FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Agent-based modeling framework for complex adaptive organizations

**Diogo Pinto**

DISSERTATION PLANNING


Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Eugénio Oliveira

Second Supervisor: Henrique Cardoso

February 14, 2016

# Agent-based modeling framework for complex adaptive organizations

**Diogo Pinto**

Mestrado Integrado em Engenharia Informática e Computação

February 14, 2016

# Abstract

Even though humans achievements are due to their particular way to reason about the world, with approximations, simplifications and derivations, complexity always amazed us, with its deceivingly simple base rules and unforeseen consequences. Many are the systems of individuals found in natures in which the result of of their activities is higher than the sum of its parts, as is the example of ants behavior when searching for food: their search method is very efficient, even though each individual only obeys a simple restricted set of rules.

Broadcasting companies find tackling the complexity that arises from their environments hard to analyze and comprehend. As such, much of the potential for improvement is out of reach of existing tools. In this context, this thesis aims to provide a tool that helps to reason over the behaviors and interactions that take place in a multimedia production environment. By answering the questions "what is happening", "what will happen", and "what would happen if", the knowledge gathered simplifies the system of interactions in such a way that insight can be harnessed, and therefore, action can be taken.

To accomplish its goals, this work subdivides into development of a generic framework and an instantiation of it. The generic framework is constituted by multiple extensible modules, and aims to be instantiable in environments where analysis of behaviors in complex systems is relevant. Thus, not only the example of media production environment is used, but the cyber-physical systems also help to showcase the general scope of the framework. The instantiation of it binds the tool to the specific environment, and is to be deployed alongside a tool already used by many major broadcasting companies.

# Resumo

*"Intelligence is not the product of any singular mechanism but comes from the managed interaction of a diverse variety of resourceful agents. "*

Marvin Minsky

*"It is a profoundly erroneous truism (...) that we should cultivate the habit of thinking of what we are doing. The precise opposite is the case. Civilization advances by extending the number of important operations which we can perform without thinking about them. "*

Alfred North Whitehead

# Contents

CONTENTS

# List of Figures

LIST OF FIGURES

# List of Tables

LIST OF TABLES

# Abbreviations

| | |
|---|---|
| AADL | Architecture Analysis & Design Language |
| ABM | Agent-Based Model(ing) |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| AOSE | Agent-Oriented Software Engineering |
| API | Application Programming Interface |
| CNN | Convolutional Neural Network |
| CPS | Cyber-Physical System |
| DSL | Domain-Specific Language |
| FIPA | Foundation for Intelligent Physical Agents |
| fMRI | functional Magnetic Resonance Imaging |
| FP | Functional Programming |
| IoT | Internet of Things |
| JADE | Java Agent Development Framework |
| JVM | Java Virtual Machine |
| LSTMN | Long Short-Term Memory Network |
| MAS | Multi-Agent System |
| ML | Machine Learning |
| MOA | Massive Online Analysis |
| NN | Neural Network |
| ODAC | Online Divisive Agglomerative Clustering |
| OOP | Object-Oriented Programming |
| RBM | Restricted Boltzmann Machine |
| RNN | Recurrent Neural Network |
| SVD | Singular Value Decomposition |
| TCP | Transmission Control Protocol |
| UI | User Interface |
| UML | Unified Modeling Language |
| UX | User Experience |
| VFDT | Very-Fast Decision Tree |
| WEKA | Waikato Environment for Knowledge Analysis |

# Chapter 1

# Introduction

## 1.1 Context

The content broadcasted to every home's television flows from video cameras to end users by traversing a complex and interleaved series of transferring and transforming processes inside a media production environment. Additionally, the data involved is aggregated in a diverse range of big media assets, that need to be efficiently transformed and transferred.

Those processes are encapsulated over the abstract concept of *ingest steps*, i.e. the process of capturing, transferring, or importing different types of video, audio, or image media (*i.e.* media assets) into a given editing or storing facility. Such processes are already efficiently tackled by modern solutions, even though in a shallow manner: there is no analysis over the actual performance of different ingest steps.

The setting where these processes happen is still very unstructured, facing many deficiencies in what regards the working methodology of content editors and producers. This leaves a wide range of optimizations (and subsequent profits) to be made, that are not yet harnessed by existing systems. But there are no straightforward solutions to any of these problems, and they mainly involve a complex conjunction of several smaller solutions.

## 1.2 MOG Technologies

MOG Technologies is a worldwide supplier of solutions for improving workflow performance of broadcaster companies. One of Mog's products, mxfSPEEDRAIL, focuses on materializing the abstraction provided by the ingest step concept, and provide a full media management system.

MOG Technologies experience led them to devise a project that, through machine learning and big data-related techniques, takes advantage of the unexplored layers of complexity over ingest steps to enhance their state of the art functionality. This enhancement is achieved through conferring a greater degree of autonomy and efficiency over the actions performed in the system, while altogether boosting the media production teams' own productivity.

## 1.3 Motivation and Goals

As it was aforementioned, there is still a lot of value to gain from analyzing the ingest operations performed over media assets in a production environment. As these operations define most of the very own environment where they take place, and current systems already manage and record these actions, this is the par excellence opportunity to introduce such optimizations.

As such, this thesis hypothesizes that, by taking advantage of the information gathered and generated in these ingest steps (*e.g.* sources and destinations, transformations, nature of the assets), it is possible to provide a versatile high-level way to reason over the behavior of the individual agents (*e.g.* people or content servers), as well as over the whole system of interactions. Concretely, this is achieved through the following dimensions:

1. To describe what is happening.
   *e.g.* time-series distribution of errors in ingest procedures; distribution of activity time throughout resources.

2. To predict what is going to happen.
   *e.g.* availability of asset repositories; spikes in activity in the servers.

3. To test what would happen if something was true.
   *e.g.* if a given resource becomes full, what is the impact in the rest of the resources.

If one abstracts away from the specifics of the context that this hypothesis targets, it is possible to understand that this environment naturally fits under the concept of a complex system, *i.e.* a system where from the interactions originated by the agents' simpler behaviors, complexer phenomena emerges[1]. Therefore, it is also possible to abstract the prior hypothesis, widening its range of applicability to many other use cases. One such example is software development teams, where the model can be fed by, *e.g.*, a version control system. As the number of programmers, system designers and architects grows, there has to be efficient ways to ease communication and organize the human resources into smaller and efficient groups. Other example is social networks. The number of users and actions taken in these systems is very high, and underneath that dimension lies great value in understanding how simple user actions map to, *e.g.* politic decisions and movements.

Next, it is briefly discussed the benefits that can be accomplished with such solution over two other use cases: enhancement of social web-based services (which involves in greater extent behavioral modeling), and analysis of cyber-physical systems.

### 1.3.1 Social Web-Based Services

There is a growing interest in enhancing the UX [2] of web-based services through experience personalization, specially in services where users interact among themselves. One heuristic for

[1] This topic is further discussed in Section 2.1
[2] User Experience

achieving this is to model the users' interaction behaviors, and then simulate scenarios, so that the system is able to adapt to them, if they ever occur in the real world. In addition to being a relevant problem for the development of machine learning in general, efforts are being made to accomplish this task even by big companies in the artificial intelligence field like Google [SHM+16], who saw in the quest of Go game mastering a bridge to enhance their own services personalization [SHM+16].

