Combining the Internet of things, complex event processing, and time series classification for a proactive business process management.

Raef Mousheimish

université PARIS-SACLAY

NNT : 2017SACLV073

Thèse de doctorat

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

Combinaison de l'Internet des objets, du traitement d'évènements complexes et de la classification de séries temporelles pour une gestion proactive de processus métier

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'université de Versailles Saint-Quentin en Yvelines

École doctorale n°580 sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : informatique

Thèse présentée et soutenue à Versailles, le 27 octobre 2017, par

M. Raef Mousheimish

Composition du Jury :

Mme. Amel Bouzeghoub Professeur, Telecom SudParis – Paris Saclay	Président
M. Florent Masseglia Chercheur HDR, INRIA	Rapporteur
M. Michael Papazoglou Professeur, Tilburg University	Rapporteur
M. Mohamed Dbouk Professeur, Université Libanaise	Examinateur
M. Mohand-Said Hacid Professeur, Université Claude Bernard Lyon 1	Examinateur
Mme. Karine Zeitouni Professeur, Université de Versailles Saint-Quentin – Paris Saclay	Directrice de thèse
M. Yehia Taher MCF, Université de Versailles Saint-Quentin – Paris Saclay	Co-Encadrant de thèse
M. Michel Dubus Ingénieur, C2RMF Paris	Invité

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Combinaison de l'Internet des objets, du traitement d'évènements complexes et de la classification de séries temporelles pour une gestion proactive de processus métier

Mots clés : Traitement des événements complexes, Fouille de séries temporelles, Classification précoce, Séries temporelles multivariées, Gestion des processus métiers

Résumé : L'internet des objets est au cœur des processus industriels intelligents grâce à la capacité de détection d'évènements à partir de données de capteurs. Cependant, beaucoup reste à faire pour tirer le meilleur parti de cette technologie récente et la faire passer à l'échelle. Cette thèse vise à combler le gap entre les flux massifs de données collectées par les capteurs et leur exploitation effective dans la gestion des processus métier. Elle propose une approche globale qui combine le traitement de flux de données, l'apprentissage supervisé et/ou l'utilisation de règles sur des évènements complexes permettant de prédire (et donc éviter) des évènements indésirables, et enfin la gestion des processus métier étendue par ces règles complexes.

Les contributions scientifiques de cette thèse se situent dans différents domaines : les processus métiers plus intelligents et dynamiques; le traitement d'évènements complexes automatisé par l'apprentissage de règles; et enfin et surtout, dans le domaine de la fouille de données de séries temporelles multivariées par la prédiction précoce de risques. L'application cible de cette thèse est le transport instrumenté d'œuvres d'art.

Title : Combining the Internet of things, complex event processing, and time series classification for a proactive business process management

Keywords : Complex Event Processing, Time Series Data Mining, Rule Discovery, Early Classification, Multivariate Time Series, Business Process Management

Abstract: Internet of things is at the core of smart industrial processes thanks to its capacity of event detection from data conveyed by sensors. However, much remains to be done to make the most out of this recent technology and make it scale. This thesis aims at filling the gap between the massive data flow collected by sensors and their effective exploitation in business process management. It proposes a global approach, which combines stream data processing, supervised learning and/or use of complex event processing rules allowing to predict (and thereby avoid) undesirable events, and finally business process management extended to these complex rules. The scientific contributions of this thesis lie in several topics: making the business process more intelligent and more dynamic; automation of complex event processing by learning the rules; and last and not least, in datamining for multivariate time series by early prediction of risks. The target application of this thesis is the instrumented transportation of artworks.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

ACKNOWLEDGEMENTS

Firstly I would like to express my sincere gratitude to my advisor Prof. Karine Zeitouni for the continuous support of my PhD. work, for her patience, dedication, and immense knowledge. Surveying and carefully going through related materials was essential to advance in my PhD., but it would have been impossible to finish if not for Prof. Karine's punctual guidance, accurate instructions, and tremendous contributions. Given her sheer knowledge, solid research experience, and great motivation, I could not imagine having smoother years or a better mentor for my Ph.D study.

Secondly, my deepest thankfulness goes to my co-advisor Dr. Yehia Taher for his indescribable help, not just on my PhD. work, but throughout many stages of my life. I thank him first as his Master and PhD. student for being a great mentor, always knowing how to best orient and add values to the work. He introduced me to the world of research in France, and if it is not for the plentiful meetings that we have had, then I would not be concluding my thesis now. I will always be honored for being his first PhD. student. I also thank him a second time, as a friend, for his kindness, assistance, and personal advices. I frequently sought his counsel because there is always something to learn from him.

