FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Models for innovative factory layout design techniques and adaptive reconfiguration in the automotive industry

José Manuel da Rocha Pimenta Soares

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Gil Manuel Magalhães de Andrade Gonçalves, PhD

February 26, 2016

Resumo

À medida que a pressão competitiva de países emergentes aumenta e se faz sentir na europa, os líderes da indústria europeia são forçados a tomar a difícil decisão de cortar não apenas nos custos não apenas dos produtos, mas também nos investimentos em equipamento, planeamento das fábricas, operação et... tudo isto sem comprometer a qualidade que a caracteriza.

O projecto ReBORN, procura desenvolver uma série de estratégias e tecnologias que suportam um novo paradigma para a reutilização de equipamentos de produção industriais no momento da concepção e planeamento de novas linhas de produção. Estas estratégias abordam questões relevantes na indústria actual, como a falta de uma metodoliogia mais abrangente que inclua métodos para aferir o custo total de vida, o impacto ambiental e a fiabilidade das máquinas a serem instaladas, no processo de planeamento das novas linha de produção.

No contexto do projecto uma ferramenta de simulação (High Performance Simulation tool) está a ser criada pelos vária parceiros envolvidos, que ajudará na decisão da escolha das melhores máquinas a integrar uma dada linha de produção. Esta ferramenta cria vários *layouts* possíveis que serão depois simulados com o intuito de aferir o seu *Life-cycle Cost* (custo de vida), *Life-cycle Assessment* (análise ao impacto ambiental) e a sua fiabilidade.

O trabalho desenvolvido nesta dissertação centrou-se em desenvolver uma ferramenta que facilite a criação de modelos de simulação das linhas de produção, de forma a aferir a sua fiabilidade.

Desta forma, foi desenvolvido um interface que controla remotamente o Flexsim - software de simulação usado. Este interface gera automaticamente o modelo das linhas de produção no Flexsim, com base na sua descrição num ficheiro XML.

Do lado do Flexsim, foi criado um novo objecto de simulação que representa as máquinas, cujas funcionalidades adicionais permitem complementar os modelos gerados pelo interface.

Abstract

With the competitive pressure form emerging countries increasing, European manufacturing industry leaders have to rethink the costs of not only products, but also investments in equipment, factory planing, ramp-up and operation. All of this without compromising the quality of its products.

The ReBORN project seeks to demonstrate technologies and strategies that support the a new paradigm for re-use of production equipment when designing installing new production lines. These strategies and technologies will address relevant issues in today's manufacturing, such as the lack of comprehensive design methodology for manufacturing systems or the nonexistence of appropriate methods for dynamic cost assessment for system design and system life-cycle optimization.

A High Performance Simulation tool is being developed by the several partners that will assist the new production line planner in making the best choices regarding machines and other production equipment. This tool creates several possible layouts for a job description based on the capabilities of the machines and equipment available, that is then simulated in order to assess the production machines regarding their Life-cycle Cost, Life-cycle Assessment and their Reliability.

The work developed in this dissertation focused on developing a tool that facilitates the creation of simulation models of the production lines in order to assess their reliability.

An Interface was developed that controls Flexsim - the simulation software used- remotely. This Interface generates the production lines' simulation models automatically from their descriptian in XML files. On Flexsim's side a new class of Processor was created, that has added features to complete the simulation model generated by the Interface.

Acknowledgments

I would like to thank Professor Gil Gonçalves, for the guidance and opportunities given.

I would like to thank Susana Aguiar from SystecFoF for the valuable input during the implementation of this work.

Finally, I would like my family, my parents in particular, for their unconditional support.

José Soares

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Scope 1
1.3 Goals 2
1.4 Document Overview 2

2 Literature Review **3**
2.1 Manufacturing Systems 3
2.1.1 Flexible Manufacturing Systems 3
2.1.2 Reconfigurable Manufacturing Systems 4
2.1.3 Intelligent Manufacturing Systems 5
2.2 Lifecycle Costing 7
2.3 Life Cycle Assessment 8
2.4 Reliability Assessment 10
2.5 Reliability Modeling 11
2.5.1 Basic models for Parts (Life Distribution Models) 12
2.5.2 Physical Acceleration 14
2.5.3 Bottom-up approach to System Reliability 16
2.5.4 Basic Models for Repairable systems 19
2.6 Overall Equipment Effectiveness 19
2.7 Manufacturon 20
2.8 Simulation 20
2.8.1 Limitation to current simulation techniques 21
2.8.2 Simulation Package - Flexsim 22
2.9 Data Format for Information Exchange 22
2.9.1 AutomationML 23
2.9.2 Core Manufacturing Simulation Data 24

3 Problem Analysis **25**
3.1 VERSONs 25
3.2 Overview of the High Performance Simulation Tool 26
3.3 Problem specification 28
3.3.1 Parameters for the metrics 31
3.3.2 Conclusion 32

4 Machine Conceptual Models **33**
4.1 Machine as a system 33
4.1.1 Socket Level 34

4.2 Lifetime modeling 34
4.2.1 Lifetime distributions 35
4.2.2 Accelerated Lifetime models 35
4.2.3 Lifetime information 36

5 Implementation **37**
5.1 Overview of Process 37
5.2 Data Structure 38
5.3 Flexsim Remote Control 40
5.3.1 Explored alternatives 40
5.3.2 Solution adopted 41
5.4 Interface 41
5.4.1 Overview of the Interface 42
5.4.2 Automatic model generation from XML 43
5.5 Flexsim Models 44
5.5.1 Basic Flexsim concepts 44
5.5.2 Implemented Models 45
5.5.3 Shortcomings in the implementation 46
5.5.4 Outcome of the Simulations 47

6 Conclusions and Future Work **49**
6.1 Conclusions 49
6.2 Future Work 50

A Common formulas and plots for lifetime models **55**
A.1 Exponential 55
A.2 Weibull 57
A.3 Lognormal 59

B Schematics for bottom-up Reliability **63**
B.1 Series Model 63
B.2 Parallel Model 64
B.3 Complex Model 65

List of Figures

2.1 Manufacturing system model . . . 4
2.2 Life Cycle stages [1] . . . 9
2.3 Phases of an LCA [1] . . . 9
2.4 Example of a system reliability model [2] . . . 11
2.5 Bathtub Curve for parts [3] . . . 13
2.6 Example of a production line model in Flexsim . . . 23
2.7 Structure of AutomationML Projects [27] . . . 24

3.1 ReBORN high performance simulation tool [4] . . . 27
3.2 Process of the subsystem being implemented at FEUP . . . 28
3.3 Comparison of 2 Reliability Assessments in the web application developed at FEUP 30
3.4 Example of a comparison of an LCC assessment in two machines . . . 30

4.1 Machine Conceptual Model . . . 34
4.2 Illustration of Renewal Process [5] . . . 34
4.3 Example of evolution of step-load levels over operating time . . . 35
4.4 Equivalent failure probability with 3 different load conditions . . . 36

5.1 Overview of the solution's methodology . . . 37
5.2 Example of a layout defined in XML . . . 39
5.3 Example of a layout defined in XML . . . 43

A.1 Exponential PDF plots [3] . . . 56
A.2 Exponential CDF plots [3] . . . 56
A.3 Examples of Weibull PDF plots [3] . . . 58
A.4 Weibull Failure Rate plots for different values of β [3] . . . 58
A.5 Examples of Lognormal PDF plots [3] . . . 60
A.6 Examples of Lognormal Failure rate plots [3] . . . 61

B.1 Example of a Series model [3] . . . 63
B.2 Example of a Series model [3] . . . 64
B.3 Example of a Complex model [3] . . . 65

Abbreviations

ABS	Agent Based Simulation
CDF	Cumulative Distribution Function
CIM	Computer Integrated Manufacturing
CIM	Computer Integrated Manufacturing
CMSD	Core Manufacturing Simulation Data
CNC	Computer Numerical Control
COM	Component Object Model
COM	Component Object Model
DES	Discrete Event Simulation
DES	Discrete Event Simulation
DLL	Dinamic-Link Library
DLL	Dynamic-link library
DMS	Distributed Manufacturing System
DOM	Document Object Model
ERP	Enterprise Resource Planning
FMS	Flexible Manufacturing System
FOM	Force of Mortality
GUI	Graphical User interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IID	Independent and Identically Distributed
IMS	Computer Aided Design
IMS	Intelligent Manufacturing System
LCA	Life Cycle Assessment
LCC	Life Cycle Costing
MAS	Muli-Agent System
MAS	Multi-agent Systems
MRP2	Manufacturing Resource Planning
MTBF	Mean Time Before Failure
MTTF	Mean Time to Failure
MTTR	Mean Time To Repair
OEE	Overall Equipment Efficiency
CIM	Computer Integrated Manufacturing
PDF	Probability Density Function
RMS	Reconfigurable Manufacturing System
SD	Computer Aided Design
VERSON	Versatile, Flexible Lifecycle Extended Devices
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

Industrial manufacturing is characterized by fierce competition on a global scale. The pressure to cut costs from emerging countries, forces the European industry to think over the costs of both the products, as well as their investments in equipment, factory planing, ramp-up and operation without compromising the quality of its products.

The ReBORN project, is an European consortium that incorporates academic and research institutions as well as companies in the industrial manufacturing sector. It envisions the demonstration of strategies and technologies that support a new paradigm for the re-use of production equipment in old, renewed and new factories, maximizing the efficiency of this re-use and making the factory design process much easier and straight forward, shortening ramp-up times and increasing production efficiency and flexibility. This paradigm will give new life to decommissioned production systems and equipment, making it possible their "reborn" in new production lines

1.2 Scope

A High Performance Simulation tool is being developed by several partners in ReBORN which will take advantage of system information for real time system adaptation and use it in improving overall system performance throughout its life-cycle. The purpose is to tackle relevant issues in today's manufacturing, such as the lack of comprehensive design methodology for manufacturing systems or the nonexistence of appropriate methods for dynamic cost assessment for system design and system life-cycle optimization.

The work being developed at FEUP is framed in a subsystem of the aforementioned tool and it aims at enabling the assessment and comparison of various potential change, upgrade, reuse, dismantle and disposal scenarios and the corresponding reconfiguration effect on the overall manufacturing system cost, performance and status throughout the life-cycle.

1.3 Goals

The work done for this thesis focuses on providing a tool that facilitates the creation of simulation models of production lines in which the equipment is detailed in a XML based data format, with the main purpose of producing various metrics that will be used in Reliability and Life Cycle Cost assessments. If the inputs are detailed enough these models may also produce useful metrics for Life Cycle Assessments.

1.4 Document Overview

The Literature Review in Chapter 2 introduces important concepts for ReBORN and the tool under development, as well as the theoretical concepts that support the models of the machines to be implemented. Chapter 3 presents an overview of ReBORN, and proceeds to detail the subsystem to be addressed. In Chapter 4 the concept of the machines that integrate the production lines is exposed. Chapter 5 describes the implementation of the tool, specifying its several components and how they were developed, their integration and shortcomings in the implementations of the models. In Chapter 6 Conclusions are drawn and Future work is suggested.

Chapter 2

Literature Review

In this chapter, an introduction is made to the concepts of LCC, LCA and Reliability, which are in the core of the process studied in this thesis. In the next section a brief account of the evolution of manufacturing systems in the 20th century is made and finally other concepts that support ReBORN like Intelligent Agents, Simulation and previous related projects are also introduced.

2.1 Manufacturing Systems

Like other business sectors, industry is driven by profit, reputation and market share. From the turn of the 20th century, with the invention of the moving assembly line by Henry Ford and the introduction of Mass Production as a paradigm until the present day, enterprises must implement manufacturing systems that are able to meet market demands and effectively and efficiently address the requirements of the environment in which they operate. Currently, these requirements are understood to be the following [6] :

- Short lead time;
- More variants;
- Low and fluctuating volumes;
- Low price.

Figure 2.1 shows a model of a manufacturing system.

In the following sections a brief description of different paradigms in industry are described, with special relevance for Intelligent Manufacturing Systems, and its related concepts.

2.1.1 Flexible Manufacturing Systems

In the 1980's Flexible Manufacturing Systems (FMS) were introduced. FMS is a programmable machining system configuration which incorporates software to handle changes in work orders,

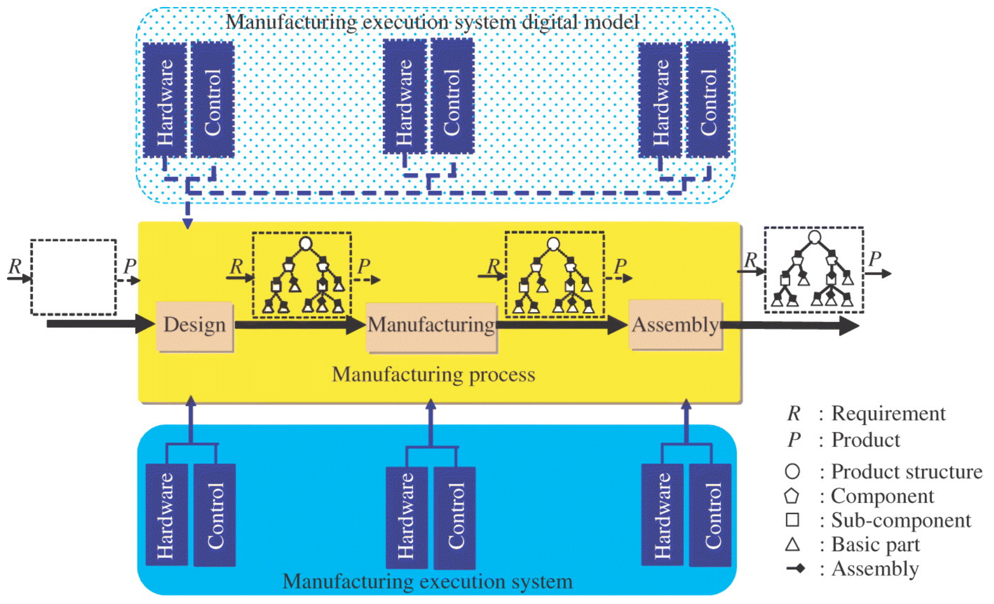


Figure 2.1: Manufacturing system model

production schedules, part-programs, and tooling for production of a family of parts. The objective of a FMS is to make possible the manufacture of several families of parts, with shortened changeover time, on the same system. [7]

2.1.2 Reconfigurable Manufacturing Systems

In the mid 1990's, a new concept was introduced, the Reconfigurable Manufacturing System(RMS). It is basically a machining system which can be created by incorporating basic process modules- both hardware and software that can be rearranged or replaced quickly and reliably. [7] Reconfiguration allows adding, removing, or modifying specific process capabilities, controls, software or machine structure to adjust production capacity in response to changing market demands or technologies. [7] RMS provides customized flexibility for a particular part-family and will be open-ended, so that it can be improved, upgraded and reconfigured, rather than replaced. RMS are marked by six core reconfigurable characteristics [8]:

- Customization
- Convertibility
- Scalability
- Modularity
- Integrality
- Diagnosability

The components of RMS are CNC machines, Reconfigurable Machine Tools, Reconfigurable Inspection Machines and material transport systems (such as gantries and conveyors) that connect the machines to form the system. Different arrangements and configurations of these machines will have an impact on the system productivity [6].

The purpose of an RMS is to provide exactly when it is needed. RMS goes beyond the objectives of FMS by permitting [7]:

- Reduction of lead time for launching new systems and reconfiguring existing systems
- The rapid modification and quick integration of new technology and/or new functions into existing systems

2.1.3 Intelligent Manufacturing Systems

In order to meet the new requirements listed in 2.1, the concept of **Computer Integrated Manufacturing** (CIM) was created. It is characterized by the centralized control of a completely automated factory, where decision-making is broadcast from higher hierarchical levels down to operational units. It integrates operational functions and business level tools, like Manufacturing Resource Planning (MRP2) and Enterprise Resource Planning (ERP). On the downside, the complexity of its structures and the high investments necessary can lead to rigid systems [9].

In order for companies to survive fierce market competition and volatility, as well as the demand for more flexibility and reactivity, they had to become more **agile**. Agility can be seen as "the ability to operate with a high level of coordination and pro-activity throughout the supply chain, simultaneously reacting to disturbances on the shop floor efficiently while taking the increasing process complexity (e.g., variabilities, high product variety, reconfiguration issues) into account" [10].

Because of CIM rigid nature and therefore inability to be agile, researchers turned to **distributed** architectures, Distributed Manufacturing Systems (DMS), where autonomous entities have the capacity to make decisions without a centralized view. These systems are called Intelligent Manufacturing Systems (IMS) and they are characterized by: [11]:

- autonomy,
- decentralization,
- reliability
- flexibility
- efficiency
- learning
- self-regeneration

Several approaches to IMS have been proposed like Fractal Manufacturing Systems, Bionic Manufacturing Systems, Holonic Manufacturing Systems, Product Driven Manufacturing Systems and Agent-Based Manufacturing [9] [10], combining elements of a centralized control e.g. its predictability, with the agility and robustness of DMS [10]. The common ground to these approaches are the concepts of Agent 2.1.3.1 and Multi-Agent System 2.1.3.2.
In general, IMS need to meet certain requirements such as: [12]

- Full integration of heterogeneous software and hardware systems within an enterprise, a virtual enterprise, or across a supply chain;
- Open system architecture to accommodate new subsystems (software or hardware) or dismantle existing subsystems "on the fly";
- Efficient and effective communication and cooperation among departments within an enterprise and among enterprises;
- Embodiment of human factors into manufacturing systems;
- Quick response to external order changes and unexpected disturbances from both internal and external manufacturing environments;
- Fault tolerance both at the system level and at the subsystem level so as to detect and recover from system failures and minimize their impacts on the workflow environment.