### 1.3.2 Cyber-Physical Systems

The field of CPS[3], usually referred to as IoT[4] or Industrial Internet, merges physical and virtual environments, being specially interesting for applying this thesis proposal. These systems are gaining ground and will certainly shape our lives as we know them. On the other hand, they are inherently complex: the possibilities for interaction between devices grow exponentially with the number of devices, therefore, this complexity has to be properly analyzed and maintained.

This thesis final hypothesis is, then, that it is possible not only to understand how higher-level behaviors emerge from lower-level ones, but also how that mapping is performed. To accomplish this, this thesis has the goal of developing a scalable framework for modeling inter-related agents' behaviors in the context of complex systems.

## 1.4 Contributions

This thesis contributions consist of not only solving problems faced everyday by media producers and editors at broadcasting companies, but also of providing a reusable and extensible tool for behavior analysis in complex systems to the scientific community (namely in the field of CPS). That system is to support:

- a real-time simulation environment for complex systems, fed from a data stream;
- induction of resource's behaviors from the data stream;
- definition of behaviors through a DSL[5];
- analysis over different layers of the complex system (individual, sub-graphs and global);
- provide predictions of future events;
- scale to reasonably-sized complex systems;
- enable features' extension through additional modules.

[3] Cyber-Physical Systems
[4] Internet of Things
[5] Domain-Specific Language

## 1.5 Document Structure

This document is structured as follows:

- **Chapter 1: Introduction:** Description of the context and motivation for the work, together with main goals and expected contributions.
- **Chapter 2: Literature Review:** Review of the work developed in the fields of Complex Systems, Cyber-Physical Systems, Agent-Based Modeling, Real-Time Data Mining, and Maintenance and Usability.
- **Chapter 3: Solution Perspective:** Using the knowledge gathered in the literature review, and after a discussion of technologies that may aid the thesis development, a solution for the problem is devised.
- **Chapter 4: Conclusions and Future Work:** Final considerations and future directions.

# Chapter 2

# Literature Review

This thesis is built upon current efforts in understanding behaviors, and scaling that analysis to complex systems' level. Some projects and research proposals are presented in this chapter, which directly or indirectly influence the decisions taken in the thesis development. Some of the content is merely introductory, being left out for further analysis as future work.

This chapter starts with an overview of complex systems in Section 2.1, materializing the concepts in the context of cyber-physical systems in Section 2.2. Multi-agent-based computing usefulness is introduced in Section 2.3, while the importance of behavioral cloning tasks is discussed in Section 2.4. Due to the nature of the environment where the system is to be deployed, a careful analysis over real-time data mining is done in Section 2.5, alongside some key algorithms. The chapter ends with some considerations on maintenance and usability questions in Section 2.6, so that the development is kept on the right tracks.

## 2.1 Complex Systems

Complex systems are systems made of many similar, interacting parts. Each part behaves according to a relatively simple set of rules, but from the whole system certain properties emerge that are greater than the sum of its parts [Sor07].

One of the richer sources of complex systems is nature itself. This happens because natural selection finds in complexity tool for communities to be able to adapt and persevere. Many animal species – such as ants, dolphins and bats – were already studied with the aim of understanding how from their simple behaviors complex societies emerge [KPS11]. Humans, being also an highly social specie, also give rise to complex properties from their (arguably) complex behavior [Sor08, SMG14].

Fundamentally, development of cyber-physical systems has been restricted by the cost of computer chips, the dissemination of internet, and computing capacity to process large amounts of data

in real-time. With the current democratization of the first two restrictions, big data-related techniques play a major role in what can be achieved with CPS. Big data is a broad concept, and is many times used in different contexts, but all its definitions have in common that it relates to the usage of vast amounts of data. It is usually described according to the following aspects (known as the seven Vs):

- **Volume:** data is in the order of terabytes or larger.
- **Velocity:** capture, transfer, computation, store and access of data is time critical, and commonly performed in real-time.
- **Variety:** data may have different types and be in various degrees of structuredness.
- **Veracity:** data may contain uncertainty or impreciseness.
- **Validity:** there is a need for ensuring that both the measurements are correct, and that the process is sufficiently transparent.
- **Value:** big data brings down past limits on what can be accomplished with vasts amounts of data.
- **Volatility:** data may vary in degree of volatility, being many times susceptible to only one usage, at its arrival.

Big data also broadens the possibilities presented by cyber-physical systems, a kind of system discussed in Section 2.2 that encompasses many properties of complex systems. But complex systems influence computing science not only through big data. For example, the fundamental aspect of neural networks is that they aggregate many cells that map linear data transformations over non-linear functions, and share the result with their neighbors. This way, they achieve the ability of modeling highly complex functions[1]. Additionally, swarm intelligence studies originated ant colony optimization, which is inspired from natural ants' behavior to minimize objective functions withing certain restrictions [DBS06].

Uncovering both the lower-level rules, and how they map into the emergent properties of complex systems proves to be a challenge. Fundamentally, this thesis objective is to devise an empirical method for understanding them, through analysis of behaviors at their different complexity levels.

## 2.2 Cyber-Physical Systems

CPS[2] are abstractly defined as the integration of virtual computation over physical systems [SGLW08]. They are expected to permeate our every-day lives in the near future, with a scope that ranges from transportation to health and energy. Due to recent improvements and democratization of data mining techniques, vast investments are being made for CPS development. As a matter of

[1] This topic is further extended in Section 2.5.2.
[2] Cyber-Physical Systems

fact, according to a study by McKinsey Global Institute, Internet of Things (IoT)[3] has an economic potential of up to $11.1 trillion a year by 2025 [MCB+15].

Nonetheless, to harness their full potential, integrated systems must be coordinated, distributed, and connected, while assuring robustness and responsiveness [Lee08].

This unique combination of both high complexity and high pervasive power gives rise to many difficulties, and not only in what regards maximizing their utility. Cyber crime is one such problem, which already represents $100 billion of loss in the United States of America alone [Jou13]. Depending on the criticalness of the role performed, CPS may also be subject to timing predictability (*i.e.* the system must react within a specific time frame), and also to energy consumption restraints. Arguably more important are even the human, social, philosophical and legal issues that arise, which demand robust assurances from the CPS. Together with the dynamism and heterogeneity of the systems that underlie CPS (*e.g.* network standards), they prove to be a major challenge to reason about.

To ease the complexity of building such systems, one of the most promising approaches is to intertwine the development of the system with a simulation of it. Zhang [Zha14] proposed an architecture for modeling CPS through an object-oriented design, where he integrates AADL[4] [Hug13], ModelicaML [Sch09], and clock theory [Jif13]. AADL is used to model and analyze both software and hardware architectures of embedded systems, which comprises hardware and software components' description. Modelica is a language for virtually defining physical systems [Ass10], and ModelicaML enables definitions that use UML[5] class diagrams syntax to be transformed into Modelica language.

One of the aims of this thesis is to not only enable the definition of resources behaviors, like the system demonstrated by Zhang, but also to introduce in these systems effective ways to induce those behaviors from execution, therefore expanding the scope of these CPS simulations to other grounds.