Besides my advisors, I would like to thank the rest of my thesis committee. Specifically, I value the efforts of Prof. Michael Papazoglou and Dr. Florent Masseglia for reviewing my dissertation. I truly appreciate their constructive feedback and the time that they spare given their busy schedules. I also deliver my sincere gratitude to Prof. Mohamed Dbouk, Prof. Mohand-Said Hacid, and Prof. Amel Bouzeghoub for examining my work. At last, I greatly acknowledge M. Michel Dubus who was honorably invited to participate in my committee. Even though, M. Michel Dubus was not officially my supervisor, but our partner, still I learned plenty from our numerous meetings and from his deductive advices. Therefore, I thank him for the time that he dedicated for me throughout the three years of my PhD. In this context, special thanks go to the association PATRIMA who funded my thesis and putted me in contact with M. Michel Dubus in the first place.

My acknowledgement would be flawed without including several other persons that had huge impacts on my life. From the university of Versailles, I want to thank all the members of DAVID laboratory, and in particular the members of ADAM team: Prof. Mokrane Bouzghoub, Dr. Béatrice Finance, Dr. Laurent Yeh, Dr. Zoubida Kedad, Dr. Stéphane Lopez, and Dr. Nicoleta Preda. I had the chance to meet them and to get their feedback on several occasions. My gratitude is doubled for Dr. Béatrice Finance, as she co-supervised my Master project and taught me

numerous research lessons along the way. Furthermore, I thank the secretary Mme. Chantal Ducoin for her kind help and assistance. From the Lebanese university, I want to re-mention Prof. Mohamed Dbouk who was my Master director. He professionally directed me at my studies, and he was the first to teach me about research methodologies, therefore and after all, I am what I am thanks to him. Other than that, his company was enjoyable and his conversations were always interesting in life, thus I am really lucky to get to know such a person. I also convey my honest gratitude to Dr. Ihab Sbeity who taught me several pleasant classes and whom I knew and admired long before meeting him at the university. He was a great teacher in life as well as in classrooms, and he was the first to believe in my capabilities, to encourage me, and to push me in the right directions.

Even though I am studying in France and away from family, but living here is a breeze for me, and for that I owe several persons. I want to thank Dr. Rafiqul Haque for his amazing company and friendship, he is one of few guys that got tremendous amounts of information, and brainstorming with him was always a pleasure. Furthermore, I could not survive the life here if not for the true support of my deer friends, who were also my lab mates, and even my neighbors. I almost shared everyday with them, inside and outside the office. Therefore, I want to deliver my purest acknowledgment to Mohammad AlShaer, Mohammad Moussawi, Ali Masri, Mohammad Rihany, Dai-Hai Ton-That, Hanane Ouksili, Maria Koutraki, Ticiana Linhares, Mayssam Rihan and others, for being true friends and supporters all the way.

My deepest gratitude goes to my precious family, who tried to support me and encourage me in any possible mean. Even though, I managed to live normally and finish my studies, their presence would have magically changed a lot of things to the best. I am entirely grateful to my father Abed and my mother Widad, not to mention my brothers, for their tremendous moral and emotional support. I am also grateful to my second family, my in-laws Adnan and Wafaa, for their genuine encouragement. Finally, my very special thank goes to my lovely wife Fatima for her pure and unconditional kindness. I happily attribute every success in my life to her.

Raef Mousheimish

TABLE OF CONTENTS

LIST OF TABLES

TABLE **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

> Change is the law of life. And those who look only to the past or present are certain to miss the future.
>
> *John F. Kennedy*

Survival of the fittest, as Darwin beautifully puts it, is a precious lesson that life wants us to learn. With every change, and in order to survive, one needs to adapt. A fact that holds since the dawn of the first species.

The technological world, where change is plainly the dominant theme, is no exception. The only difference is that extinction happens at a faster pace. Technologies that were trends a couple of years ago could be on the verge of disappearing, should they abstain to cope with the current changes.

Nowadays, data and events are being continuously and enormously collected in different domains, all objects are becoming more and more connected, and things and interlinked to form a gigantic network called the Internet of Things (IoT). Digital representations that were unimaginable before, are now real and waiting to be exploited. Harnessing such representative and finely descriptive data has cultivated in the fact that prediction and proactivity are becoming more and more requirements rather than options. Tremendous efforts are spent on technologies that promote proactivity and artificial intelligence such as data analysis and data mining. In fact, it is unsurprising how the science of data is reaching to every other domain. Technologies that are striving to survive are obliged to integrate some data science techniques within their cores. On the other hand, odds are against technologies that hold massive advantages and opportunities, but are sticking to their reactive natures.

Complex Event Processing (CEP) by design and definition **reacts** to situations, i.e., CEP engines are oriented to respond to events that **have already happened**. The fact that puts this technology on the weak spot, which may eventually lag behind when prediction becomes an obligation. Logically, other technologies that count on CEP, such as Business Process Management (BPM), are indeed occupant of the same slow train as well. They still did not fit with the current change by incorporating data mining and prediction within their essence, and at the end, survival is only for the fittest.

1.1 Motivation

Thanks to its countless and undeniable advantages, prediction is an enticing term that could instantly capture anyone's attention. In any domain or business that one could think of, prediction can always comes in handy. If the predicted situation is negative, then it could be avoided, and if avoidance is not an option, then risk could be efficiently mitigated and reduced. Else if it is positive, then involved parties could take advantage of it, and maximize their gains.