2.1.3.1 Agents

An agent is a software entity that has a set of protocols which governs the operations of the manufacturing entity, a knowledge base, an inference mechanism and an explicit model of the problem to solve [13]

From a point of view of a Distributed Manufacturing System, an agent has the following characteristics [9]:

- is autonomous;
- can represent physical resources (e.g. robots);
- can represent logical objects (e.g. schedulers, orders);
- has intelligence to make own decisions and act in order to achieved its goals (process planning, scheduling, etc);
- has the capability to interact with other agents (also with humans) and cooperate if it doesn't possess knowledge and skills to reach alone its objectives;

- can interact in the environment where is inserted (e.g. production environment) feeling and changing it based on the knowledge that it contains;
- reacts to context incentives and defines actuation plans based in his knowledge;
- can decide if it accepts or rejects a service requested by other agent, based in its knowledge and skills;
- has capacity to acquire and to memorize new knowledge.

2.1.3.2 Multi-Agent Systems

A Multi-Agent System (MAS) is a system composed of different agents that interact with each other in planning and executing their responsibilities. In manufacturing, MAS view the supply chain as composed of a set of intelligent (software) agents. Each agent is autonomous, goal-oriented software process that operates asynchronously, communicating and coordinating with other agents as needed [11]. Almeida et. al. identified the major barriers to the adoption of MAS by the manufacturing industry as being the deficient security of agent execution and communication, the complexity of the system, the low level of scalability, the need of standardization and the need of a better integrated human-machine interaction.

2.2 Lifecycle Costing

Life Cycle Costing (LCC) is a process to determine the sum of all the costs associated with an asset or with part of an asset. These costs include acquisition, installation, operation, maintenance, refurbishment, and disposal. LCC can be carried out during any or all phases of an asset's life cycle. LCC process usually includes steps such as [14]:

1. Life Cost Planning: concerns the assessment and comparison of options/alternatives during the design/ acquisition phase;
2. Selection and development of LCC model (e.g. designing cost breakdown structure, identifying data sources and uncertainties;
3. Application of LCC model;
4. Documentation and review of LCC results;
5. Prepare Life Cost Analysis;
6. Implement and Monitor Life Cost Analysis.

The ultimate goal for carrying out LCC calculations is to aid decision making in:

- Assessing and controlling costs and identifying cost significant items.
- Producing selection of work and expenditure planning profiles.

LCC allows for economic justification for the sustainability considerations, as implementing LCC in planning for construction projects shows that, over a project's life, incorporation of sustainable elements proves cost-effective as well as environmentally beneficial.

Early identification of acquisition and ownership costs enables the decision-maker to balance performance, reliability, maintainability, maintenance support and other goals against life cycle costs. Decisions made early in a asset's life cycle have a much greater influence on Life Cycle Costing than those made late in a asset's life cycle, leading to the development of the concept of discounted costs.

2.3 Life Cycle Assessment

Life Cycle Assessment (LCA) is a methodological framework for estimating and assessing the environmental impacts attributable to the life cycle of a product, such as climate change, stratospheric ozone depletion, tropospheric ozone (smog) creation, eutrophication, acidification, toxicological stress on human health and ecosystems, the depletion of resources, water use, land use, and noise—and others. [1]

Life cycle assessment is a "cradle-to-grave" approach for assessing industrial systems. "Cradle-to-grave" begins with the gathering of raw materials from the earth to create the system and ends at the point when all materials are returned to the earth. LCA evaluates all stages of a product's life from the perspective that they are interdependent, meaning that one operation leads to the next. LCA enables the estimation of the cumulative environmental impacts resulting from all stages in the product life cycle, often including impacts not considered in more traditional analyses (e.g., raw material extraction, material transportation, ultimate product disposal, etc.). By including the impacts throughout the product life cycle, LCA provides a comprehensive view of the environmental aspects of the product or process and a more accurate picture of the true environmental trade-offs in product and process selection.

The LCA process is a systematic, phased approach and consists of four components as illustrated in Figure 2.2, page 9:

- **Goal Definition and Scope -** Define and describe the product, process or activity. Establish the context in which the assessment is to be made and identify the boundaries and environmental effects to be reviewed for the assessment;
- **Inventory Analysis -** Identify and quantify energy, water and materials usage and environmental releases (e.g., air emissions, solid waste disposal, waste water discharges);

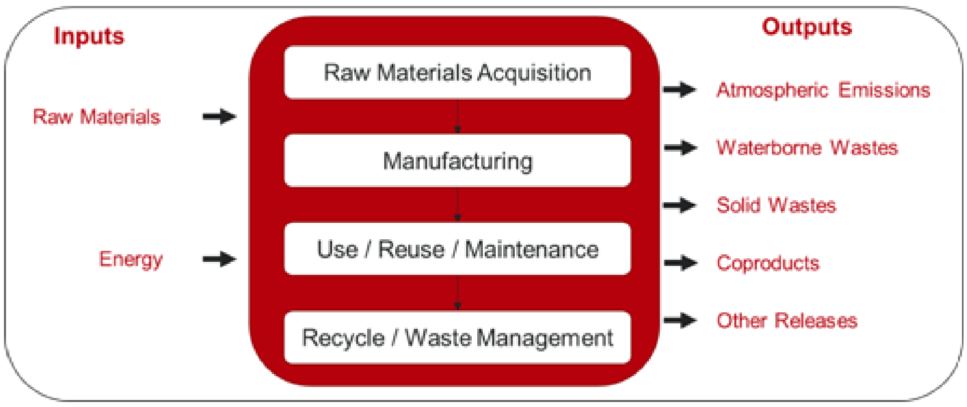


Figure 2.2: Life Cycle stages [1]

- **Impact Assessment -** Assess the potential human and ecological effects of energy, water, and material usage and the environmental releases identified in the inventory analysis. Inputs and outputs are categorized in different midpoint (climate change, land use,...) and endpoint (human health, resource depletion,...) impact categories;
- **Interpretation -** Evaluate the results of the inventory analysis and impact assessment to select the preferred product, process or service with a clear understanding of the uncertainty and the assumptions used to generate the results;

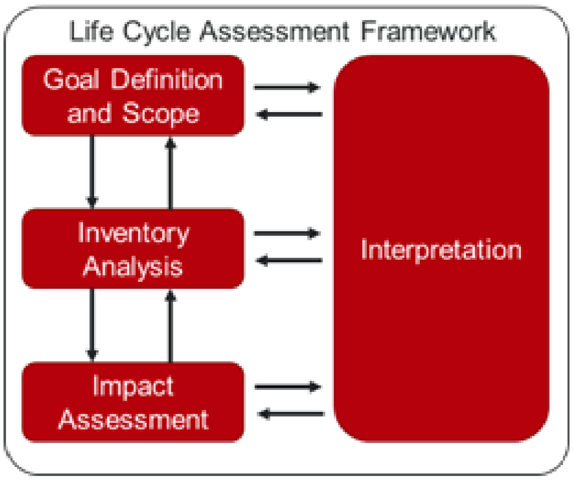


Figure 2.3: Phases of an LCA [1]

2.4 Reliability Assessment

Two of the most important factors that demonstrate the capability of any manufacturing system to deliver its uninterrupted and on-time service are the **Reliability** and **Maintainability** of its machinery and equipment. [15] But before proceeding the study of those two concepts, one should understand what *failure* is. Failure can be essentially be defined in two ways [16]:

- The termination of the ability of the product as a whole to perform its required function.
- The termination of the ability of any individual component to perform its required function. but not the termination of the ability of the product as a whole to perform.

Reliability measures the frequency of failures, interruptions, or needed adjustments and corrections, whereas the availability measures the duration of those events.

The less often a piece of equipment fails, and the less time it requires to be brought back to an operable state should it fail, the more reliable and maintainable it is in comparison with a machine that fails more frequently and that takes a longer period of time to repair.

Reliability and maintainability of any machinery or equipment are functions of the reliability and maintainability of its components, subsystems, and systems. Improved reliability and maintainability leads to lower total life-cycle costs that are necessary to maintain the competitive edge. [15]

Equipment reliability can be quantified by a measure of Mean Time Between Failures (MTBF) (eq.2.1), which is the average time between failure occurrences. Equipment maintainability can be quantified by Mean Time To Repair (MTTR), which is the average time it takes to fix a failure. Together, MTBF and MTTR make up the equipment's availability (A), which is the percentage of time the equipment is available for production.

$$MTBF = \frac{total\ uptime}{no.\ of\ breakdowns} \tag{2.1}$$

$$MTTR = \frac{total\ downtime}{no.\ of\ breakdowns} \tag{2.2}$$

$$Reliability: \quad R = e^{-(\frac{time}{MTBF})} \tag{2.3}$$

$$Availability: \quad A = \frac{MTBF}{(MTBF + MTTR)} \tag{2.4}$$

More formally, reliability is the probability that machinery /equipment can perform continuously without failure, for a specified interval of time when operating under stated conditions. Increased reliability implies less failure and consequently less downtime and loss of production.

Maintainability is a characteristic of design, installation, and operation, usually expressed as the probability that a machine can be restored to specified operable condition (returned to a serviceable state) within a specified interval of time when maintenance action is performed in accordance with prescribed procedures and resources. [15]

2.5 Reliability Modeling

When modeling the Reliability and Lifetime of products, it is important to understand the object under study and distinguish between parts, components, systems and subsystems. A level must be chosen at which no more detailed information about the object of analysis is known or needs to be considered. At that point, the object of analysis as a "black box" [2]. The selection of this level (e.g., component, subassembly, assembly or system) determines the detail of the subsequent analysis.

System reliability analysis focuses on building models that represent the times-to-failure of entire systems based on the life distributions of its components, subassemblies and/or assemblies ("black boxes") [2], as illustrated in the figure 2.4.

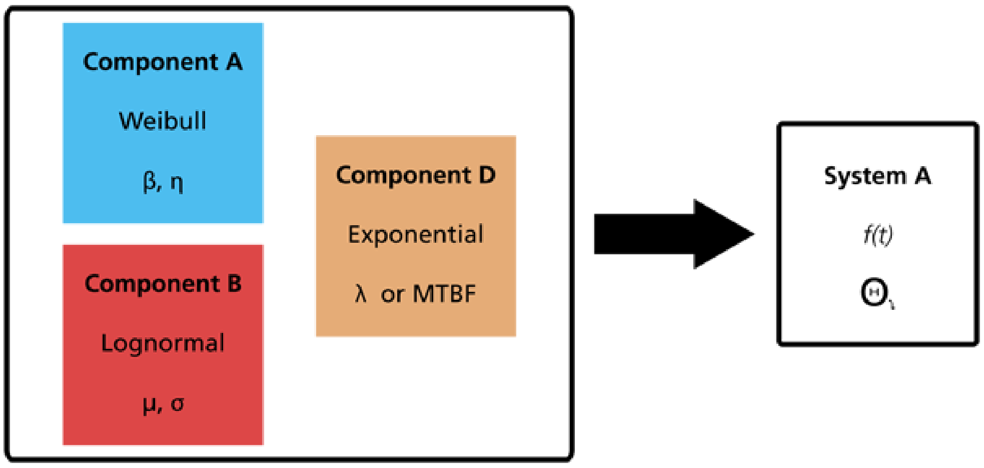


Figure 2.4: Example of a system reliability model [2]

In other words, a system is a collection of parts, components, subsystems and/or assemblies arranged to a specific design in order to achieve desired functions with acceptable performance and reliability.

Asher and Feingold in [17] define the basic terminology used in systems reliability as follows:

1. *Part* - An item which is not subject to disassembly and hence, is discarded the first time it fails.

2. *Socket* - A circuit or equipment position which, at any given time, holds a part or component of a given type.

3. *System* - A collection of two or more sockets and their associated parts, interconnected to perform one or more functions.

4. *Nonrepairable system* - A system that is discarded the first time that it ceases to perform satisfactorily.

5. *Repairable system* - A system which, after failing to perform at least one of its required functions, can be restored to performing all of its functions by any method, other than replacement of the entire system.

2.5.1 Basic models for Parts (Life Distribution Models)

Life Distribution Models describe how populations of non-repairable units (i.e. Parts) fail over time. If T is defined as the random variable "time to failure of a part", then a lifetime distribution model can be any probability density function (PDF) $f(t)$ defined over the range of time from $t = 0, \ldots, \infty$. The corresponding cumulative distribution function (CDF) $F(t)$ gives the probability that a randomly selected unit will fail by time t. $F(t)$ is also called **Failure probability**, function and is given by:

$$F(t) = \int_{-\infty}^{t} f(s)ds \tag{2.5}$$

The **Reliability** (or Survival) function is the probability a part survives beyond time t and can be defined by

$$R(t) = 1 - F(t) = P(T > t) \int_{t}^{\infty} f(s)ds \tag{2.6}$$

If $f(t)$ is absolutely continuous , then

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{R(t)} = \text{the } \textbf{instantaneous Failure rate}. \tag{2.7}$$

The probability that a part will fail in the time interval $[t, t + \Delta t]$ when it is known that the part is working at time t is:

$$P(t < T \leqslant t + \Delta t | T > t) = \frac{P(t < T \leqslant t + \Delta t)}{R(T > t)} = \frac{F(t + \Delta t) - F(t)}{R(t)}$$

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t < T \leqslant t + \Delta t | T > t)}{\Delta t}$$

$$h(t) = \lim_{\Delta t \to 0} \frac{1}{R(t)} \frac{F(t + \Delta t) - F(t)}{\Delta t}$$

$$h(t) = \frac{f(t)}{R(t)}$$

Failure rate in time interval $[t_1, t_2]$ is $\frac{R(t_1)-R(t_2)}{(t_2-t_1)R(t_1)}$. The quantity $h(t)dt$ is the conditional probability that a part of age t will fail in the interval $[t, t+\Delta t]$.

The **instantaneous Failure rate** is also known as **Hazard function** or **Force of Mortality** (FOM).

The name **Cumulative Hazard** function is derived from the fact that

$$H(t) = \int_0^t h(s)\,ds \tag{2.8}$$

which is the "accumulation" of the hazard over time. The avarage hazard rate between two times is $AHR(t_1, t_2) = \frac{H(t_2)-H(t_1)}{t_2-t_1}$.

The Mean Time to Failure (MTTF) of a population of parts with PDF f is

$$\text{MTTF} = E(T) = \int_0^\infty t f(t)\,dt \tag{2.9}$$

The expected time to failure of a part that has already survived to t, its **Mean Residual life**, is given by

$$r(t) = E(T-t|T>t)$$

$$r(t) = \frac{1}{R(t)}\int_t^\infty (s-t)f(s)\,ds = \frac{1}{R(t)}\int_t^\infty R(s)ds \tag{2.10}$$

If enough parts from a given population are observed operating and failing over time, it is relatively easy to compute periodic estimates of the failure rate $h(t)$. It is common for the plot of $h(t)$ over time to yield a curve similar to the one in figure 2.5.

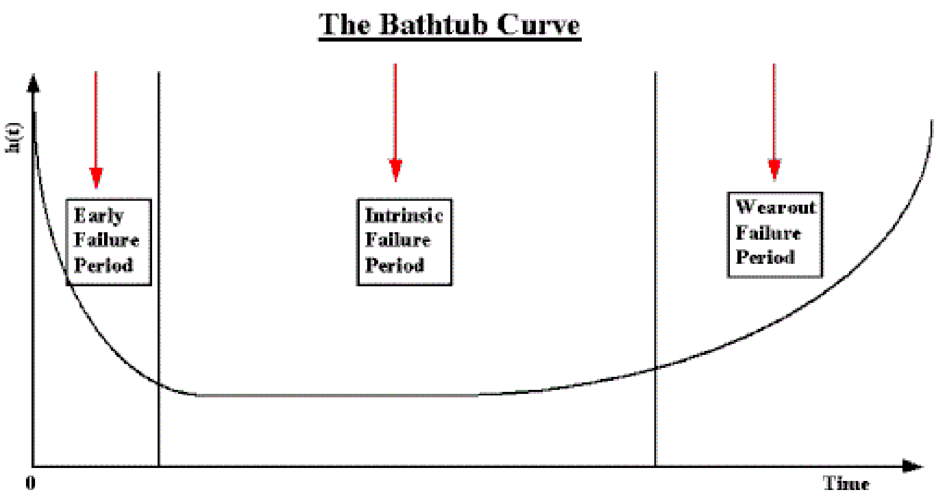


Figure 2.5: Bathtub Curve for parts [3]

The initial region, **Early Failure Period** (or **Infant Mortality Period**) that begins at $t = 0$ and is characterized by a high but rapidly decreasing failure rate. This behavior is attributed to several factors that are related to manufacturing errors, imperfections, deviations from standards, poor quality, and human factors.