## 2.3 Multi-Agent-Based Computing

Software systems vary widely in range of complexity. Numerous ways have been discussed and concretely implemented that aim to improve the capacity of programmers and system designers to cope with such complexity, as is the example of the OOP[6] paradigm, from which a series of well-known design patterns have been derived [GHJV95], and UML [Omg98].

As levels of complexity grow, there is a trade-off between relaxation of optimal solution and computational power needed. To answer those questions, MAS[7] aims to divide complexity into

[3] the coarse term for CPS
[4] Architecture Analysis & Design Language
[5] Unified Modeling Language
[6] Object-Oriented Programming
[7] Multi-Agent System

manageable computational entities, with well-defined responsibilities [Bon02]. These computational entities interact between themselves according to specific protocols. This division of responsibilities leads also to an efficient distribution over multiple computing nodes.

This intuitive way to deal with complexity is extended even to the software design stage, in the form of AOSE[8]. For that, there exist several well-defined methodologies, like Tropos [BPG+04] and Gaia [WJK00]. These methodologies are successfully used in various examples, like the managing of an airline operations control centre by Oliveira and Castro [CO08]. Muller and Fischer, while reviewing the actual impact of MAS [MF14], present one other example of Google, which uses a MAS for automated bidding for their advertisement slots.

These fundamental principles affect this thesis in two ways:

- AOSE concepts are used to cope with the architectural complexity of the system.
- Each entity that is mapped into the system is modeled as an agent. That agent learns its behavior from the external entity, and reproduces it in the simulation environment. This results in a merge of ABM[9] and behavioral cloning[10], as a way to cope with behaviors in complex systems.

## 2.4 Behavioral Cloning

With the democratization of machine learning techniques, behavioral analysis became very relevant: such knowledge may lead to great and effective improvements in systems' UX[11]. This analysis can be accomplished mainly in two ways: by describing the behavior, or by really modeling (*i.e.* cloning) it, which is a task whose nature represents more of a challenge, but also holds the pottential for greater rewards, as it is possible to both have access to a user's behavior on-demand, as well as to synthesize new behaviors.

Recommender systems essentially accomplish the task of describing users' behaviors. These systems can be divided into two groups: content based recommendation and collaborative recommendation [Par13]. The content based approach creates a profile for each user based on her characteristics and his past. The utility of a new item $s$ for user $u$ is estimated by looking at items $s_i$, assigned to the users that have similar profile to $u$. On the other hand, the collaborative approach predicts the utility of a new item $s$ by looking at the utility of the same item $s$ assigned to users with similar assigned items in the past.

Both methods have their strengths and weaknesses. The collaborative approach is considered to be more accurate than the content-based one, but suffers from "cold start problem" due to the need for a large base of users in order to perform valid recommendations.

A significant boost to research in recommender systems was due to NetFlix. This company offered a prize of $1,000,000 to the first person or team to beat their recommender system,

[8] Agent-Oriented Software Engineering
[9] Agent-Based Model(ing)
[10] Further discussed in Section 2.4.
[11] User Experience

called CineMatch, by 10%. The winning entry of the competition was a combination of several algorithms that had been developed independently. They used several SVD[12] models, including SVD++ (a slight modification to the original SVD algorithm), blended with RBM[13] models [Gow14].

Other recent efforts aim to model behaviors in a deeper level. One such recent effort was employed by Google, with the task of building AlphaGo, a system built to master the game Go using deep learning[14] [SHM+16]. Go is a game known not only to require high reasoning and logic skills from the players, but also demands creativity and imagination. This happens because of the enormous number of possible board states, roughly approximated to be $2.082 \times 10^{170}$ [TF07]. One of the essential steps on the training stage was to mimic the behavior of professional players through reinforcement learning, and only after improving through playing the game against itself. With this approach, by copying how humans play, there is no need to parse the whole search tree for optimal moves. This confers a certain degree of "intuition" to the algorithm itself.

Probabilistic programming is one other category that holds the potential to further improve behavioral cloning performance. Lake *et al.* introduced a method to induce probabilistic programs when learning new concepts [LST15]. This process achieves learning behaviors more similar to what humans accomplish: generalizing from few examples, and learning to learn, *i.e.* increasing the learning performance of new concepts from past experience.

Video-games industry have already been modeling behaviors for a long time. In this environment, AI[15] reasoning has to be encoded in a way both familiar to designers and programmers.

Behavior trees were devised due to the lack of expressiveness and of reusability of state machines in the job of describing AI reasoning [CH07, SGJ+11]. They are tree-like structures constituted by three kinds of nodes: predicate nodes, sequence nodes and select nodes; where predicate nodes may have a mix of interactions and conditional checks with the world, sequence nodes are parents of other nodes that are executed in a specific order and that must return *True* for this to return *True*, and select nodes execute all its child nodes, returning *True* if one of them returns *True*. An example is showcased in Figure 2.1.

Due to the natural properties of trees representation, discussed here and expanded in Section 2.5.1, the adaptation of decision trees' induction algorithms to the structure of behavior trees presents an useful algorithm to use in this thesis context.

## 2.5 Real-Time Data Mining

Traditional data mining techniques were developed and applied having static datasets in mind, *i.e.* all the data is available at the learning stage. That is the direct result of two assumptions: the data is generated from a stationary distribution, and we are able to gather enough data before the

[12] Singular Value Decomposition
[13] Restricted Boltzmann Machine
[14] Topic further discussed in Section 2.5.2
[15] Artificial Intelligence

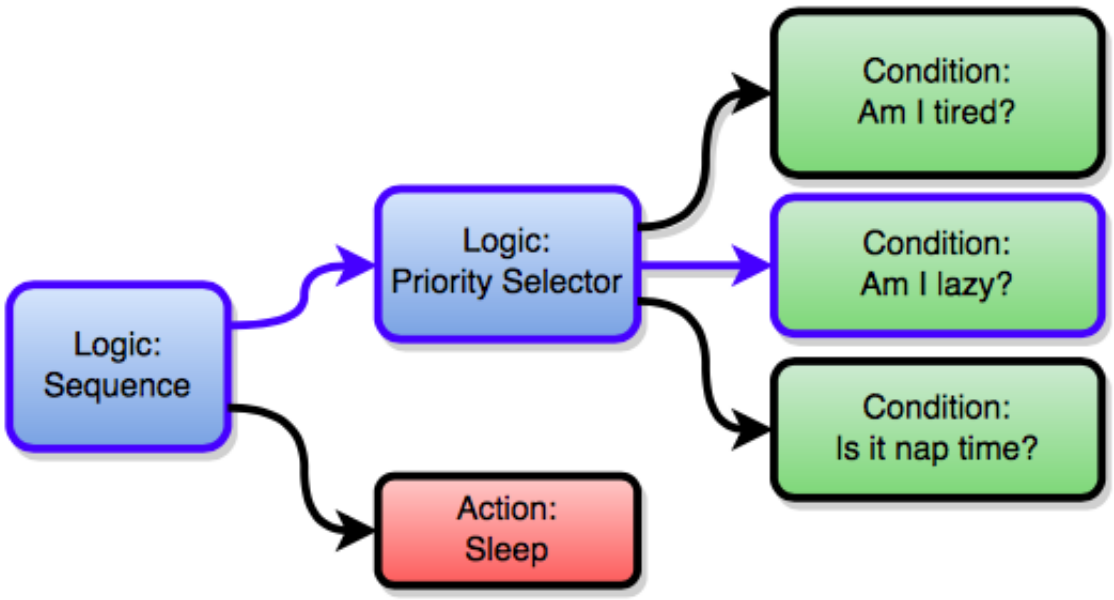


Figure 11: Selector Node Evaluating "Should I Sleep" Logic

Figure 2.1: An example behavior tree (Source: \hyphenation{https://gamedevdaily.io/managing-ai-in-gigantic-523dc84763cf})

system is deployed into production. This led to the common methodology of retraining a classifier from time to time, as its performance deteriorates.