This has always been the case, and prediction has hold an unbounded trait of attractiveness since the beginning of humanity, where it was referred to as prophecies or visions. However the big difference is that it is no more mere fantasies, and it now has scientific grounds. In fact, every knowledgeable individual could emphasize that it is being used in many domains throughout our everyday life, and that it will propagate to all other domains —that are willing to keep up— in the near future. We argue that the main ingredient that brings this crystal ball into reality nowadays, is **data**.

With every passing year, sensors are becoming cheaper and smaller, and they are continuously being embedded into lifeless objects, such as cars, fridges, and buildings, to count a few. These advances in the hardware world, combined with the omnipresence of fast broadband connection, in addition to the doubtless improvements in data handling frameworks, are serving as a prologue to an era full of intelligent objects. It is no secret that the names of our "once-were-dumb" objects, one after the other, are being prefixed with the word smart, like smart phones, smart houses, smart TVs, etc.

These ever-evolving worlds of hardware, network, and software, are giving momentum to this phenomenon to the extent that it obliged scientists to grant it a name: **The Internet of Things (IoT)**. Thanks to the ubiquity of IoT devices, data is generated in a pace that was unwitnessed before. All kinds of events, that were otherwise unattainable, are now available for acquisition, processing, and analysis, the thing that opens new perspectives for scientific-based predictions. With that being said, harnessing this abundance of data merely for reactive purposes is no longer sufficient. Both academia and industry are striving to maximally exploit this massive amount of real life events in order to achieve one primary goal: **proactivity**.

In principle, prediction methods are already laid out by fields such as predictive analytic and data mining, but IoT is now offering data as the best-ever inputs for these methods in order to

efficiently exercise prediction in real world applications. This forces technologies to proceed in this direction, and incorporate data mining techniques within their cores. Other technologies that stick to their reactive nature are being slowly left behind. Therefore we are witnessing a change, or a paradigm shift from reactivity to proactivity, which is compelling us to alter the way we are processing data.

As briefly mentioned before, two main promising technologies are being affected by this change, **complex event processing** and **business process management**. Both technologies have gained a substantial amount of popularity after their inceptions, however we doubt that this popularity will hold given the paradigm shift we have just discussed.

CEP is currently sticking to its reactive quality, as achieving proactivity in this domain has proven to be difficult in practice [55, 53]. On the other hand, BPM is activity-based, coarse-grained, and it counts on CEP to achieve a level of event-based processing, even though the integration of these two fields is not evident [76, 78]. In summary, and putting aside some timid attempts, data mining and the availability of IoT data are not yet harnessed for CEP & BPM. Subsequently, it is clear that these two technologies are fastened to the reactive side of the paradigm, and indeed those who cannot adapt to the change will not survive it.

1.2 Aim

In recent years, a noticeable amount of enterprises have shifted toward BPM in order to better understand, manage, organize, and visualize their business processes. This extensive adoption in different fields has devised processes of various types. To account for all the diversity, the active BPMN[1] standard, and in its latest version, has introduced many forms of tasks and events. For instance, one could differentiate between automated and monitorable tasks. The former is instantaneous, executed in a controllable environment, and could be monitored by software engines. The latter refers to tasks that are executed in the real world, they probably span over a long interval of time, and they may be affected by their dynamic surrounding. Therefore, it makes sense to monitor the behavior of monitorable tasks by integrating different sensors, IoT devices, and stream processing techniques.

However, the main problem with current BPM systems is that monitoring and analyzing the behavior of monitorable tasks are beyond their reach [51, 14]. In other words, BPM allows to differentiate between these tasks on the design level, but when it comes to execution, there isn't much difference on how they are being handled. Typically, start and end times are recorded, while with the potential of IoT, much more could be achieved. Therefore one of the main goals of this work is to promote the management of processes, and exploit the availability of data to better manage monitorable tasks.

The flaw of activity-based BPM engines has been extensively targeted by researchers [14, 76, 78, 26, 25, 120, 77, 4, 102, 127, 187] that sought the help of CEP. The aforementioned approaches,

[1] http://www.bpmn.org/

and through the CEP technology, have reached a level of fine-grained processing by connecting real-life events with business process models. This practice helped to create more visibility for monitorable tasks, and offered a mechanism to monitor and react to situations whenever they happen. However, the integration of BPM and CEP was never evident. BPM users who seek to monitor their tasks will quickly find themselves in need to master yet another complex technology to work out the integration, and to state how situations of interest will be detected in CEP jargons. This hints at another dilemma as CEP is a reactive technology, and by employing it, situations will only be detected rather than predicted. Therefore, the jump into the proactive side of the paradigm will hardly occur.

