating the applicability of the generic modelling method in the domain of factory automation, it is first applied to testbed, which is a conveyor-based pallet transferring system for assembling electronic devices located in FAST lab. The results present that the generic method is capable to describe the whole production process and to compute characteristics of a manufacturing system, which is vital in design and maintenance phases of system engineering.

The research is extended to the use case of Pultrusion Process for manufacturing composite material in ACCIONA R&D centre located in Madrid, Spain, in which material from creel racks are passing through a certain sequence until desired productions are finalized. The system could exemplify SoS from the horizon of heterogeneity, since a social system (human operators) is integrated to the large-scale technical systems. The system was analyzed according to the generic method based on which the whole production process can be described.

The results of the thesis show 1) a generic SoS modelling method for the domain of factory automation is introduced; 2) the applicability of the generic method to a system with distributed control where every module can be considered as a subsystem; and 3) the applicability of the generic method to a complex manufacturing system.

Contents

List of figures

List of tables

Abbreviations and notation

A	refers to Actions in the generic modelling method
C	Conveyor in testbed
CC	Cross Conveyor in testbed
CSD	Constituent System Description in the generic modelling method
DEVS	Discrete EVent Systems specification
DEVSJAVA	A java-based tool for simulation of discrete event systems specification
DD	Dynamic Description for a constituent system utilized in the generic SoS modelling method
EB	End Block in testbed
EC Formalism	Event Calculus Formalism
G	refers to Goals in the generic modelling method
L	Lifter in testbed
LL	Lower Line in testbed
MAS	Multi-Agent Systems
MB	Middle Block in testbed
RFID	Radio Frequency IDentification
SB	Start Block in testbed
SD	Static Description for a constituent system utilized in the generic SoS modelling method
SoS	System of Systems
SMS	State Modelling Software
UL	Upper Line in testbed
XML	eXtensible Makeup Language

1 INTRODUCTION

Manufacturing systems are facing new challenges due to customized production, shortened production life cycle and frequent process reengineering [1]. In order to solve these problems, an increasing number of new technologies have been integrated to traditional manufacturing systems, such as Internet based communication and information exchange, open source operating system and others [2]. A proliferation of this kind of integration for hardware, software and network systems has enhanced the complexity of architecture and communication for manufacturing systems, and lead to a new kind of system – System of System (SoS) [3] - [5]. SoS are differentiated with very large, complex, but monolithic systems in several properties, which were first introduced in [4] and they are stated as follow:

1) Operational independence of the constituent systems;
2) Managerial independence of the constituent systems;
3) Geographic distribution;
4) Emergent behaviours;
5) Evolutionary and adaptive development.

System modelling is a technique to express, visualise, analyse and transform the architecture of a system. It is essential for system engineering especially for SoS engineering. In order to capture the specific SoS properties presented above, the modelling methods for traditional systems in factory automation domain should be modified to satisfy the up-to-date need in SoS engineering. It is assumed that computational modelling method for SoS can increase reliability and meantime decrease development cost facilitating the construction.

The thesis has two objectives: 1) to introduce a computational method for SoS modelling; and 2) to show the applicability of the method to the domain of factory automation.

1.1 Background

Up till now, there is no commonly accepted definition for SoS and the SoS concept is still at its developing stage [6]. Collections of exiting definitions have been presented in

[7] - [9]. Definition of SoS utilized in the thesis was firstly proposed by Jamshidi: "System of systems is a large-scale integrated systems that are heterogeneous and independently operable on their own, but are networked together for a common goal" [10], as far as it presents the majority of SoS specific properties with convincing generalization.

Plus to the properties which differentiate SoS from traditional systems introduced in [4], Boardman and Sauser emphasized five characteristics related to the ability of constituent systems to fulfil their goals. These five characteristics are: Autonomy, Belonging, Connectivity, Diversity and Emergence [11]. The differences between each component system can be observed by measuring these characteristics with a computation method as introduced in [12].

A progress was achieved through numbers of publications which were dedicated to SoS modelling and analysis (see for example [3], [13] - [15]). The main thrust behind is to obtain more capabilities and better performance than would be possible with traditional systems approaches [6]. Despite of the great interest to the topic on academy and industry [10], the research on computational modelling method is in its early stage and no application to factory automation was proposed yet [16]. In order to capture the specific SoS properties presented above, modelling methods for traditional systems should be modified.

Assuming that SoS can be decomposed to elements which are traditional system themselves, Clark [17] proposed traditional system modelling methods such as Building Block and V-Model to model SoS. However, this approach is ignoring essential characteristics of SoS like communication and cooperation between component systems, and this appeals to develop a more SoS-specific methodology.

In order to treat SoS separately and differently with traditional systems, a new modelling method was introduced in [18]. The method employs taxonomy and related lexicon which describes the properties and connection between constituent systems specifically in a SoS level. As a drawback, the method needs hierarchy description of the target systems, lacks readability in a current stage and needs a further generalization.

In [12], characteristics of SoS are modelled by a computational method, providing a new perspective which allows to directly observe the properties of SoS components. The interfaces between components in SoS are described through actions [19].

In military domain, modelling methods such as State Modelling Method [14] and DEVS Modelling Language [6] were applied in military system and threat detection system. However, the utility of the two approaches to manufacturing system modelling is questionable. Furthermore, there is a lack of general computational methods for the factory automation (FA) domain.

In order to model the whole production process in FA domain, the thesis proposes a generic SoS modelling method based on system characteristics and description of interface activities which were first introduced in [12] and [19] respectively. Combing these two methods, it is possible to describe the characteristics of each component system and the interface activities between component systems simultaneously. Thus the whole production process can be modelled in the perspective of SoS with better approximation compared to the SoS modelling methods proposed already.

1.2 Objectives

Two objectives of this thesis are 1) to develop a generic SoS modelling method that can be utilized in FA domain; and 2) to evaluate the applicability of the modelling method with two use cases, which are a testbed of a conveyer based production line in FAST lab (Tampere University of Technology, Finland) and a composite material manufacturing system (ACCIONA R&D centre in Madrid, Spain).

1.3 The Generic SoS Modelling Method

The generic SoS modelling method is developed basing on two SoS-specific modelling methods which were described in [12] and [19]. According to the paradoxes of SoS engineering (see chapter 2), components in SoS can be seen as a whole and as a part of a superior system simultaneously, and thus in order to reflect both situations, two different concepts are introduced: Static Description (SD) and Dynamic Description (DD). The former concept treats each component system as a whole while the latter concept focuses on cooperation between constituent systems in a SoS.

The modelling method by definition of SoS characteristics is for the *description of properties for each component* [12] (see section 2.4.2.3) and the modelling method by definition of SoS interface is for *the communication* between components [19] (see section 2.4.2.4). In order to provide a more completed model of the whole production process, these two modelling methods are integrated to the generic method.

1.4 Use Case Studies

To evaluate the applicability of the generic SoS modelling method to manufacturing systems in FA domain, two use cases are studied in the thesis.

The first use case is a testbed, which is a conveyer based manufacturing line for assembling electronic devices. Although the testbed cannot be seen as an application of SoS, it was selected for the first phase of the study with two reasons: 1) the modelling method for SoS should also be compatibly working for SoS-parts including the traditional systems; 2) the selected system can be seen as a multi-agent system (MAS), and due to operational independence of agents and decentralized control in MAS (both are important features of SoS), the testbed can be used as a simplified model for the method application.

In order to evaluate the applicability of the approach to a module manufacturing system in the FA domain, the use case was extended to Pultrusion Process, a composite material manufacturing system located in ACCIONA R&D workshop in Madrid, Spain. Various materials from steel racks are treated in a certain sequence consisting of several processes: Resin Impregnation, Heated Die, Pulling and Cutoff so as to produce desired final production. Cooperation and communication exist amongst constituent systems in order to make the whole system function normally. It is a more complex application with a greater number of the constituent system's types and more various communications inside the system. The Pultrusion Process can be considered as a SoS application because of its heterogeneity: it is an integration of technical systems and a social system (human operators).

1.5 Thesis Outline

Thesis is organized as follows: Chapter 2 provides a literature review on concept and characteristics of SoS, differences between traditional system engineering and SoS engineering, and approaches for SoS modelling. Chapter 3 introduces the generic modelling method. Chapter 4 and Chapter 5 are dedicated to two use cases and results obtained from the application of the generic modelling method. Chapter 6 wraps up the thesis with conclusions and recommendations for future developments.

2 LITERATURE AND TECHNOLOGY REVIEW

This chapter revises related works and consists of three parts. First part is dedicated to the concept of SoS and its characteristics which differentiate SoS with traditional systems. Second part presents the requirements posed by SoS characteristics in SoS engineering compared with traditional system engineering. A review of some existing approaches and methods for modelling of SoS which are potentially applicable to FA domain is introduced in the last part of this chapter.

2.1 The Concept of SoS

The SoS engineering perspective was presented in 1991 as an approach to solve the problems related to architecting and managing complex systems [3], [5]. Since then, the term was adopted and discussed in a number of fields such as transportation [20], [21], combat missions [22], health care [23], electric power grids [24], swarm robotics [25], and others. Although the SoS concept has been used in many areas, yet no commonly accepted definition for the term was yet introduced [8]. Several reviews and collections of definitions were published in recent decade [8], [26].

Some existing definitions highlight the cooperative behaviours, role of interaction and networking between constituent systems (usually it is also assumed the systems are complex) in a way of achieving common goals [7], [9], [27]. Another commonly shared point is the geographical distribution of SoS [8], [10], [28], [29]. Some definitions stress interdisciplinary nature and impact of SoS [28], [30]. Then SoS might be defined through wanted feasibility such as “interoperability and synergism of command, control, computers, communications, information and intelligence, surveillance, and reconnaissance” [31].

Five principles that distinguish SoS from very large, complex, but monolithic systems were firstly introduced by Maier (1996) in [4]:

1) Operational independence of the constituent systems;
2) Managerial independence of the constituent systems;
3) Geographic distribution;

4) Emergent behaviours;
5) Evolutionary and adaptive development.

Principle 2-5 distinguish among SoS and Multi-Agent Systems (MAS) which are complex systems composed of autonomous agents who, while operating on local knowledge and possessing only limited abilities, are nonetheless capable of enacting the desired global behaviours [32]. However, MAS approach can benefit to SoS engineering, for instance to avoid hierarchical analysis which becomes inefficient in modelling when the target system is complex.

This thesis utilizes the following definition of SoS: systems of systems are large-scale integrated systems that are heterogeneous and independently operable on their own, but are networked together for a common goal [10]. Because it is assumed that the definition explicitly or implicitly summarizes the majority of the SoS essentials mentioned in previous works, and thus it is accepted in this paper as the working definition.

2.2 The Characteristics of SoS

Boardman and Sauser (2006) described five characteristics that differentiate SoS with traditional systems in [11] and they are: Autonomy, Belonging, Connectivity, Diversity and Emergence. This section concentrates on explanations and meanings of the five characteristics.

2.2.1 Autonomy

For a SoS, each constituent system belonging to it should be entitled with Autonomy for two reasons. First, each constituent system should have its right to pursue motive for being an individual system, having operational and managerial independence. Another reason is Autonomy helps to achieve the global goals through different behaviours performed by the constituent systems [33]. Though the constituent systems have the property of Autonomy, there should be constrains for this quality which would keep the SoS linked. However, these constrains should not overwhelm the nature of constituent systems to perform [11]. Here, the nature means the reason why the constituent system exists, and “constrain” was exemplified with a relationship between an automobile and its brake [11]. Autonomy is refereeing to the managerial and operational independence of constituent systems and this characteristic promotes an individual performance of a component system with aim to contributing to the global goals of SoS.

2.2.2 Belonging

In traditional systems, component systems cannot choose whether they belong to the system or not as long as they have been integrated together. The reason is that without any of the parts, the system will not function at all. Thus the parts have no reason for existence if they do not belong to a system, and belonging is one of their natures [33]. In SoS, the property of Belonging is dynamic in corresponding to the holistic goals. The ground for Belonging of a constituent system is the costs or benefits those the system may bring to the whole SoS [11].

In SoS, Belonging refers to the ability of a constituent system to extend its goals to the holistic goals of the whole SoS or of a network of constituent systems. The ability could be considered from two different perspectives. Firstly, from the perspective of the whole SoS, when the global goals have to be completed through cooperation between constituent systems, some of them have to change their belonging to accomplish the mission with respects to benefits expected. Secondly, for a component system, under the condition that individual goals cannot be completed by the system alone through the autonomy, some new connections and cooperation may be formed within SoS. Thus, Belonging can be considered as “shared mission” [12], and, in this paper, Belonging is evaluated by the percentage of contribution to the global goals.

2.2.3 Connectivity

Connectivity, widely discussed with interoperability, is determined in the design phase in traditional system engineering. In SoS Connectivity may not be predesigned, yet the constituent systems are expected to obtain communication and interoperability by supporting interactions between them [33]. From this perspective, a language between constituent systems in SoS has great importance and should support the efficient communication and interaction. As discussed above, Belonging of a constituent system may vary with respect to the goal, and thus the Connectivity may also be dynamic. These two characteristics are tightly intertwined together because the connections will not be formed if the constituent systems do not belong [34]. In relation to Connectivity of SoS, the crucial role of Autonomy is to perform the task regardless to the status of the connection [11]. In SoS the Connectivity can be considered as the capability to form connections between components to accomplish common goals.

2.2.4 Diversity

Diversity is the noticeable heterogeneity, having distinct or unlike elements or qualities in a group [35]; example is the variation of social and cultural identities among people existing together in an operational setting.

In traditional systems, the functions and capabilities of a system are decided in the design phase, while in SoS, the constituent systems may be not supposed to work together before the formation of SoS. Even after formation, the function of SoS is more than a simple summation of constituent systems due to emerging connections, communication and interoperations. Comparing Diversity and Autonomy, Diversity refers to the abilities of a system to perform different actions while Autonomy includes actions needed to fulfil the holistic missions [12].

2.2.5 Emergence

Emergence is the appearance of new properties in the course of development or evolution [33]. For a traditional system, emergence is intentionally and deliberately designed and thus, the bad behaviours can be tested out [35]. While the emergence of a SoS cannot be foreseen through analysis because it comes from the collaborations and Autonomy of constituent systems. From this view, it is a feedback from the characteristics of Autonomy and Diversity, which can be utilized in addressing the degree of these two characteristics [34]. Furthermore, it is possible to detect or even eliminate unwanted behaviours by modifications of characteristics before the real formation of a SoS [13].

2.3 SoS Engineering and Traditional System Engineering

In this section, requirements for SoS engineering are discussed in comparison with the traditional system engineering. A traditional system is defined as a collection of things or elements working together which, produces a result not achievable by the things alone [8]. The definition of SoS has been introduced in section 2.1.

The SoS characteristics discussed in the previous section transform the traditional system qualities to the qualities of SoS in a paradoxical way [8]. The paradoxes are summarized in the table 2-1 as follow:

Table 2-1. Paradoxes of SoS [8]:

Traditional system		**SoS**
Conformance	Autonomy ⟷	Independence
Centralization	Belonging ⟷	Decentralization
Platform-centric	Connectivity ⟷	Network-centric
Homogeneous	Diversity ⟷	Heterogeneous
Foreseen	Emergence ⟷	Indeterminable

Addressing to theoretical and practical problems in engineering, these paradoxes reveal the differences between perspectives of the traditional systems engineering and SoS engineering [8]. A summary of the differences is given in table 2-2:

Table 2-2. Traditional System Engineering vs SoS Engineering [16]:

Critical Point	**System Engineering**	**SoS Engineering**
Focus of Analysis	Single System	Integration of Systems
Focus of Improvement	Optimization	Realistic Cost and Scheduling
Target	End Product	Initial Deployment
System Requirements	Fixed	Evolving
System Boundaries	Well-defined	Indefinable

The differences between traditional systems and SoS, the SoS paradoxes and specific characteristics appeal for complex approach to SoS engineering. In addition to the system engineering knowledge, SoS engineering of different application domains could get value from other disciplines like operational analysis, decision analysis, modelling and simulation, value engineering, cognitive modelling, and theory of collaboration [36]. The applicability of Artificial Life approach was shown for analysis of financial markets as to an example of self-organizing systems [37].

Systems which are too simple are static and those that are too active are chaotic, thus only on the edge between these two extremes can a system undertake productive, and thus the reasonable balance between traditional system engineering and SoS engineering

should be found in order to capture the specific nature of SoS without losing the practical perspective [38].

A methodology to manage and to engineer SoS from the 'Holarchical view' was proposed based on the five characteristics of SoS [39]. Holarchy was created to describe a hierarchical structure of holons, and the word "holon" means both to be a whole and at same time to be a part of the superior system [39]. This concept is similar with the role played by constituent systems in a SoS. Using 'Holarchical view' in SoS, it means engineering, managing and observing a SoS through a dualism perspective: traditional system perspective and SoS perspective. Earlier the holon dynamics was proposed to assist the model formulation for a Port Security System considering benefits and limitations of multi-methodological approach to SoS engineering [40].

Concluding this section, while engineering SoS, the key is to discover the 'boundary' in each characteristic through a dualism perspective. At this boundary, SoS is both relatively stable and productive, while losing the balance would lead to inefficiency or instability.

2.4 SoS Modelling Methods

Amongst the available SoS modelling methods proposed already (such as [3], [13]-[15]), there are coexisting two polar opinions on SoS modelling, one is advocating system engineering standards and guides as they are necessary and sufficient for SoS modelling [17], while another is reckoning that traditional approaches are not enough and there should be developed some new appropriate methods [12], [18], [24].