A widely-spread study performed by DOMO [DOM15] on several well-known providers of web-services tries to alert to the fact that data is growing in size and speed at exponential rates: in 2015, every minute, an approximate average of 350 thousand tweets are published on Twitter and 4 million posts on Facebook.

In this context of data streaming, traditional data mining strategies suffer from starting with the assumption that it is possible to gather the needed data for training before deploying: if we try to use vasts amounts of data, it is difficult to scale the methodologies and obtain result in a reasonable time-frame, but if we downscale the data size, the models we train may be nowhere near useful.

Altogether, data streams characterize by:

- the data arriving sequentially, fast, dynamically and asynchronously;
- the streams are generally considered to be infinite;
- the data distributions change over time;
- the objects that arrive are unlabeled.

To answer the need for dealing with that different nature of data, real-time data mining emerged as a natural evolution of traditional data mining. Due to the nature of data streams, the developed techniques start off with the following assumptions:

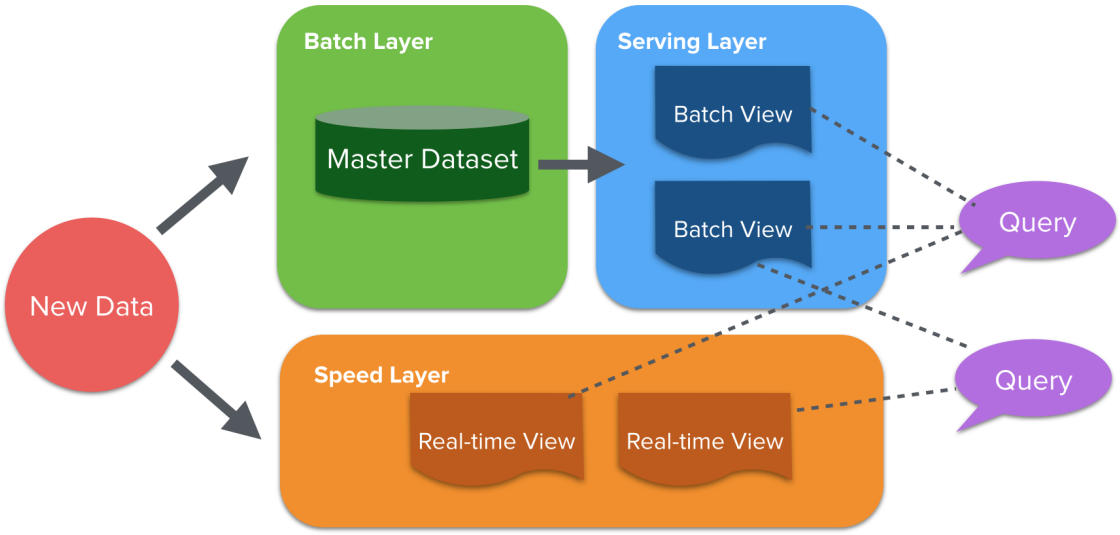


Figure 2.2: Generic schema for Lambda Architecture deployments (Adapted from: http://lambda-architecture.net/

- data must be accessed only once;
- data cannot be stored in (main) memory;
- decision models must be continuously updated.

This leads, essentially, to decision models that must be able to incorporate new information, detect changes and adapt accordingly, and to forget outdated information.

Currently, many deployments of streaming systems is performed according to the Lambda Architecture, whose schema is shown in Figure 2.2. Originally proposed by Nathan Marz [MW13], it focus on managing the trade-off between batch systems, properly optimized but with higher latency, and streaming systems, with lower latency but also lower performance, combining their results in an adaptable service for its clients.

Gama *et al.* [GŽB+14], with the schema presented in Figure 2.3, demonstrate the architecture of a robust system, solely based in online learning. As a consequence of these systems' target of being deployed only once, they must be highly adaptable in their core. Therefore, not only the prediction part is relevant in the architecture, but a self-diagnosis part that provides feedback also represents a key feature for them to be reliable.

### 2.5.1 Decision Trees

Decision trees are widely used due to producing results that are very intuitive to interpret. Therefore, their use to model behaviors may provide a deeper insight into why certain decisions are performed.

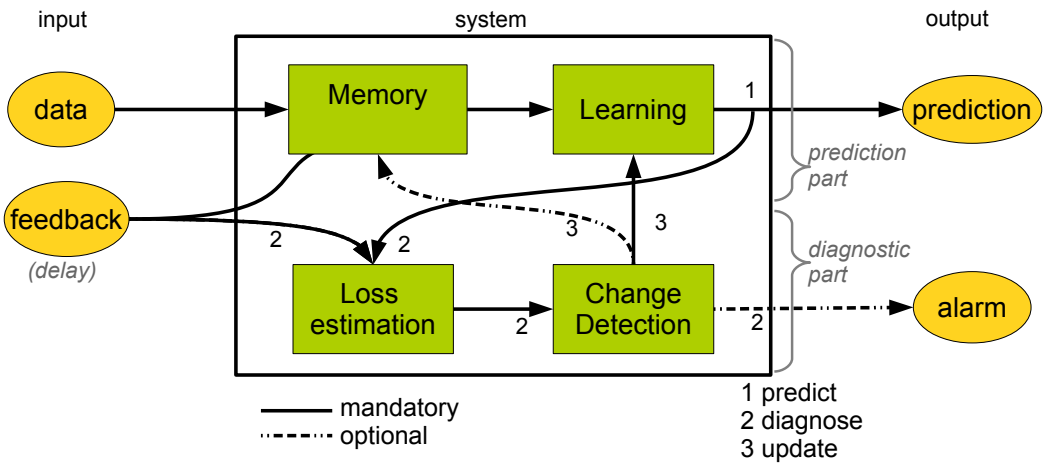


Figure 2.3: Generic schema for online adaptive learning algorithms [GŽB+14]

They are one of the most common and traditional algorithms in AI, and the most common specifications are ID3 [Qui86] and C4.5 [Qui92]. Even though, these specifications are designed to work in a batch setting: all the data is available at model's creation time.

For the task at hand, the behavioral information arrives in a continuous stream of data. For that setting, Domingos and Hulten introduced VFDT[16] [DH00]. As they use the concept of Hoeffding bound in tree construction stage, the performance is guaranteed to converge asymptotically to the one of a batch algorithm fed with infinite data.

Nonetheless, this method still has the disadvantage of considering that the distribution from which the data is obtained is static; in order to address that, VFDT was adapted into CVFDT [HSD01], introducing the notion of concept drift into the algorithm.