The previous section carries us to plainly put the other main goals of this work. On one part, we aim at equipping CEP with some data mining weapons at its core, in order to activate the hidden **predictive** potentials of this technology. Moreover, and since we are targeting the essence of CEP, then we require that our solution will be easy-to-use and generic. Mainly, filling a gap between data mining and complex event processing is a first step to reach a solution. On the other part, since we emphasized on the easy-to-use requirement, we intend for our predictive CEP technology to be integrated seamlessly within the BPM world. Therefore, we aim at sparing BPM users the hassle of learning another complex language and keep them working within their comfort zone.

In summary, from an abstract point of view, the major objectives are to adapt the CEP and the BPM technologies to the current change, and to provide the essential cornerstones to flip them into the proactive side.

1.3 Scope

The research and contributions of this work stretches over a wide range of different domains. It reaches to the data mining field on the low level, going through complex event processing, and into business process management on the application level.

From a data mining perspective, we are interested in predictive analytic and specifically over classified time series. To this end, our proposal exploits a relatively new primitive for time series data mining, which is called **shapelet** [207]. In supervised learning, each time series within the available training set is annotated with a given class, shapelets are basically discriminative subsequences within the time series that are likely correlated to these classes. In other words, the occurrence of a given shapelet could be enough to tell which class the time series in question belongs to. This practice supports real-time stream classification, and saves classification time as it is not obligatory to scan the whole time series (like KNN classifiers do [205]) in order to detect its class. On the other hand, depending on the size of the training set and the lengths of shapelets, learning time could be expensive. Shapelets must be accurate, so they need to be long enough to capture the features of time series, but at the same time to promote earliness and prediction, the shorter the shapelet the better.

On the other hand, we repeatedly highlighted the reactive nature of CEP, and we argue that this is the case because of two facts: the only inference mechanism in this domain is **CEP rules**, plus these rules must be specified manually by human experts. Naturally, it is extremely hard, even for experts, to manually write rules that **predict** situations. In contrast, this is why domains such as data mining and machine learning [19, 72] were created in the first place, i.e., by using data mining techniques, machines could learn (predictive) patterns from histories that humans are not able to find themselves. This clearly shows a gap between the two fields, data mining and CEP, and it also proves that filling this gap will potentially be very advantageous.

On the application level, the BPM technology is the main focus. Specifically, the monitorable tasks within business processes. The environment where these tasks are executed is highly dynamic, and contextual events may break out stochastically and not as expected. Therefore prediction and context-aware management are a must at this point.

From a global point of view, the shapelet-based data mining part will pull deep predictive CEP capabilities to the surface. Afterward the outcome of this integration will be proven beneficial for BPM applications.

1.4 Problem Definition

Throughout the previous sections, we have looked at main challenges from different perspectives. By motivating, then directing, and finally mapping the scope of our research, we paved a way to better define the problems that we are targeting.

At this point, it should be apparent that IoT is generating an abundance of data, and the CEP technology is still away from exploiting these data for predictive purposes. From another perspective, as long as descriptive and detailed data are being generated, data mining methods will excel in their results, after all these methods thrive on data. Therefore, we argue that it is the best time to think of a bridge that can connect some data mining techniques with the CEP core. Should this connection takes place in a seamless manner, solutions for one of the most intractable problems will start to rise. This problem is: **How to better exploit the abundance of data in the CEP world? Or how to make best use of them in order to introduce predictive CEP capabilities?**

The aforementioned questions could be linked to a more fundamental problem, i.e., **how to adapt the CEP technology to the paradigm shift that is occurring?** As we already discussed, with the ever-developing hardware (to sense very exhaustive data), software (to better handle these data), and connections (to easily transmit them), prediction will become a requirement rather than an option. Therefore, asking this question, in an explicit way, does make sense.

Essentially, BPM extensively counts on CEP to hide its inability while managing long-running real-world tasks (i.e., what we are calling monitorable tasks). Therefore the problems that were highlighted before, and by a cascading effect, are totally appropriate in the BPM world. Still,

other problems emerge at this level as well. On one hand, the BPM/CEP integration is known to be complex, therefore **how to turn this complexity into straightforwardness?** On the other hand, BPM systems tend to favor models over instances, even though monitorable tasks from different instances, may be executed in distinct environments with unlike contexts. Therefore, a rising problem is **how to manage instances in a context-aware manner? And how (predictive) CEP could come handy in this situation?**

The merging of all aforementioned issues will give rise to one global problem: **how to devise a complete, a generic, and a dedicated approach that will address the questions, going from low level data mining techniques, into an intermediate level of complex event processing, and then concluding with business process management on the application level?**

1.5 Research Questions

In order to go in more details and fetch solutions to the problems that we stated in the prior section, some research questions must be asked on a higher level of concreteness. These questions may be overlapping and answering one may answer the other, however all kind of questions that may be addressed by our work are laid out explicitly.