Next, the failure rate levels off and remains roughly constant for the majority of the useful life of the part. This is known as the **Intrinsic Failure Period** (also called the **Stable Failure Period**).

If parts from the population remain in use long enough, the failure rate begins to increase as materials wear out and degradation failures occur at an ever increasing rate. This is the **Wearout Failure Period**.

Note: It is important to mention that the bathtub curve applies **only for a population of parts**. A similar curve can be applied to Repairable Systems (see 2.5.4), but the statistical **interpretation differs** greatly. Asher and Feingold dedicate a section in [17] (p. 136) to clear this common misconception.

Due to its flexibility, the Weibull distribution is often used to model the lifetime of parts. Other typical PDF's used are the Exponential and Lognormal [3]. For more information on the formulas, parameters and plots of each distribution see Appendix A

2.5.2 Physical Acceleration

Physical Acceleration (sometimes called **True Acceleration** or just **Acceleration**) means that operating a unit at high stress (i.e., higher temperature or voltage or humidity or duty cycle, etc.) produces the same failures that would occur at typical-use stresses, except that they happen much quicker.

When there is true acceleration, changing stress is equivalent to transforming the time scale used to record when failures occur. The transformations commonly used are linear, which means that time-to-fail at high stress just has to be multiplied by a constant (the **acceleration factor**) to obtain the equivalent time-to-fail at use stress. If t_s is time-to-fail at stress and t_u is corresponding time-to-fail at use, then an acceleration factor AF between stress and use means the following relationships hold [3]:

Time-to-Fail:	$t_u = AF \times t_s$
Failure Probability:	$F_u(t) = F_s(t/AF)$
Reliability:	$R_u(t) = R_s(t/AF)$
Failure Rate:	$h_u(t) = (1/AF)h_s(t/AF)$
PDF :	$f_u(t) = (1/AF)f_s(t/AF)$

A model that predicts time-to-fail as a function of stress is an **Acceleration model**. If $t_f = G(S)$, with $G(S)$ denoting the model equation for an arbitrary stress level S, then the acceleration factor between two stress levels S_1 and S_2 can be evaluated simply by $AF = G(S_1)/G(S_2)$. [3]

Acceleration models are usually based on the physics or chemistry underlying a particular failure mechanism.

Two of the most common acceleration models are presented in the next sections. These two models as well as several others are described in [3].

2.5.2.1 Arrhenius

The Arrhenius model predicts failure acceleration due to temperature increase. It takes the form [3]:

$$t_f = A \cdot \exp\left[\frac{\Delta H}{kT}\right] , \tag{2.11}$$

where

- T is the temperature measured in degrees Kelvin at the point when the failure process takes place
- k is Boltzmann's constant (8.617e-5 in ev/K)
- A is a scaling factor that drops out when calculating acceleration factors
- ΔH is the activation energy, which is the critical parameter in the model

The acceleration factor between a higher temperature T_2 and a lower temperature T_1 is given by

$$AF = \exp\left[\frac{\Delta H}{k}\left(\frac{1}{T_1} - \frac{1}{T_2}\right)\right] . \tag{2.12}$$

2.5.2.2 Eyring

The Eyring model has a theoretical basis in chemistry and quantum mechanics and can be used to model acceleration when many stresses are involved. The model for temperature and one additional stress takes the general form:

$$t_f = AT^{\alpha}\exp\left[\frac{\Delta H}{kT} + \left(B + \frac{C}{T}\right)\cdot S_1\right] , \tag{2.13}$$

where,

- S_1 could be any relevant stress, like voltage or current
- $\alpha, \Delta H, B$ and C determine acceleration between stress combinations
- k as with the Arrhenius model, is the Boltzmann's constant and temperature is in degrees Kelvin.

Another non-thermal stress term can be added, hence the model becomes:

$$t_f = AT^{\alpha}\exp\left[\frac{\Delta H}{kT} + \left(B + \frac{C}{T}\right)\cdot S_1 + \left(D + \frac{E}{T}\right)\cdot S_2\right] , \tag{2.14}$$

2.5.3 Bottom-up approach to System Reliability

Several simple models and methods can be used to compute system reliability, starting with failure rates for failure modes within individual system parts and component configurations. Different component configurations means different system reliability. The configuration models presented in this reference are:

- Competing Risk Model;
- Series configuration;
- Parallel configuration;
- r out of n Model;
- Complex Model.

The schematics for the Series, Parallel and Complex configurations as well as their reduction to an equivalent single component system can be seen on Appendix B.

2.5.3.1 Competing Risk Model

Assume a (replaceable) component or part has *k* different ways it can fail. These are called **failure modes** and underlying each failure mode is a **failure mechanism**.

The **Competing Risk Model** evaluates component reliability by "building up" from the reliability models for each failure mode. It is applicable when the following conditions are met:

- Each failure mechanism leading to a particular type of failure (i.e., failure mode) proceeds independently of every other one, at least until a failure occurs;
- The component fails when the **first** of all the competing failure mechanisms reaches a failure state;
- Each of the k failure modes has a known life distribution model $F_i(t)$.

If c refers to the component and i to the i-th failure mode, then then the competing risk model formulas are [3]:

$$R_c(t) = \prod_{i=1}^{k} R_i(t)$$

$$F_c(t) = 1 - \prod_{i=1}^{k} [1 - F_i(t)]$$

$$h_c(t) = \sum_{i=1}^{k} h_i(t)$$

2.5.3.2 Series Model

The Series Model is used to build up from components to sub-assemblies and systems. It only applies to non replaceable populations (or first failures of populations of systems). The assumptions and formulas for the Series Model are identical to those for the Competing Risk Model, with the k failure modes within a component replaced by the n components within a system. [3]

The following three assumptions are needed:

- Each component operates or fails independently of every other one, at least until the first component failure occurs;
- The system fails when the first component failure occurs;
- Each of the n (possibly different) components in the system has a known life distribution model Fi(t).

If the Series Model assumptions hold, then [3]:

$$R_S(t) = \prod_{i=1}^{n} R_i(t)$$

$$F_S(t) = 1 - \prod_{i-1}^{n} [1 - F_i(t)]$$

$$h_S(t) = \sum_{i=1}^{n} h_i(t)$$

with the subscript S referring to the entire system and the subscript i referring to the i-th component. Note that the above holds for any arbitrary component life distribution models, as long as "independence" and "first component failure causes the system to fail" both hold.

2.5.3.3 Parallel Model

The opposite of a series model, for which the first component failure causes the system to fail, is a parallel model for which all the components have to fail before the system fails. If there are n components, any $(n-1)$ of them may be considered redundant to the remaining one (even if the components are all different). When the system is turned on, all the components operate until they fail. The system reaches failure at the time of the last component failure.

The assumptions for a parallel model are:

- All components operate independently of one another, as far as reliability is concerned;
- The system operates as long as at least one component is still operating. System failure occurs at the time of the last component failure;
- The CDF for each component is known.

For a parallel model, the CDF $F_S(t)$ for the system is just the product of the CDFs $F_i(t)$ for the components, or [3]

$$F_S(t) = \prod_{i=1}^{n} F_i(t) \ .$$

$R_S(t)$ and $h_S(t)$ can be evaluated using basic definitions, once $F_S(t)$ is known.

2.5.3.4 R out of n model

An "r out of n" system contains both the series system model and the parallel system model as special cases. The system has n components that operate or fail independently of one another and as long as at least r of these components (any r) survive, the system survives. System failure occurs when the $(n-r+1)$-th component failure occurs.

When$r = n$, the r out of n model reduces to the series model. When r = 1, the r out of n model becomes the parallel model.

The simple case where all the components are identical is treated here.

Formulas and assumptions for the r out of n model (identical components):

- All components have the identical reliability function $R(t)$.
- All components operate independently of one another (as far as failure is concerned).
- The system can survive any $(n-r)$ of the components failing. The system fails at the instant of the $(n-r+1)$-th component failure.

System reliability is given by adding the probability of exactly r components surviving to time t to the probability of exactly (r + 1) components surviving, and so on up to the probability of all components surviving to time t. These are binomial probabilities (with $p = R(t)$), so the system reliability is given by [3]:

$$R_S(t) = \sum_{i=r}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) [R(t)]^i \, [1 - R(t)]^{n-i} \; .$$

2.5.3.5 Complex Systems

Many complex systems can be diagrammed as combinations of Series components, Parallel components, R out of N components and Standby components. By using the formulas for these models, subsystems or sections of the original system can be replaced by an "equivalent" single component with a known CDF or Reliability function. Proceeding like this, it may be possible to eventually reduce the entire system to one component with a known CDF [3].

2.5.4 Basic Models for Repairable systems

Asher and Feingold in [17] analyze Repairable Systems as stochastic point processes. A **stochastic point process** is a mathematical model for a physical phenomenon characterized by highly localized events distributed randomly in a continuum. Applied to repairable systems, the continuum is time and the highly localized events are failures which are assumed to occur at instants within the time continuum [18].

In [17] the following basic models (and variants) are discussed:

- Homogeneous Poisson Process (HPP);
- Non-Homogeneous Poisson Process (NHPP);
- Non-Homogeneous Poisson Process (NHPP);
- Branching Poisson Process (BPP);
- Renewal Process (RP);
- Superimposed Renewal Process (SRP).

For more information on the study of Repairable Systems as stochastic point processes, refer to [17], [19] and [18].

2.6 Overall Equipment Effectiveness

Overall Equipment Effectiveness (OEE) is essentially the ratio of Fully Productive Time to Planned Production Time. In practice, however, OEE is calculated as the product of its three contributing factors:

$$OEE = Availability \times Performance \times Quality \tag{2.15}$$

In the context of OEE the three factors can be defined [20] as:

Availability is the ratio between the actual run time and the scheduled run time. The scheduled run time does not included breaks, lunches and other pre-arranged time a production line or process may be down.

$$Availability = \frac{operating\ time}{scheduled\ time} \tag{2.16}$$

Performance is the ratio between the actual number of units produced and the number of unit that theoretically can be produced and is based on the standard rate. The standard rate is rate the equipment is designed for.

$$Performance = \frac{Parts\ Produced \times Ideal\ Cycle\ Time}{scheduled\ time} \tag{2.17}$$

Quality is the ratio between good units produced and the total units that were stated.

$$Quality = \frac{Units\ Produced - defective\ units}{units\ produced} \tag{2.18}$$

2.7 Manufacturon

The *XPRESS* project [21] one of the projects that forms the basis for ReBORN, introduces the concept of Manufactron. "A Manufactron is an agent based equipment which has all the knowledge to perform a certain task, and that only needs to be told what and when to do it. This knowledge based concept integrates the complete process chain: production configuration, multi-variant production line and 100% quality monitoring." [22]

2.8 Simulation

In order to tackle the challenges that come with the increasing complexity of modern manufacturing systems, such as resource and layout planning, simulation has been used successfully for decades as a tool to support decision-making in this process with clear benefits, as it is cheaper and faster to build a virtual system and experiment with different scenarios and decisions before actually implementing it [23]. Simulation models can be classified according to several independent pairs of attributes, such as:

- Stochastic or deterministic;
- Steady-state or dynamic;
- Continuous or discrete;
- Dynamic system simulation or dynamics simulation of field problems;
- Local or distributed.

For manufacturing systems simulation, one of the following three methods is used [24]:

System Dynamics (SD) simulation where the models are continuous based on differential or difference equations. The model is represented in the form of causal loops and stock and flow diagrams with positive or negative feedback relations. The mathematical equations can be exact, approximate or empirical. The drawbacks are that it uses deterministic mathematical equations while the real world does not and the structure of the system and the governing rules are constant [24].

Discrete Event Simulation (DES) is the most widely used method for manufacturing systems simulation, the systems components are modeled as objects with attributes. The states of the objects change in response to specific events that can occur at random and several events can occur at the same time. The main advantages of DES are [24]:

- the ease of use,
- the ability to include stochastic elements,
- the ability to track individual system components and get several performance measures for them.

Agent Based simulation (ABS) is a particularization of DES, where agents, see 2.1.3.1, are used instead of objects as the system components. The agents interaction with other agents is described by simple rules but resulting into complex interactions. Their behavior can change, based on their experience [24].

2.8.1 Limitation to current simulation techniques

There are two fundamental limitations to current simulation techniques, one of which being the need for multiple simulation runs, which is not an issue when the decision maker is faced with few scenarios to choose from, but can quickly escalate as the number of possible decisions increases, for example: if the system consists of 10 components and each component has only two possible alternatives, then there are $2^{10} = 1024$ alternatives to be simulated.

The other limitation is the impossibility to change the system under simulation. Although some specific variables may be altered, overall the parameters and structure under simulation are both static and rigid, much unlike real systems which are dynamic. [24]

2.8.2 Simulation Package - Flexsim

There are several simulation packages in the market, each with its own features and purposes. The simulation software to be used in this project is Flexsim. It is a powerful discrete event simulation (DES) software, that can be applied in many different fields, from manufacturing to logistics to healthcare and others. The structure uses object-oriented design and objects are part of super-classes, e.g. Fixed Resource - with objects like Processor, Sink and Source - or Task Executer - with objects like Operator or Transporter. Flexsim has a 3D environment and its very easy to create models by drag &dropping objects directly in them. Once the objects have been dropped into the model they can then be edited via dialog box. The model can also be viewed in a hierarchical tree view. It is possible for the user to create his own classes and libraries from scratch or customize the existing ones.
The access to the data tree structure, makes it possible for the experienced user to interact with the model via C++ or flexscript (Flexsim's scripting language) in run-time. Flexsim models can be saved in ".fsm" format or in a ".fsx" (a XML based format) file that can be edited directly (but not in real-time).
There is the possibility to connect Flexsim to user developed DLL's, which opens the way for external control in run-time via COM objects. Socket communication is also supported by Flexsim. Figure 2.6 shows a 3D view of a model and some of the default libraries available on the left, from where the user can drag the objects into the model.

2.9 Data Format for Information Exchange

XML is an effective method of transporting data from one location to another. Data structure is its the main advantage. Its syntax allows storing each piece of data in user defined markup tags. The markup tags are arranged in a hierarchy that represents the ordering of the data in a parent/child relationship. XML is a neutral format that simplifies data sharing, data transport, platform changes, and makes data more available [25].

Data exchange standardization is being implemented in many areas like process engineering, process control engineering, oil&gas industry, and in manufacturing automation engineering [25]. Boulonne et al. in [25] describe 3 norms: **(1)** AutomationML, **(2)** IEC/ISO 62264 and **(3)** Core Manufacturing Simulation Data. In this document special attention is given to AutomationML.

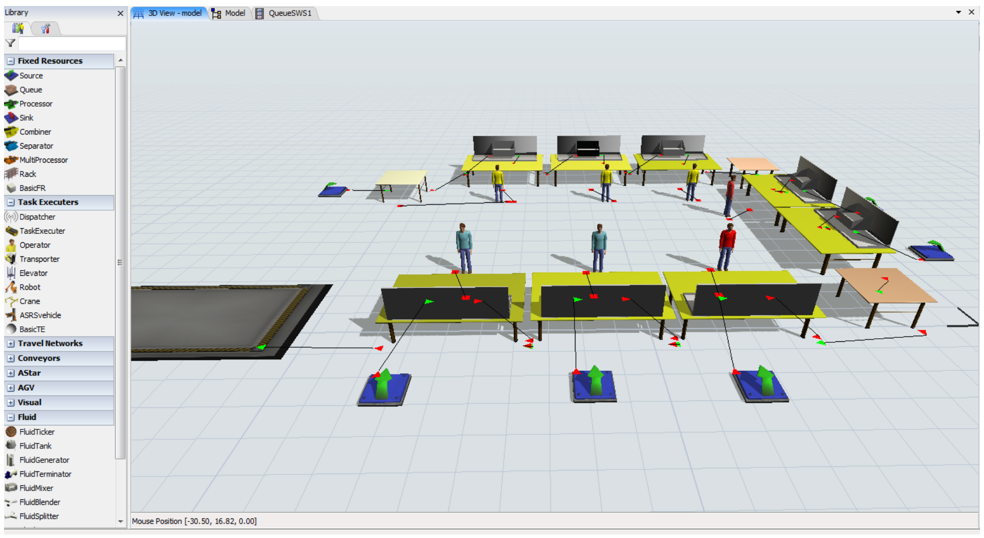


Figure 2.6: Example of a production line model in Flexsim

2.9.1 AutomationML

Automation Markup Language (AutomationML) has been developed by AutomationML e.V. (see [26]) as an open, vendor neutral, XML based neutral free data format for storing and exchanging plant engineering information. The aim is to interconnect the heterogeneous tool landscape of engineering tools in mechanical plant engineering, electrical design, HMI development, PLC, and robot control fields. AutomationML works as the glue between all the factory planning tools. It implements an object oriented paradigm; real factory components are described as data objects [25].