### 2.5.2 Deep Learning

Neural networks started as crude approximations of the mechanisms found in neuroscience. Even though, their value was only discovered when computing power enabled both the processing of high volumes of data, and also the construction of bigger neural networks. This gave rise to the concept of *deep learning*.

One of the earliest use cases that started this interest was in computer vision: LeCun developed a CNN[17] named LeNet, to recognize handwritten characters in documents [LBBH98]. This neural network used multiple convolutions to abstract classification from position and scale in spatial representations.

Due to their training being performed using stochastic updates, neural networks are also naturally suited for online learning. Nonetheless, the application of deep learning to data streams demanded also that they were able to produce decisions not only according to each sample of data

[16] Very-Fast Decision Tree
[17] Convolutional Neural Network

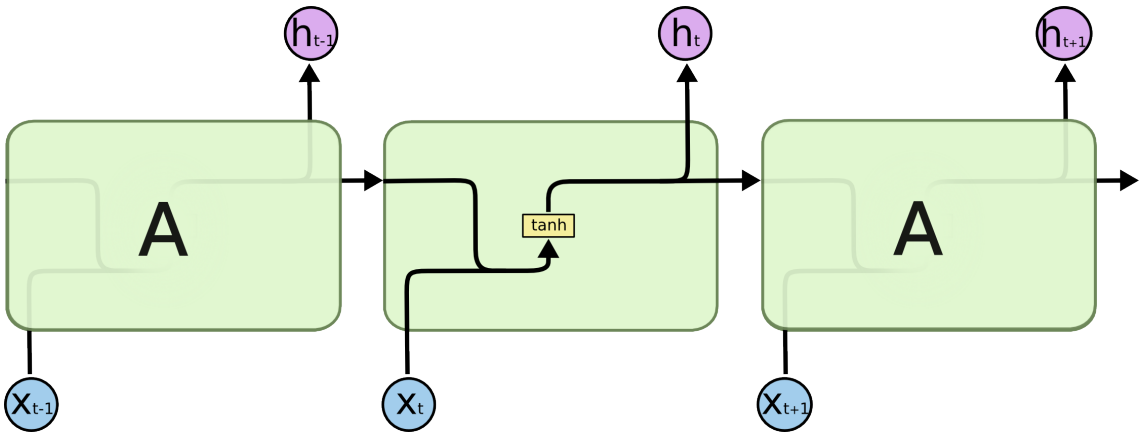


Figure 2.4: The repeating module in an RNN. (Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

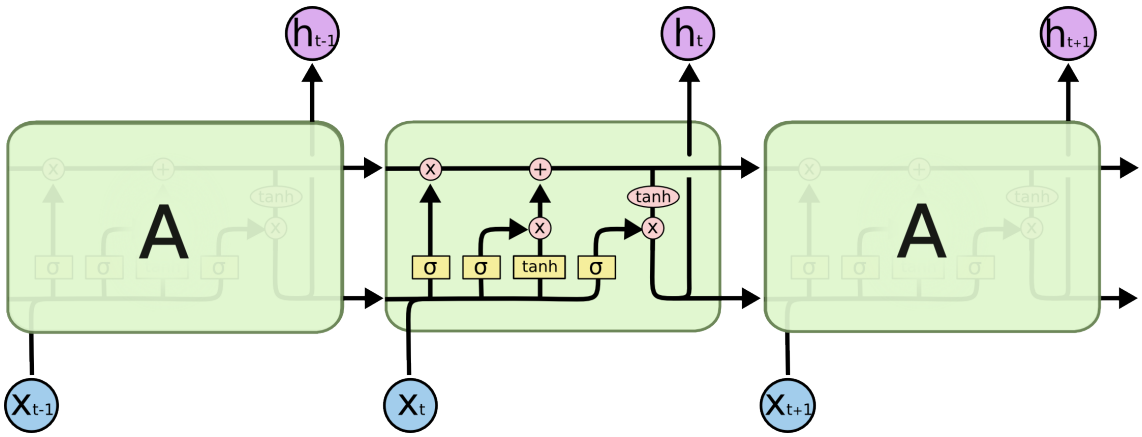


Figure 2.5: The repeating module in an LSTM. (Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

that arrives, but using past information as well. The answer for that was found in recurrence, with the proliferation of RNN[18] usage. Nonetheless, the architecture of these networks, showcased in Figure 2.4, was only competent in producing decisions when only the near past of data samples are relevant.

In that context, Hochreiter and Schmidhuber introduced LSTM[19] Networks [HHSS97], whose original architecture is showcased in Figure 2.5. They introduced into the cell's definition a long memory component, alongside four gates, which enabled the reliable usage of old knowledge from the time series data into making decisions. These neural networks are also naturally adaptable to reinforcement learning tasks in a streaming setting, and are, therefore, useful for the project at hands.

### 2.5.3 Graph Analysis

Behind a complex system is a network that defines the interactions between its components. By analyzing the underlying network, then, we can better understand the original system.

Traditionally, most of the analysis performed on graphs were either at node and edge level, or at global level. As an example of this, Brin and Page incorporated in the Google search engine the node classification into hubs or authorities [BP98].

But nowadays, graphs commonly take dimensions ranging from $10^9$ to $10^{23}$ nodes, representing social networks, chip designs or even the human brain. With this increasing relevance of complex systems, higher-level concepts have the potential to provide greater insight when used in the analysis.

Sub-graphs are being widely used as the building blocks of the larger complex systems they define. Milo *et al.* used sub-graphs distributions to characterize and discriminate graphs from different natures [MIK+04]. Based on that, Milo *et al.* introduced the concept of network motifs [MSOIK02], defining them as recurring (*i.e.* with high frequency) and significant (*i.e.* that are more frequent than expected) patterns of interconnections (*i.e.* the induced sub-graphs). Ribeiro and Silva further introduced G-tries as an efficient data structure for discovering network motifs [RS10].

These abstraction ideas prove to be very useful, and they are even found in the human brain's architecture, as a way to manage complexity. Through analysis of fMRI[20] experiments, Taylor *et al.* found that abstract thoughts are hierarchically dependent between themselves, where thoughts with higher-level abstract nature result from information aggregation, combination and refinement of lower-level thoughts [THBS15].

When analyzing a graph in its whole, one relevant analysis is community detection. Also referred to as clustering, it essentially bifurcates into inclusive clustering (*i.e.* a single node may belong to more than one cluster) and exclusive clustering (*i.e.* a single can only belong to one of the clusters).

[18] Recurrent Neural Networks
[19] Long Short-Term Memory
[20] functional Magnetic Resonance Imaging

Traditional exclusive clustering algorithms, like K-Means [Mac67], only work in batch setting, and therefore are not well-suited to the problem at hands. One of the first works is from Fisher [Fis87], who introduced CobWeb, a method for incremental clustering, which organizes the observations into a classification tree, enabling the prediction of the class for a new observation. Additionally, Aggarwal *et al.* introduced CluStream [AWC+03], which divided clustering process into two stages: an online one, where summary statistics were being updated according to the data stream, and another one, this time offline, which consists on answering the user queries. P. Rodrigues *et al.* introduced ODAC[21], which uses hierarchical clustering as the basis to perform progressive merges and partitions of clusters.