- *Q01: What is the-state-of-the-art on this matter? And how to benefit from them?*
  At the beginning, and as the law of research dictates it, the literature needs to be inspected. Thus, its necessary to answer this question. As clearly shown in the scope section, we are touching three separate domains, and this leads us to ask the subsequent questions.
- *Q02: Why shapelets are chosen as the data mining technique to be adopted for CEP?*
  This may be a starting question to address from a data mining perspective.
- *Q03: What are the latest advancements on shapelets?*
- *Q04: Do these state-of-the-art approaches fit to be integrated in the CEP core?*
- *Q05: What extensions into the multivariate world exist?*
  In principle, shapelets are proposed to work on univariate time series.
- *Q06: Could these extensions take advantage of the expressiveness of CEP, should we decided to adopt them?*
- *Q07: How to go from manually specifying CEP rules into automatically defining them?*
  On the CEP level, this is an important concern to address.
- *Q08: What work existed before on the integration of data mining and CEP?*
- *Q09: Is there some predictive approaches on this matter?*
- *Q10: How shapelet-based solutions could help CEP to adapt to the paradigm shift?*

- *Q11: What is the state-of-the-art on integrating CEP? and handling monitorable tasks?* Some questions need to be researched as well from a BPM point of view.
- *Q12: To what extent CEP capabilities are exploited in BPM systems?*
- *Q13: Is there real predictive practices when it comes to managing business process instances?*
- *Q14: Are instances (especially those who contain monitorable tasks) being managed in a context-aware fashion?*

After we have answered the state-of-the-art issues in our work, we were confronted with new research questions to address.

- *Q15: How to exploit CEP capabilities for prediction rather that reaction?*
- *Q16: How to transform shapelets into CEP rules in an automatic manner?*
- *Q17: How to go directly from pure data mining into the CEP world?*
- *Q18: How to extract shapelets with advanced knowledge from multivariate time series?* This question should be addressed in case shapelet transformation takes place. With advanced knowledge, we mean patterns that may predict and benefit from the expressiveness of CEP rules when they will be transformed, such as sequences, time windows, and correlations over different dimensions.

Connecting data mining and CEP is a major step toward the solution, but still, we need to consider some questions:

- *Q19: How to go into the BPM field in an easy way?*
- *Q20: With no expertise required in the CEP domain, how to make predictive CEP functions available to BPM engines?*
- *Q21: How to wrap the solutions in an easy-to-use and generic framework?*

1.6 Contributions

Typically, contributions will carry solutions to the unanswered research questions. In this section, and to list our contributions in every domain, we will implicitly focus on two parts. In the first part, we will analyze the domain and list shortcomings of current state-of-the-art approaches. In the second, we will provide our added-values.

1.6.1 Time Series Data Mining

In the realm of time series, different types of prediction may be distinguished. Notably **always-on forecasting** (ARIMA, IBM Cognos, etc.) that foresees values at every time step, and **rule-based prediction** (the focus of our work) that monitors the stream of events and triggers predictions only on the detection of specific shapes.

1.6.1.1 Limitations of Current Approaches

After surveying the current related research efforts on rule discovery and prediction, we argue that they have not matured enough, especially when it comes to consider predictive rules over **multivariate** time series. On one hand, a handful of approaches explicitly targeted the extraction of predictive rules but they only operate on univariate time series like in [169, 196, 179, 44]. On the other hand, some recent research initiatives [115, 65, 66, 151] exploited shapelets to this end.

In univariate settings, shapelets by themselves could be considered and implemented as rules [135]. However, when it comes to multivariate time series, things tend to be more complicated, as predictive rules should take into account complex combinations and temporal correlations among the time series dimensions. Although the aforementioned approaches have extended the shapelet discovery algorithms to operate on multivariate time series, yet they disregard such relationships between dimensional shapelets.

1.6.1.2 Our Contributions

In this work, we propose a novel two-step approach, called **USE** & **SEE**, which is built to deal with *supervised classification* problems. At the first step, the algorithm learns shapelets from multivariate time series. Afterward, it constructs accurate predictive rules that could classify **as early as possible** a given new unclassified instance. Our proposition advances from the available multivariate shapelet learning algorithms by overcoming their limitations.
Firstly, unlike the existing work, our algorithm allows the learning of predictive rules from the minimum needed combinations of shapelets. In other words, the outputted rules are not supposed to include shapelets from all dimensions. Secondly, our rule discovery process explores all possible correlations (synchronous, sequential with various time gaps) between time series dimensions. As far as we know, this is the only early prediction approach on multivariate time series that discovers such advanced knowledge.

1.6.2 Complex Event Processing

The CEP technology is applied in various domains: environmental monitoring [24], fraud detection [172], RFID-based inventory management [194], and financial analysis [46]. Many great books [118, 119, 55, 2] are totally dedicated to discuss CEP, and considerable amount of research articles [23, 172, 185, 41, 42] have targeted different aspects of the CEP world—like scalability, latency, distribution and performance. Such writings emphasize the undeniable advantages of event-driven systems in general and complex event processing in particular. However, it is no secret that all these efforts have focused on the reactive nature of CEP. Numerous scenarios and examples were discussed by authors, yet they all tend to have reactive traits, like the **detection** of high/low temperature, fraud, or traffic jams.