2.4.1 Modelling SoS Using Traditional System Modelling Methods

In this section, two modelling methods from traditional system engineering perspective are presented as follow.

If SoS can be seen as an integration of multiple individual parts which have the right to choose whether they belong to the membership of common domain or not, then SoS could be decomposed to the elements which are traditional systems, and then the traditional system engineering principles could be applied [17]. Thus system engineering standards and guides would provide engineers with a complete process for modelling a SoS and there is no need developing new modelling principles. In this subsection, two

methods are presented in order to illustrate how SoS could be modelled using traditional system engineering method: Building Block and V-Model.

2.4.1.1 Building Block Method

The Building Block [17] of a traditional system is comprised of three classes: Product, Process and People. Each of three classes could be further divided to subclasses: Sub-Product, SubProcess and SubPeople respectively as illustrated in the figure 2.1.

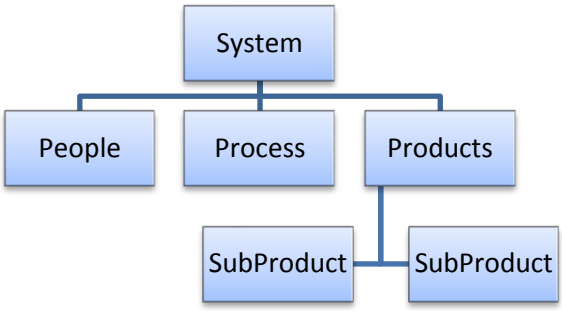

Figure 2.1. Building block of traditional systems [17].

Using Building Block method, a SoS could be decomposed to subsystems until every subsystem is an individual traditional system, and then the methods in system engineering can be applied. This process is presented in figure 2.2.

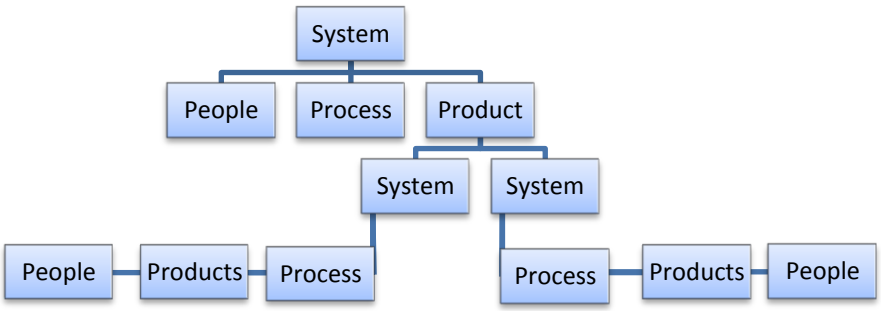

Figure 2.2. Building Block of SoS [17].

2.4.1.2 V-Model Method

V-Model (see Fig. 2.3) describes nearly the whole system life cycle [17] and has got its name because of "V"-like visual representation of the consecutive steps in process of

system modelling. System requirements are listed on the left side of the "V", which is called *specification stream*. The validation process is listed on the right side of "V" which is *testing stream*. Modelling of every system starts with the requirement process and ends at validation process, forming a life cycle. In modelling SoS, this method can be used to each subsystem until every subsystem is validated according to its requirements.

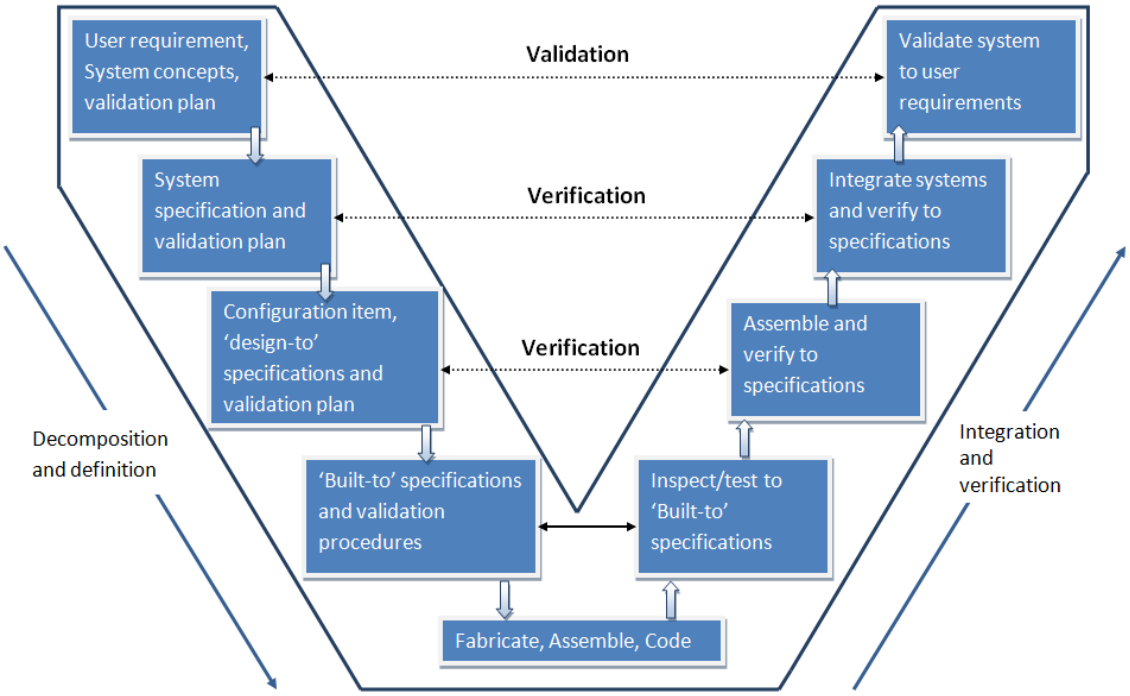

Figure 2.3. Original V-Model [17].

As a conclusion for this section, two traditional system modelling methods discussed above have clear representations and seem handy for practical execution. However, essential characteristics of SoS such as communication and cooperation between component systems are ignored, which appeals to development of more SoS-specific methodologies.

2.4.2 Modelling Methods Specific for SoS

In this section, several methods that created specifically for SoS are introduced. These methods concentrate on the properties possessed by SoS, which have been introduced in chapter 1 already.

2.4.2.1 SoS State Modelling Method

Sandia National Laboratories presented a methodology for developing a SoS model, defining state models and simulating a system of state models over time [14]. In [14], it is believed that the method of state modelling provides a more convenient way to model SoS. The modelling and simulation are executed in State Modelling Software (SMS) which are components of SyOp, a Sandia developed systems analysis and optimization tool. The principle of the state modelling method is presented as follows.

In this method, statecharts were used in specifying real systems and it works well for modelling reactive systems based on the structured analysis paradigm [42]. *State* can be seen as an assigned value for each attribute of an object. Then all objects have state. Statechart shows a network of state and event in which there are several notations that are explained as follow.

Transitions are enabled when the state is "on" and disabled otherwise, and every transition has an *event* that acts as a trigger and meantime it may have a guard. The function of a trigger is to activate a transition while the guard is to allow the transition. When a transition is enabled, it fires if the trigger event occurs and its guard is true. *Events* are occurrences of stimuli that can trigger an object to change its state.

This method can be utilized to the systems whose status and functions can be determined based on which states are occupied. The transition of the states can be triggered by the occurrence of the failure events associated with the state.

The basics of statecharts should include [14]:

1) A state model contains a hierarchical system of states.
2) There is one root state which does not have a parent state.
3) The user must define a subset of the states as initial states.
4) The system transitions from one set of states to the next set one step at a time. This is exemplified with the figure 2.4 shown as follows. The source state for the transition X is State S and the destination state is State D. G and T represent guard and trigger respectively. When the State S is occupied by this system and the trigger T and the guard G are true simultaneously, the transition will occur, leading to a new State D.

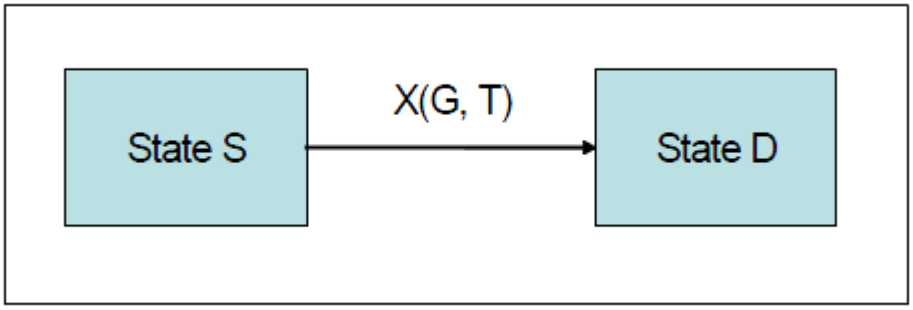

Figure 2.4. Illustration of transition from State S to State D [14].

If at some step, a system occupies state S, the trigger T is true, and the guard G is true; then the system will transition from state S to state D. Both of them have to be true to guarantee the state transition.

The states for each component in a SoS are represented according to the structure, and meantime initial states are set. SMS then finds each possible path from the initial states to the goal state and a path which consists of the sequence states must be passed through [15]. Meantime the corresponding set of transitions must fire along the path according to the state change. Here the event of interest refers to the events that cause the trigger to be true and they become variables in the Boolean expression describing the set of events that have to occur for achieving the goal state. The Boolean expression is converted to an algebraic expression for quantifying performance measures for the state model [15].

As mentioned already, this method can be used to represent the systems whose functionality can be described by the states of the system's elements. Sandia National Laboratories developed this program to enhance an integrated modelling and simulation environment that addressed complex modelling and analysis needs of SoS. This approach has been applied to a Future Combat Systems, which can be seen as a SoS, with programmatic success [14].

2.4.2.2 Modelling SoS by Employing Taxonomy And Related Lexicon

Revealing the SoS properties such as operational independence, a method by employing taxonomy and related lexicon was proposed to solve problems in designing and modelling of SoS [18]. It is assumed that various types of SoS need the type-specific solutions. A taxonomy based on the distinction between families-of-systems was introduced

in order to facilitate the SoS-type identification which is shown as follows in the figure 2.5.

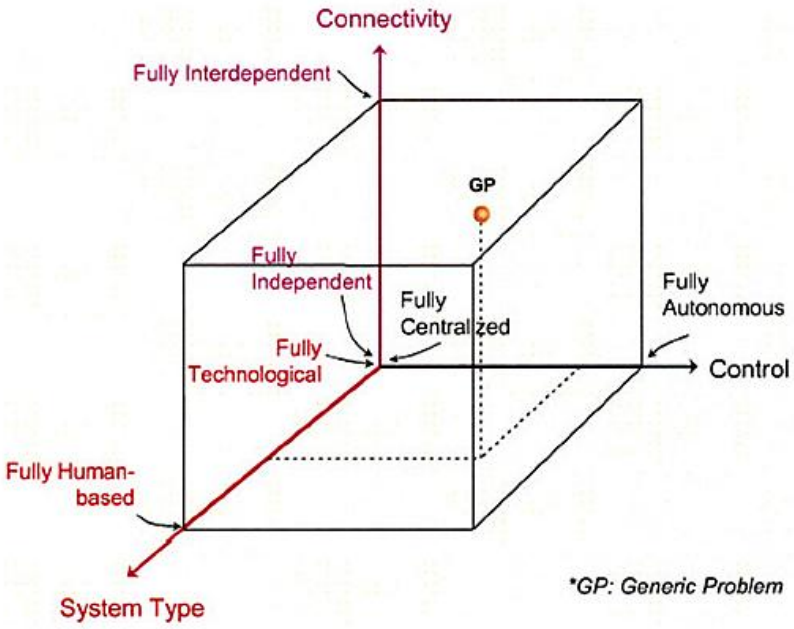

Figure 2.5. Taxonomy dimension of SoS [18].

The taxonomy has three dimensions:

- *System Type*, which shows the dominant factor in the composition of SoS. For instance, if the main components in a SoS are mechanical (hardware) or computational (software) artefacts, this SoS is closer to technological pole, and if the main components are human beings as operators, service provider or service consumers, this SoS is closer to human-based pole.
- *Degree of Control*, this dimension is related to the properties of operational and managerial independence.
- *Connectivity*, which refers to the interactions between constituent systems within a SoS.

After the SoS-type has defined with the taxonomy, the domain lexicon should be also considered. The lexicon contains four categories of systems (*Resources* – physical entities, *Stakeholders* – non-physical entities, *Operations* directing physical or non-physical entities, and *Politics* which are external functions having impact to previous three categories) and hierarchy of system organization (systems of the lowest order α are networking to a system β, network of systems β produces a system γ, and so on) [18].

Important is that the physical manifestation may be most obvious at the α-level, but the behaviour of the SoS is dominated by the structure and organization at higher levels [18]. The most important design problem expected to be solved in SoS engineering seems to be related to this point to a great extent: how preferred or observed behaviours of a SoS affect the entities in the lower lever including its connectivity, autonomy etc.

In [18], the dynamic connectivity is exemplified using the lexicon given above (see Fig. 2.6). As can been seen, after a certain time interval, system α_3 is replaced by system α_4 in the connection with the systems α_1 and α_2 in the level of upper system β_1; and latter the system α_4 is replaced with α_5. Meantime, not only the components have changed but also the format of the connections between them.

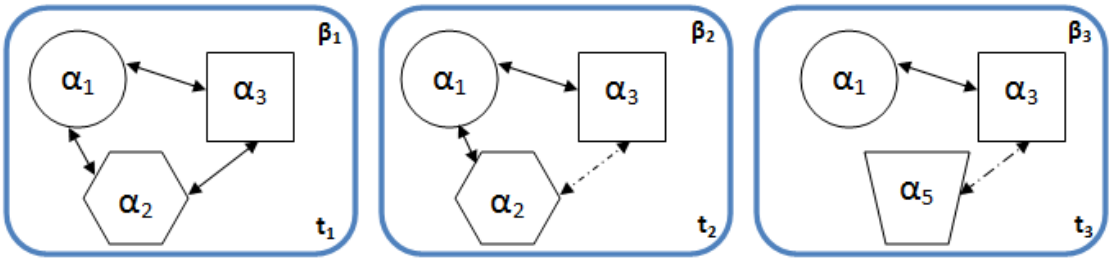

Figure 2.6. The expression of SoS dynamic connectivity [18].

This modelling method has been used in three study cases: shared-autonomy in automotive swarm, air transportation architectures and space transportation architectures. However, because only taxonomy and related lexicon are introduced, more effort should be done to make this method more generic, detailed and readable. From this perspective, language description for modelling SoS is a preferred direction.

2.4.2.3 Modelling SoS Based On Characteristics

Another approach for modelling SoS is based on specifics of SoS's characteristics introduced in [12]. The characteristics differing SoS with traditional systems are Autonomy, Belonging, Connectivity, Diversity, and Emergence whose meanings have already been discussed above. A SoS can be considered as a constitution of several elements which are traditional systems or SoS. Baldwin and Sauser (2009) defined each element in the SoS as [12]:

$$S_i = \{A_i, G_i, E_i\}, G_i \neq \emptyset, i \in Z+ \quad (2\text{-}1)$$

In which A_i represents different actions that can be executed by the element S_i, and a number of goals for S_i are symbolized by G_i. E_i stands for the expression of the charac-

teristic *Emergence*. *Diversity* is the characteristic that ensures different actions can be undertaken by this system for different goals, which indicates that there are multiple components in A_i and there are multiple sets of A_i for different goals.

Thus, Autonomy = $|A_i|$ (2-2)

Diversity $\equiv A_i \neq \emptyset \wedge |A_i| > 1, i \in Z+ > 1$ (2-3)

A SoS, being a type of system, is denoted by S* here.

$S^* = \{S_1, \ldots S_n, G^*\}, n \in Z+ > 1, G^* \neq \emptyset$ (2-4)

The *Belonging* is referring to subsystems ability to involved or not to SoS and it is evaluated on cost or benefit basis in the global goals. Baldwin and Sauser (2009) considered it as the percentage of contribution it has to do in the global goals [12]. As a result, the definition of Belonging can be expressed as:

$$\text{Belonging} \equiv \frac{|f(A_i) \cap G*|}{|A_i|} \geq \beta_i > 0 \quad (2\text{-}5)$$

For *Connectivity*, it is assumed that connection takes place in two situations:

- the connectors for two elements are disjoint, or
- the computation for the characteristic Belonging of either of the two connected systems is under threshold.

C_i and C_j are both subsets of static connectors for two constituent systems. These two systems must have a common connector in order to be connected. This can be represented as follow:

$$\text{Connectivity}_{i,j} = \begin{cases} \text{true: } \exists k < c_k, c_k > \in C_i \times C_j \\ \text{false: } \vee k < c_k, c_k > \notin C_i \times C_j \end{cases} \quad (2\text{-}6)$$

No expression for Emergence was introduced, but in the real implementation the model of SoS should inherently possess this one.

2.4.2.4 Modelling SoS by Definition of Interface

In [19], there was introduced a method to model SoS by defining descriptive interfaces between constituent systems which contain the obligations that should be complied. A SoS is depicted as a multi-agent system, and each agent is capable of releasing a certain

number of obligations. These obligations are accomplished either by actions taken by the agent itself, or by commanding other agent(s) to complete some certain actions.

The *Obligation* is moral impositions to oneself to reach a state of affairs, demanded by a social force [19]. An obligation can be represented as $O(agt_1:S, f)$, which means that the state of affairs f is reached by the agent agt_1 who belongs to S through obligations. Reversely, the agent agt_2 of system S which is prohibited to reach the state of affairs f through obligations, can be symbolized as $O(agt_2:S, \neg f)$. If all the agents in the system S are expected to attain the state of affairs f, O(S, f) can represent this process.