Analyzing graphs' clusters in the context of data streams presents an even more interesting aspect: the study of clusters' evolution. Palla *et al.* lists 6 different cluster events [PBV07]: growth, contraction, merging, splitting, birth, and death. Grene *et al.* provided a concrete implementation, based on those principles [GDC10].

## 2.6 Maintenance and Usability

Despite this thesis major focus on theoretical questions on how to cope with complexity, the resulting artifacts of this thesis are to be instantiated into an existing tool, already with a large volume of clients. Therefore, both maintenance and usability of the system should be key aspects from which to devise design and implementation decisions.

Sculley *et al.* warns about the tendency of easy and fast deployment of ML[22] systems, but difficult and expensive maintenance over time [SHG+15]. He argues that ML systems incur in an high technical debt by combining existing problems in software engineering with specific ones to ML, and this debt is not easily paid by usual methods, *e.g.* refactoring or improvement of unit tests. ML brings to the equation erosion of system's boundaries, data dependencies, and other problems with it, that must be carefully checked for anti-patterns.

Nonetheless, the major evaluation metric for a given software project is how large the degree of the target users' usability is.

Visualizations are a major vehicle for transmitting data and interacting with the users, and through careful planning of elements' visual arrangement, it is possible to control better how the end user interacts with the system [Shn96].

Dietvorst *et al.* [DSM15] alerts to other phenomena that threatens data-based systems: algorithm aversion. Empirically, he demonstrates that humans lose confidence in algorithms faster than in humans, even if the algorithms' error-rate is lower. For any reasonably sized and interesting problem, we can only asymptotically approximate a system's error to zero, but there are other ways to prevent consequences' escalation. Ideally, the system should persuade, by providing explanations and insight over the reason of its outputs, and also adapt, not only to new data, but also to users' feedback. Regarding the first aspect, a tree-like structure, discussed in Section 2.5.1,

[21] Online Divisive Agglomerative Clustering
[22] Machine Learning

is expected to enable higher persuasion than, *e.g.*, a NN[23]. As of the latter aspect, the architecture by Gama *et al.* shown in Figure 2.3 addresses it, reinforcing the great need for adaptability in streaming environments.

## 2.7 Conclusion

This chapter reviewed the theory, algorithms and tools that support this thesis development.

While complex systems pose a big challenge, their value is very prominent, and by using the correct approximations, it is possible to tackle them in feasible time. On the other hand, behaviors are a delicate subject to model, but the adaptation of robust techniques that work in streaming environments may provide a useful insight to make concrete decisions. Additionally, the existing options for CPS augments the potential that inducing the resource's behaviors brings to CPS simulations is highlighted.

Both older but more robust knowledge, like agent-based computing, and very recent research, as is the example of online learning, are merged into a tool that is expected to provide new ways to analyze behaviors over complex systems.

[23] Neural Network

# Chapter 3

# Solution Perspective

This chapter assembles the knowledge gathered in the literature review into a solution proposal. It is presented an overview of the solution, with respect to its main design decisions, using the context of media production environments.

Firstly, technologies that aid the development are reviewed in Section 3.1. Then, the methodological approach is presented in Section 3.2, which is to be subject to validation and evaluation procedures presented in Section 3.3. Lastly, the activities planning to achieve the solution is presented in Section 3.4.

## 3.1 Technological Review

In this section is is discussed the utility of two tools, Scala and Python, and how they can aid the development of the solution.

Python is expected to be mostly used for prototyping. On the other hand, Scala will be used for the core of the solution, only delegating to Python certain machine learning tasks for which it is more robust.

### 3.1.1 Scala's Environment

Scala is a language that seamlessly combines functional and object-oriented programming [OSV08]. Together with keeping up-to-date with the research in programming languages, it achieves a very expressive, scalable and reliable environment on which to develop applications.

Due to its minimal syntax and powerful compiler with various implicit transformations, it is adapted to defining DSL, which enables the development of an intuitive framework. Its functional nature also eases the tasks of dealing with data flows concurrently, and enable the native support for parser generation.

One other advantage of working in Scala is that we do not have to compromise libraries availability, as it is able to run on top of the JVM[1]. Therefore, this project may benefit from the following libraries:

- **JADE**[2], which simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA[3] specifications [BCG07]. In this context, this library may ease the managing of communication protocols between the various agents, alongside enabling the distribution of computation through various nodes.
- **WEKA**[4], a collection of machine learning algorithms for data mining tasks [HFH+09].
- **MOA**[5], an open source framework for data stream mining, with its roots in WEKA [BHKP10].

As an example of Scala's applicability to the problem at hands, in "Programming in Scala", by Odersky *et al.* [OSV08], it is explained the development of a simulation framework, based on the work of Bernardinello and Bianchi [BB12]. Here, the power of Scala's scheme for parallelization with Actors, and reactive-style of programming are enhanced, both being key aspects for modeling complex CPS.

Altogether, Scala encompasses an environment that has been growing both in size and adoption, as is the example of Spark's implementation migration to Scala [ZCF+10]. Due to these reasons, it is the tool of choice for implementing this project.

### 3.1.2 Python Machine Learning Stack

Python is a general-purpose scripting language that has been establishing itself as a language of choice in the data science field, together with R. Even though it is not as suited for large-scale development as other languages, Python's environment is very diverse and useful for rapid development, encompassing libraries like Numpy, Scipi, Science-kit Learn and Pandas.

Due to its visibility and adoption, it is becoming also a relevant tool for deep learning development. Theano is one such example [BBB+10], being already widely used for these demanding tasks. More recently, TensorFlow was made available by Google [AAB+15], reinforcing the relevance of this field in the (near) future of machine learning.

### 3.1.3 System's API Discussion

An essential component of the system at hands is the real-time reporting to client applications. This reporting has to be done in a streaming manner, so that it answers the clients' needs.

HTTP does not enable real-time communications, due to being half-duplex, and as such does not suit the needs. WebSockets [LG10] was established as a middleware layer to answer the current needs for bi-directional, full duplex, persisting communications, using reliable, TCP channels.

[1] Java Virtual Machine
[2] Java Agent Development Environment
[3] Foundation for Intelligent Physical Agents
[4] Waikato Environment for Knowledge Analysis
[5] Massive Online Analysis

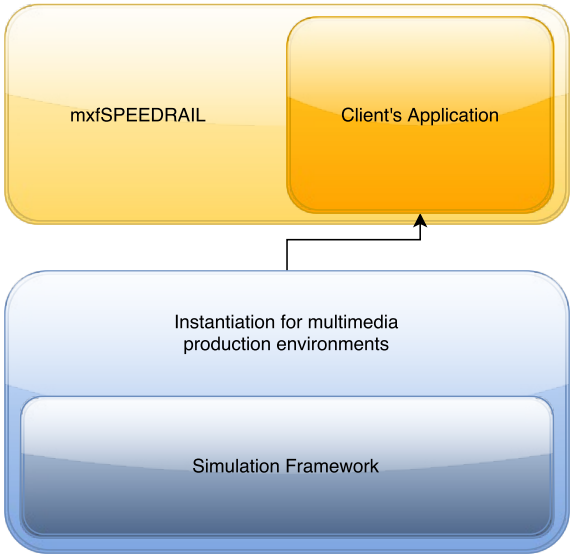


Figure 3.1: Architecture of the system for simulation of behaviors in complex systems in multimedia environments

## 3.2 Methodological Approach

The system to be developed aggregates three main layers:

- the framework for simulation of behaviors in complex systems;
- the instantiation of the framework in the multimedia production environment;
- a client application, that connects to the previous layer through an API, and integrates into the existing mxfSPEEDRAIL solution.