AutomationML follows a modular structure by integrating and enhancing/adapting different already existing XML-based data formats combined under one roof the so called top level format (see 2.7). These data formats are used on an "as-is" basis within their own specifications and are not branched for AutomationML needs. Logically AutomationML is partitioned in [27]:

- Description of the component topology and networking information including object properties expressed as a hierarchy of AutomationML objects and described by means of CAEX following IEC 62424 [28];

- Description of geometry and kinematics of the different AutomationML objects represented by means of COLLADA 1.4.1 and 1.5.0 (ISO/PAS 17506:2012) [29];

- Description of control related logic data of the different AutomationML objects represented by means of PLCopen XML 2.0 and 2.0.1 [30];

- Description of relations among AutomationML objects and references to information that is stored in documents outside of the top level format using CAEX means.

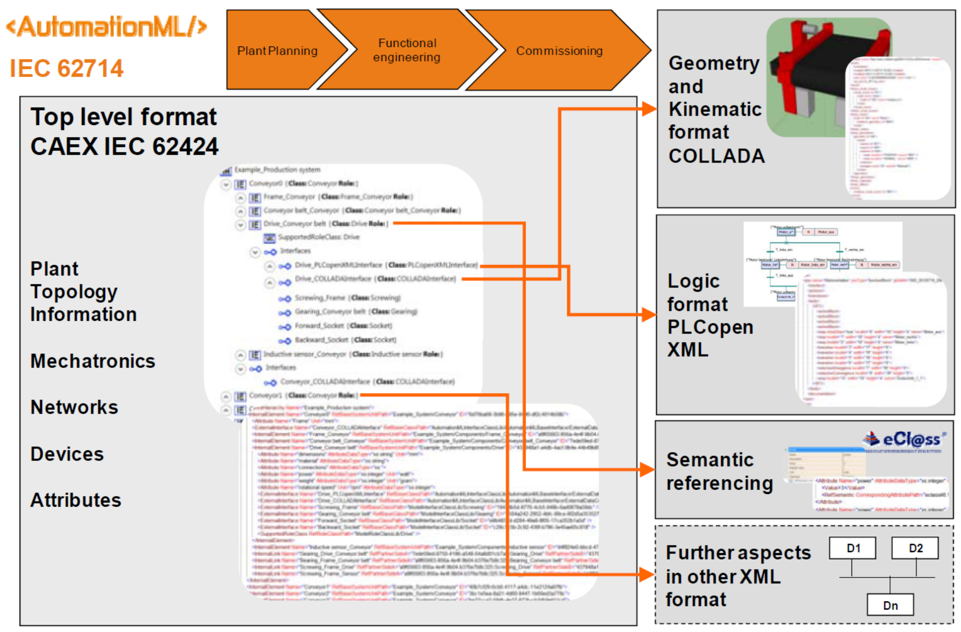


Figure 2.7: Structure of AutomationML Projects [27]

2.9.2 Core Manufacturing Simulation Data

In order to increase interoperability between manufacturing applications and simulation systems, the National Institute of Standards and Technology (NIST) is working on a standards development effort under the guidelines, policies and procedure of the Simulation Interoperability Standards Organization (SISO). The CMSD Information Model defines a data specification for efficient exchange of manufacturing data in a simulation environment [25]. CMSD aims at:

- fostering the development and use of simulations in manufacturing applications;
- facilitating data exchange between simulation model and other manufacturing software applications;
- increasing manufacturing software application interoperability.

Chapter 3

Problem Analysis

The strategies and technologies to be developed during execution of ReBORN, will address relevant issues in today's manufacturing, such as the inability to re-use production equipment after a production line is decommissioned, the lack of comprehensive design methodology for manufacturing systems or the nonexistence of appropriate methods for dynamic cost assessment for system design and system life-cycle optimization. It will use system information for real time system adaptation and will take advantage of its use in improving overall system performance throughout its life-cycle.

In this chapter the concept of VERSON is introduced, as well as the overall architecture of the system under development in the Work Package 3 of ReBORN is presented, with special focus on the subsystem being developed at FEUP.

3.1 VERSONs

The decision on the re-use of production equipment requires knowledge about the actual status of the machine and its components after the past use and the adjustment and/or prediction of the operating, service and maintenance characteristics, which have changed due to wear, exchange, update and other influences during the past operation. In addition, information might be useful in predicting possible changes of these characteristics during a foreseen operating period. The creation of this information requires an assessment of the operation so far (with the initial conditions as a base line), which is characterized by the executed processes, the deployment conditions, the supply data, sensor data, quality data and logs about failures, faults and service, maintenance and calibration measures.

ReBORN will address these needs by the introduction of **Versatile, Flexible and Lyfecycle Extended Devices (VERSON)**

VERSONs are agents which can have a physical or virtual representation of production equipment. The virtual representation is mainly used for simulation purposes. In the physical representation, VERSONs wrap existing equipment and turn it into modular, agent-based, task-driven, plug & produce devices for smart factories, which can be exchanged and adapted for both new production goals and structures. The devices are always aware of their own state of capabilities and remaining lifetime, which they offer to the production network.

The versons shall have:

- **analytical capabilities** to determine their own state, to find the best practical operational parameters;
- **intelligence** to derive a lifetime prognosis, maintenance requirements, refurbishment plans;
- state-dependant cost model estimation related to:
  - task execution
  - maintenance
- **communication capabilities** to describe and optimize themselves towards their environment by providing knowledge and models about their properties, abilities, constraints and re-use abilities (device self-description).

3.2 Overview of the High Performance Simulation Tool

In order to fulfill its vision, ReBORN will provide a high performance simulation tool, Figure 3.1, with a dynamic self-learning environment and a modular structure. Each software module represents and simulates a specific production process under real conditions through the connection to a knowledge database, including all the methods for the performance of this specific process.

The *ReBORN method* can be divided in two processes. The first is triggered when the user defines a set of requirements and constraints for a production process. Those definitions are captured and the formalized job description is sent to the Layout Configurator, which will match the requirements with the capabilities provided by VERSONs and other equipment to identify potential candidates to integrate the layout and will eliminate those that do not fit the constraints, like size or geometry.

Once all the possible layouts have been configured, they are sent in AutomationML format, see 2.9.1, to the Layout Simulator which will combine the results of three different assessments and present the result back to the user, providing him with information for an educated decision in designing the production line.

The knowledge from operational VERSONs and other equipment deployed in production environments is captured and used to update the VERSONs and the network so the models are enhanced.

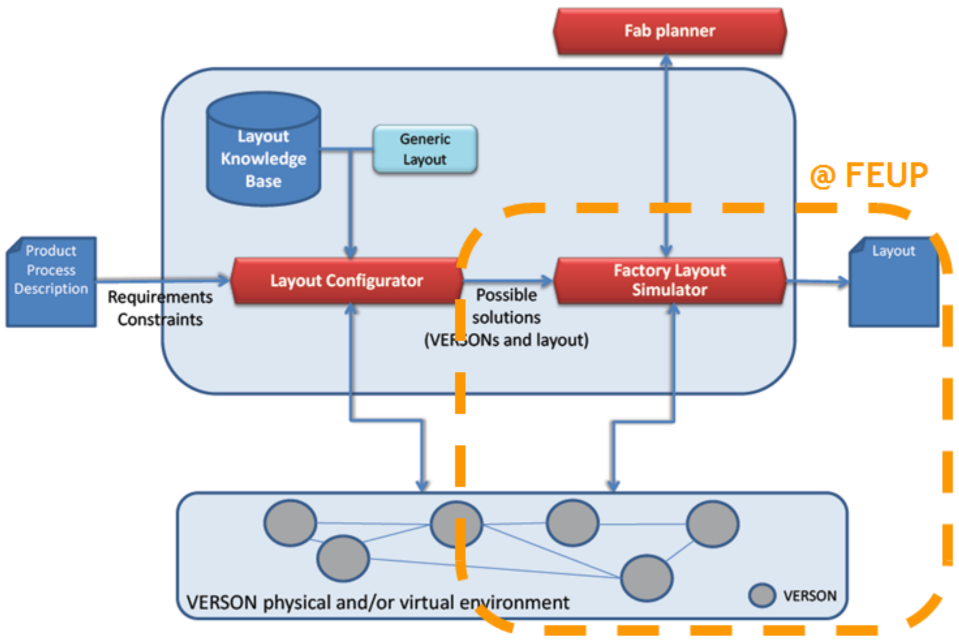


Figure 3.1: ReBORN high performance simulation tool [4]

3.3 Problem specification

The part of ReBORN being developed at FEUP is framed in the subsystem outlined by the dashed orange line in Figure 3.1 and it aims at enabling the assessment of various potential change, upgrade, reuse, dismantle or disposal scenarios and the corresponding reconfiguration effect on the overall system cost, performance and status throughout their life-cycle.

These scenarios constitute different layouts, sent by the Configurator and described in an AutomationML (see Section 2.9.1) format, which are then simulated and the results used in the assessment of their life-cycle impact.

The detail and amount of information available about the possible layouts and the VERSONs that compose them will influence the the output, which is the result of three assessments:

- LCC (section 2.2);
- LCA (section 2.3);
- Reliability Assessment (section 2.4).

Here lies a **key innovation** in the design of manufacturing systems: the integration of reliability and life-cycle status information in its conception.

The process is depicted in figure 3.2.

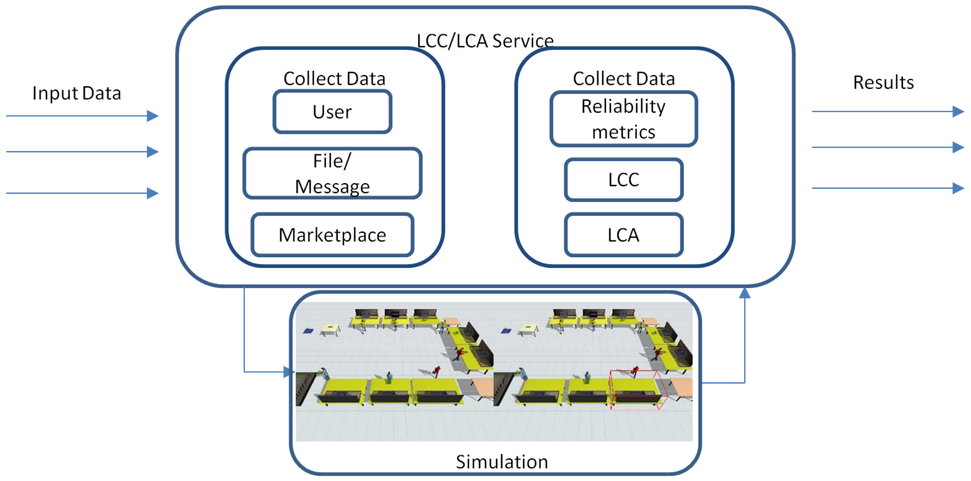


Figure 3.2: Process of the subsystem being implemented at FEUP

The results of the Reliability Assessment are a set of metrics, previously described in Section 2.4:

- Failure rate;
- MTBF;
- MTTR;

- Reliability;
- Availability;
- Performance;
- Quality;
- OEE.

The result of the LCC (concerning each machine) is:

- Future Value: $FV = OriginalInvestment \times (1 + (InterestRate \times ReviewPeriod))$
- Present Value: $PV = FV/(1 + DiscountRate)^{ReviewPeriod}$
- Net Present Cost: $NPC = PV \times DiscountRate \times \sum Cashflow$
- Net Present Value with initial costs: $NPV = InitialCosts + (\sum cashflow/(1 + DiscountRate)^{ReviewPeriod-1})$

Note: the development of the LCC model, while out of the scope of this dissertation, was developed at FEUP nonetheless.

The results obtained will be displayed in graphs by a web application, comparing the possible layouts at machine level for each metric for a stronger visual impact on the user. Figure 3.3 shows a comparison of the results of Reliability Assessments between two machines displayed in the aforementioned tool. Figure 3.4 shows a comparison of LCC assessments in two machines. On the upper frame two machines are compared in terms of the NPV for a period of two years and five years time. On the lower frame the same machines are compared for the same period of time in terms of their FV, PV and NPC.

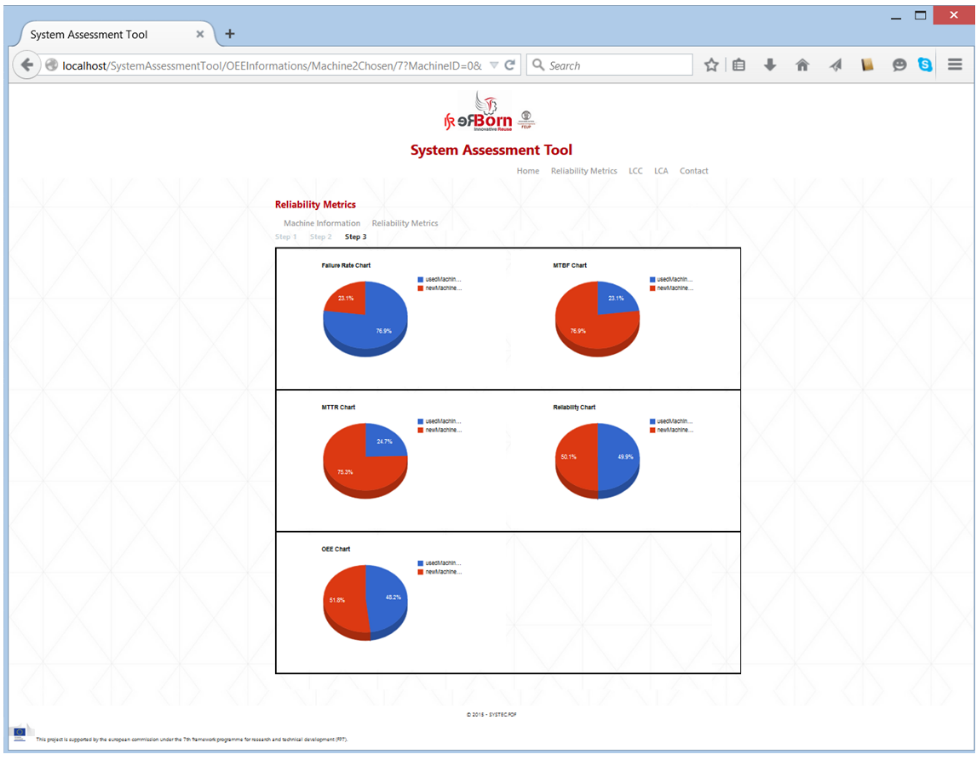


Figure 3.3: Comparison of 2 Reliability Assessments in the web application developed at FEUP

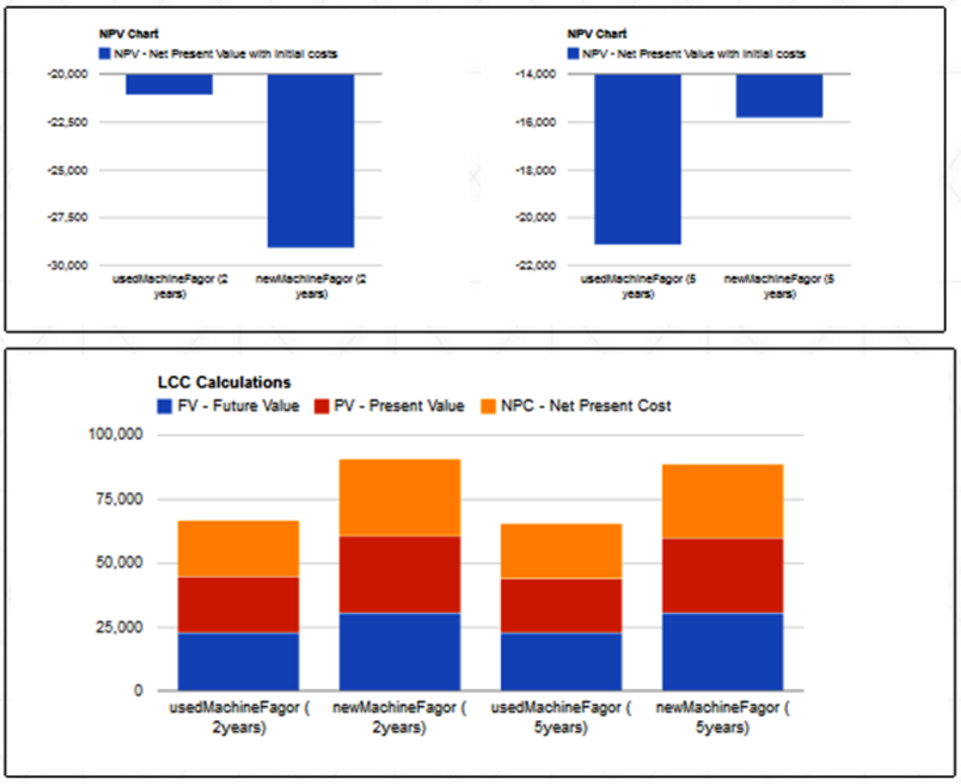


Figure 3.4: Example of a comparison of an LCC assessment in two machines

3.3.1 Parameters for the metrics

In order to achieve the results mentioned above, an extensive list of parameters was identified, not only for the Reliability Assessment and LCC but also for LCA. Some of these parameters are taken from the **marketplace**, like:

- Cashflows;
- % of recyclable material;
- Emission flow type;
- Emission flow value;
- Input flow type;
- Input flow value;

or even costs for:

- Hardware Acquisition;
- Software Acquisition;
- Service Contracts;
- Administrative;
- Set Up & Installation;
- Disposition Cost;
- Equipment Upgrade;
- Default Annual Energy Consumption;

Other parameters on the other hand, are more reliable when derived from **simulation**, like:

- Operating Hours per Year;
- Unscheduled Downtime;
- cycle time;
- Estimated Downtime in a Year;
- Expected remaining Life ;
- Number of machine Failures per Year;

- Power Consumption per unit in "Active" mode;
- Power Consumption in "Sleep/Stand By" Mode;
- Power Consumption in "Off" Mode;
- % of time turned-off during workday;
- % of time in stand-by during workday.