The next important concept introduced in [19] is Event Calculus (EC) formalism, which contains three elements: an action a(), a fluent f that stands for a property value influenced by the action a() at a certain point of time t. Examples of formulas and their meanings are presented in the table 2-3 and they are used for creating obligations, releasing obligations, and definition of SoS.

Table 2-3. EC Formalism [19]:

Formula	Meaning
Initiates(a(), f, t)	Fluent f holds after the execution of a() at time t
Terminates(a(), f, t)	Fluent f terminates after execution of the action a() at time t
HoldsAt(f, t)	Fluent f holds at time t
Happens(a(), t)	action a() is executed at time t

A. Creating Obligations

The creation of an obligation to reach a state of affairs f, by executing an action a() at time point t, is shown as follow:

$$CreateO(a(agt_1:S), O(agt_1:S,f), t) \quad (2\text{-}7)$$

Where a(agt:s) means that this action is performed by the agent agt_1 belonging to S; and $O(agt_1:S,f)$ denotes that the state of affairs f is reached by agt_1 when this action is taken place.

There is another way to create obligations, in which the action is commanded by the agent age_1 and taken by the agent agt_2, with the state of affairs f reached also by agt_2. This process can be defined:

$CreateO(a(agt_1{:}S), O(agt_2{:}S,f), t)$ (2-8)

B. Releasing Obligations

The releasing of an obligation needs to be executed when the state of affairs f has already been reached by the agent which is supposed to do so. This process can be represented as follow:

$ReleaseO(a(agt_1{:}S), O(agt_1{:}S,f), t)$ (2-9)

Similar to the creation of obligations discussed above, obligations can also be released by another agent:

$ReleaseO(a(agt_1{:}S), O(agt_2{:}S,f), t)$ (2-10)

C. Definition of SoS

In [19], SoS is defined as a 4-tuple (Agent-Community, Constituent-Systems, Interface, SoS-State). Where the *Agent-Community* is the set of agents that represent the interest of the stakeholder, for each system S ∈ *Constituent-Systems* and the *Interface* is the union of all the interaction contexts of the *Constituent-Systems*, and finally, *SoS-State* keeps a record of the interaction.

Summarily, the interactions between the constituent systems within a SoS are expressed on an obligation basis, which can be resolved by the EC formulism (see table 2-3) and SoS is defined using an agent-based method.

2.4.2.5 Modelling SoS by the Platform DEVSJAVA

In [6], an architecture modelling method and simulation software is developed based on discrete event systems specification (DEVS) simulation tools and Extensible Makeup Language (XML). Constituent systems in a SoS sends and receives data from each other and then makes decisions to fulfil their goals. In the real use case, different hardware and software systems may be utilized by these constituent systems. If they are not compatibly working, huge barrier will exist between constituent systems in data aggregation and fusion. To solve this problem, a new method aiming at creating a common language to describe data is presented in [6].

Each component system in a SoS is described as a model:

$M = X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda$, and t_a

Where X is a set of input values, S is a set of states and Y is a set of output values. δ_{int} represents internal transition function, δ_{ext} stands for external transition function and t_a is defined as the time advance function.

The model description then uses (or discards) the message in the event to do the computation and delivers an output message on the out-port and makes a state transition [6]. The messages between models are described in XML, containing the information about system properties, data properties, and data. Three different scenarios are studied in [6] using the platform of DEVSJAVA with huge progress.

As for concluding this chapter, the main characteristics and principles of SoS challenge system engineering with new requirements. Although there is a call for SoS-specific modelling methods, very few computational methods were introduced and the existing methods cannot guarantee their applicability in FA domain. Thus, there is a room for further research on SoS computational modelling method in a more dedicated and generic level. Next chapter presents the description of a generic SoS modelling method which is developed in this thesis.

3 THE GENERIC SOS MODELLING METHOD

As it was concluded in state-of-the-art, essential characteristics of SoS cannot be modelled with traditional methods. Since SoS deals with dynamic requirements, Connectivity and Belonging of constituent systems may also vary with respect to the requirements or global goals. The agent-based approach treats each constituent system equally and can benefit to SoS modelling. A language description method can make SoS modelling more understandable and efficient if each component system is able to identify and execute the language.

As mentioned in section 2.3, constituent systems in SoS can be treated as a whole and as a part simultaneously. In order to reflect these two situations, two concepts are introduced first: *Static Description (SD)* and *Dynamic Description (DD).*

SD describes the actions and goals which can be achieved by a component system alone no matter whether the system is a legal constituent system in a SoS or not. Properties related to SD are denoted with letter σ. Actions and goals are represented as $A\sigma_i$ and $G\sigma_i$ respectively, where i is the ordeal notation. For completing each $G\sigma_i$, there should be a set(s) of actions to perform. Consequently the SD for each component-system (S) can be presented as follows:

$$S\sigma_i = \{A\sigma_i, G\sigma_i\}\ i \in Z \tag{3-1}$$

This concept describes the essential properties of a component system.

The other concept is *DD* which is used when a system has become a legal constituent system in a SoS concerning networked nature presented in the SoS concept by Jamshidi [10]. The letter δ is utilized for expressing the properties related to DD. The DD for system i is denoted by $S\delta_i$. Similarly, there are two components in this description and they are Actions ($A\delta_i$) and goals ($G\delta_i$). The difference is that goals can only be achieved through cooperation with other constituent systems. This description focuses on communication and collaborations between the constituent systems in a SoS. The concept of DD for a system is similar to that in [12] and is presented as follows:

$$S\delta_i = \{A\delta_i, G\delta_i\}\ i \in Z \tag{3-2}$$

$A\delta_i$ refers to the actions the system i has to perform in order to fulfil the goals. $G\delta_i$ represents the goals that have to be obtained for this constituent system to achieve its functionality in the whole SoS. $G\delta_i$ is a function of $A\delta_i$, because goals are achieved through actions.

For a whole constituent system in a SoS, it has the properties included in SD and DD, and thus can be described as:

$$S_i = \{S\sigma_i, S\delta_i\}\ i \in Z \tag{3-3}$$

Though the following representation (see equation 3-4) is not sufficient for depicting Autonomy, it still can be seen as an indicator. The characteristic Diversity ensures that there is more than one element in $A\sigma_i$. For each component system, there is more than one set of $A\sigma$ that can be executed.

The computational methods for Autonomy and Diversity are similar as defined in [12]:

$$\text{Autonomy (i)} = |A\sigma_i| \tag{3-4}$$

$$\text{Diversity (i)} = A\sigma_i \tag{3-5}$$

For a SoS, the characteristic of Belonging can be seen as the ability for a constituent system to extent its goal to the holistic goal, and thus Belonging is considered as the ability to ‘share mission’. For measuring this characteristic, method is similar to that in [12] and is presented as follows:

$$\text{Belonging (i)} = \frac{|(G\sigma_i \cup G\delta_i) \cap G|}{|G\sigma_i \cup G\delta_i|} \tag{3-6}$$

The result of Belonging should be in the range [0, 1], and the closer to 1, the greater contribution the component system has done for the global goals.

Then the whole SoS is defined as:

$$SoS = \{S_i, \ldots S_n, C, G\},\ n \in Z+ > 1,\ G \neq \emptyset \tag{3-7}$$

In which $S_1, \ldots S_n$ refer to the description presented in (3-3), C stands for the connection situation for these systems, and G represents global goals. Standing for the characteristic Connectivity of a SoS, C is an n*n table for the system with n constituent systems. C is exemplified in Figure 3.1, which shows a 6*6 table for the characteristic of Connectivity for a SoS made up of 6 constituent systems. The upper triangular part of the table denotes the current situation for connection while the lower triangular part stands for the

minimum direct connection among these 6 components for completing their dynamic goals (see Fig. 3.1). This table plays an important role when displaying Connectivity using graph theory. The function for updating connect situation from time t to t' is shown as follows:

$$C_{t'} = \text{Connect}(C_t, t') \tag{3-8}$$

$C_{t'}$ represents the new table for Connectivity at a new time t', and C_t represents the old table in the previous time t. The table for Connectivity will be updated when the operation of (3-8) has run.

	$S\sigma_1$	$S\sigma_2$	$S\sigma_3$	$S\sigma_4$	$S\sigma_5$	$S\sigma_6$
$S\sigma_1$	0	1	0	0	0	0
$S\sigma_2$	1	0	1	0	1	0
$S\sigma_3$	1	1	0	0	1	0
$S\sigma_4$	0	1	1	0	0	0
$S\sigma_5$	0	0	1	1	0	0
$S\sigma_6$	1	1	0	0	1	0

Figure 3.1. A 6*6 table for displaying Connectivity.

The interfaces between component systems can be represented using similar expression like EC Formalism presented in [19]. Specifically the components of each independent action $A\sigma_i$, can be controlled as combinations of formulas such as CreateO and ReleaseO in order to represent the necessary steps for completing the actions.

The method described above can be depicted in the figure 3.2 as follows: each constituent system has actions and goals to complete, systems are connected with each other by actions or goals that should be achieved by cooperation which is related to the characteristic Connectivity and dynamic goals affect the characteristic Belonging. Using this method, constituent systems, their actions and goals, and the whole SoS are connected together to achieve a new model, which can describe SoS in a more detailed level and nearly the whole production process can be specified.

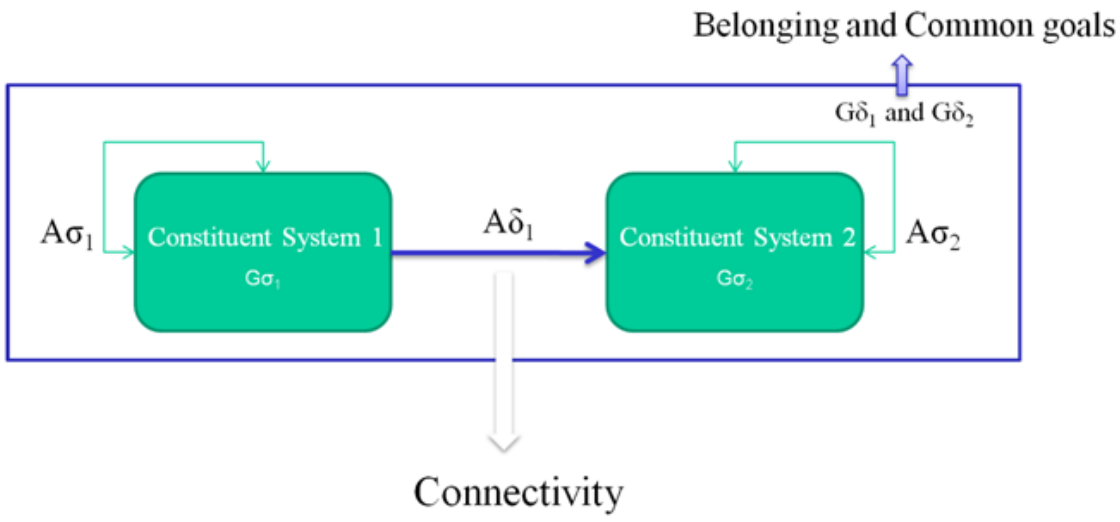

Figure 3.2. Frame work of the generic method.

Dynamic Connectivity Expressed by the Generic Method

As it was mentioned in review section for the characteristic Belonging which may be dynamic with respect to the actual goals, there is an opportunity to share the goals for the constituent systems connected. The process of the connectivity building and the meaning of the table for displaying connectivity (see Fig. 3.1) are exemplified as follows.

The figure 3.3 considers the situation similar to that which was described earlier in [18] (see Fig. 2.6). If a goal should be shared between $S\sigma_2$ and $S\sigma_1$, before the execution starts, $S\sigma_2$ should find a way to communicate with $S\sigma_1$. First, the lower triangular part in the figure 3.1, which stands for the maximum direct connection ability among the 6 components, should be inquired. Since $S\sigma_2$ and $S\sigma_1$ are connected directly, it is possible for them to complete a common goal if both of them are able to execute the actions requested by the goal (see Fig. 3.3-1). Then the position which shows the connectivity status for these two components in the upper triangular part in the figure 3.1 should be marked with 1.

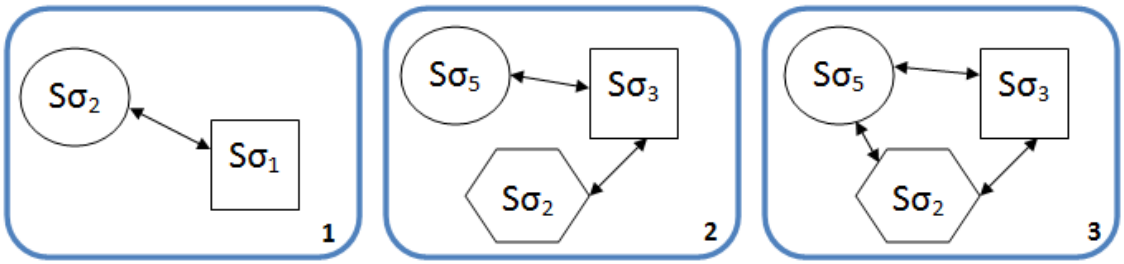

Figure 3.3. The depiction of connection changes: 1) connection situation at t_1; 2) connection situation at t_2; 3) connection situation at t_3.

If the two components that are supposed to achieve interoperability are not connected directly like $S\sigma_2$ and $S\sigma_5$ in the lower triangular of the table, then a new route should be searched out for their communication. Firstly $S\sigma_5$ searches in the lower triangular part of the table for all the components that are connected with it directly which are $S\sigma_3$, $S\sigma_4$ and $S\sigma_6$. Then same searching is done with the three components and the results for $S\sigma_3$ are $S\sigma_1$, $S\sigma_2$, $S\sigma_5$ and $S\sigma_6$. Then a new route has been discovered which are through $S\sigma_5$, $S\sigma_3$ to $S\sigma_2$ (see Fig. 3.3-2), making cooperation between $S\sigma_2$ and $S\sigma_5$ possible (see Fig. 3.3-3). Finally, the connection status for the route should be displayed in the upper triangular part of the table as long the systems have shared mission through this route. After running the operation in (3-8), the connectivity table updates according to the changes mentioned above.

Comparing to the taxonomy method reviewed in the section 2.5.2, the connection description proposed here does not require dividing SoS into several levels, which makes work simpler but meantime makes description more detailed. Much work should be done still to create a simple but effective language to enable the communication between different components.

As a conclusion for this chapter, the generic SoS modelling method is firstly introduced together with two new specific concepts which are SD and DD according to actions and goals. The computational method for SoS characteristics is also included. In the next two chapters, the generic modelling method is applied to two use cases which are testbed and Pultrusion process together with the results for evaluation of the applicability.

4 CASE STUDY 1: A CONVEYOR-BASED MANUFACTURING SYSTEM

This chapter is intended to present applicability of the generic SoS modelling method to testbed, a conveyor-based manufacturing system. Section 4.1 is dedicated to description of the testbed based on the generic modelling method, and section 4.2 presents the results obtained.

4.1 Testbed

This section concentrates on analyzing testbed according to the generic SoS modelling method. Testbed is a conveyor-based pallet transferring system for automatically assembling electronic devices. Although the testbed cannot be seen as a SoS, it is assumed that the modelling method for SoS should also be compatibly working for traditional systems because this type of transferring system has given up centralized control approach and become distributed where each module can be considered as a subsystem of a SoS nowadays. Thus the work presented has provided a good use case environment.

Testbed is presented in the figure 4.1. The system can be divided into three parts laterally: Start Block (SB), Middle Block (MB) and End Block (EB). Meantime two lines are built vertically and they are Upper Line (UL) and Lower Line (LL). Three kinds of conveyors are utilized in this system: Lifer (L), Conveyor (C) and Cross Conveyor (CC), and they are depicted in figure 4.2 with direction codes for transferring pallet internal and external the conveyors (or lifter). Some of these conveyors have the functionality of Radio Frequency Identification (RFID) to read the detail information on the pallet, as it can be seen in the figure 4.1 as follows:

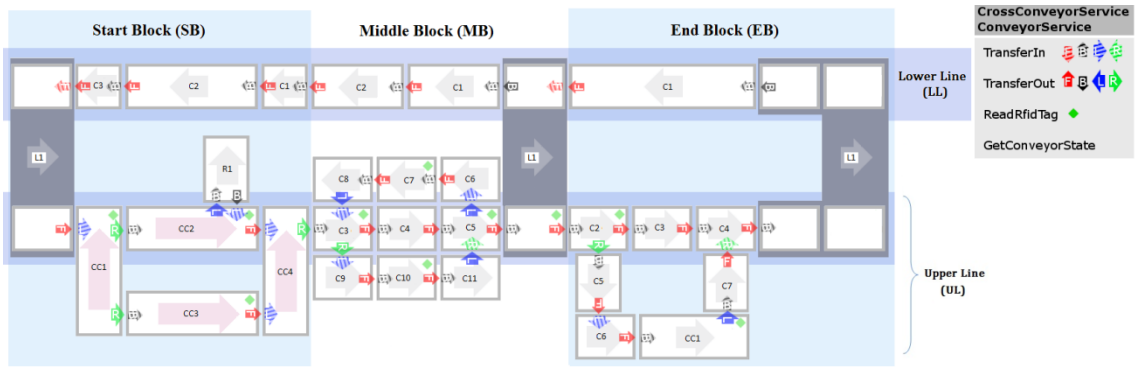

Figure 4.1. Testbed service description.