1.6.2.1 Limitations of Current Approaches

Proactive event processing is already mentioned in the literature, but it was always looked at as an emerging and future direction. For example, it was discussed in [55, 53] where authors stated that going from reactive to proactive is a must, because this could offer limitless advantages. However, and until now, proactive CEP remains a vision and it is still away from being realized. We argue that this continues to be the case, because the standard way to define CEP rules is by writing them **manually**. To this day, deep domain expertise allows manual correct rule specifications that **detect** situations of interest. Nonetheless, regardless of the level of experience, it remains highly challenging and tedious for human experts to manually write rules that can **predict** these situations. Consequently, we deem the ultimate fact that experts are in charge of writing rules as the main barrier for the prosperity and diffusion of CEP, especially that it holds the CEP technology from keeping up with the emerging science of data where prediction is a requirement.

To cope with this problem, our work intends to overtake the de facto approach regarding the definition of CEP rules. Therefore, we bridge a gap between time series data mining and complex event processing to automatically extract rules that are driven by data and learned from history. Few and recent approaches [122, 146, 185, 173] have also proposed this course of actions and focused on the automatic generation of rules. However they either make the unrealistic assumption of having one and only one rule that could lead to a specific situation [122], or they are very user-dependent and cannot learn complete rules [146, 185, 173]. Furthermore, no existing approach learns CEP rules for predictive purposes, rather they are focused on detecting situations.

1.6.2.2 Our Contributions

In our work, we introduce a novel data mining-based approach to tackle the problem of automatic predictive CEP rules generation. The proposed framework mainly counts on **USE & SEE** for the learning, and then **autoCEP** is in charge of automatically translating rules into the CEP jargon. More specifically, the whole approach **USE & SEE** with **autoCEP**, combines Early Classification on Time Series (ECTS) techniques with the technology of complex event processing, and the result is a complete framework that addresses a big part of the problems discussed so far.

In general, **autoCEP** carries these contributions in the domain of complex event processing: **(1)** The first approach to integrate time series pattern mining techniques within the CEP domain. Thanks to this, real-valued events are now supported, which was not the case in current state-of-the-art rule generation approaches. **(2)** It works on multivariate time series, and so it learns complex CEP rules by correlating events coming from multiple sensors. **(3)** The first approach to automatically learn **predictive** CEP rules. **(4)** The involvement of the user is optional, and so any user can now employ an out-of-the-box configured CEP engine without the requirement of being a technical expert in the domain. **(5)** The overall solution is generic and could be applied in any domain where classified (multivariate) time series exist.

1.6.3 Business Process Management

By definition, existing BPM engines manage the whole process in an activity-oriented fashion. This is very advantageous to administer the flows of procedures, keep them compliant with agreements, log, and monitor the beginning and end of tasks. However, engines give little to no knowledge about what is happening inside long-running tasks. From the viewpoint of analysts, these tasks are just black boxes that being executed, and they need to be signaled manually to the engine whenever they start and finish. Even though, most agreement violations could potentially stem from the inside of long-running tasks, such as traffic jams while performing a truck delivery [134], or a machine failure while producing goods [151]. BPM users have no possibility to predict these violations or even monitor the inside of such tasks.

1.6.3.1 Limitations of Current Approaches

As already discussed, researchers in the domain have noticed this limitation in activity-based BPM engines, and the obvious solution is to compensate by integrating Complex Event Processing (CEP) techniques. Therefore, the BPM/CEP integration is storming this research area in the past couple of years [14, 76, 78, 26, 25, 120, 77, 4, 102, 127, 187]. Even though these approaches proposed various conceptual and practical solutions to the BPM/CEP integration, and some of them even leveraged the predictive capabilities of event processing, all of them —as argued in [27] as well—have mainly focused on design-time integration. In other words, they annotate process models with all-purpose events at design-time. This practice is not able to support specific process instances with special constraints, it may disregard contextual relevant events coming from external services and sensors, and it is also not functional to perform predictions while tasks are executing.

Up to this day, no real attempt was made to integrate predictive CEP capabilities inside executing long activities. We argue that the explicit reason for this is the overwhelming complexity that this integration could yield. Designing a process and managing its instances through the help of a BPM engine are one thing, but manually writing predictive CEP rules and linking them to the instances are on a completely different level. Since, and as explained, in the CEP world rules are written manually, an easy-to-use and a generic solution is not evident.

1.6.3.2 Our Contributions

This work introduces a novel and the **easiest yet** approach to fulfill a BPM/CEP integration. Moreover, this integration is specifically designed to predict in-activities violations, and therefore helping to realize a proactive business process management. Next to its predictive capabilities, our approach is also generic, i.e., it could be employed easily in different application domains. Specifically, we propose the notion of **contextual templates**, in addition to a conceptual framework called **the Butterfly**, to tackle context-dependent monitorable tasks, and then we link to **autoCEP** that will easily provide real-time predictions to BPM systems.