3.3.2 Conclusion

The work done for this thesis focuses on providing a tool that facilitates the creation of simulation models of the layouts - described in an AutomationML format - sent by the Layout Configurator, with the purpose of calculating metrics to be used in their LCC and Reliability Assessments. The simulation models should at least produce the parameters stated above, but are not restricted to them, depending on the detail of the Inputs. These models may also produce useful metrics for an LCA, if the inputs are detailed enough.

Chapter 4

Machine Conceptual Models

Given that the purpose of this work is to provide a tool that facilitates the creation of simulation models of the production lines sent by the Layout Configurator, with the main goal of assessing the Reliability of these layouts, VERSONs (see 3.1) will be perceived as machines in a production line and therefore only some of the VERSONs' features are contemplated. Furthermore, at this level of abstraction the actual jobs performed by the machines at tool level, like cutting or welding are not considered. The physical/geometrical representation of the model is not relevant for the assessment either, so physical dimensions of the machines, their coordinates on the shopfloor, the distances between workstations and other objects, the distances traveled by the product being assembled or produced, as well as the corresponding travel times are not considered in the simulations.

It is also important to note that at the time of the realization of this work there was yet limited information about the AutomationML structure adopted by the other partners in ReBORN and the data contained in it, so assumptions were made as to what the inputs of the simulation were.

Over the next chapter, the models and concepts behind the individual machines /VERSONs that make the production lines to be simulated, will be presented.

4.1 Machine as a system

Following the concepts introduced in 2.5.3, the machines will be modeled as systems composed of any given number of parts and/or components, which are placed individually in sockets (see 2.5). At the highest level, the **components** are arranged in a **Series configuration** 2.5.3.2, which means a machine fails when any component that level fails. Figure 4.1 illustrates this architecture.

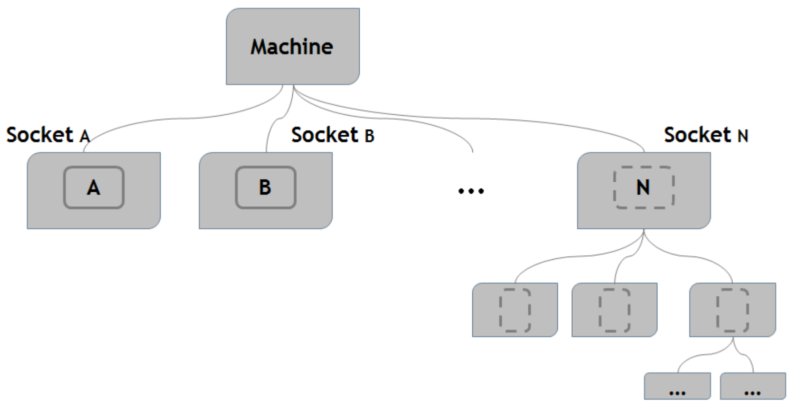


Figure 4.1: Machine Conceptual Model

4.1.1 Socket Level

When a component fails, it is replaced immediately with a new one of the same kind. In other words, after each replacement the socket is put back into an "as good as new" condition. Each component has a time-to-failure that is determined by the underlying distribution. Once again, a distribution relates to a single failure (see 2.5.1) . The sequence of failures for the socket constitutes a random process called a **Renewal process** (see 2.5.4). A renewal process is defined as a nonterminating sequence of independent, identically distributed (IID) non-negative random variables, $X_1, X_2, ...$, which with probability 1 are not all zero [17].

In figure 4.2, the component life is X_j, and t_j is the system time to the j^{th} failure. Each component life X_j in the socket is governed by the same distribution $F(x)$.

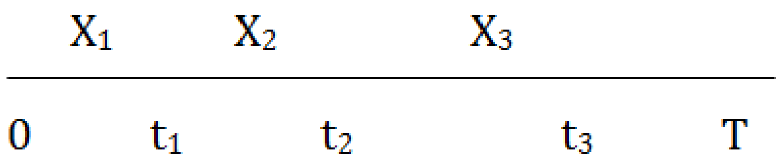


Figure 4.2: Illustration of Renewal Process [5]

4.2 Lifetime modeling

As already stated in previous sections, lifetime models apply only to the parts in sockets. The following assumptions were made in regards to the models to be implemented in this solution:

- Lifetime will be accelerated or decelerated by load conditions referred to normal load conditions. For the basic principles see 2.5.2;
- Physics of failure mechanism does not change under changed load conditions;

- The system under consideration exhibits cumulative damage, i.e. the remaining life depends on future load and the future distribution function, starting at the previously accumulated fraction of failure probability.

4.2.1 Lifetime distributions

Lifetime of parts can be modeled by any PDF, as explained in section 2.5.1. The most common pdf used are the Weibull distribution and the Exponential distribution. Their formulas and plots are presented in Appendix A.

4.2.2 Accelerated Lifetime models

The concept of "physical acceleration" explained in 2.5.2 is used in these models, so that higher loads imply faster equivalent time evolution (refered to normal load). This is explained with the following example.

In figure 4.3 an example an evolution of step-load levels l_i over operating time is presented. The normal load condition (reference) is l_0. Acceleration factor is then:

$$AF_i = \frac{life(l_o)}{life(l_i)}$$

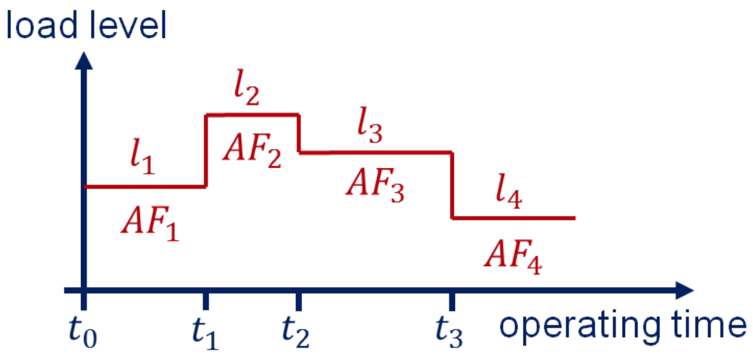


Figure 4.3: Example of evolution of step-load levels over operating time

$t_{eq}(t)$ is the equivalent lifetime, i.e. accumulated operating time weighted by acceleration factors. It is given by:

$$t_{eq}(t) = \begin{cases} AF_1(t-t_0) & \text{for } t_0 < t \leq t_1 \\ AF_1(t_1-t_0)+AF_2(t-t_1) & \text{for } t_1 < t \leq t_2 \\ \vdots & \\ AF_1\Delta t_1 + \cdots + AF_n\Delta t_n + AF_{n+1}(t-t_n) & \text{for } t_n < t \end{cases}$$

When load conditions are arbitrary, $\Delta t_i = \Delta t$, then $t_{eq}(t)$ is

$$t_{eq} = \int_{t_0}^{t} AF(t')dt'$$

$t_{eq}(t)$ contains the whole impact of historic load conditions.

Consider $f(t)$ is a Weibull PDF (see A.2), with characteristic lifetime under normal load conditions η_0. The characteristic lifetime for arbitrary load conditions is given by:

$$\eta(t, L(t)) = \frac{\eta_0}{AF(t, L(t))}$$

Figure 4.4 depicts the equivalent failure probability, when 3 different load conditions are experienced over the operating time.

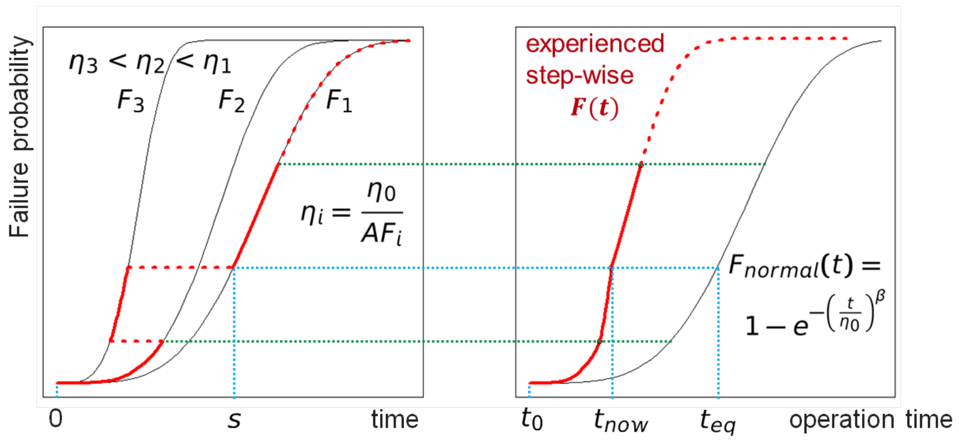


Figure 4.4: Equivalent failure probability with 3 different load conditions

From figure 4.4, it is possible to derive

$$F_1(s) = F(t_{now}) = F_{normal}(t_{eq})$$

4.2.3 Lifetime information

For each failure mode, the following information should be provided:

- t_0 - operating time at last repair, maintenance, replacement or birth;
- $t_{eq,0}$ - describes pre-damage at t0;
- t_{eq} - describes accumulated history;
- $AF_m(t, L_m)_{m=1}^{M}$ - functional or numerical representation of acceleration factors for every load variable L_m; includes normal load conditons;
- PDF and parameters under normal load conditions for each component.

Chapter 5

Implementation

One of the obstacles in the usage of simulation models as a "decision support system" when planning the layout of a new production line, is that the creation of these models is time consuming and prone to errors. The work developed under this thesis (see 3.3) tackles this issue.

This Chapter describes the implementation of the tool that facilitates and automatizes the process of creating simulation models (conceptualized in Chapter 4) of the production lines in the context of ReBORN (see 3) and the integration of Flexsim (see 2.8.2) with the layout description in XML.

5.1 Overview of Process

The models are built semi-automatically via an interface developed for this purpose that controls Flexsim (see 2.8.2) remotely by using DLL's. It parses the layout specified in a xml file stored in a server and automatically builds the models. As the node tree is traversed in the XML document, the values of each element are copied to variables used in predefined Flexscript (Flexsim's scripting language) scripts which are "injected" into Flexsim, instanciating objects like machines, sources, sinks, etc., setting their variables and connecting them, thus creating the models automatically. At this time some manual adjustments may have to be made, namely defining the time table (scheduled working hours) of the production line. The process is illustrated in figure 5.1.

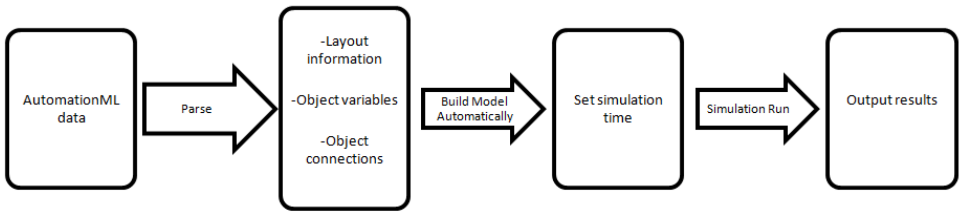
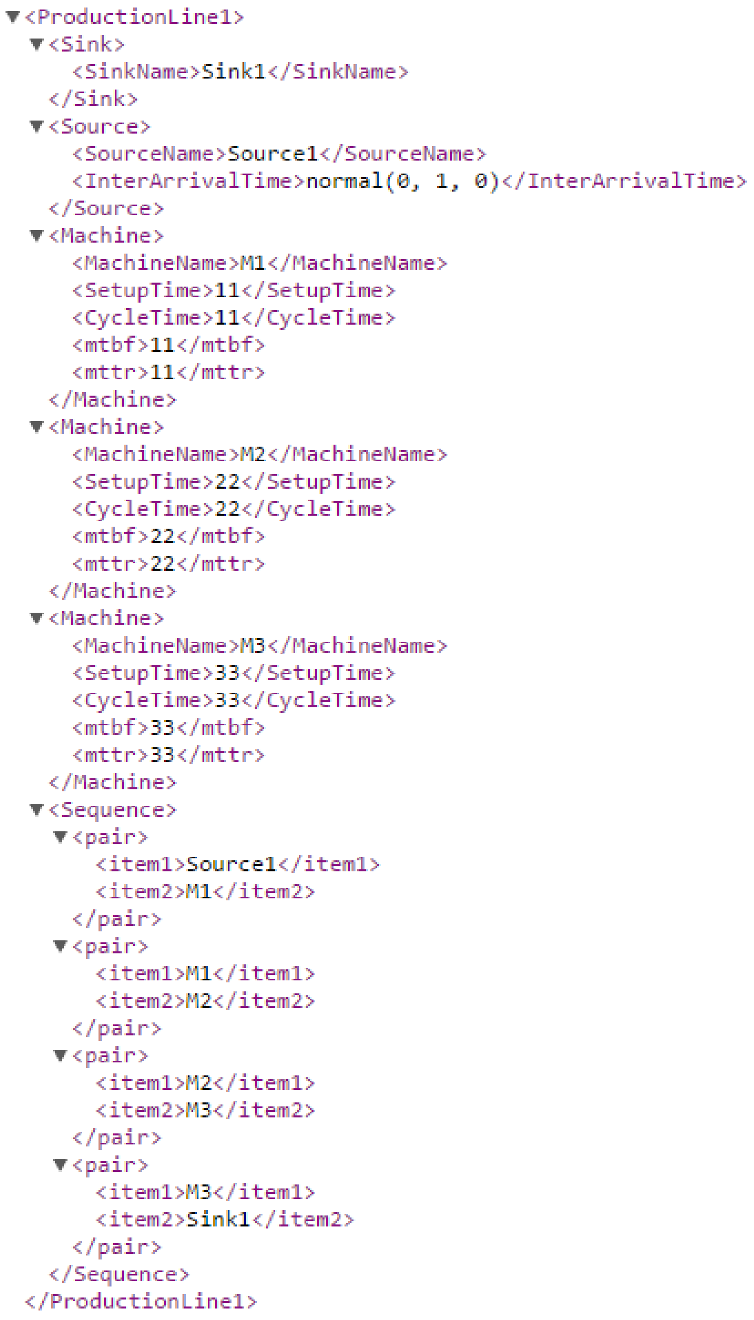


Figure 5.1: Overview of the solution's methodology

5.2 Data Structure

AutomationML was the data format adopted to be used across all levels of hierarchy of the project's solution. Because it is a standardized data format and the structure to be used was yet to be defined by the other partners upstream in ReBORN at the time of the writing of this dissertation, another XML format was used for the development and experiments. An example is shown in Figure 5.2.

The structure is kept simple for development and experiments and only the necessary information for the layout is represented since additional tags with other information can be ignored. In Figure 5.2, the layout described consists of 3 machines: M1, M2 and M3 a source (Source1) from where parts or products enter the model and a sink (Sink1) where the final product exits the model. The inputs of the machines are: Name, Setup Time, Cycle-time, MTBF and MTTR. The child element of <Sequence> - <pair> - describes the connections between objects in the model.



```
▼<ProductionLine1>
  ▼<Sink>
     <SinkName>Sink1</SinkName>
   </Sink>
  ▼<Source>
     <SourceName>Source1</SourceName>
     <InterArrivalTime>normal(0, 1, 0)</InterArrivalTime>
   </Source>
  ▼<Machine>
     <MachineName>M1</MachineName>
     <SetupTime>11</SetupTime>
     <CycleTime>11</CycleTime>
     <mtbf>11</mtbf>
     <mttr>11</mttr>
   </Machine>
  ▼<Machine>
     <MachineName>M2</MachineName>
     <SetupTime>22</SetupTime>
     <CycleTime>22</CycleTime>
     <mtbf>22</mtbf>
     <mttr>22</mttr>
   </Machine>
  ▼<Machine>
     <MachineName>M3</MachineName>
     <SetupTime>33</SetupTime>
     <CycleTime>33</CycleTime>
     <mtbf>33</mtbf>
     <mttr>33</mttr>
   </Machine>
  ▼<Sequence>
    ▼<pair>
       <item1>Source1</item1>
       <item2>M1</item2>
     </pair>
    ▼<pair>
       <item1>M1</item1>
       <item2>M2</item2>
     </pair>
    ▼<pair>
       <item1>M2</item1>
       <item2>M3</item2>
     </pair>
    ▼<pair>
       <item1>M3</item1>
       <item2>Sink1</item2>
     </pair>
   </Sequence>
 </ProductionLine1>
```

Figure 5.2: Example of a layout defined in XML

5.3 Flexsim Remote Control

The process of creating simulation models is time consuming and error-prone. Therefore, finding a way to **(1)** automate this process and **(2)** make the simulation software "invisible" to the user was essential. In order to achieve these two goals it was necessary to control the simulation software and the models externally. This was the most difficult part of the work and where most of the time was spent.