Before applying the generic method to testbed, the whole system should be configured first. The functionality of the configuration includes:

1. Check the working status (able or disable) to ensure each component conveyor is working normally;
2. Inform each component of the neighbour information (name of the neighbour, connection direction).

For each analyzed system, based on the characteristic Belonging and the availability of the cooperation between the component systems, there are SD and DD. Because testbed is a traditional system, the component systems cannot choose whether they belong to the whole system or not as long as they have been integrated together, consequently the quantification of the characteristic Belonging for each component system is 1 according to the equation 3-6. In other words, the conveyors cannot do anything else except fulfil the holistic goals of testbed.

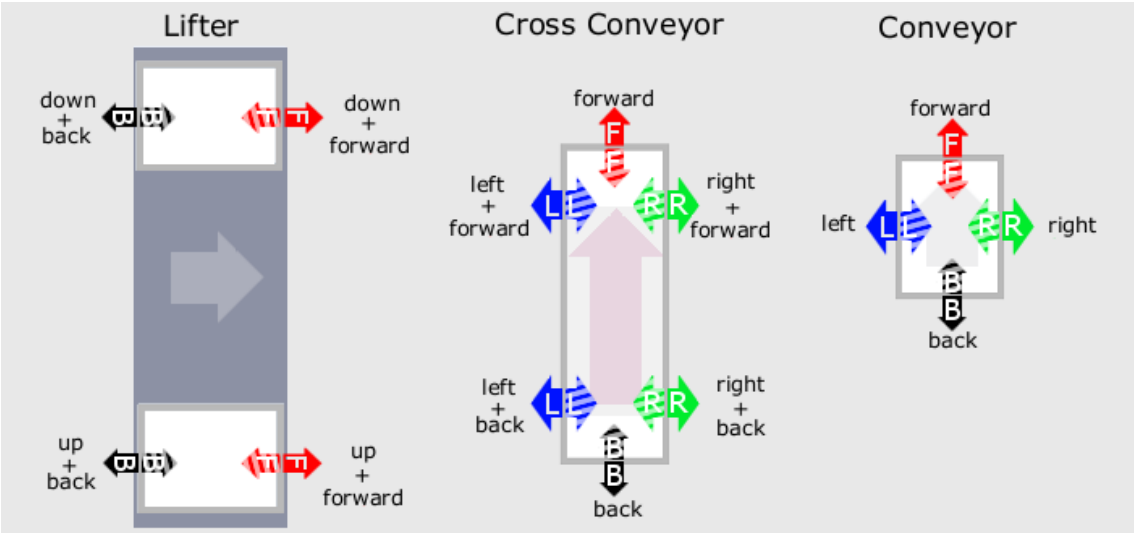

Figure 4.2. Three types of conveyors with the direction codes.

There are three types of conveyors as presented in the figure 4.2. For each of them, it is capable to implement several actions without collaboration with other conveyors which meantime is an element in SD. These actions are symbolized and specified in the table 4-1 as follow:

Table 4-1. Actions Can Be Implemented without Collaborations:

Action Symbol	Specific actions	Action Symbol	Specific actions
A_1	Transfer pallet forward	**A_3**	Transfer pallet up
A_2	Transfer pallet backward	**A_4**	Transfer pallet down

For each type of conveyors introduced in the figure 4.2, the actions that can be accomplished are presented as follow:

For Lifters: $A\sigma_L = \{A_3, A_4\}$ (4-1)

For Cross Conveyors: $A\sigma_{CC} = \{A_1, A_2\}$ (4-2)

For Conveyor: $A\sigma_C = \{A_1, A_2\}$ (4-3)

The goals ($G\sigma$) each component conveyor can realize without cooperation with other conveyors are through executing the actions in (3-9) (3-10) and (3-11). They can be expressed using the combination of direction codes as follows:

$$Block\_Conveyor\_G\sigma_i = \{DirectionCode_x, \ldots DirectionCode_z\} \quad (4\text{-}4)$$

These goals reveal the internal flow direction of a pallet within a conveyor (or a lifter) and can be accomplished by the actions introduced in the equation 4-1, 4-2 and 4-3.

As a result, the SD for each component can be expressed as:

$$S\sigma_i = \{A\sigma_i, G\sigma_i\} \quad (4\text{-}5)$$

For a conveyor (or a lifter), as long as its neighbours and neighbours' positions are fixed, the actions that can be executed through collaborations are fixed as well. As mentioned above, these conveyors are capable to transfer pallets in or out from different directions after operations. These actions are related with the component $A\delta_i$ in the DD and they are shown in the table 4-2 as follow:

Table 4-2. Actions Can Be Executed by Collaborations:

Action Symbol	Specific actions (TransferOut or TransferIn actions)	Direction Code (DC)
Ac_1	Transfer out to left	L (left)
Ac_2	Transfer in from left	
Ac_3	Transfer out to right	R (right)
Ac_4	Transfer in from right	
Ac_5	Transfer out to forward	F (forward)
Ac_6	Transfer in from forward	
Ac_7	Transfer out to back	B (back)
Ac_8	Transfer in from back	
Ac_9	Transfer out to up	U (up)
Ac_{10}	Transfer in from up	
Ac_{11}	Transfer out to down	D (down)
Ac_{12}	Transfer in from down	

For each type of conveyors introduced in the figure 4.2, the actions can be accomplished through cooperation are presented as follow:

For Lifters:

$$A\delta_L = \{Ac_1, Ac_2, Ac_3, Ac_4, Ac_9, Ac_{10}, Ac_{11}, Ac_{12}\} \quad (4\text{-}6)$$

For Cross Conveyors:

$$A\delta_{CC} = \{Ac_1, Ac_2, Ac_3, Ac_4, Ac_5, Ac_6, Ac_7, Ac_8\} \quad (4\text{-}7)$$

For Conveyors:

$$A\delta_C = \{Ac_1, Ac_2, Ac_3, Ac_4, Ac_5, Ac_6, Ac_7, Ac_8\} \quad (4\text{-}8)$$

Goals achieved by each component can be expressed as:

$$\text{Block\_Conveyor\_G}\delta_n = \{\text{Previous\_Conveyor\_Name, Next\_Conveyor\_Name}\} \quad (4\text{-}9)$$

Each goal is specified as a combination of two conveyors' names, which are the name of the previous conveyor from which the pallet was transferred in, and the name of the next conveyor to which the pallet will be transferred out. Goals can be achieved by a set (or sets) of actions presented in the table 4-1 and the table 4-2, and they reveal the flow direction of the pallet internal and external the conveyors. Assembling process can be done with the pallet during the flow. For each conveyor, there may be more than one

goals can be carried out, and these goals make up the component $G\delta_n$. Thus the DD for each conveyor system can be expressed as:

$$S\delta_i = \{A\delta_i, G\delta_i\} \quad (4\text{-}10)$$

The description for the whole conveyor contains SD and DD simultaneously and thus can be expressed as:

$$S_i = \{S\sigma_i, S\delta_i\} \quad (4\text{-}11)$$

For the whole testbed:

$$S = \{S_1, S_2 \dots S_n, C, G\} \quad (4\text{-}12)$$

In which S_1 to S_n represent all the conveyor systems that make up testbed, and C displays the characteristic of Connectivity. G stands for the global goals testbed are able to achieve, and the content of it are sets of recipes. In other word, G contains sets of conveyor names forming different routes to complete various production processes.

As for observing SoS characteristics for testbed, the computation for each characteristic was first introduced in [12], and chapter 3 has presented the computation approach for this generic modelling method.

For each goal the component conveyor can achieve, it consists of one set of TransferIn action and one set of TransferOut action. For each action that makes up the $A\sigma$, it should be accomplished by the cooperation of two adjacent conveyors. Besides modelling the actions and SoS characteristics for each component conveyor, it is necessary to model the interface for describing the whole production process as well. These actions can be executed only when the current conveyor has successfully ‘negotiated’ with the corresponding neighbour about the action. This process is depicted in the figure 4.3 as follows.

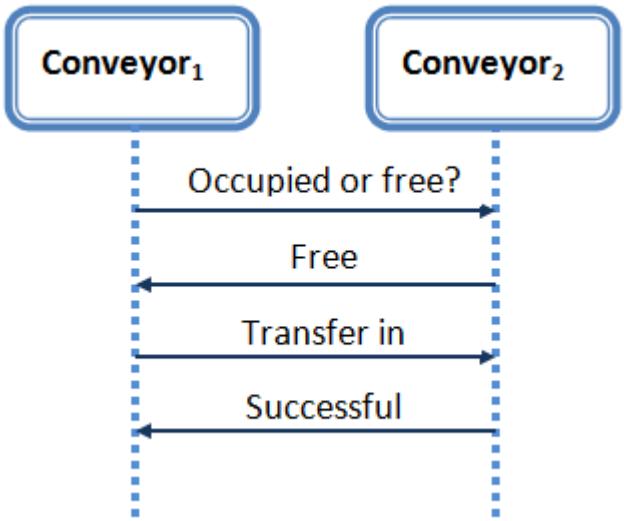

Figure 4.3. Interface description of conveyors when executing action.

As presented in the figure 4.3, the order for 'transfer in Conveyor$_2$' is made by Conveyor$_1$. The state 'Transfer in' indicates that the corresponding actions for a set of adjacent conveyors have started and the state 'transfer in successfully' indicates that this action has been accomplished by cooperation. This process can be modelled in the following approach which is modified according to [19].

$$\text{Start}(A_i{:}S_1, A_j{:}S_2, \text{state}_1) \quad (4\text{-}13)$$

$$\text{End}(A_i{:}S_1, A_j{:}S_2, \text{state}_2) \quad (4\text{-}14)$$

In which S_1 and S_2 represent Conveyor$_1$ and Conveyor$_2$, and A_i stands for the action 'Transfer out from Conveyor$_1$', A_j for 'Transfer in to Conveyor$_2$'. State$_1$ refers to the order 'transfer in' and State$_2$ refers to 'transfer in successful'.

As remarked above in this section, not only each conveyor in testbed is modelled using the generic method, but also the interface activities between conveyors are described. The result is illustrated with an example in the following section.

4.2 Application of the Generic SoS Modelling Method to Testbed

After configuration, each conveyor is ensured to 1) work normally; 2) acquire its SD; and 3) obtain the information about its neighbours which helps each conveyor in testbed to develop its own DD. This process can be depicted in the figure 4.4. After the whole

system has been modelled, the production process can be modelled in the following approach.

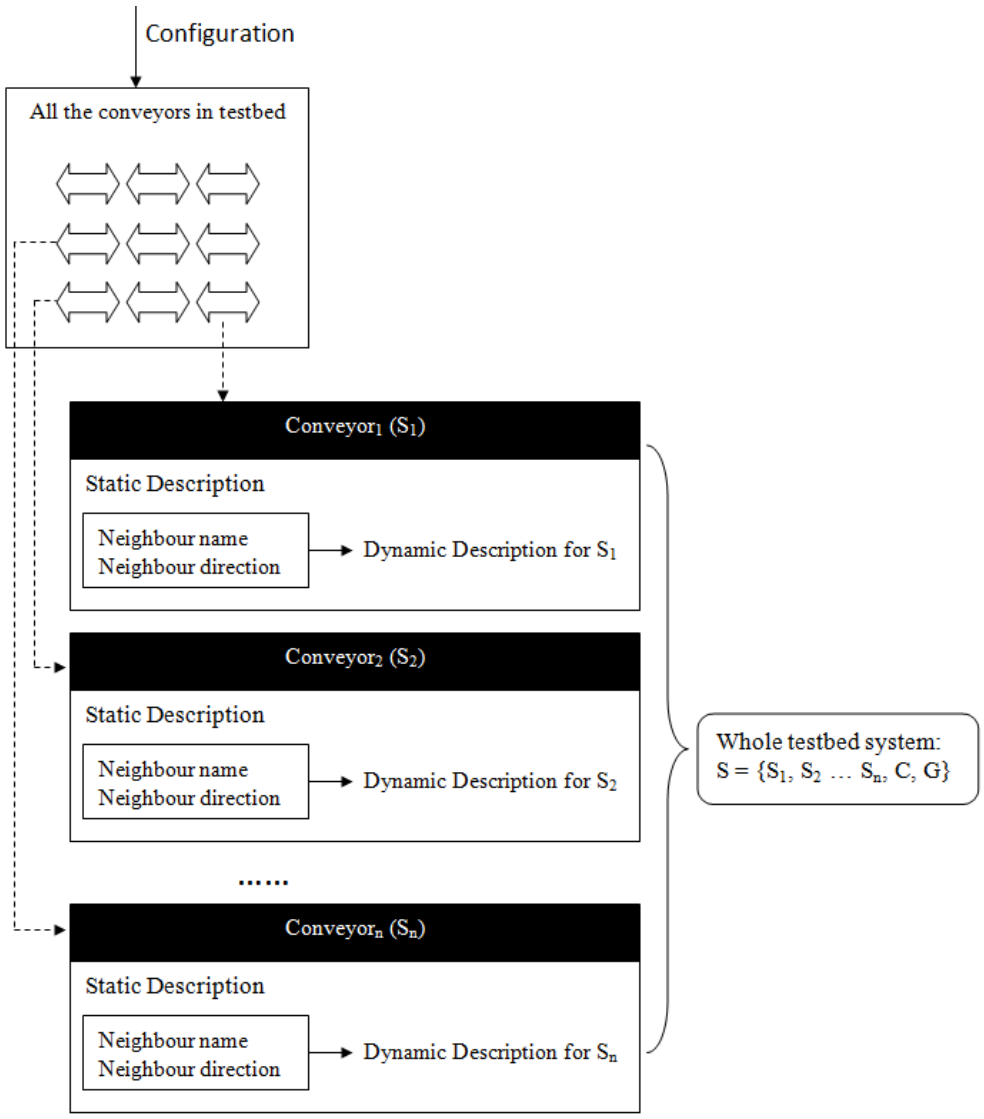

Figure 4.4. Configuration process for testbed.

As long as the global goal is fixed, which is a recipe containing the names of all the conveyors in a certain route, the starting conveyor receives the whole recipe and identifies the next conveyor and its direction in the neighbour list. Firstly the conveyor is instructed to execute the corresponding actions in SD according to the information, which includes the current direction of the pallet and the joint position of the next conveyor in the recipe. When pallet reaches the joint position, communication in the figure 4.3 is performed to transfer the pallet from one conveyor to another. Meantime the recipe is modified to delete the name of the past conveyors and transmitted to the next conveyor. In this way, the recipe each conveyor receives always starts from itself. Same process is

carried out for all the corresponding conveyors until there is only the name of the current conveyor on the recipe.

Next an example is presented to show the result of the application of the generic SoS modelling method to the testbed.

Example: transferring pallet from SB_L1 (see Fig. 4.1) in the lower line to MB_C3 (see Fig. 4.1). For each conveyor, after recognizing its own goal to achieve, corresponding actions will be executed. Interface activities between conveyors are also described.

For SB_L1, its goal is a sequence of conveyor names in the whole recipe of the process. The conveyor reacts according to the information on the following conveyor in the recipe (see A_3 as follows). After the pallet reaches the joint between two conveyors, interface activities start functioning (see Start and End function). Same procedure happens for all conveyors whose names are on the recipe, and through this approach, the whole production process can be described as follows:

- **For SB_L1:**

  G = {SB_L1, SB_CC1, SB_CC3, SB_CC4, MB_C3}

  A_3

  Start($(Ac_9+Ac_5):S_{SB\_L1}$), $(Ac_2+Ac_6):S_{SB\_CC1}$, $state_{StartTranferIn}$)

  End($(Ac_9+Ac_5):S_{SB\_L1}$), $(Ac_2+Ac_6):S_{SB\_CC1}$, $state_{TransferInSuccess}$)

- **For SB_CC1:**

  G = {SB_CC1, SB_CC3, SB_CC4, MB_C3}

  A_2

  Start($(Ac_3+Ac_7):S_{SB\_CC1}$, $Ac_8:S_{SB\_CC3}$, $state_{StartTrasnferIn}$)

  End($(Ac_3+Ac_7):S_{SB\_CC1}$, $Ac_8:S_{SB\_CC3}$, $state_{TransferInSuccess}$)

- **For SB_CC3:**

  G = {SB_CC3, SB_CC4, MB_C3}

  A_1

  Start($Ac_5:S_{SB\_CC3}$, $(Ac_2+Ac_8):S_{SB\_CC4}$, $state_{StartTrasnferIn}$)

  End($Ac_5:S_{SB\_CC3}$, $(Ac_2+Ac_8):S_{SB\_CC4}$, $state_{TransferInSuccess}$)

- **For SB_CC4:**

G = {SB_CC4, MB_C3}

A_1

Start($(Ac_5+Ac_3):S_{SB\_CC4}$, $Ac_8:S_{MB\_C3}$, $state_{StartTrasnferIn}$)

End($(Ac_5+Ac_3):S_{SB\_CC4}$, $Ac_8:S_{MB\_C3}$, $state_{TransferInSuccess}$)

- **For MB_C3:**

  G = {MB_C3}

Meantime, the quantifications of the SoS characteristics can also be deduced during this process for observation of each component conveyor in testbed.

The results of the application indicate that the generic method applied is capable to describe the production process based on the sub-systems actions and goals. The computation of SoS characteristic allows to modify the whole SoS in design and maintenance phases. To evaluate the applicability of this generic SoS modelling method in FA domain, the application is extended to a more complex use case (a composite material manufacturing system), which is presented in the next chapter.