1.6.4 The Overall Picture

As detailed in the previous three subsections, we are proposing a complete and complementary work going from data mining into business process management. In every domain, we tried to solve problems by tracing them to their roots. Starting with BPM, we saw that it is complex to integrate CEP because this integration needs to be done manually. Moreover, CEP rules are to be specified by humans, the thing that makes it even harder to use for predictive purposes. Subsequently, we turned our eyes into the CEP world, and we found that rules indeed need to be specified by users and they are in no manner driven by data. This is where we sought the help of data mining, and we analyzed the gap that exists between the two fields. Therefore, we continued into the data mining domain, and searched for proposals that could bridge the gap, however no solution was easily compatible and applicable. Finally, at this point we proposed **USE & SEE** as new pure data mining algorithms. Then the puzzle pieces started to fit together. **AutoCEP**, which is a novel CEP rule generation approach, counts completely on **USE & SEE**. At the end, the BPM solution is made possible because of **autoCEP**. So in our chain of contributions, each proposition, whenever feasible, counts on the previous and supports the next. Figure 1.1 illustrates it.

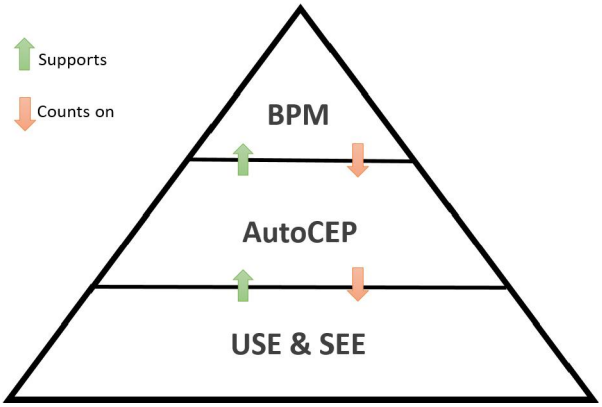

Figure 1.1: The Big Picture

It is worth mentioning that even though our propositions are complementary, but as previously discussed, each has its contributions within its domain, and of course they could be used without the framework that they support (fig. 1.1). For example, **USE & SEE** could be employed for pure data mining tasks, and equally, it is not mandatory for **autoCEP** to be used only in BPM applications.

1.7 Structure of the Dissertation

Chapter 1 already introduced our work and approached different subjects with various level of details. At this point, the motivation behind our research, our aim, and our contributions should be clear. The subsequent chapters convey the following purposes:

- **Chapter 2**: This chapter gives more insight on related work. It first discusses the overall domains that we touch in this dissertation, the different techniques that exist in these domains, and therefore it helps to position the specific parts that we are dealing with. It later presents the specifics of all appropriate state-of-the-art approaches.
- **Chapter 3**: This section includes an exhaustive list of terms and definitions that are going to be used throughout the dissertation. This chapter could be read at any point, and it should be referenced several times from subsequent chapters.
- **Chapter 4**: The business process management part is the first to be discussed in this chapter. Specifically, scenarios are included, contextual templates are defined, and the application face of our work is shown.
- **Chapter 5**: This chapter will provide all necessary information on the data mining part. The two algorithms USE & SEE will be explained, and all technical details will be included.
- **Chapter 6**: AutoCEP is going to be discussed in this chapter, where the mechanism of how data mining patterns could be algorithmically translated into CEP rules will be exposed.
- **Chapter 7**: Different tests, evaluations, and comparisons to stress the good and weak points of our approach are presented in this chapter.
- **Chapter 8**: Finally, the work is concluded in this chapter, the contributions will be revisited, and we will shed some light on future and potential perspectives.

For consistency purposes, we do recommend that the chapters be read in the specified order. However and as Figure 1.2 shows, chapter 3 could be read at any point, and it could be referenced from any other chapter.

Aside from the main flow of chapters, this dissertation also contains four additional appendices. These appendices constitute extra information, they could be examined at anytime, and they are not necessary for the main chapters.

- **Appendix A**: This part presents with fine details the early published version of **autoCEP** that deals with univariate time series. It also contains all specifics about learning shapelets in univariate environments, in addition to all experiments and results achieved.
- **Appendix B**: This appendix contains the motivation and the overall story behind our research from the very beginning where the thesis started. Specifically, it includes the research methodology and all different steps that were followed to produce this work.
- **Appendix C**: This appendix exhaustively lists all publications till the date of writing this dissertation. Submissions that are still being reviewed are not included, and only the publications in international journal and conferences are listed.
- **Appendix D**: For clarity purposes, this appendix contains the list of all acronyms used throughout the dissertation. It could be visited at any point should readers have any ambiguity about a specific abbreviation.