5.3.1 Explored alternatives

While a lot of capability exists in Flexsim, knowledge is still limited in various areas as to controlling it remotely. Several ways were explored to meet the objectives:

- Sockets;
- CMSD; see sec. 2.9.2
- using *.fsx*;
- Microsoft Excel;
- use of DLL's;
- Batch Files and VBS/VBA command-line execution;
- Component Object Model.

As stated in section 2.8.2 Flexsim supports socket communication; however, documentation is scarce and no examples were found to use as a starting point. Another method tried was through the execution of shell commands to open a Flexsim model and passing command-line parameters through the shell that the Flexsim model could interpret and make decisions from, using .bat files, VBScript or VBA, but these proved to be ineffective. Flexsim allows the connection to user made DLL's and this was tried as a method to remotely pass parameters into a model but was inefficient. Another failed approach was through Excel tables that the Flexsim model would extract values from, but was inefficient and the amount of variables and their complexity would rapidly escalate, making it impossible to use. Several mentions to the use of CMSD 2.9.2 in automatically generating the Flexsim model [31] [32] [33] were found, but these would still require a translator for the corresponding XML format which was not available.

The use of Flexsim's XML based format (.fsx) was examined also. It proved easy to insert objects like machines or conveyors in the model, but the main problem was in following the logic behind the notation that set the connections between object models, making it impossible to connect all the objects correctly. It also did not allow for run-time model editing, since any modifications to the .fsx file would implicate the closing of the model and its re-opening. Another disadvantage of this method is that sometimes its necessary to edit multiple nodes when a simple command line in Flexscript or C++ would do it.

5.3.2 Solution adopted

The solution adopted was the use of a Component Object Model (COM). A beta release of a COM object for Flexsim 4 was made available by Flexsim developers. The initial project was created with C++ in Visual Studio 2008 for 32bit platforms. The project had numerous bugs that were preventing it from compiling successfully. Once the source code was debugged and other changes were made to adapt it to newer versions of Flexsim, namely the Registry Key for Flemxsim 4 to the version in use, the object worked as expected. Initally, Flexsim 5 had to be used instead of the latest release, Flexsim 7.5, due to abnormal behavior observed on the Flexsim model when testing the COM object with Flexsim 7.5. As development continued and a more basic use of the COM was adopted, simply using it to "inject" scripts instead of directly changing the nodes in Flexsim's model data tree, the use of Flexsim 7.5 was successful.

From the Client side, some of methods can be called that make it possible to access and edit the model data tree in run-time. Most of these methods mimic Flexscript (Flexsim's scripting language) commands. Given that the COM is only a beta version, the number of methods that can access the model data tree is very limited in comparison with all the Flexscript commands. This limitation is overcome by using a particularly useful method *FlexsimAXApp::ExecuteString(BSTR execString, double* retval)* which executes the string passed as parameter on the Flexsim model as Flexscript, basically allowing for any Flexscript command to be called remotely.

Apart from the methods that mimic Flexscript, other methods that are called from the client side worth of notice:

- FlexsimAXApp::StartApplication(BOOL) - opens a new Flexsim thread;
- FlexsimAXApp::ExitApplication(void) - kills the FLexsim thread;
- FlexsimAXApp::OpenLibrary(BSTR path) - opens a Flexsim library specified in the path;
- FlexsimAXApp::OpenModel(BSTR path) - opens the Flexsim model specified in the path.

Before using the COM, the Client DLL has to be registered through the command line.

5.4 Interface

An HTML page was created as the interface to link all the components of the solution together. This web page will, from now on, be called Interface. It uses the ActiveXObject in Javascript to interact with the COM object and call the methods on the client side. Due to the use of ActiveX, a Microsoft's proprietary software framework, the web page only works with Microsoft's Internet Explorer.

Once the Interface is open, the user can start a new instance of Flexsim, which will open a Flexsim user library as well, that has some components of the COM server installed, as well as other elements used for the simulation models described in the next section. This instance serves as a foundation on which the user will work, either by opening other saved models, using FlexsimAXApp::OpenModel(BSTR path), inserting objects in the model individually and connecting them or by building the model from a description in a XML file.

5.4.1 Overview of the Interface

Interaction with the Interface is kept simple, buttons and html <textarea> for passing parameters are the only forms of user interaction with it.

All the functions and logic of the Interface were written in Javascript. The methods from the COM object that control Flexsim remotely are instantiated inside Javascript functions that are called by the user when certain buttons are pressed. Early in the development of the Interface, instantiation of Flexsim objects in the model was done "manually", where the user would submit a form with the input data in the <textarea>. Several functions were built to implement the different objects. They were later used to instantiate objects taking node values of a XML file as parameters.

Some functions combine the instantiation of multiple objects. For instance, when inserted individually, objects of the class FixedResource (see 2.8.2) (e.g. Processors, i.e. machines) cannot simulate breakdown events. Another object - MTBF/MTTR - has to be instanciated, where the MTBF and MTTR are defined. These MTBF/MTTR objects can have several members (other objects like Processors) and each member may belong to other MTBF/MTTR objects. In this solution, when each machine is added to the model, a MTBF/MTTR object, of which the machine is a member, is added automatically as well.

Due to the limited number of methods implemented on the COM Client and the difficulty of implementing new ones, the method FlexsimAXApp::ExecuteString(BSTR execString, double* retval) was used to overcome this since it executes the string it receives as a parameter as Flexscript. In other words, it allows the "injection" of Flexscript into the model, just like running a script with Flexsim's script console. This function was used in fact to implement a Flexscript console in the Interface, particularly useful during development as the console in the software is only available when the licenses are activated.

The interface should be updated regularly, in order to stay aligned the Flexsim models.

5.4.2 Automatic model generation from XML

This is one of the most important features of the Interface, as it solves the problem of building the model manually which is, as stated in sec. 5.3, a time consuming and error-prone process. For security reasons client-side Javascript cannot access the localhost, so the XML files were stored in FEUP's SAMBA server during development.

All modern browsers have a built-in XMLHttpRequest object to request data from a server. One of the characteristics of this object is that it enables the possibility to request and receive data from a server without reloading the page. One of the properties of XMLHttpRequest is the responseXML. This property returns the response as a XML DOM object. A XML DOM (Document Model Object) is a standard programming interface for XML and it defines a standard way for accessing and manipulating XML documents.

In the Interface, xmlhttp is an instance of XMLHttpRequest. When xmlhttp.responseXML is called, the result is a XML DOM named xmlDoc, which is traversed and the node values are passed as parameters in the same functions created in early stages of the development of the Interface, that allow the instantiation of several types of objects in the model and their connection. Figure 5.3 shows the model created by the Interface with the layout defined in the XML of Figure 5.2.

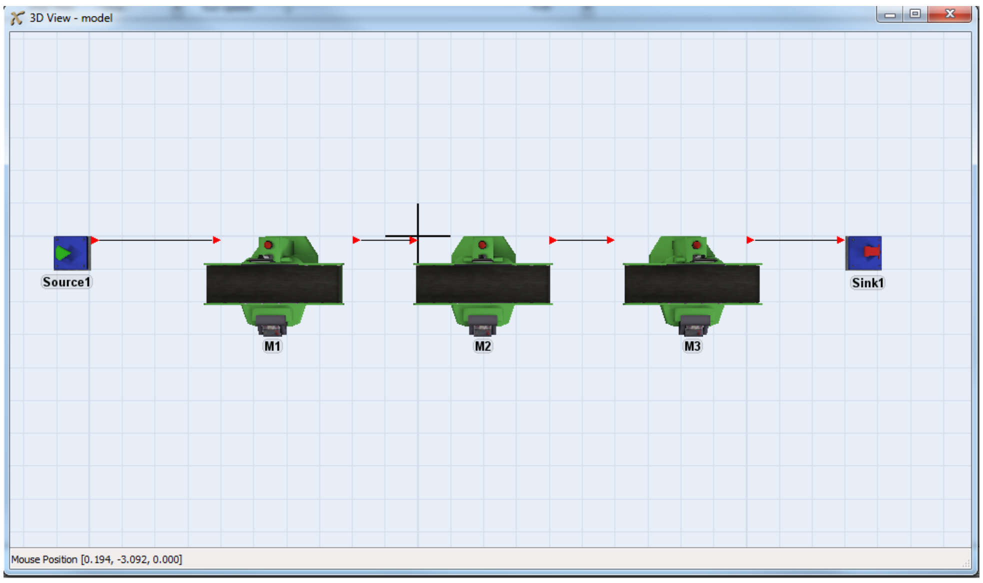


Figure 5.3: Example of a layout defined in XML

It should be noted that the display of the objects in the 3D view of Figure 5.3 was altered manually for illustration purposes. When objects are instantiated they are all in the location (0,0,0). As stated in Chapter 4 the physical location of the machines is not relevant in this context; however, placing the object in the right coordinates if they are passed as input, could be easily done with a simple script.

5.5 Flexsim Models

In order to implement the simulation models that realize the concept in Section 4 for the machines, the use of several Flexsim objects has to be combined. Also, given that the concept is the same for all the machines, the standard machines in Flexsim (the Processors) were edited and saved in a user library so they can be re-utilized.

Standard objects in Flexsim come with very little features. The basic Processor allows the user to define a setup time and a cycle time. All the other features have to be added to the objects, like labels to store values, what actions to execute on each trigger etc.

5.5.1 Basic Flexsim concepts

5.5.1.1 Labels

Any number of labels can be defined for any object. Labels can of type *number* or *string*. Labels can be set at any time. In this case for each machine a number of labels where added, some of them are used to increment the value of a metric, like the *number of breakdowns*, or *number of good* or *bad pieces* produced in each machine. Other labels are useful when defining the logic of the events or actions to perform at any given time.

The basic labels to store values of metrics in a machine are:

- good pieces produced;
- bad pieces produced;
- number of breakdowns;

If more than one type of pieces or products are produced, then each product or piece (*Flowitem*) has its own type number, stored in a label as well. This allows, for example, the machines to keep track of how many pieces of each type were produced and define different cycletimes for each type of Flowitem.

5.5.1.2 Triggers

Each object has a different set of predefined triggers. Triggers allow an increase in complexity in the models. When a certain event happens (triggered), the functions associated with that event are called. Triggers for the machines are:

- On Reset;
- On Message;
- On Entry;
- On Exit;

- On Setup Finished;
- On Process Finish.

5.5.1.3 Breakdowns

Breakdowns in machines are modeled with a special kind of object called "MTBF/MTTR" objects. Each "MTBF/MTTR" can have many different machines as its members, and each machine can have several "MTBF/MTTR" associated with it. The inputs of these objects are :

- First Failure Time;
- MTBF;
- MTTR;

Note: The notation MTBF and MTTR used in these objects is misleading, as it defines the time to the **next failure** and time to the **next repair** and not Mean Time Between Failures and Mean Time to Repair.

The triggers associated with these objects are:

- On BreakDown;
- On Repair.

5.5.2 Implemented Models

When the model is created and objects instantiated the values for **cycletime**, **setup time**, **quality factor** for each machine are defined as well. For each *itemtype* (given by the *type number* label of a *Flowitem*) different cycle times and quality factors may be defined.

When the model is running and an item enters a machine, the sequence of steps are as follows:

- OnEntry trigger the machine reads the *Flowitem's type number* label;
- It then processes the item for the duration of the cycletime as defined;
- OnProcessFinish, it performs a Bernoulli test, where, with a certain probability (given by the **quality factor**) it may change the *item type number*, meaning the process produced a defect and the *Flowitem* will be rejected;
- OnExit, if $itemtype == 2$, meaning the item is *not good*, the itemtype exits the system through a special Sink. If $itemtype! = 2$, then it proceeds to the next machine.

OnExit, labels that keep record of produced pieces (*good or bad*) are incremented. OnReset *Good pieces* label, *Bad pieces* label and *Number of breakdowns* labels are set to 0.

On the model data tree, a record is kept of the time spent by the machine in each state. There are 50 different states that is kept a record e.g. Processing Time, Scheduled downtime, Breakdown time or Setup time.

5.5.2.1 Breakdowns in the implemented models

As explained in Chapter 4 and Figure 4.1, each machine is composed of several sockets, each socket has a part or component and they are arranged in a series configuration.

This can be modeled with the "MTBF/MTTR" objects presented in 5.5.1.3, where each socket is represented by one "MTBF/MTTR" object. The lifetime distribution of its parts is directly defined in the input *MTBF* of the object. The value in this input is given by a function that returns a random number with the PDF and the parameters defined. In each new cycle (meaning a part has died and was replaced) a new value for the next time-to-failure is generated, just like in Chapter 4.

OnBreakdown, the *Number of Breakdowns* label for that machine is incremented.

5.5.3 Shortcomings in the implementation

The solution implemented to simulate the behaviour of the sockets, does not include the accelerated lifetime, as described in section 4.2.2.

The reason is that the input *MTBF* in the "MTBF/MTTR" object is read at every OnRepair trigger and although the value can be altered in the input, it will not be read until the next OnRepair trigger. Several ways to force the OnRepair trigger were attempted, but were not successful.

The only viable alternative explored would be to change the models entirely, instead of "MTBF/MTTR" objects create a *User event* for each socket that:

- periodically performs a Bernoulli test ($B(F,1,0)$) which has probability F of retuning 1;
- if $B(F,1,0) == 1$ then the event sends a message (sendmessage()) and a delayed message (senddelayedmessage()) to the machine;
- OnMessage (the machine receives the first message) execute stopobject();
- OnMessage (the machine receives the delayed message) execute resumeobject().

Probability F is the CDF ($F(t)$) of the lifetime distribution for that socket, with $t = t_{eq}$ and the parameters of the PDF are updated to the corresponding load level 4.2.2.

Moreover, this solution implies a change in the way labels of the machine are incremented. Instead of using OnRepair and OnBreakdown (current solution) labels would have to be updated OnMessage.

Due to lack of time these changes were only studied, but not implemented.

5.5.4 Outcome of the Simulations

From the values stored in the labels, and the internal variables of the machines that record the time spent in each state (accessible in the model data tree), the following results can de computed for each machine:

- Planned Production Time (hours/year);
- Current Planned Production Time (hours);
- Current Operating Time (h);
- unscheduled Down Time year;
- Cycletime (per *itemtype*) (average);
- Combined Cycletime (average);
- No. Good Pieces (per *itemtype*) ;
- No. Good Pieces total;
- No. Bad Pieces (per *itemtype*) ;
- No. Bad Pieces total;
- Quality factor (per *itemtype*) ;
- Quality factor (total);
- No. machine failures;
- No. repairs (per socket);
- Total Power Consumed (MW);
- Availability;
- Performance;
- Quality;
- OEE.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis. A brief review on the developed work and the results it achieved is contained in it. At the end of this chapter some suggestions for further research are exposed.

6.1 Conclusions

This dissertation was framed in the context of ReBORN project, which envisions the demonstration of strategies and technologies that support the a new paradigm fo re-use of production equipment in old, renewed and new factories.

A simulation tool is being developed by several partners in the project which will allow for the integration of reliability and life-cycle status information early in the conception of manufacturing systems.

The purpose of the work developed for the thesis, was to provide a tool to facilitate the creation of simulation models of production line layouts, with the purpose of assessing their Reliability and Life Cycle Cost.

The first step into the development of this tool was to find a way to control Flexsim - the simulation software- remotely. This is done via a COM object. The Interface was developed in HTML/Javascript since they support COM objects.

The next step was to automate the process of building simulation models. Given that the information about the production layouts comes in a XML document, the Interface makes a xml-HttpRequest to the server where the file is stored. The resulting XML DOM is traversed and the values in the nodes are parameters of functions developed in the previous step, that "inject" scripts into Flexsim, enabling the instantiation of objects in the simulation model.

The third step was perfecting the models on Flexsim. The standard Flexsim Processor was edited, upgraded and added to a user library to be reused. It now keeps record of the number of Good and Bad parts produced, as well as the number of breakdowns it experienced. It also has some built-in logic to handle the effects of defects or losses upstream in the production line.

The biggest difficulty encountered during the development, and where most of the time was spent, was on the first step - external control of Flexsim. Several technological options were tested but only the adopted one worked.

The choice for COM, HTML and Javascript impose some limitations the Interface, as it only works with Internet Explorer, because of ActiveX. Also, the COM object did not work on 64-bit Operating Systems.