5 CASE STUDY 2: PULTRUSION PROCESS

This chapter is dedicated to evaluation of the applicability of the generic SoS modelling method to Pultrusion process, which is a composite material manufacturing system in ACCIONA R&D centre located in Madrid, Spain. This use case allows applying the generic modelling method to a heterogeneous system which integrates social systems (human operators) into context of the technical systems and thus can be considered as a SoS application. Section 5.1 concentrates on description of the whole process. Section 5.2 analyzes the manufacturing system according to the generic method and section 5.3 presents the results for application.

5.1 Pultrusion Process

In the ACCIONA R&D centre in Madrid, Spain, real size prototypes of civil and building structures were built to compete with the structures built by conventional materials, such as steel and concrete. The main advantages of using composite material in construction are high relation between strength and weight, simple setup and reduction of logistic means.

Different process methods including Resin Infusion, RTM, Filament Winding and Pultrution are applied to build the structures depending on the required applications, sizes and materials. Amongst the mentioned processes, Pultrution is utilized to produce different types of profiles with various materials, geometries and dimensions.

Composite materials have many properties that make them attractive for structural projects (see for example [44]), but it is difficult to manufacture. Traditional composite manufacturing methods are very labour intensive, and thus the development of manufacturing has focused on automated processes that can produce large quantities of finished product. Pultrusion is one of the most efficient manufacturing techniques for manufacturing composite material with constant cross-sections (the whole process can be seen in the figure 5.1 as follows). Dry fibres are first pulled through an impregnation station. Then they are passed through a die for forming and curing. Thousands of yards of finished structure can be made in a continuous process with the major limitation that

the finished product must be straight with a constant cross section. Typical speed of the Pultrusion process ranges from 30 mm/min (1.18 in/min) to 30 m/min (98 ft/min).

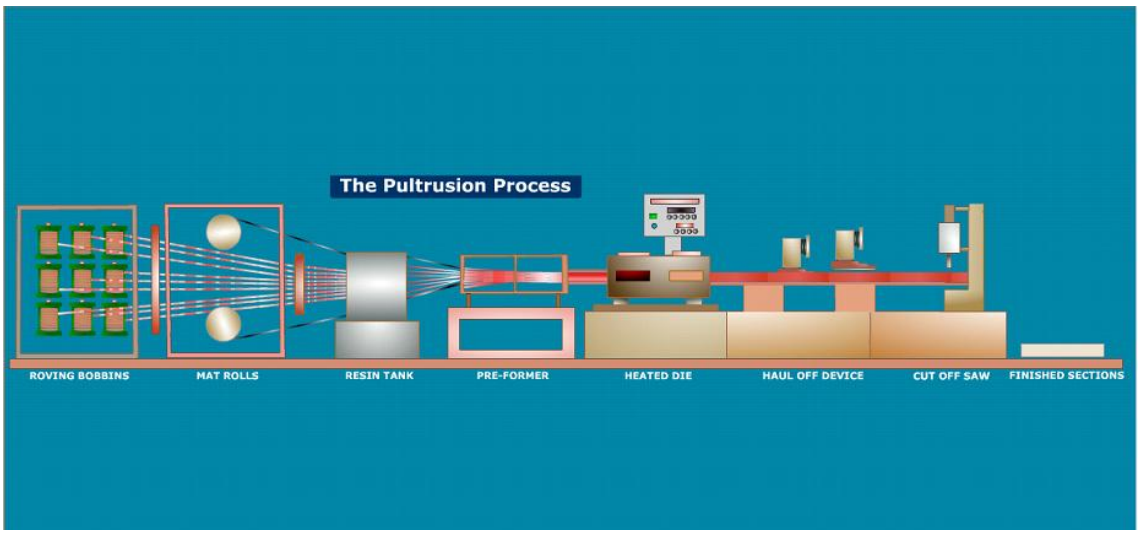

Figure 5.1. Depiction of the Pultrusion Process.

The Pultrusion Process can be divided in four main parts and they are Reinforcements, Resin Impregnation, Pultrusion Die and Pulling and Cutoff which are described as follow.

5.1.1 Reinforcements

The process of Reinforcements is the first step in Pultrusion and the main responsibility is to setup the materials for pulling. In unidirectional pultrusions, roving may be used with fabrics to add off-axis fibres. In the composite material process, the entire cross section is pulled at once and hence a large number of spools are required.

Typically glass, aramid and carbon fibres are utilized, which are stored in the dry form on creel stands. Although woven and multi-axial fibres can also be introduced into the mold, the majority of the material in the process is still unidirectional fibres which require more pulling forces by Pulling System.

Because the number of the stands is very large, thus it is easy for fibres to form tangles. In order to solve the problem, fibres are passed through an alignment card immediately after leaving the creels to prevent twisting of the roving.

Additionally, this system has to: 1) give the operator an alarm to indicate that a tow or tows are almost finished, with enough time to change; and 2) inform the operators if any of the tows is broken.

5.1.2 Resin Impregnation

Once all of the fibres have been aligned, they are going through a resin bath. To make sure the fibres are fully wetted, the strands are passed through a series of rollers which flatten and spread out the individual roving. This process is similar to impregnation during filament winding, but the baths are usually much longer.

Many other material including polyesters, vinyl esters, and epoxies can be also utilized in impregnating resin, which are specially enhanced to assure fast curing. However, required properties restrict the chosen material. Pultrusion resins should have a fairly low viscosity (for good, rapid impregnation), a long pot life (for continuous processing), and a short cure time (for full cure during the time Pultrusion is in the die).

Additionally it is important to verify the level of the resin in the bath, because if the level is low, the impregnation of the fibre cannot be guaranteed. The best way to ensure a good level is through the weight of the resin bath. The system should give an alarm when the weight of the resin is low, in order to re-fill the bath.

Meantime the resin temperature in the bath is one of the most critical parameters in the process, because low temperature produces high viscosity, and therefore leads to bad impregnation. While high temperature speeds up the cure process in the bath. Thus it is important that the system is capable to monitor the bath temperature, and to send a signal to switch on/off the heater.

Production rate influences directly in the quality of the final piece, and thus it is important to quantify and to control. In fact, the rate is determined by the kinetic of resin polymerization reaction and the magnitude of the frictional forces generated into the mold. Therefore, a wrong adjustment of this parameter can turn all the production process ineffective, losing run time and materials.

5.1.3 Pultrusion Die

The key to the Pultrusion Process is the die. Loose fibres enter the die and exit as a cured part. The die must maintain fibre alignment, compress the fibre to the desired volume fraction, and cure the composite in a relatively short period of time. This must be done without damaging the fibres. The design of the die is probably the most difficult part for establishing a Pultrusion process.

The heated and cooling sections of the forming die have inserts, which can be shaped according to the form of the produced profiles. These inserts are mainly made out of steel and are chrome-plated for wear resistance. For hollow profiles, special tool such as a floating core is required. Meantime it has to be at the right temperature, to avoid an uncured material or on the other hand a burned piece.

5.1.4 Pulling and Cutoff

The pulling station is located at the end of the Pultrusion line. A set of padded clamps grips the cross section and pulls the section horizontally. In order for the process to be continuous, two sets of grips are used: when one set is pulling, the other set travels back to its initial position. Typical pulling forces are on the order of six to eight tons, but can reach higher. However, to avoid extra stress, it is important to control and monitor the force that is applied to the profile.

5.1.5 Controlled Parameters in the Production Process

During the whole process, several parameters should be controlled to ensure the quality of the final production.

a. Fibre breaking

In order to have the same fibre volume fraction it is important to control if any tow of roving breaks. If there are any of them breaking, there should be an alarm for further operation.

b. Humidity and environment temperature

The operation of mixing components should be carried out in a controlled environment because the parameters, such as humidity and temperature will affect the reaction rate. It is important to ensure that all the steps of the productive process are complied correctly, and the element is manufactured according to good criteria of quality.

c. Dimension, weight and quantities control

The quality of the parts could be checked by controlling dimensions, weight and quantities, in this way the repeatability of the process can be guaranteed.

d. Mistakes in the selection of raw materials

An exhaustive control of raw materials such as fibres, resin, catalytic agent, etc, avoids mistakes in the process. The system should be able to indicate the operators the right materials that should be used.

5.2 Description of the Pultrusion Process Based On the Generic Method

This section concentrates on the analysis for the composite material manufacturing system based on the generic SoS modelling method, whose cores are two concepts: SD and DD. Specifically, section 5.2.1 focuses on SD analysis for the individual subsystem and section 5.2.2 is dedicated to DD analysis. This section provides a detailed analysis on the constituent systems in the Pultrusion process which helps to understand it more thoroughly.

5.2.1 SD for the Pultrusion Process

There are two elements in the SD of the generic SoS modelling method; one is Action and the other is Goal. The component Action contains actions that can be executed by the system and Goal indicates a set(s) of goals that can be accomplished without cooperation with other systems. This concept reveals the reasons why individual systems are created and analyzes functions each component system can perform.

5.2.1.1 Reinforcements

The system Reinforcement is expressed using the notation S_1 in analysis based on the generic modelling method. According to the analysis in the last section, Reinforcements (see Fig. 5.2) is the first subsystem in the whole process of Pultrusion. Spools of graphite fibre held by the creel racks are pulled out by the force generated by Pulling System. The actions can be executed by Reinforcement solely are concluded as follow:

- **A_{11}**: Indicate right material.
  This action indicates the operators the right material that should be utilized according to the type of final productions.

- **A_{12}**: Indicate when the material is going to be finished.
  This action generates a signal that the material on any of the creel racks is going to be finished, leaving enough time for the operators to change. It stops when the material returns to normal state.

- **A_{13}**: Indicate when any of the tows is broken.
  The creel racks are capable to sense the situation when any of the tows is broken and then send a signal to inform the operators, which meantime stops the pulling force, leading to stopping the whole production line. Further operations should be carried out before this signal disappears and the whole system starts to run again.

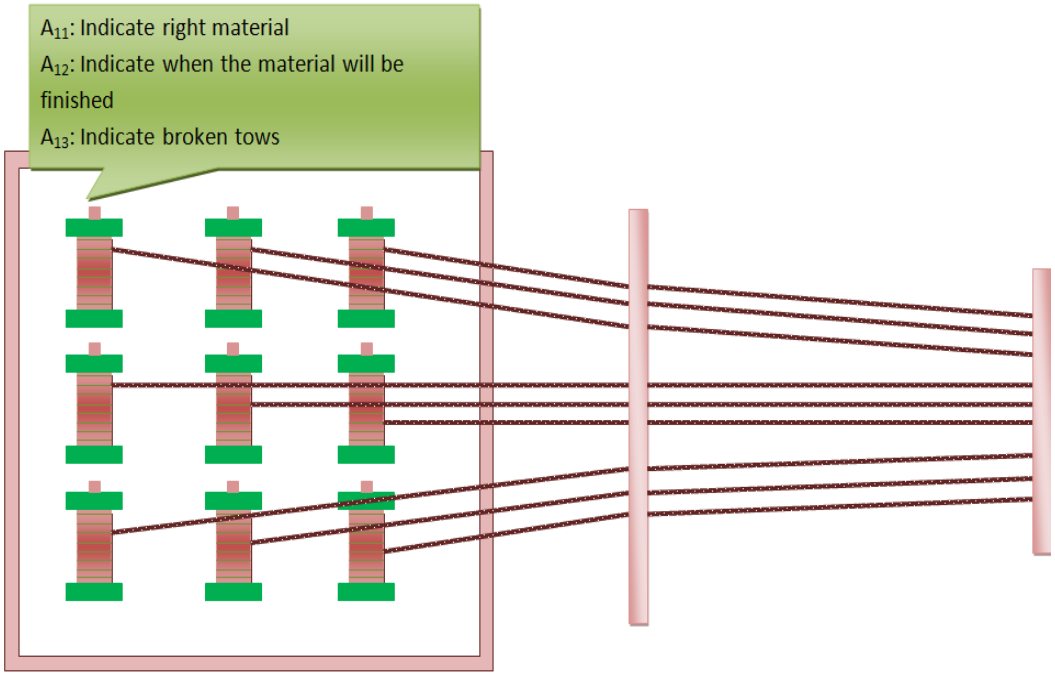

Figure 5.2. Depiction of the Reinforcements subsystem.

Using the expression existing in the generic method, the component 'action' in SD can be presented as:

$$A\sigma_1 = \{A_{11}, A_{12}, A_{13}\} \quad (5\text{-}1)$$

There is no goal can be completed by this subsystem solely and therefore,

$$G\sigma_1 = \{\varnothing\} \quad (5\text{-}2)$$

$$S\sigma_1 = \{A\sigma_1, G\sigma_1\} \quad (5\text{-}3)$$

5.2.1.2 Resin Impregnation (S_2)

The system Resin Impregnation is expressed using the notation S_2. Aligned fibres are passing through a resin bath shown in the figure 5.3. This subsystem is more complex and actions that Resin Impregnation can accomplish are:

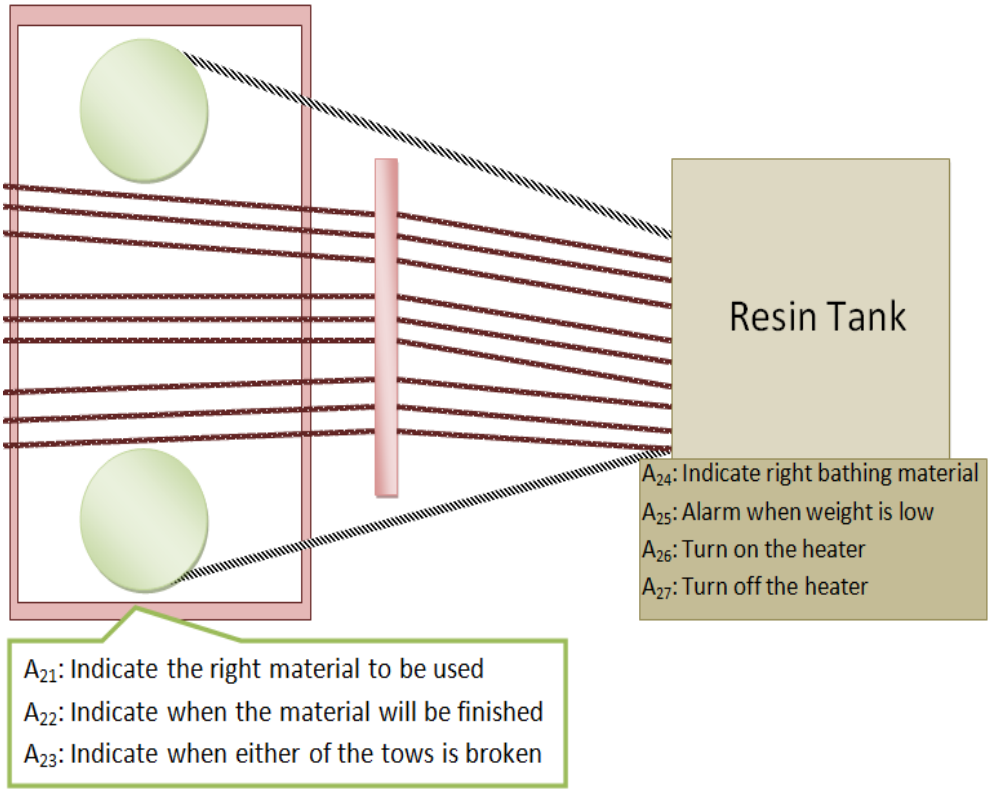

Figure 5.3. Depiction of the Resin Impregnation subsystem.

- **A_{21}**: Indicate the right material.
  Similar with the action executed by Reinforcements, this subsystem should also indicate operators which material should be added on both racks before whole process starts to run.

- **A_{22}**: Indicate when the material will be finished
  Similar with A_{12}.

- **A_{23}**: Alarm when any of tows is broken.
  The creel racks that are located in this subsystem should also alarm operators when any of the tows is broken, which will stop the pulling force and then the whole production process will pause. The alarm disappears when further operations have done and the whole system returns to normal.

- **A_{24}**: Indicate right bathing materials.
  This subsystem should be able to indicate operators the right material in resin tank before the whole system is running.

- **A_{25}**: Alarms when the weight of resin tank is low.
  To guarantee the impregnation of the fibre, a certain level in resin tank should be controlled, which could be told from the weight of the resin tank. The alarm disappears after operators have carried out further operations.

- **A_{26}**: Turn on the heater.
  Temperature in the resin tank should be kept with a certain range to guarantee the quality of the final production. When the temperature is below the range, this action will start functioning to increase the temperature in the tank.

- **A_{27}**: Turn off the heater.
  This action is similar with A_{26}, with the difference that A_{27} intends to decrease the temperature in the resin tank when it is higher than the range.

So in SD,

$$A\sigma_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}, A_{26}, A_{27}\} \quad (5\text{-}4)$$

Goal(s) can be achieved by the subsystem of Resin Impregnation is:

- **G_{21}**: Adjust temperature in the resin tank
  Goals are accomplished by executing actions of A_{26} and A_{27} to keep the temperature in resin tank constant according to the desired final product.

Similarly, the component ‘Goal’ in SD can be presented as:

$$G\sigma_2 = \{G_{21}\} \quad (5\text{-}5)$$

Consequently, the whole SD can be expressed as follows:

$$S\sigma_2 = \{A\sigma_2, G\sigma_2\} \quad (5\text{-}6)$$

5.2.1.3 Pultrusion Die

The system Pultrusion Die is expressed using the notation S_3. After Resin Impregnation, the material is first entering a pre-former machine (see Fig. 5.4 as follows), which aggregates the loose material and then the material will enter the heated die to form the final products. Actions can be executed by this subsystem are presented as follow:

- **A_{31}**: Shape inserts.

According to the desired final production and the running material, the pre-former should be able to adjust the cross section of inserts, preparing for the movement of entering the heated die.