The disposition of these layers is showcased in context in Figure 3.1. Next, these components are further described.

### 3.2.1 Simulation Framework

This underlying framework is the most complex component of the overall system. It divides itself into five different modules: data stream processor, simulation engine, learning module, analysis module, and client's interface. The organization is shown in Figure 3.2

The system is fed from the stream of information about events happening in the physical world. The data stream processor is responsible to parse this data according to a set of rules defined by the framework, and others defined by the framework's concrete instantiation. This information is then fed to the simulation module.

The simulation engine aggregates a set of agents, that represent and are bound to the external entities, and manages their interactions. It uses the information provided by the data stream processor both to feed the learning process of the agents, as well as to evaluate its own progress. Additionally, it provides support for changes in its running clock, for past or future points in time.

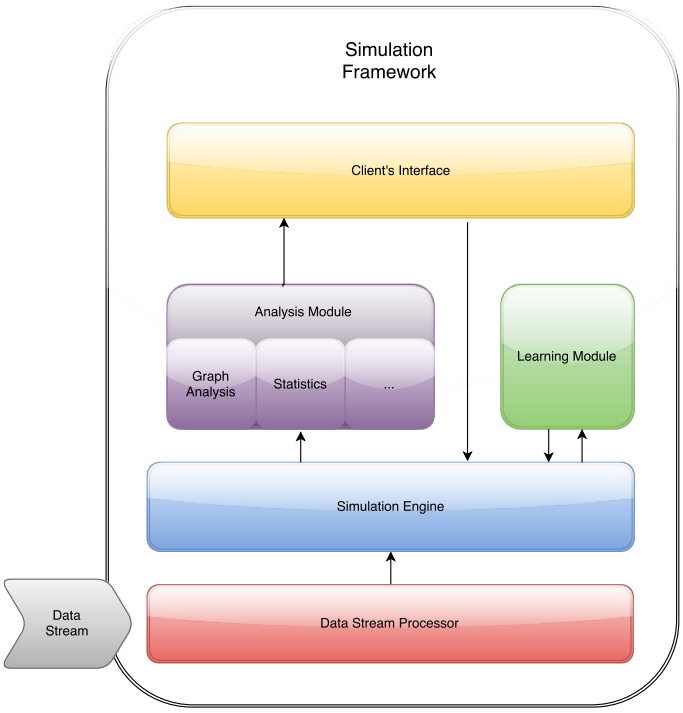


Figure 3.2: Architecture of the simulation framework

At its core, the distributed framework discussed in Section 3.1.1 is responsible of the coordination of the actions performed.

The learning module coordinates the agents' behavioral cloning process. The agents' behavioral models evolve according to the data stream that arrives. These models essentially consist on a mapping from sensors criteria to actions, which can be nested within themselves. These actions may involve triggering sensors in other agents. It also supports requests to override agents' behavior with certain rules. In the context of this thesis, the objectives are to test the usage of LSTMN (adapted to behavior trees) and CVFDT, both discussed in Section 2.5.

The analysis module continuously gathers information from simulation engine, providing descriptive and inferential knowledge about what happens. In it, there is an extensible environment of smaller modules, *e.g.* graph analysis and statistics.

Finally, the client's interface gathers data from the simulation engine and analysis module, and provides it to the outside world through an API. This communication is established using WebSockets, as discussed in Section 3.1.3. It also redirects incoming requests to the simulation engine, being them to edit agents' behavior, or to simulate different clock times.

### 3.2.2 Instantiation of the Framework

This component binds the framework to the context of a given media production environment. Its responsibilities are:

- define a set of sensors and actions for agents;

- implement additional modules in the analysis module;
- bind the data stream processor to a given stream.

### 3.2.3 Client's Application

The client's application essentially retrieves information from the framework's interface, namely the one that originates in the analysis module. This information is then presented to the user, which in the context of media production environments is performed through an web UI[6]. It also sends requests to override agents' behavior and to simulate different clock times.

## 3.3 Validation and Evaluation

The achieved design's flexibility aims to meet with the following validation requirements:

- ability to simulate present and future points in time;
- ability to simulate different hypothesis;
- ability to describe events a given point in simulation time;
- ability to predict future events at a given point in simulation time.

Concrete use cases originate from the merge of different items of the previous list (*i.e.* hypothesis take place at a given present or future point in time, and at each time we want to be able to describe and predict events).

Due to time constraints, evaluation will take place against simulated and real broadcasting clients' databases, which is initially the main target for the system.

In order to evaluate the system's performance, as explained in Section 2.6, the main evaluation parameters are usability and decision improvement. These are fundamentally subjective experiences, and are, thus, subject to careful planning. Nonetheless, the framework that underlies the complete system is to be evaluated according to two parameters:

- error in estimating future points in time;
- error estimating hypothesized scenarios.

## 3.4 Thesis Planning

The tasks planning to accomplish this thesis' objectives is showcased in Figure 3.3 as a Gantt Chart [Wil03]. The remainder actions mostly concern about the implementation, validation and evaluation of the system, culminating with the thesis' final documentation.

[6] User Interface

| | Nov | Dec | Jan | Feb | Mar | Apr | May | Jun |
|---|---|---|---|---|---|---|---|---|
| Tasks | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **Dissertation Preparation** | | | | | | | | |
| Study of Media Production Environments | | | | | | | | |
| Experimentation with mxfSPEEDRAIL | | | | | | | | |
| Literature Review | | | | | | | | |
| Monograph Writing | | | | | | | | |
| **Dissertation** | | | | | | | | |
| Personal Page Development | | | | | | | | |
| Development of the system's supporting structure | | | | | | | | |
| Iterative development of the simulation framework | | | | | | | | |
| Iterative development of the framework's instantiation and client's application | | | | | | | | |
| System's validation | | | | | | | | |
| System's evaluation | | | | | | | | |
| Scientific article writting | | | | | | | | |
| Dissertation Writting | | | | | | | | |

Figure 3.3: Gantt Chart for the Thesis

# Chapter 4

# Conclusions and Future Work

In this document, it was hypothesized that a system capable of inducing behaviors and providing simulation and analysis support in complex systems, not only solves some of the problems faced in media production environments, but can be extrapolated to other fields, where behaviors of entities and interaction between those entities are key aspects.

## 4.1 Discussion

The utility of the system discussed in this document is very prominent: with the advent of IoT and growing scale of web-based services, complex interactions have to be properly analyzed to extract concrete knowledge, before any real decision takes place.

This is a difficult problem to tackle, and there is no definitive solution. By combining the methodologies discussed in this thesis it is expected to achieve a first iteration of a possible solution. It is expected that this system will improve over the very own insight it will provide, in an iterative manner.