The shortcomings of the Flexsim models do not render the whole methodology implemented invalid, since improvements on Flexsim models do not change the structure of the solution.

The solution is valid as a proof of concept.

6.2 Future Work

Further developments can be de done to this work. For one hand, the continuation of this solution starting by validating the proposed models for Flexsim. The Interface can have installed the feature of generating a XML document to export with the results of the Reliability Assessment and simulation.

Regarding the shortcomings exposed in 5.5.3, the alternative method should be implemented, so that the simulation models are accurate and coherent with their concept.

The models proposed will have to be reviewed to adapt to the inputs when the AutomationML structure is defined by the other partners.

Degradation of a machine does not only affect its Reliability, but also its capabilities. Models for degradation of machines capabilities should be considered

A different approach to the solution could be taken however, by translating the AutomationML files to .fsx (Flexsim XML) and creating the simulation models directly. This solution could increase the variety of the models; on the other hand, the complexity of changing all the nodes required in the Flexsim XML tree may render it impractical, as sometimes changing an object alters the structure of other objets in the tree.

References

[1] G. Rebitzer, T. Ekvall, R. Frischknecht, D. Hunkeler, G. Norris, T. Rydberg, W.-P. Schmidt, S. Suh, B. P. Weidema, and D. W. Pennington, "Life cycle assessment: Part 1: Framework, goal and scope definition, inventory analysis, and applications," *Environment International*, vol. 30, no. 5, pp. 701–720, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0160412003002459

[2] Reliasoft Corporation, *System Reliability {&} Maintainability Reference*. Tucson, Arizona: ReliaSoft Publishing, 2015.

[3] M. Natrella, *NIST/SEMATECH e-Handbook of Statistical Methods*, C. Croarkin, P. Tobias, J. J. Filliben, B. Hembree, W. Guthrie, L. Trutna, and J. Prins, Eds. NIST/SEMATECH, 2010. [Online]. Available: http://www.itl.nist.gov/div898/handbook/

[4] "Annex I – "Description of Work". Project name: Innovative Reuse of modular knowledge Based devices and technologies for Old, Renewed and New factories," 2013.

[5] Reliasoft Corporation, *Reliability Growth and Repairable System Analysis Reference*. Tucson, Arizona: ReliaSoft Publishing, 2015. [Online]. Available: https://books.google.pt/books?id=iy9bQwAACAAJ

[6] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," *International Journal of Production Research*, vol. 46, no. 4, pp. 967–992, 2008.

[7] M. G. Mehrabi, a. G. Ulsoy, and Y. Koren, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 13, no. 2, pp. 135–146, 2002.

[8] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130–141, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jmsy.2011.01.001

[9] C. Christo and C. Cardeira, "Trends in Intelligent Manufacturing Systems," *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pp. 3209–3214, 2007.

[10] A. Thomas, D. Trentesaux, and P. Valckenaers, “Intelligent distributed production control,” *Journal of Intelligent Manufacturing*, vol. 23, no. 6, pp. 2507–2512, 2012.

[11] F. L. Almeida, B. M. Terra, P. Dias, and G. M. Goncalves, “Adoption Issues of Multi-agent Systems in Manufacturing Industry,” *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, 2010.

[12] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, “Applications of agent-based systems in intelligent manufacturing: An updated review,” *Advanced Engineering Informatics*, vol. 20, no. 4, pp. 415–431, 2006.

[13] M. Uppin and S. Hebbal, “Multi Agent System Model of Supply Chain for Information Sharing,” *Contemporary Engineering Sciences*, vol. 3, no. 1, pp. 1–16, 2010. [Online]. Available: http://m-hikari.com/ces/ces2010/ces1-4-2010/uppinCES1-4-2010.pdf

[14] New South Wales Treasury, “Life Cycle Costing Guideline,” *Total Asset Management*, no. September, p. 14, 2004.

[15] a. Houshyar, “Reliability and Maintainability of Machinery and Equipment, Part 1: Accessibility and Assessing Machine Tool R&M Performance,” *International Journal of Modelling and Simulation*, vol. 24, no. 4, 2005.

[16] B. W. Torell and V. Avelar, “Mean Time Between Failure : Explanation and Standards,” *Power*, vol. 78, pp. 1–10, 2004. [Online]. Available: http://support.casit.net/Portals/0/NTForums{_}Attach/VAVR-5WGTSB{_}R0{_}EN.pdf

[17] H. Ascher and H. Feingold, *Repairable Systems Reliability - modeling, inference, misconceptions, and their causes.* New York : Marcel Dekker, 1984.

[18] O. Basile, P. Dehombreux, and F. Riane, “Identification of reliability models for non repairable and repairable systems with small samples,” *Proceedings of IMS 2004*, 2004.

[19] B. H. Lindqvist, “Statistical Modeling and Analysis of Repairable Systems 2 \ Major events " in the history of repairable systems relia-,” pp. 1–18, 1997.

[20] “inductive automation - 3 Factors That Define OEE,” pp. https://inductiveautomation.com/mes–software/oee–s. [Online]. Available: https://inductiveautomation.com/mes-software/oee-software/oee-calculations/oee-definition

[21] “XPRESS Project.” [Online]. Available: http://www.xpress-project.eu/

[22] T. Ribeiro and G. Gonçalves, “Formal methods for reconfigurable assembly systems,” *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2010*, 2010.

[23] C. Jia, W. Ma, and X. Xia, “Research on modeling and simulating optimization for digital factory,” *2010 International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2010*, vol. 3, pp. 115–119, 2010.

[24] a. Seleim, A. Azab, and T. AlGeddawy, “Simulation methods for changeable manufacturing,” *Procedia CIRP*, vol. 3, no. 1, pp. 179–184, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.procir.2012.07.032

[25] A. Boulonne and A. Skoogh, “Simulation Data Architecture for Sustainable Development,” pp. 3435–3446, 2010.

[26] “AutomationML e.V.. AutomationML.” [Online]. Available: www.automationml.org

[27] N. Schmidt, “AutomationML in a Nutshell,” no. November, 2015.

[28] “International Electrotechnical Commission: IEC 62424 - Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools.” [Online]. Available: www.iec.ch

[29] “International Organization for Standardization: ISO/PAS 17506:2012 - Industrial automation systems and integration – COLLADA digital asset schema specification for 3D visualization of industrial data,” 2012. [Online]. Available: www.iso.org

[30] “Association PLCopen: PLCopen XML,” 2012. [Online]. Available: www.plcopen.org

[31] P. Barlas, G. Dagkakis, C. Heavey, B. Gaffney, P. Young, and J. Geraghty, “Test Implementation and Initialisation of a Simulation model using CMSD,” *Procedia CIRP*, vol. 25, no. Idm, pp. 276–282, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.procir.2014.10.039

[32] J. Fournier, “Model building with core manufacturing simulation data,” *Proceedings - Winter Simulation Conference*, no. Wu 2005, pp. 2214–2222, 2011.

[33] S. Leong, Y. Lee, and F. Riddick, “A core manufacturing simulation data information model for manufacturing applications,” *Simulation Interoperability Workshop, Simulation Interoperability and Standards Organization*, pp. 1–7, 2006. [Online]. Available: http://www.mel.nist.gov/div826/library/doc/sim{_}model.pdf

Appendix A

Common formulas and plots for lifetime models

Formulas and plots in this Appendix from [3]

A.1 Exponential

The exponential model, with only one unknown parameter, is the simplest of all life distribution models. The key equations for the exponential are shown below: [3]

PDF:	$f(t,\lambda) = \lambda e^{-\lambda t}$
CDF:	$F(t) = 1 - e^{-\lambda t}$
Reliability:	$R(t) = e^{-\lambda t}$
Failure Rate:	$h(t) = \lambda$
Cumulative Hazard Rate:	$H(t) = \lambda t$
MTTF:	$\frac{1}{\lambda}$
Median:	$\frac{\ln 2}{\lambda} \cong \frac{0.693}{\lambda}$
Variance:	$\frac{1}{\lambda^2}$

Note that the failure rate reduces to the constant λ for any time. The exponential distribution is the only distribution to have a constant failure rate.

The plots for the exponential PDF and CDF are shown in figures A.1 and A.2 respectively.

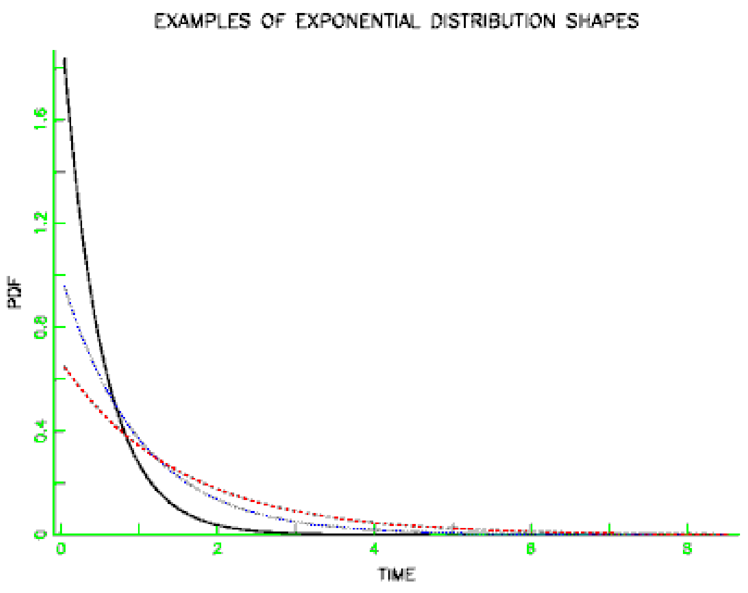


Figure A.1: Exponential PDF plots [3]

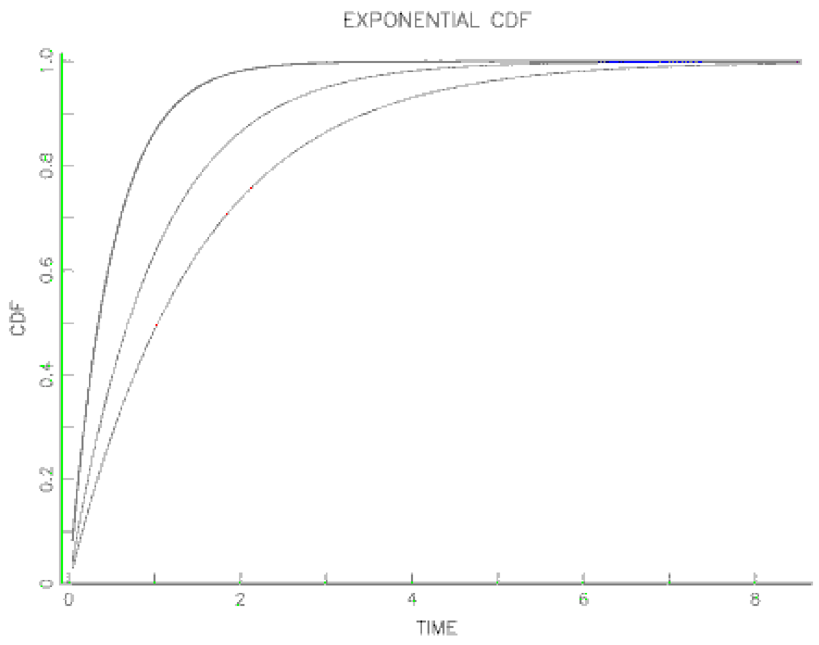


Figure A.2: Exponential CDF plots [3]

A.2 Weibull

The Weibull is a very flexible life distribution model with two parameters. Its CDF and PDF and other key formulas given by: [3]

PDF:	$f(t,\beta,\eta) = \frac{\beta}{t}\left(\frac{t}{\eta}\right)^{\beta} e^{-\left(\frac{t}{\eta}\right)^{\beta}}$
CDF:	$F(t) = 1 - e^{-\left(\frac{t}{\eta}\right)^{\beta}}$
Reliability:	$R(t) = e^{-\left(\frac{t}{\eta}\right)^{\beta}}$
Failure Rate:	$h(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}$
Cumulative Failure Rate:	$H(t) = \left(\frac{t}{\eta}\right)^{\beta}$
Mean:	$\eta\Gamma\left(1+\frac{1}{\beta}\right)$
Median:	$\eta(\ln 2)^{\frac{1}{\beta}}$
Variance:	$\eta^2\Gamma\left(1+\frac{2}{\beta}\right) - \left[\eta\Gamma\left(1+\frac{1}{\gamma}\right)\right]^2$

with η the Scale parameter (the Characteristic Life), β the Shape Parameter, and Γ is the Gamma function with $\Gamma(N) = (N-1)!$ for integer N.

A more general three-parameter form of the Weibull includes an additional waiting time parameter γ (sometimes called a shift or location parameter). The formulas for the 3-parameter Weibull are easily obtained from the above formulas by replacing t by $(t-\gamma)$ wherever t appears. No failure can occur before γ hours, so the time scale starts at γ, and not 0. If a shift parameter γ is known (based, perhaps, on the physics of the failure mode), then all you have to do is subtract γ from all the observed failure times and/or readout times and analyze the resulting shifted data with a two-parameter Weibull. [3]

Special Case: When $\beta = 1$, the Weibull reduces to the Exponential Model, with $\eta = 1/\lambda =$ the mean time to fail (MTTF).

Depending on the value of the shape parameter β, the Weibull model can empirically fit a wide range of data histogram shapes. This is shown by the PDF example curves in figure A.3.

From a failure rate model viewpoint, the Weibull is a natural extension of the constant failure rate exponential model since the Weibull has a polynomial failure rate with exponent $\beta - 1$. This makes all the failure rate curves shown figure A.4 possible.

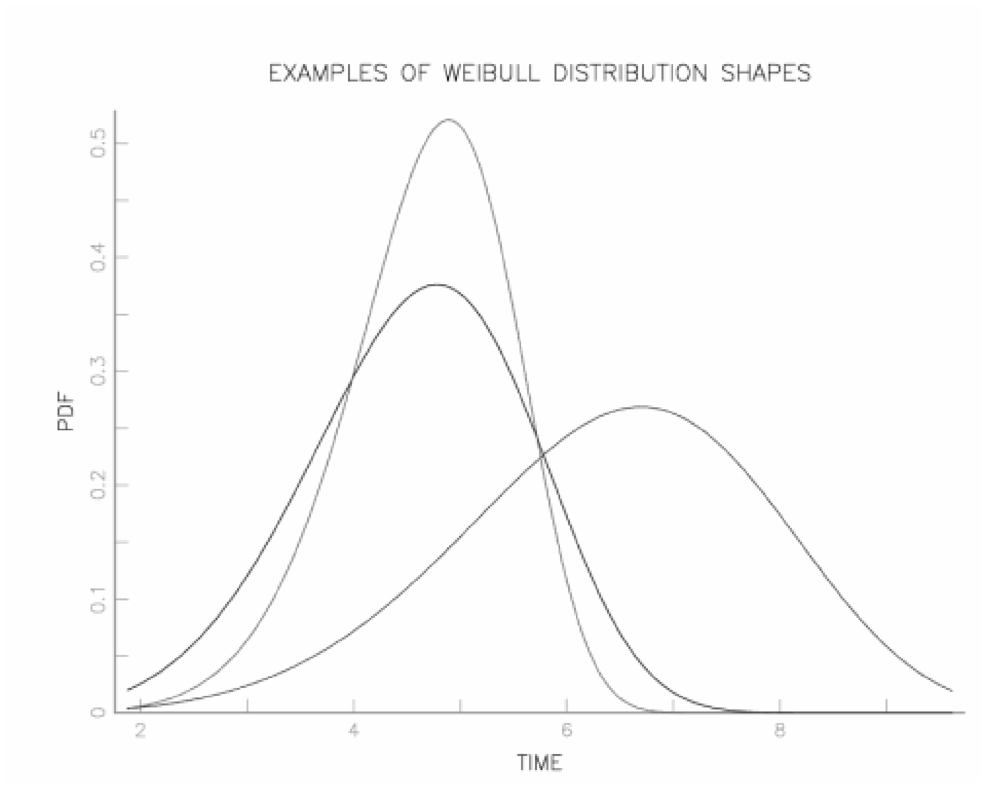


Figure A.3: Examples of Weibull PDF plots [3]

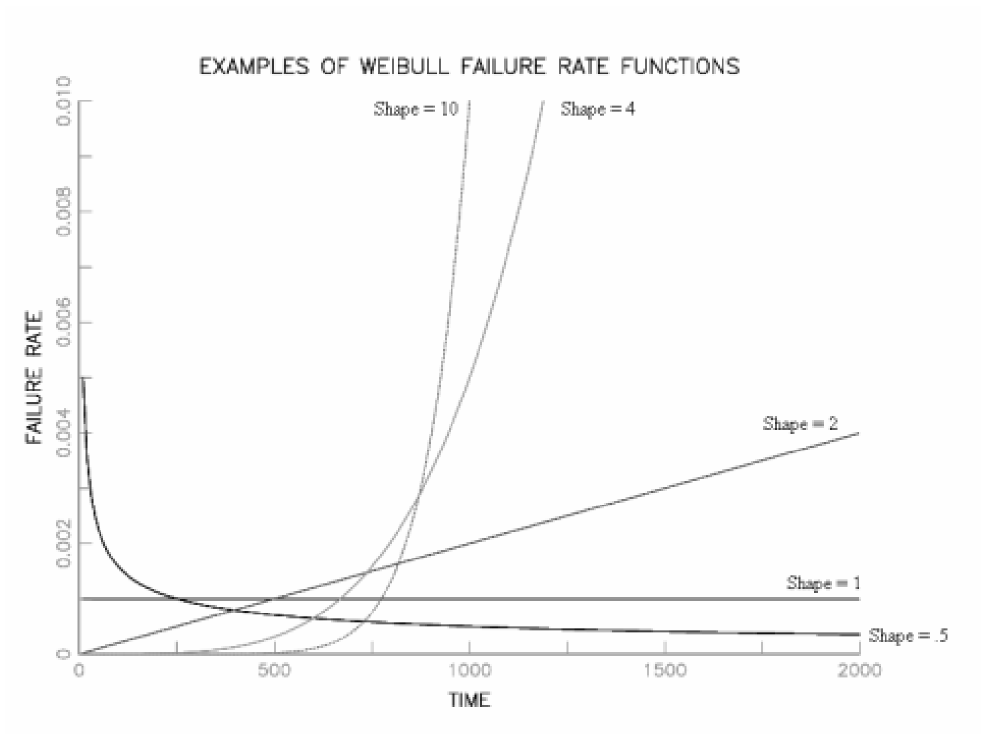


Figure A.4: Weibull Failure Rate plots for different values of β [3]

A.3 Lognormal

The lognormal life distribution, like the Weibull, is a very flexible model that can empirically fit many types of failure data. The two-parameter form has parameters σ is the **shape** parameter and T_{50} is the **median** (a **scale** parameter).