- **A_{32}**: Decide special tools.
  For hollow profiles special tools should be used.

- **A_{33}**: Adjust diameter.
  The diameter of the heated die decides the dimension of the final production and plays a vital role in the whole process. It should be adjusted according to the demanded final production.

- **A_{34}**: Turn on the heater.
  Temperature in the heated die should be kept with a certain range to guarantee the quality of the final production. When the temperature is below the range, this action will function to increase the temperature.

- **A_{35}**: Turn off the heater.
  This action intends to decrease the temperature in the heated die when it is higher than the range.

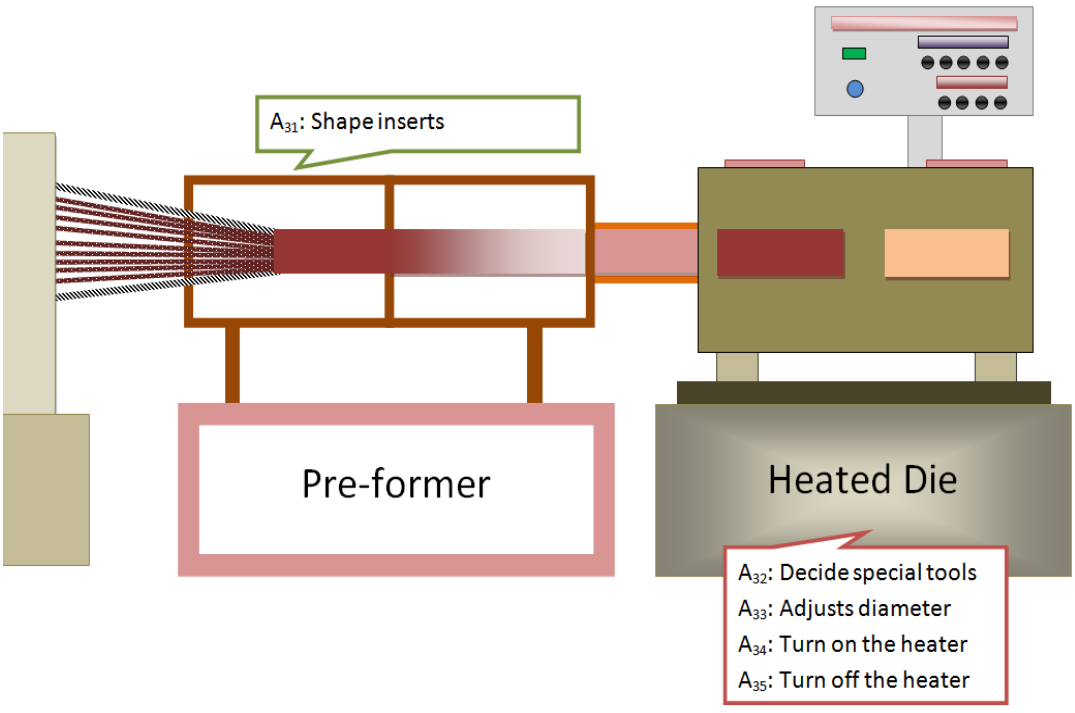

Figure 5.4. Depiction of the Heated Die subsystem.

So $A\sigma_3 = \{A_{31}, A_{32}, A_{33}, A_{34}, A_{35}\}$ (5-7)

Goals can be achieved by the subsystem of Pultrusion Die are:

- **G_{31}**: Prepare pre-former for the material.
  This goal can be accomplished by executing actions A_{31}.

- **G_{32}**: Prepare heated die for the material.
  Through actions A_{32} and A_{33}, heated die can be prepared before the whole process starts running.

- **G_{33}**: Keep the temperature in the heated die constant.
  Through executing A_{34} and A_{35}, temperature is able to stay constant.

Similarly, the component 'Goal' in SD can be presented as:

$G\sigma_3 = \{G_{31}, G_{32}, G_{33}\}$ (5-8)

The SD for Pultrusion Die can be expressed as:

$S\sigma_3 = \{A\sigma_3, G\sigma_3\}$ (5-9)

5.2.1.4 Pulling System

The Pulling System is expressed using the notation S_4. Two clamps grip the cross section and pull the profile to the end, and meantime the pulling force is the only factor that keeps the whole system running. When it stops pulling, all the subsystems in this production line should stop functioning. The actions this subsystem is capable of executing solely are introduced as follow together with an illustration in the figure 5.5 as follow:

- **A_{41}**: Pull the cross section.
  The pulling force generated by the components keep the whole system running.

- **A_{42}**: Adjust the pulling force.
  The applied force should be under supervisory so that it will not generate extra stress for the material (bad effect).

- **A_{43}**: Stop pulling.

When there is an alarming or some signals that should stop the whole system, the first step is to eliminate the pulling force, which leads to stopping the whole production line.

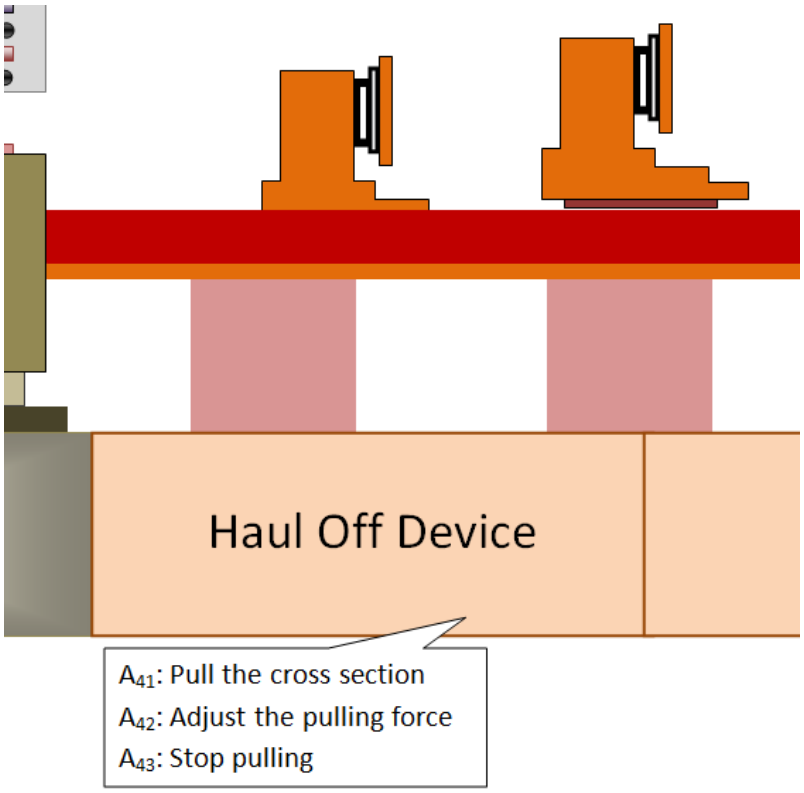

Figure 5.5. The depiction of the Pulling subsystem.

So $A\sigma_4 = \{A_{41}, A_{42}, A_{43}\}$ (5-10)

Goal can be achieved by the Pulling System is:

- **G_{41}**: Pull the material to the end.
  By executing A_{41} and A_{42}, the haul off device is capable of pulling the material consecutively using a certain force.

The component 'Goal' in SD can be presented as:

$G\sigma_4 = \{G_{41}\}$ (5-11)

Thus the whole SD for Pulling System can be written as:

$S\sigma_4 = \{A\sigma_4, G\sigma_4\}$ (5-12)

5.2.1.5 Cutoff System

The Cutoff System is expressed using the notation S_5. This is the final subsystem of the whole production line, and the main functionality is to cut the product using a certain cutting frequency so that the final products have same length according to the need. Actions that can be completed can be concluded as follows (see Fig. 5.6):

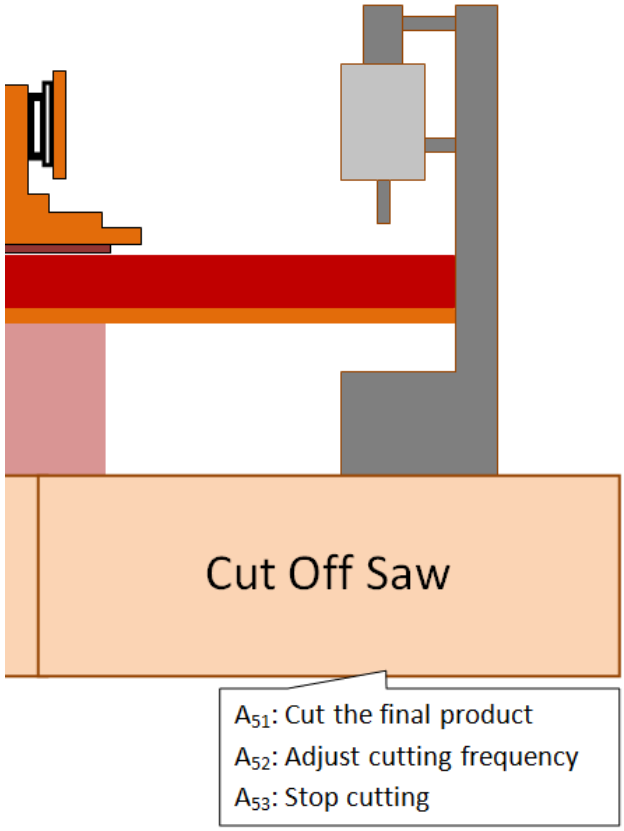

Figure 5.6. The depiction of the Cutoff subsystem.

- **A_{51}**: Cut the product.
  This action cuts the final product into comparatively small pieces and they are automatically stored in the finished section.

- **A_{52}**: Adjust cutting frequency.
  The cutting frequency should be adjusted according to the length of the desired final production.

- **A_{53}**: Stop cutting.

Thus $A\sigma_5 = \{A_{51}, A_{52}, A_{53}\}$ (5-13)

Goal(s) can be achieved by the subsystem of Cutoff is:

- **G_{51}**: Cut the product into a certain length.
  A_{54} and A_{55} guarantee that the final products have same length using a certain cutting frequency.

Therefore, the component 'Goal' in SD can be presented as:

$$G\sigma_5 = \{G_{51}\} \quad (5\text{-}14)$$

SD for Cutoff System is:

$$S\sigma_5 = \{A\sigma_5, G\sigma_5\} \quad (5\text{-}15)$$

5.2.1.6 Subsystem of Environmental Controller

The subsystem of Environmental Controller is expressed using the notation S_6. As can be seen from the description part of the composite material manufacturing system, temperature and humidity in the environment during the production process should be controlled in order to guarantee good qualities of the final products. Thus the environmental controller should also be considered as a subsystem and analyzed based on the generic modelling method.

The goals can be achieved by this subsystem are:

- **G_{61}**: Reach the desired environmental temperature.
- **G_{62}**: Reach the desired environmental humidity.

Actions should be done to accomplish the specific goals are:

- **A_{61}**: Adjust the environmental temperature.
- **A_{62}**: Adjust the environmental humidity.

$$\text{So } A\sigma_6 = \{A_{61}, A_{62}\} \quad (5\text{-}16)$$

The component 'Goal' in SD can be presented as:

$$G\sigma_6 = \{G_{61}, G_{62}\} \quad (5\text{-}17)$$

$$S\sigma_6 = \{A\sigma_6, G\sigma_6\} \quad (5\text{-}18)$$

5.2.1.7 Subsystem of Human Operators

The subsystem of Human Operators is expressed using the notation S_7. Actions executed by human operators play an important role in running the whole production line and they are discussed as follow:

- **A_{71}**: Load right material for Reinforcement.
  After being indicated by the subsystem of Reinforcement, operators have to load the right material on the creel racks to until the indication from Reinforcement has disappeared.

- **A_{72}**: Change the material that will be finished in Reinforcement.
  An indication from Reinforcement that the material on the creel racks will be finished will lead the operators to executing this action.

- **A_{73}**: Fix the broken tows for Reinforcement.
  The Reinforcement will alarm the operators when there are any of the tows broken. Then the whole production system will have to halt till further operations have been taken into effect to fix the broken tows.

- **A_{74}**: Load right material for Resin Impregnation.
  Several creel racks exist in this subsystem and the material on them should be loaded manually before running the whole production line.

- **A_{75}**: Change the material that will be finished in Resin Impregnation

- **A_{76}**: Fix the broken tows for the subsystem of Resin Impregnation.

- **A_{77}**: Load required material for bathing.
  According to the desired final product and the material introduced by A_{74}, the corresponding bathing material is utilized in the process of Resin Impregnation.

- **A_{78}**: Increase the weight of the resin tank.
  The subsystem of Resin Impregnation will indicate the operators when the weight of the resin tank is lower than it should be. Operators take actions immediately after being indicated until the indication disappears.

- **A_{79}**: Start the whole production line.
  This action should be executed when every subsystem in the Pultrusion Process is in the working status. A_{79} leads Pulling System to start pulling the material, resulting in running the whole production line.

The component 'action' in SD can be presented as:

$$A\sigma_7 = \{A_{71}, A_{72}, A_{73}, A_{74}, A_{75}, A_{76}, A_{77}, A_{78}, A_{79}\} \quad (5\text{-}19)$$

Additionally there is no goal can be completed by human operators solely and therefore:

$$G\sigma_7 = \{\varnothing\} \quad (5\text{-}20)$$

Thus SD for Human Operators is:

$$S\sigma_7 = \{A\sigma_7, G\sigma_7\} \quad (5\text{-}21)$$

5.2.2 DD for the Pultrusion Process

In DD, there are also two components: action and goal. Goals could only be completed through cooperation between constituent systems. In this section, the composite material manufacturing system is analyzed in the perspective of cooperation and communication with detailed explanations.

In Pultrusion process, there are four subsystems that can achieve goals by cooperation with other subsystems, and they are Reinforcement (S_1), Resin Impregnation (S_2), Pulling System (S_4) and Human Operators (S_7). The analysis is presented as follows from the table 5-1 to the table 5-4. In the tables, Dynamic Goals refer to the aim of the cooperation among systems, Involved Systems and Actions refer to the actions that should be completed for reaching the Dynamic Goals by certain systems and Relationship between Actions describe the interface activities.

5.2.2.1 Dynamic Analysis for Reinforcement (S_1)

Goals that can be achieved through cooperation between Reinforcement and other subsystems are:

- **Gc_{11}:** Introduce material on creel racks:

When the action A_{11} is being executed, indicating operators the right material that should be utilized according to the required final product. As long as the right material has been added (A_{71}), this action terminates.

- **Gc_{12}:** Change the material that will be finished:
  A_{12} is executed when the material will be finished, leaving enough time for operators to change. After changing the material (A_{72}), this action (A_{12}) terminates.

- **Gc_{13}:** Fix the broken tows:
  When A_{13} is being executed to warn there is tow(s) broken, the pulling system will first to stop (A_{43}) in order to stop the whole production process. After fixing the broken tow(s) (A_{73}) by human operators, the warning stops.

The component in DD for the subsystem Reinforcement can be concluded in table 5-1 as follows. The notations in the column Relationship between Actions combine the actions that should be completed by Involved Systems in a certain sequence for reaching the Dynamic Goals.

Table 5-1. Goals Can Be Completed by Reinforcement through Cooperation with Other Subsystems:

No.	Dynamic Goals	Involved Systems and Actions	Relationship Between Actions
Gc_{11}	Introduce material on creel racks	Reinforcement: A_{11}	Start(A_{71}:S_7, state(A_{11}:S_1))
		Human operators: A_{71}	End(A_{71}:S_7, state($\neg A_{11}$:S_1))
Gc_{12}	Change the material that will be finished	Reinforcement: A_{12}	Start(A_{72}:S_7, state(A_{12}:S_1))
		Human Operators: A_{72}	End(A_{72}:S_7, state($\neg A_{12}$:S_1))
Gc_{13}	Fix the broken tows	Reinforcement: A_{13}	Start(A_{73}:S_7, A_{43}:S_4, state(A_{13}:S_1))
		Pulling System: A_{43}	
		Human Operators: A_{73}	End(A_{73}:S_7, state($\neg A_{13}$:S_1))

The component Action and Goal in DD can be presented as follow:

$$A\delta_1 = \{A_{11}, A_{12}, A_{13}\} \quad (5\text{-}22)$$

$$G\delta_1 = \{Gc_{11}, Gc_{12}, Gc_{13}\} \quad (5\text{-}23)$$

Thus the DD for S_1 is expressed:

$S\sigma_1 = \{A\sigma_1, G\sigma_1\}$ (5-24)

5.2.2.2 Dynamic Analysis for Resin Impregnation (S_2)

Goals that can be achieved through cooperation between Resin Impregnation and other subsystems can be introduced as follows:

- **Gc_{21}:** Introduce material on creel racks:
  When the action A_{21} is being executed by Resin Impregnation (S_2), indicating operators the right material that should be utilized according to the required final product. Then the operator starts to load required material for Resin Impregnation: Start(A_{74}:S_7, state(A_{21}:S_2)). After the indication terminates, the operator stops loading material: End(A_{74}:S_7 state($\neg A_{21}$:S_2)).

- **Gc_{22}:** Change the material that will be finished:
  The action A_{22} is running when the material is going to be finished, leaving operators enough time for changing which can be done by action A_{75} by the subsystem of human operators (S_7).

- **Gc_{23}:** Fix the broken tows:
  When there is any broken tow that is detected by the subsystem of Resin Impregnation (A_{23}), the whole system has to pause first by running the action A_{43} by stopping Pulling System (S_4). Human operators start fixing the broken tows by executing A_{76} till the broken tow has been mended already.

- **Gc_{24}:** Introduce material in the resin bath:
  This process is similar with Gc_{21} explained already.