## 4.2 Future Work

At this stage it is already possible to devise a few improvements over the system that will be developed. Two of which would be the integration of carefully crafted curriculum learning (*i.e.* to start with a simple version of the problem, and progressively increase its complexity) and adaptive learning (*i.e.* incorporate human decision in training when the model has low confidence). These techniques focus on improving the learning process, and achieving better results in the end.

# References


[AAB+15] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow : Large-scale machine learning on heterogeneous distributed systems. 2015.

[Ass10] Modelica Association. Modelica ® - a unified object-oriented language for physical systems modeling language specification. *Interface*, 5(6):250, 2010.

[AWC+03] Charu C. Aggarwal, T. J. Watson, Resch Ctr, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. *Proceedings of the 29th international conference on Very large data bases*, pages 81–92, 2003.

[BB12] Luca Bernardinello and Francesco Adalberto Bianchi. A concurrent simulator for petri nets based on the paradigm of actors of hewitt. *CEUR Workshop Proceedings*, 851:217–221, 2012.

[BBB+10] James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math compiler in python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, (Scipy):1–7, 2010.

[BCG07] Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. 2007.

[BHKP10] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[Bon02] Eric Bonabeau. Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl. 3):7280–7287, 2002.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a search engine, 1998.

[BPG+04] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

REFERENCES


[CH07] Robert Colvin and Ian J Hayes. A semantics for behavior trees. (April), 2007.

[CO08] António Castro and Eugénio Oliveira. The rationale behind the development of an airline operations control centre using gaia-based methodology. *International Journal of Agent-Oriented Software Engineering*, 2(3):350, 2008.

[DBS06] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[DH00] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. *Proceedings of The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[DOM15] DOMO. Data never sleeps 3.0, 2015.

[DSM15] Berkeley J Dietvorst, Joseph P Simmons, and Cade Massey. Algorithm aversion: People erroneously avoid algorithms after seeing them err. *Journal of Experimental Psychology: General*, 144(1):114–126, 2015.

[Fis87] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[GDC10] Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *Proceedings - 2010 International Conference on Advances in Social Network Analysis and Mining, ASONAM 2010*, pages 176–183, 2010.

[GHJV95] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*, volume 206. 1995.

[Gow14] Stephen Gower. Netflix prize and svd. pages 1–10, 2014.

[GŽB+14] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. ... *Computing Surveys ( ...*, 46(4):1–37, 2014.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: An update. *ACM SIGKDD Explorations*, 11(1):10–18, 2009.

[HHSS97] Sepp Hochreiter, S Hochreiter, Jürgen Schmidhuber, and J Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–80, 1997.

[HSD01] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, 18:97–106, 2001.

[Hug13] Jérôme Hugues. Aadlib, a library of reusable aadl models. 2013(September), 2013.

[Jif13] He Jifeng. A clock-based framework for construction of hybrid systems. pages 32–51, 2013.

[Jou13] The Wall Street Journal. Annual u.s. cybercrime costs estimated at $100 billion, 2013.

[KPS11] Gerald Kerth, Nicolas Perony, and Frank Schweitzer. Bats are able to maintain long-term social relationships despite the high fission-fusion dynamics of their groups. *Proceedings. Biological sciences / The Royal Society*, 278(1719):2761–7, 2011.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[Lee08] Edward A. Lee. Cyber physical systems: Design challenges. page 8, 2008.

[LG10] P Lubbers and F Greco. Html5 web sockets: A quantum leap in scalability for the web. *SOA World Magazine*, (1), 2010.

[LST15] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tnenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[Mac67] J B MacQueen. Kmeans some methods for classification and analysis of multivariate observations. *5th Berkeley Symposium on Mathematical Statistics and Probability 1967*, 1(233):281–297, 1967.

[MCB+15] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. The internet of things: Mapping the value beyond the hype. *McKinsey Global Institute*, (June):144, 2015.

[MF14] Jorg P. Muller and Klaus Fischer. Application impact of multi-agent systems and technologies: A survey. *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, 9783642544:27–53, 2014.

[MIK+04] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science (New York, NY)*, 303(5663):1538–1542, 2004.

[MSOIK02] R Milo, S Shen-Orr, S Itzkovitz, and N Kashtan. Network motif: Simple building blocks of complex networks. *Science*, 824(2002):298., 2002.

[MW13] Nathan Marz and James Warren. Big data - principles and best practices of scalable realtime data systems. . . . . *Harvard Bus Rev*, 37:1 – 303, 2013.

[Omg98] Omg. Unified modeling language (uml). *InformatikSpektrum*, 21(2):89–90, 1998.

[OSV08] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. 2 edition, 2008.

[Par13] Shameem Ahamed Puthiya Parambath. Matrix factorization methods for recommender systems. 2013.

[PBV07] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui92] J Ross Quinlan. *C4.5: Programs for Machine Learning*, volume 1. 1992.

[RS10] Pedro Ribeiro and Fernando Silva. G-tries: an efficient data structure for discovering network motifs. *Proceedings of the 2010 ACM Symposium on . . .*, pages 1559–1566, 2010.

[Sch09] Wladimir Schamai. Modelica modeling language (modelicaml): A uml profile for modelica. *Last Accessed*, pages 1–49, 2009.

[SGJ+11] Alexander Shoulson, Francisco Garcia, Matthew Jones, Robert Mead, and Norman I. Badler. Parameterizing behavior trees. *Proceedings of the 4th International Conference on Motion in Games (MIG '11)*, pages 144–155, 2011.

[SGLW08] Lui Sha, Sathish Gopalakrishnan, Xue Liu, and Qixin Wang. Cyber-physical systems: A new frontier. *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 1–9, 2008.

[SHG+15] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, and Dan Dennison. Hidden technical debt in machine learning systems. *Nips*, pages 2494–2502, 2015.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Shn96] Ben Shneiderman. The eyes have it:a task by data type taxonomy for information visualizations. *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[SMG14] Didier Sornette, Thomas Maillart, and Giacomo Ghezzi. How much is the whole really more than the sum of its parts? 1+1 = 2.5: Superlinear productivity in collective group actions. *PLoS ONE*, 9(8):1–29, 2014.

[Sor07] D Sornette. Probability distributions in complex systems. *Arxiv preprint arXiv07072194*, physics.da:27, 2007.

[Sor08] D. Sornette. Nurturing breakthroughs: Lessons from complexity theory. *Journal of Economic Interaction and Coordination*, 3(2):165–181, 2008.

[TF07] John Tromp and Gunnar Farnebäck. Combinatorics of go. *Computers and Games: 5th International Conference*, pages 84–99, 2007.

[THBS15] P. Taylor, J. N. Hobbs, J. Burroni, and H. T. Siegelmann. The global landscape of cognition: hierarchical aggregation as an organizational principle of human cortical networks and functions. *Scientific Reports*, 5(November):18112, 2015.

[Wil03] James M. Wilson. Gantt charts: A centenary appreciation. In *European Journal of Operational Research*, volume 149, pages 430–437, 2003.

[WJK00] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[ZCF+10] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark : Cluster computing with working sets. *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, page 10, 2010.

[Zha14] Lichen Zhang. A framework to model big data driven complex cyber physical control systems. *Automation, International Conferenceon Science, Computer Guangzhou, Technology*, pages 12–13, 2014.