Note: If time to failure, t_f, has a lognormal distribution, then the (natural) logarithm of time to failure has a normal distribution with mean $\mu = lnT_{50}$ and standard deviation σ. This makes lognormal data convenient to work with; just take natural logarithms of all the failure times and censoring times and analyze the resulting normal data. Later on, convert back to real time and lognormal parameters using σ as the lognormal shape and $T_{50} = e^{\mu}$ as the (median) scale parameter.

Below is a summary of the key formulas for the lognormal.

PDF: $f(t) = \frac{1}{\sigma t\sqrt{2\pi}} e^{-\left(\frac{1}{2\sigma^2}\right)(\ln t - \ln T_{50})^2}$

CDF: $F(t) = \int_0^T \frac{1}{\sigma t\sqrt{2\pi}} e^{-\left(\frac{1}{2\sigma^2}\right)(\ln t - \ln T_{50})^2} dt$

$F(t) = \Phi\left(\frac{\ln t - \ln T_{50}}{\sigma}\right)$

$\Phi(z)$ denotes the standard normal CDF.

Reliability: $R(t) = 1 - F(t)$

Failure Rate: $h(t) = \frac{f(t)}{R(t)}$

Mean: $T_{50}\, e^{\frac{1}{2}\sigma^2}$

Median: T_{50}

Variance: $T_{50}^2 e^{\sigma^2}\left(e^{\sigma^2} - 1\right)$

Note: A more general three-parameter form of the lognormal includes an additional **waiting time** parameter θ (sometimes called a **shift** or **location** parameter). The formulas for the three-parameter lognormal are easily obtained from the above formulas by replacing t by $t - \theta)$ wherever t appears. No failure can occur before θ hours, so the time scale starts at θ and not 0. If a shift parameter θ is known (based, perhaps, on the physics of the failure mode), then all you have to do is subtract θ from all the observed failure times and/or readout times and analyze the resulting shifted data with a two-parameter lognormal.

Examples of lognormal PDF and failure rate plots are shown below. Note that lognormal shapes for small sigmas are very similar to Weibull shapes when the shape parameter β is large and large sigmas give plots similar to small Weibull β's. Both distributions are very flexible and it is often difficult to choose which to use based on empirical fits to small samples of (possibly censored) data.

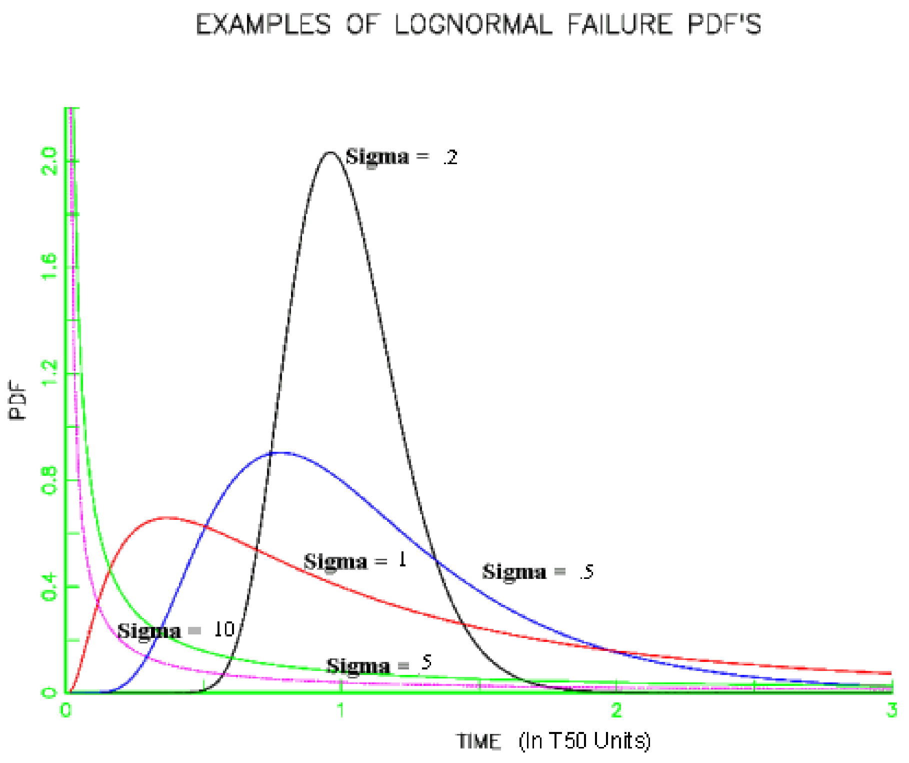


Figure A.5: Examples of Lognormal PDF plots [3]

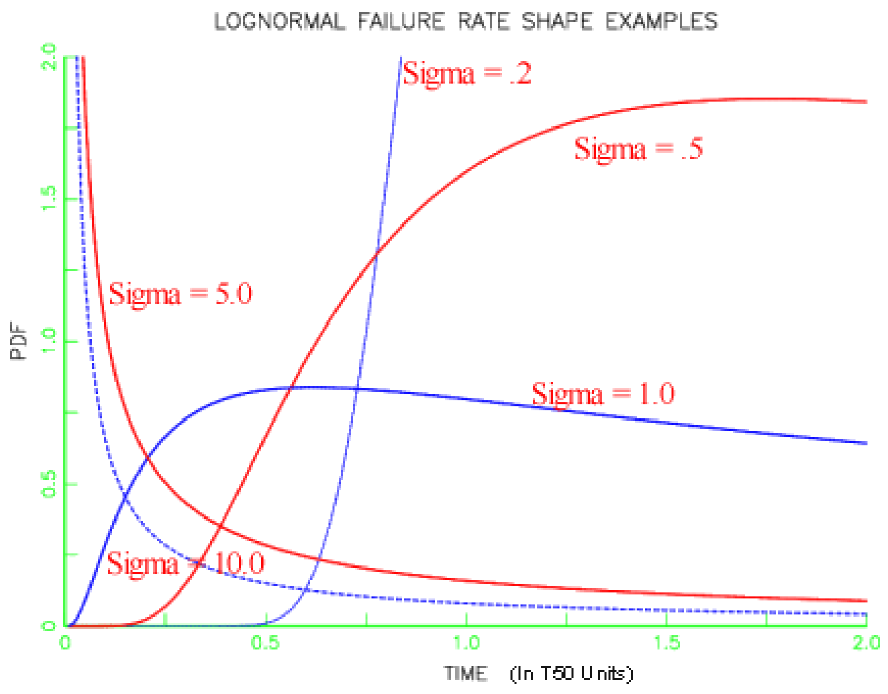


Figure A.6: Examples of Lognormal Failure rate plots [3]

Appendix B

Schematics for bottom-up Reliability

Schematics presented in this Appendix from [3].

B.1 Series Model

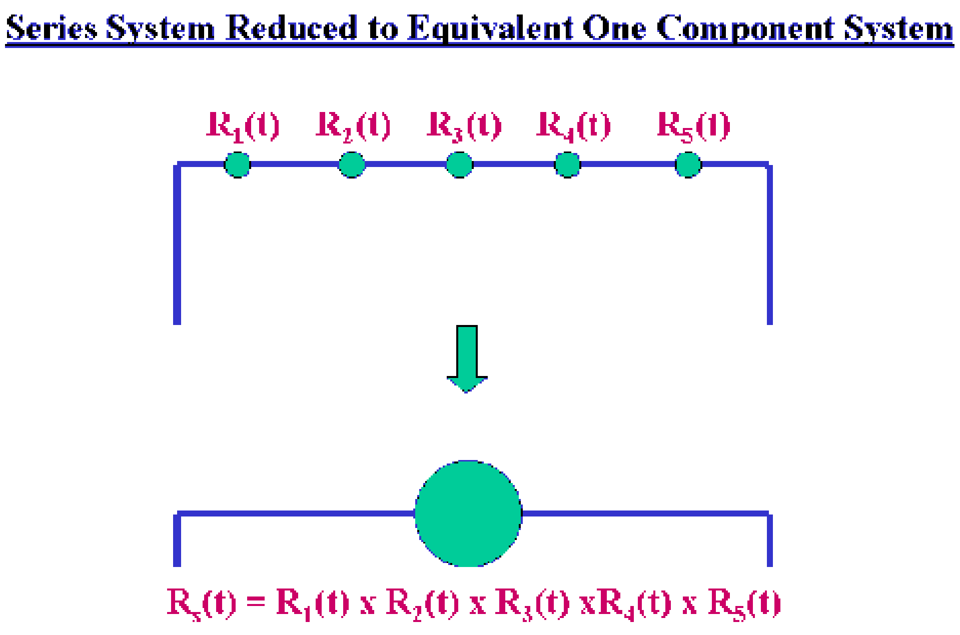

Series System Reduced to Equivalent One Component System
$R_1(t)$
$R_2(t)$
$R_3(t)$
$R_4(t)$
$R_5(t)$
$R_s(t) = R_1(t) \times R_2(t) \times R_3(t) \times R_4(t) \times R_5(t)$

Figure B.1: Example of a Series model [3]

B.2 Parallel Model

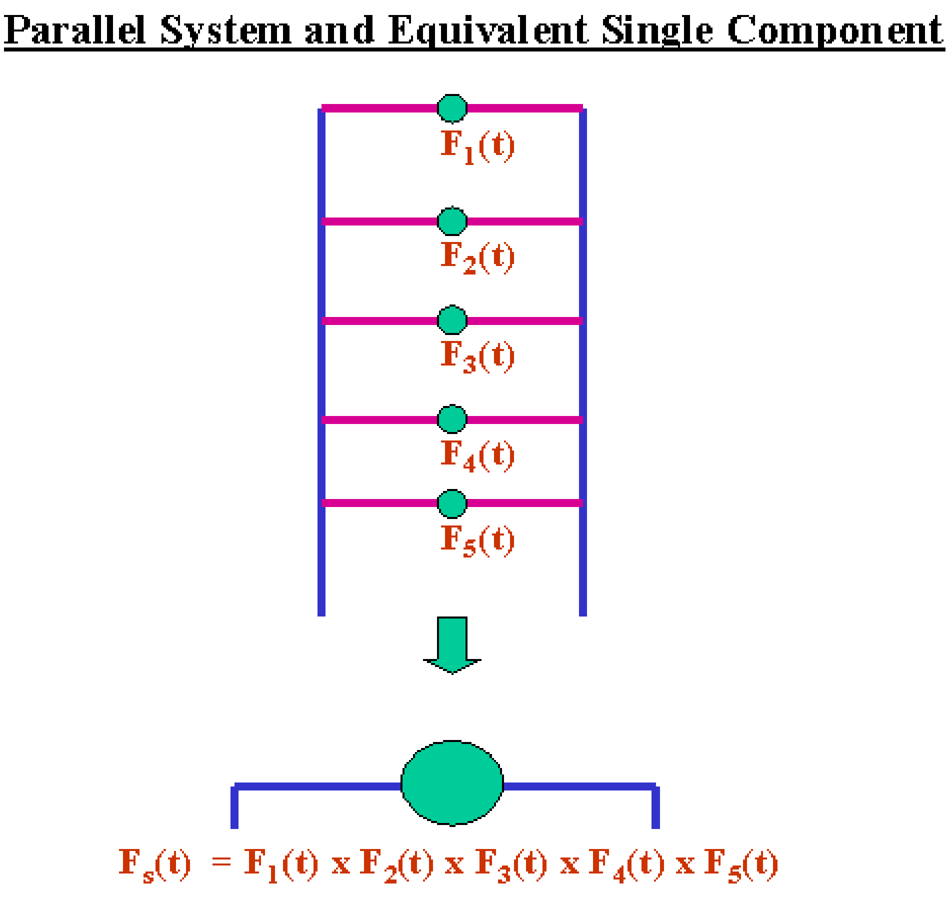

Parallel System and Equivalent Single Component
$F_1(t)$
$F_2(t)$
$F_3(t)$
$F_4(t)$
$F_5(t)$
$F_s(t) = F_1(t) \times F_2(t) \times F_3(t) \times F_4(t) \times F_5(t)$

Figure B.2: Example of a Series model [3]

B.3 Complex Model

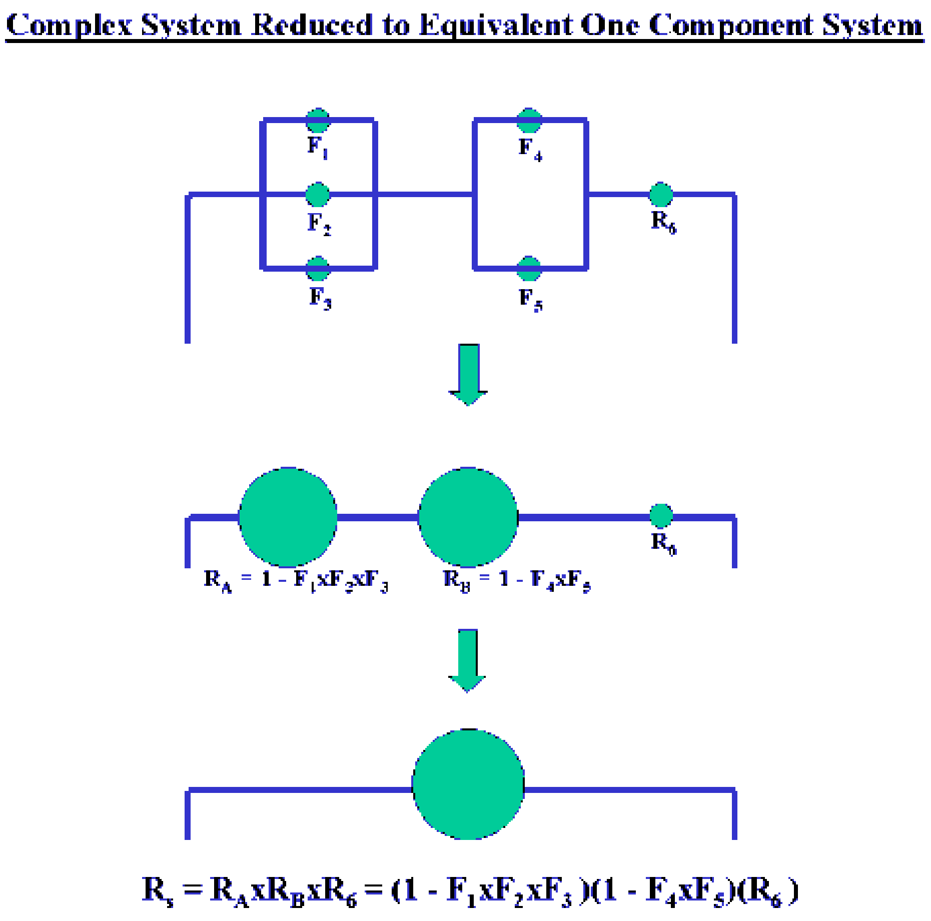

Complex System Reduced to Equivalent One Component System
F_1
F_2
F_3
F_4
F_5
R_6
$R_A = 1 - F_1xF_2xF_3$
$R_B = 1 - F_4xF_5$
R_6
$R_s = R_AxR_BxR_6 = (1 - F_1xF_2xF_3)(1 - F_4xF_5)(R_6)$

Figure B.3: Example of a Complex model [3]