- **Gc_{25}:** Control the weight of resin bath:
  When the action (A_{25}) alarms that the weight of the resin tank is lower than required, first the whole production process should be stopped by running A_{43}. After weight has been adjusted by human operators, the alarm terminates.

All the goals and their relationships can be concluded in the table 5-2 as follow:

Table 5-2. Goals Can Be Completed by Resin Impregnation through Cooperation with Other Subsystems:

No.	Dynamic Goals	Involved Systems and Actions	Relationship Between Actions
Gc_{21}	Introduce material on creel racks	Resin impregnation: A_{21}	Start(A_{74}:S_7, state(A_{21}:S_2)) End(A_{74}:S_7 state($\neg A_{21}$:S_2))
		Human operators: A_{74}	
Gc_{22}	Change the material that will be finished	Resin impregnation:A_{22}	Start(A_{75}:S_7, state(A_{22}:S_2)) End(A_{75}:S_7, state($\neg A_{22}$:S_2))
		Human Operators: A_{75}	
Gc_{23}	Fix the broken tows	Resin impregnation:A_{23}	Start(A_{76}:S_7, A_{43}:S_4, state(A_{23}:S_2)) End(A_{76}:S_7, state($\neg A_{23}$:S_2))
		Pulling System: A_{43}	
		Human Operators: A_{76}	
Gc_{24}	Introduce material in the resin bath	Resin impregnation:A_{24}	Start(A_{77}:S_7, state(A_{24}:S_2)) End(A_{77}:S_7, state($\neg A_{24}$:S_2))
		Human operators: A_{77}	
Gc_{25}	Control the weight of resin bath	Resin impregnation:A_{25}	Start(A_{43}:S_4, state(A_{25}:S_2)) End(A_{78}:S_7, state($\neg A_{25}$:S_2))
		Human Operators: A_{78}	
		Pulling System: A_{43}	

The component Action and Goal in DD can be presented as follow:

$$A\delta_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}\} \quad (5\text{-}25)$$

$$G\delta_2 = \{Gc_{21}, Gc_{22}, Gc_{23}, Gc_{24}, Gc_{25}\} \quad (5\text{-}26)$$

Thus the DD for S_2 is expressed:

$$S\sigma_2 = \{A\sigma_2, G\sigma_2\} \quad (5\text{-}27)$$

5.2.2.3 Dynamic Analysis for Pulling System (S_4)

Goals that can be achieved through cooperation by Pulling System and other subsystems can be introduced as follow:

- **Gc_{41}:** Start the whole production process once the Pulling System starts pulling the profile:
  Once the Pulling System (S_4) starts to pull the material (A_{41}), it is an indication that there is no warning or alarm and every subsystem in the whole process has a

normal working state. Then each system will start functioning by running corresponding actions:
Resin Tank (S_2): Turn on the heater (A_{26});
Pultrusion Die (S_3): Turn on the heater (A_{34});
Cutoff System (S_5): Start cutting (A_{51}).

- **Gc_{42}:** Stop the whole production process once the Pulling System stops pulling the material:
  When the whole production process needs to be stopped, the first action is to stop pulling (A_{43}). Once the Pulling System pauses, all other systems should stop functioning by running the corresponding actions:
  Resin Tank (S_2): Turn off the heater (A_{27});
  Pultrusion Die (S_3): Turn off the heater (A_{35});
  Cutoff System (S_5): Stop cutting (A_{53}).

This section can be concluded in table 5-3 as follows:

Table 5-3. Goals Can Be Completed by Pulling System through Cooperation with Other Subsystems:

No.	Dynamic Goals	Involved Systems and Actions	Relationship Between Actions
Gc_{41}	Start the whole production process once the Pulling System starts pulling the material.	Resin Tank: A_{26}	Start(A_{26}: S_2, A_{34}:S_3, A_{51}:S_5, state(A_{41}:S_4))
		Pultrusion Die: A_{34}	
		Pulling System: A_{41}	
		Cutoff System: A_{51}	
Gc_{42}	Stop the whole production process once the Pulling System stops pulling the material.	Resin Tank: A_{27}	Start(A_{27}: S_2, A_{35}:S_3, A_{53}:S_5, state(A_{43}:S_4))
		Pultrusion Die: A_{35}	
		Pulling System: A_{43}	
		Cutoff System: A_{53}	

The component Action and Goal in DD can be presented as follow:

$$A\delta_4 = \{A_{41}, A_{43}\} \quad (5\text{-}28)$$

$$G\delta_4 = \{Gc_{41}, Gc_{42}\} \quad (5\text{-}29)$$

Thus the DD for S_4 is expressed:

$S\sigma_4 = \{A\sigma_4, G\sigma_4\}$ (5-30)

5.2.2.4 Dynamic Analysis for Human Operators (S_7)

Goals that can be achieved through cooperation by Human Operators and other subsystems can be introduced as follow together with the table 5-4:

- **Gc_{71}:** Operate on Pulling System so as to start the whole production line:
  As long all the systems are in normal state, the operators (S_7) make Pulling System start functioning (A_{79}), which results in running the whole system (is connected with Gc_{41}).

Table 5-4. Goals Can Be Completed by Human Operators through Cooperation with Other Subsystems:

No.	Dynamic Goals	Involved Systems and Actions	Relationship Between Actions
Gc_{71}	Operate on Pulling System so as to start the whole production line.	Human Operators: A_{79} Pulling System: A_{41}	Start(A_{41}:S_4, state(A_{79}:S_7))

The component Action and Goal in DD can be presented as follow:

$A\delta_7 = \{A_{79}\}$ (5-31)

$G\delta_7 = \{Gc_{71}\}$ (5-32)

Thus the DD for S_7 is expressed:

$S\sigma_7 = \{A\sigma_7, G\sigma_7\}$ (5-33)

5.2.2.5 Dynamic Analysis for Pultrusion Die (S_3) and Cutoff System (S_5)

For the subsystems of Pultrusion Die and Cutoff System, there is no dynamic goal that can be reached by these two component systems directly. They are only capable of achieving other systems' dynamic goals by executing their own actions. For these two systems, their DDs can be presented as follow:

- For Pultrusion Die (S_3):

  $A\delta_3 = \{A_{34}, A_{35}\}$ (5-34)

  $G\delta_3 = \{Gc_{41}, Gc_{42}\}$ (5-35)

  Thus the DD for S_4 is expressed:

  $S\sigma_3 = \{A\sigma_3, G\sigma_3\}$ (5-36)

- For Cutoff System (S_5):

  $A\delta_5 = \{A_{51}, A_{53}\}$ (5-37)

  $G\delta_5 = \{Gc_{41}, Gc_{42}\}$ (5-38)

  Thus the DD for S_5 is expressed:

  $S\sigma_5 = \{A\sigma_5, G\sigma_5\}$ (5-39)

5.2.3 Constituent System Description for the Pultrusion Process

For a component system in a SoS, it can be considered as a whole and a part simultaneously as presented in 'Holarchical View'. Once a system has been a legal constituent system in a process, it has the properties described in SD and DD at the same time. There are two components in both SD and DD, and they are actions and goals which have already been discussed in the previous sections. The CSD (Constituent System Description) in a SoS is expressed by S_i. In S_i, it contains two components and they are SD and DD. This section concludes the discussion in previous sections and adds to the dynamic description the goals that cannot be completed by a subsystem but have to contribute actions. The results are presented in the table 5-5 as follow:

Table 5-5. The Description for Each Subsystem in Pultrusion Process:

No	System Name	System Description	
1	Reinforcement	SD	$A\sigma_1 = \{A_{11}, A_{12}, A_{13}\}$, $G\sigma_1 = \{\emptyset\}$ $S\sigma_1 = \{A\sigma_1, G\sigma_1\}$
		DD	$A\delta_1 = \{A_{11}, A_{12}, A_{13}\}$ $G\delta_1 = \{Gc_{11}, Gc_{12}, Gc_{13}\}$ $S\sigma_1 = \{A\sigma_1, G\sigma_1\}$
		CSD	$S_1 = \{S\sigma_1, S\delta_1\}$
2	Resin Impregnation	SD	$A\sigma_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}, A_{26}, A_{27}\}$,

			$G\sigma_2 = \{G_{21}\}$ $S\sigma_2 = \{A\sigma_2, G\sigma_2\}$
		DD	$A\delta_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}, A_{26}, A_{27}\}$ $G\delta_2 = \{Gc_{21}, Gc_{22}, Gc_{23}, Gc_{24}, Gc_{25}, Gc_{41}, Gc_{42}\}$ $S\sigma_2 = \{A\sigma_2, G\sigma_2\}$
		CSD	$S_2 = \{S\sigma_2, S\delta_2\}$
3	Heated Die	SD	$A\sigma_3 = \{A_{31}, A_{32}, A_{33}, A_{34}, A_{35}\}$ $G\sigma_3 = \{G_{31}, G_{32}, G_{33}\}$ $S\sigma_3 = \{A\sigma_3, G\sigma_3\}$
		DD	$A\delta_3 = \{A_{34}, A_{35}\}$ $G\delta_3 = \{Gc_{41}, Gc_{42}\}$ $S\sigma_3 = \{A\sigma_3, G\sigma_3\}$
		CSD	$S_3 = \{S\sigma_3, S\delta_3\}$
4	Pulling System	SD	$A\sigma_4 = \{A_{41}, A_{42}, A_{43}\}$, $G\sigma_4 = \{G_{41}\}$ $S\sigma_4 = \{A\sigma_4, G\sigma_4\}$
		DD	$A\delta_4 = \{A_{41}, A_{43}\}$ $G\delta_4 = \{Gc_{41}, Gc_{42}, Gc_{13}, Gc_{23}, Gc_{41}, Gc_{71}\}$ $S\sigma_4 = \{A\sigma_4, G\sigma_4\}$
		CSD	$S_4 = \{S\sigma_4, S\delta_4\}$
5	Cutoff System	SD	$A\sigma_5 = \{A_{51}, A_{52}, A_{53}\}$, $G\sigma_5 = \{G_{51}\}$ $S\sigma_5 = \{A\sigma_5, G\sigma_5\}$
		DD	$A\delta_5 = \{A_{51}, A_{53}\}$, $G\delta_5 = \{Gc_{41}, Gc_{42}\}$ $S\sigma_5 = \{A\sigma_5, G\sigma_5\}$
		CSD	$S_5 = \{S\sigma_5, S\delta_5\}$
6	Environmental Controller	SD	$A\sigma_6 = \{A_{61}, A_{62}\}$, $G\sigma_6 = \{G_{61}, G_{62}\}$ $S\sigma_6 = \{A\sigma_6, G\sigma_6\}$
		DD	$A\delta_4 = \{\emptyset\}$, $G\delta_4 = \{\emptyset\}$ $S\delta_4 = \{A\delta_4, G\delta_4\}$
		CSD	$S_6 = \{S\sigma_6, S\delta_6\}$

7	Human Operators	SD	$A\sigma_7 = \{A_{71}, A_{72}, A_{73}, A_{74}, A_{75}, A_{76}, A_{77}, A_{78}, A_{79}\}$ $G\sigma_7 = \{\emptyset\}$ $S\sigma_7 = \{A\sigma_7, G\sigma_7\}$
		DD	$A\delta_7 = \{A_{71}, A_{72}, A_{73}, A_{74}, A_{75}, A_{76}, A_{77}, A_{78}, A_{79}\}$ $G\delta_7 = \{Gc_{11}, Gc_{12}, Gc_{13}, Gc_{21}, Gc_{22}, Gc_{23}, Gc_{24}, Gc_{25}, Gc_{71}\}$ $S\sigma_7 = \{A\sigma_7, G\sigma_7\}$
		CSD	$S_7 = \{S\sigma_7, S\delta_7\}$

The characteristics of Autonomy and Diversity for each constituent system can be computed by the generic modelling method as follow:

$$\text{Autonomy (i)} = |A\sigma_i| \tag{5-40}$$

$$\text{Diversity (i)} = A\sigma_i \tag{5-41}$$

As for the characteristic of Belonging, it should be computed according with a certain number of global goals in order to evaluate the approximate contribution a constituent system has done for the whole SoS using the following equation, whose result ranges from 0 to 1:

$$\text{Belonging (i)} = \frac{|(G\sigma_i \cup G\delta_i) \cap G|}{|G\sigma_i \cup G\delta_i|} \tag{5-42}$$

As for the characteristic of Connectivity, it is widely discussed with interoperability which is usually related with actions. With respect to different goals for a SoS, the Connectivity situation might vary. Connectivity is built when there is a need for cooperation in order to complete dynamic goals through actions. As presented in the figure 5.7, all dynamic goals are marked with connectivity situations. Take connectivity situation between S_1 (Reinforcement) and S_7 (Human Operators) as an example: Gc_{11}, Gc_{12} and Gc_{13} are completed through cooperation, which leads to building connectivity between these two component systems. Thus the connectivity situation displayed in the figure 5.7 has presented the minimum direct connections for completing goals between constituent systems which should be shown in the lower triangular part in the Connectivity table.

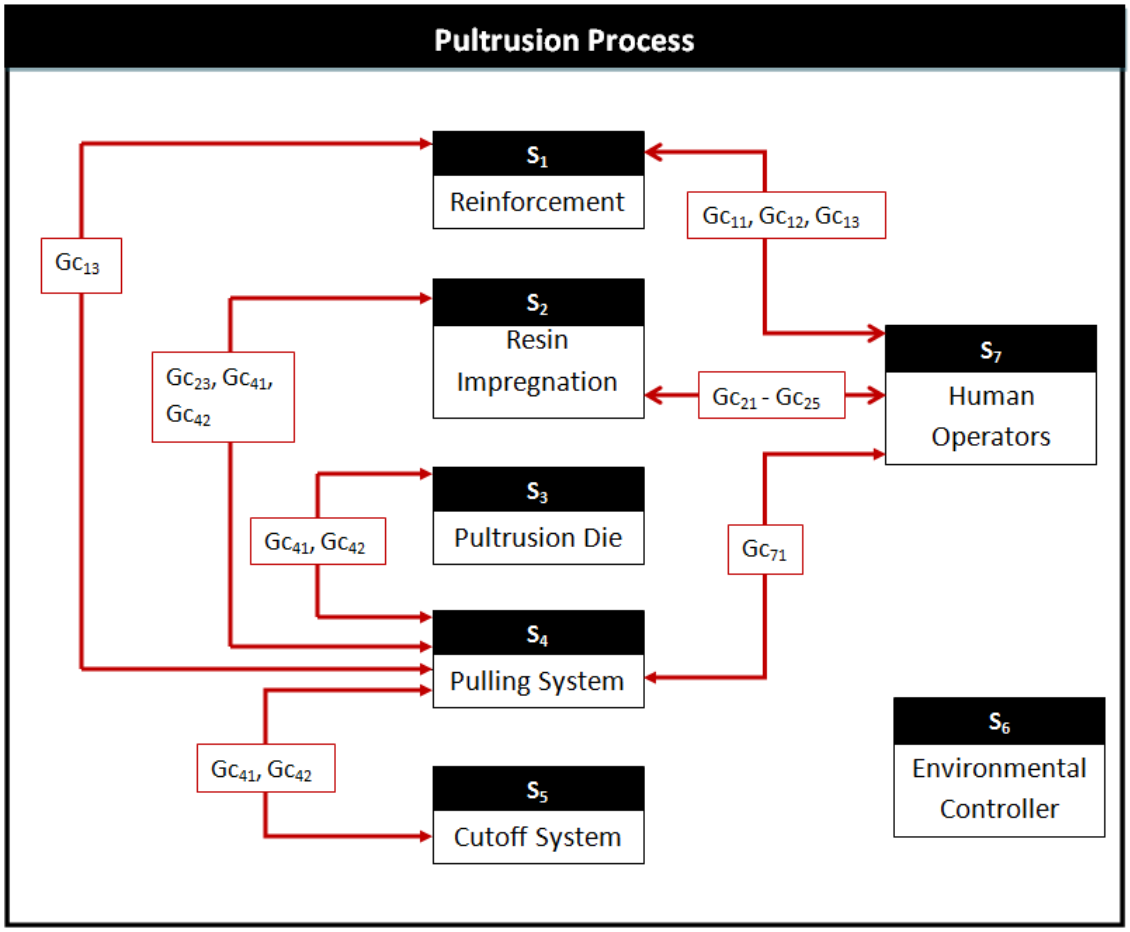

Figure 5.7. Connectivity depiction between seven systems in Pultrusion Process.

From the figure 5.12, the lower triangular part of the characteristic Connectivity can be depicted as follows:

C (Connectivity) =

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	na	na	na	na	na	na
S_2	0	0	na	na	na	na	na
S_3	0	0	0	na	na	na	na
S_4	1	1	1	0	na	na	na
S_5	0	0	0	1	0	na	na
S_6	0	0	0	0	0	0	na
S_7	1	1	0	1	0	0	0

In the next section in this chapter, the generic SoS modelling method is applied to the Pultrusion Process. The results for evaluation of applicability of the generic modelling method are presented.

5.3 Application of the Generic SoS Modelling Method to Pultrusion Process

In this section, the results for the application of the generic SoS modelling method to composite material manufacturing system (Pultrusion process) are exemplified with a possible scenario. Production processes are described and meantime the characteristics mentioned in the previous sections are evaluated.

The Goal G_1 is exemplified as the scenario when every system is running normally in the Pultrusion process. It should be the combination for the goals discussed in dynamic goals or static goals. According to the discussion in previous sections in this chapter, G_1 is presented as follows:

- $G_1 = \{Gc_{11}, Gc_{21}, Gc_{24}, Gc_{25}, Gc_{71}, Gc_{41}\}$

Then the whole system can be expressed as follows:

- $S = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, C, G_1\}$

The corresponding actions that should be executed are presented as follow (after the indication by Reinforcement (S_1), the operator starts loading the right material till the indication terminates):

- $Start(A_{71}{:}S_7, state(A_{11}{:}S_1))$
- $End(A_{71}{:}S_7, state(\neg A_{11}{:}S_1))$

The Connectivity table is displayed as:

C (Connectivity) =

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	0	0	0	0	0	1
S_2	0	0	0	0	0	0	0
S_3	0	0	0	0	0	0	0
S_4	1	1	1	0	0	0	0
S_5	0	0	0	1	0	0	0
S_6	0	0	0	0	0	0	0
S_7	1	1	0	1	0	0	0

Load the required material for the racks in Resin Impregnation (S_2):

- Start(A_{74}:S_7, state(A_{21}:S_1))
- End(A_{74}:S_7, state(¬A_{21}:S_1))

Introduce the required material for the resin bath in Resin Impregnation (S_2):

- Start(A_{77}:S_7, state(A_{24}:S_1))
- End(A_{77}:S_7, state(¬A_{24}:S_1))

Adjust the weight of the resin bath in S_2 before running the whole system (if necessary):

- Start(A_{78}:S_7, state(A_{25}:S_1))
- End(A_{78}:S_7, state(¬A_{25}:S_1))

The connectivity table is presented as follows:

C (Connectivity) =

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	0	0	0	0	0	0
S_2	0	0	0	0	0	0	1
S_3	0	0	0	0	0	0	0
S_4	1	1	1	0	0	0	0
S_5	0	0	0	1	0	0	0
S_6	0	0	0	0	0	0	0
S_7	1	1	0	1	0	0	0

The operators start pulling in Pulling System (S_4) when all the systems for reaching the goals (G_1) are all in the normal working state. As long as the pulling system starts functioning, Resin Inpregnation (S_2), Pultrusion Die (S_3) and Cutoff System (S_5) all start working to keep the whole process working.

- Start(A_{41}:S_4, state(A_{79}:S_7))
- Start(A_{26}: S_2, A_{34}:S_3, A_{51}:S_5, state(A_{41}:S_4))

The Connectivity table is presented as follow:

C (Connectivity) =

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	0	0	0	0	0	0
S_2	0	0	0	1	0	0	0
S_3	0	0	0	1	0	0	0
S_4	1	1	1	0	1	0	1
S_5	0	0	0	1	0	0	0
S_6	0	0	0	0	0	0	0
S_7	1	1	0	1	0	0	0

As discussed in the description part, during the Pultrusion process, Resin Impregnation (S_2) should keep the temperature in the resin tank above the requirement (G_{21}). G_{21} is achieved by executing the actions (Turn on (A_{26}) / Turn off (A_{27}) the heater).

- S_2: G_{21}

For Pultrusion Die (S_3), three goals should be guaranteed during the whole process. They are: 1) prepare the pre-former according to the desired final product (G_{31}); 2) prepare the shape of the heated die for the material (G_{32}); 3) keep the temperature in the die constant (G_{33}). G_{31} is completed by running the action A_{31} (shape insert in the pre-former machine), G_{32} is by the action A_{32} (decide whether there will be a special tool) and A_{33} (adjust diameter of the heated die according to the desired final product), and G_{33} is achieved by Turning on (A_{34}) /Turning off the heater (A_{35}).

- S_3: G_{31}, G_{32}, G_{33}

In Pultrusion Process, Pulling System (S_4) has to complete the goal G_{41} (pull the material) by running the action A_{41} (pull the cross section) and A_{42} (adjust the pulling force).

- S_4: G_{41}

For Cutoff System (S_5), by executing A_{51} (start cutting the final product) and A_{52} (Adjust the cutting frequency), S_5 is able to cut the final product into pieces with desired length (G_{51}).

- S_5: G_{51}

The whole process should be executed under a certain environment with desired temperature (G_{61}) and humidity (G_{62}) guaranteed by Subsystem of Environmental Control-

ler (S_6). These two goals are crucial for the whole process especially for the quality of the final product.

- S_6: G_{61}, G_{62}

When there is any action happened triggering the Pulling System (S_4) to stop pulling, the whole system will stop functioning by running the actions: 1) turn off the heater in the resin bath (A_{27}) by Resin Impregnation (S_2); 2) turn off the heater in heated die (A_{35}) by Pultrusion Die (S_3); 3) stop cutting (A_{53}) by Cutoff System (S_5). Then the operators will fix the malfunctioned part and run the whole process again.

- Start(A_{27}: S_2, A_{35}:S_3, A_{53}:S_5, state(A_{43}:S_4))

Through the approach above, the whole production process can be described based on actions and goals each system should complete and their interface activities. Meantime the SoS characteristics could be evaluated which is useful in the phases of design and maintenance in understanding the whole system.

As a conclusion to this chapter, the use case of Pultrusion Process can be considered as a SoS application because of its heterogeneous structure: integration of human operators and technical systems. It was analyzed from two perspectives: SD and DD, which reflect an important view (Holarchical view introduced in state-of-the-art) in SoS engineering. Combing the results obtained in this chapter and the previous one, applicability of the generic modelling method to manufacturing systems was evaluated and presented.

6 CONCLUSIONS

Despite of great interest to the topic of SoS modelling on academy and industry, the research on computational modelling methods is still in its early stage and no application to factory automation was proposed yet [16]. New properties are emerging with integration of hardware, software, network systems and other new technologies to the domain of factory automation, making traditional system modelling methods ineffective. Consequently modelling methods which are capable to capture the properties (such as operational and managerial independence of the constituent systems, etc) are required for analysis of SoS.

Basing on the state-of-the-art, this thesis accepted the SoS definition which emphasises certain properties of SoS (like large-scale, integration, heterogeneous and independently operable, networked together and common goals) [10] and utilized the characteristics which differentiate traditional systems with real SoS (like Autonomy, Diversity, Belonging, Connectivity and Emergence [11]) as a theoretical background for the generic modelling method for SoS. The generic method is based on computation of SoS characteristics and interface description. Two concepts are introduced and they are Static Description and Dynamic Description, which are related to ‘Holarchical view’ that SoS can be treated as a whole and as a part simultaneously. Through actions, goals and activities between constituent systems, the generic method is assumed to describe the whole production process.

Two use cases are studied to evaluate applicability of the generic method to the domain of factory automation, and they are testbed and composite material manufacturing system. Through implementation, assumption is approved that the generic SoS modelling method is able to describe the whole production process and to evaluate SoS characteristics which can be utilized in design and maintenance phases in system engineering. The generic SoS modelling method was found applicable to model manufacturing systems.

Contributions

The following outcomes result from the work:

- SoS concept and characteristics are reviewed, analyzed and defined on basis of state-of-the-art.

- SoS modelling methods are reviewed in the thesis.

- A generic modelling method for SoS is introduced. It is based on computation of SoS characteristics and interface description which can be used in the phases of designing and maintenance in system engineering, to understand: 1) the number of functionalities each constituent system is able to execute; 2) the contribution each constituent system has done for the whole SoS with respect to different commissions; 3) connection situations; 4) relationships between system states and actions.

- Two use cases (testbed and composite material manufacturing system) are studies indicating the applicability of the generic modelling method to the domain of factory automation.

Limitations and Lessons Learned

- The generic method has shown its applicability to the domain of factory automation, however, the extension of the method to other domains should be further investigated.

- The use case studies have shown that the realization of the method depends on design of a particular system. Consequently more research should be done to make the method more generic.

Future Research Directions

- To deal with large amount of components or complicated deployment within a SoS, the proposed method can be further reinforced by the method of the deployment complexity reduction proposed in [45].

- Development has to be done for further maturing and generalizing the generic SoS modelling method. The domain of factory automation should be studied on a more detailed level and common properties should be concluded based on the

perspective of System of Systems. Related software platform has to be developed for analysis and simulation.

REFERENCES

[1] C. H. Kuo and H. P. Huang, Integrated manufacturing system modelling and simulation using distributed coloured timed petri net," in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, 12-15 Oct 1999, Tokyo, Japan, pp. 781-786.

[2] J. Wang, "Factory automation systems: evolution and trends," in *IEEE AUTOTESTCON*, 2002, pp. 880-886.

[3] H. Eisner, "A Systems Engineering Approach to Architecting a Unified System of Systems," in *Proc. of IEEE International Conference on Systems Man and Cybernetics*, 02-05 Oct 1994, San Antonio, USA, pp. 204-208.

[4] M. W. Maier, "Architecting Principles for Systems -of-Systems," in *Proc. of the 6th Aunnual Symposium of INCOSE*, 1996, pp 567-574, Also online URL: http://www.infoed.com/Open/PAPERS/systems.htm

[5] S. C. Cook, "On the acquisition of systems of systems," in *Proc. of the INCOSE 2001 Annual Symposium*, July 2001 Melbourne, Australia, 9 pp.

[6] T. Nanayakkara, M. Jamshidi, and F. Sahin, "System of Systems Simulation," in *Intelligent control systems with an introduction to system of systems engineering*, CRC Press Taylor & Francis Group, Boca Raton, 2010, pp. 190-200.

[7] V. Kotov, "Systems-of-Systems as Communicating Structures," in *Hewlett Packard Computer Systems Laboratory Paper* HPL-97-124, pp. 1–15.

[8] G. D. Wells and A. P. Sage, "Engineering of a System of Systems," in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, New Jersey, 2009, pp. 46-59.

[9] W. A. Crossley, "System of systems: An introduction of Purdue University schools of engineering's signature area," in *Proc. of Engineering Systems Symposium*, 31 Mar 2004, Cambridge, USA, pp. 1-10.

[10] M. Jamshidi, "Introduction to system of systems," in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, New Jersey, 2009, pp. 7-20.

[11] J. Boardman and B. Sauser, “System of systems – the meaning of OF,” in *Proc. of 2006 IEEE/SMC International Conference on System of Systems Engineering*, 24-26 Apr 2006, Los Angeles, USA, 6 pp.

[12] W. C. Baldwin and B. Sauser, “Modeling the characteristics of system of system,” in *Proc. of IEEE International Conference on System of Systems Engineering*, May 30 2009-Jun 3 2009, Albuquerque, USA, pp. 1-6.

[13] A. Gorod, R. Gove, B. Sauser, and J. Boardman, “System of systems management: A network management approach,” in *Proc. of IEEE International Conference on System of Systems Engineering 2007*, 16-18 Apr 2007, San Antonio, USA, pp. 1-5.

[14] J. E. Campbell, D. E. Longsine, D. Shirah, and D. J. Anderson, "System of systems modelling and simulation," in *Sandia National Laboratories*, March 23-25, 2005, NJ, USA, 12 pp.

[15] J. E. Campbell, D. E. Longsine, D. Shirah, and D. J. Anderson, "System of Systems Modelling and Analysis," in *Sandia National Laboratories*, January 2005, NJ, USA, 135 pp.

[16] B. Zhou, A. Dvoryanchikova, A. Lobov, and J.L. Matenez-Lastra, "Modelling System of Systems: A Generic Method Based on System Characteristics and Interface," in *Proc. of IEEE 9th International Conference on Industrial Informatics (INDIN 2011),* 26-29 July 2011, Lisbon, Portugal, 8 pp.

[17] J. O. Clark, “System of systems engineering and family of systems engineering from a standards, V-Model, and dual-V model perspective,” in *Proc. of 3rd Annual IEEE on Systems Conference*, 23-26 Mar 2009, Vancouver, Canada, pp. 381-387.

[18] D. A. DeLaurentis, "Appropriate modeling and analysis for systems of systems: Case study synopses using a taxonomy," in *Proc of IEEE International Conference on System of Systems Engineering 2008*, 2-4 Jun 2008, Singapore, pp. 1-6.

[19] J. O. Gutirrez-Garcia, F. F. Ramos-Corchado, and J.-L. Koning, "Obligations as constrainers, descriptors, and linkers of open system of systems," in *Proc. of IEEE International Conference on System of Systems Engineering 2009*, May 30-Jun 3 2009, Albuquerque, USA, pp. 1-6.

[20] M. Mansouri, B. Sauser, and J. Boardman, "Applications of Systems Thinking for Resilience Study in Maritime Transportation System of Systems," in *Proc. of*

3rd Annual IEEE on Systems Conference 2009, 23-26 Mar 2009, Vancouver, Canada, pp. 211-217.

[21] M. Duffy, B. Garrett, C. Riley, and D. Sandor, *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, New Jersey, 2009, pp. 409-442.

[22] J. E. Campbell, D. J. Anderson, C. R. Lawton, D. Shirah, and D. E. Longsine, "System of System modeling and simulation," in *Proc. of 3rd Conference on System Engineering Research (CSER 2005)*, March 23-25, Hoboken, NJ, USA, pp. 1-11.

[23] Y. Hata, S. Kobashi, and H. Nakajima, (2009), "Human Health Care System of Systems," in *Systems Journal, IEEE*, Jun 2009, pp. 231 - 238.

[24] P. Korba and I. A. Hiskens, "Operation and Control of Electrical Power Systems," in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, New Jersey, 2009, pp. 385-408.

[25] F. Sahin, "Robotic swarms as system of systems," in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, New Jersey, 2009, pp. 482-519.

[26] M. Jamshidi, *System of Systems engineering – principles and applications*, Taylor Francis CRC Publishers, Boca Ranton, FL, USA, 2009.

[27] R. Pei, "Systems of Systems Integration (SoSI) - A Smart Way of Acquiring Army C412WS Systems," in *Proc. of Summer Computer Simulation Conference*, 2000, pp. 574-579.

[28] P. G. Carlock and R. E. Fenton "System of systems (SoS) enterprise systems engineering for information-intensive organizations," in *Systems Engineering*, vol. 4, 2001, pp. 242.

[29] M. W. Maier, "Architecting principles for systems-of-system," in *Systems Engineering*, 4th Ed, Vol. 1, Willey, 1998, pp. 267-284.

[30] S. J. Luskasik, "Systems, Systems of Systems, and the competition," *Engineering Design, Analysis, and Manufacturing*, Vol. 12, No. 1, 1998, pp. 55-60.

[31] W. H. J. Manthorpe, "The emerging joint system of systems: A systems engineering challenge and opportunity," for APL, *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 17, 1996, pp. 305.

[32] M. V. José, "Fundamentals of multi-agent systems with netlogo examples," unpublished.

[33] B. Sauser, J. Bordman, and A. Gorod, "System of Systems Management," in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. John Willey & Sons Inc, 2009, New Jersey, USA, pp. 201-207.

[34] M. J. DiMario, J. T. Boardman, and B. J. Sauser, "System of Systems Collaborative Formation," *IEEE Systems Journal*, Sep. 2009, pp. 360-368.

[35] J. Boardman and B. Sauser, "From prescience to emergence: Taking hold of system of systems management," in *Proc. of 27th Annual National Conference of American Society for Engineering Management*, 2006, Huntsville, USA, pp. 447-452.

[36] A. Meilich, "System of systems (SoS) engineering & architecture challenges in a net centric environment," in *Proc of IEEE International Conference on System of Systems Engineering 2006*, 24-26 April 2006, CA, USA, 5 pp.

[37] N. Kilicay-Ergin and C. Dagli, "Executable modeling for system of systems architecting: an artificial life framework," in *Proc. of IEEE International Systems Conference (SySCon 2008)*, April 7-10, Montreal, Canada, pp. 1-5.

[38] H. M. Joh and E. P. Scott, "Complex adaptive social systems in one dimension," in *Complex Adaptive Systems*, Princeton University Press, New Jersey, USA, 2007, pp. 129-131.

[39] A. Gorod, B. Sauser, and J. Boardman, "Paradox: Holarchical view of system of systems engineering management," in *Proc. of IEEE International Conference on System of Systems Engineering 2008*, 2-4 Jun 2008, Singapore, pp. 1-6.

[40] Y. Correa and C. Keating, "An approach to model formulation of system of systems," in *Proc. of IEEE International Conference on Systems Man and Cybernetics (SMC'03)*, October 5-8, Washington D.C., USA pp. 3353-3358.

[41] D. A. DeLaurentis, "Appropriate modelling and analysis for systems of systems: Case study synopses using a taxonomy, " in *Proc of IEEE International Conference on System of Systems Engineering 2008*, 2-4 Jun 2008, Singapore, pp. 1-6.

[42] D. Harel and A. Naamad, “The STATEMATE Semantics of Statecharts,” in *Transactions on Software Engineering and Methodology (TOSEM),* vol. V, Issue 4, ACM, NY, USA, 1996, pp. 293-333.

[43] B. Zhou, A. Dvoryanchikova, A. Lobov, J. Minor and J.L. Martinez-Lastra, "Application of the Generic Modelling Method for System of Systems to Manufacturing Domain," in Proc. of *the 37th Annual Conference of the IEEE Industrial Electronics Society (IECON2011),* 7-10 Nov 2011, Melbourne, Australia, 7 pp.

[44] I. Vilovic, N. Burum and R. Nadj, "Estimation of dielectric constant of composite materials in buildings using reflected fields and PSO algorithm," in Proc of *the Fourth European Conference on Antennas and Propagation (EuCAP),* 12-16 April 2010, Barcelona, Spain, 5 pp.

[45] J. J. Simpson and M. J. Simpson, “System of Systems Complexity Identification and Control,” in *Proc. of IEEE International Conference on System of Systems 2009,* May 30 2009-Jun 3 2009, Albuquerque, USA, pp. 1-6.