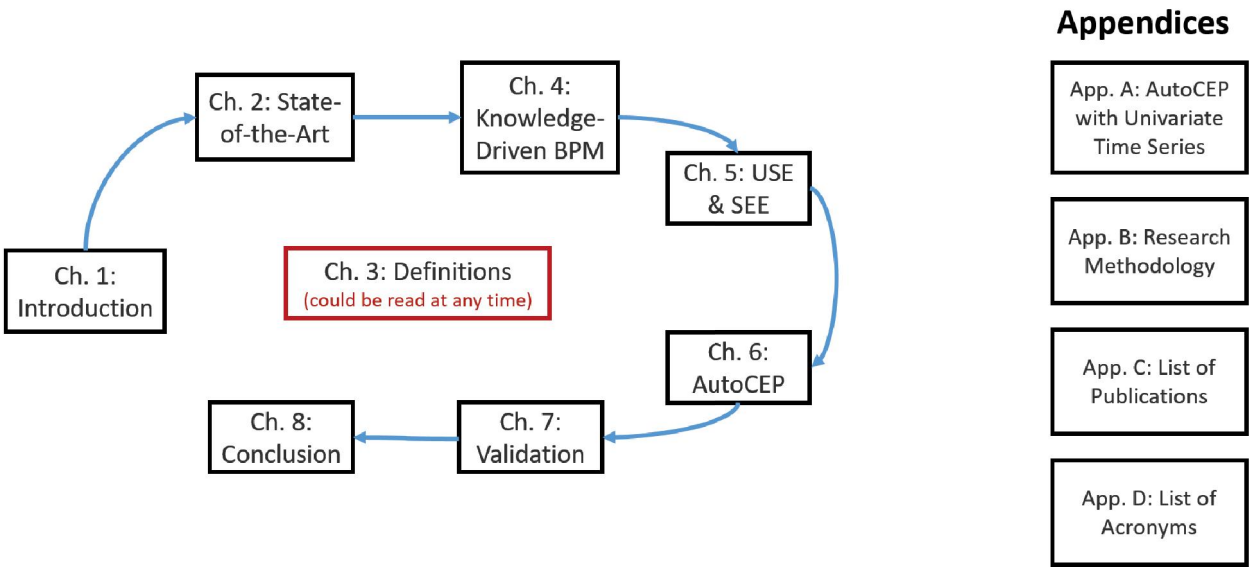

Figure 1.2: Structure of the Dissertation

CHAPTER 2

STATE-OF-THE-ART

> Literature adds to reality, it does not simply describe it.
>
> *C. S. Lewis*

This chapter serves as a pillar in order to provide an enough picture about the domains that we are working with. It discusses related work with fine levels of details. First this work will be broadly examined from three perspectives, and there will be an explicit differentiation between data mining, complex event processing, and business process management. This broad separation will help to position our work compared to the general domain. Next, complete sections will be dedicated to discuss tightly related state-of-the-art approaches that deal with the same problems that we are dealing with.

After going through the introduction of this dissertation, it should be already obvious that the content of this work is very dense. Therefore, this chapter will help to give an enough background, and to keep a coherent flow before going into more technical details in the following chapters.

2.1 Introduction

A fundamental research step is to thoroughly inspect the literature, and check whether it answers our concerns or not. In section 1.6, where we discussed our contributions, we already shed some light on state-of-the-art approaches, and highlighted their shortcomings. This section will give fine details about this related body of work.

At this point, it is clear that we touch three different domains, and for concreteness, we are interested in *early classification on time series*, *complex event processing rule learning*, and finally

the integration of complex event processing and business process management. However, before going in details on these specific points that interest us, we will zoom out in this section and talk about the main concepts when it comes to time series data mining and complex event processing. This discussion will help to give the bigger picture and to clearly position the work.

2.1.1 Time Series Data Mining

A time series is a collection of observations that is made chronologically and in a periodic fashion. It is characterized by its numerical and continuous nature. Based on the fact that time series represent a popular data model that could be obtained easily from different scientific and financial applications, we witnessed lately a remarkable deal of time series research from data mining perspectives. The article proposed in [59] serves as an interesting review on the subject.

Whenever time series data mining is mentioned, and before discussing any data mining technique, two main concepts need to be made clear: *representation methods* and *distance measures*. Time series data are known to have high dimensionality, so they are very large in size. This fact imposes many challenges when it comes to storing, querying, mining, visualizing, and any other form of analyzing this type of data. Efficiency in representation and distance measures would pave the way to a better exploitation of time series. Different types of representation methods and distance measures exist. From an abstract point of view, representation methods deal with dimensionality reduction, transforming data from one domain to another, and keeping representative features rather than storing the whole time series. Distance measures on the other hand dictates the strategy to measure the similarities between two given time series. The following sections will give more details about these concepts. At this point, it is essential to know that they make a complementary step for the time series data mining techniques discussed below.

A plethora of approaches exist to mine time series data, they started to flourish by the end of the 20th century [1, 17, 57], and they could be divided into three broad categories:

1. **Indexing:** Given a database of stored time series, a query time series at hand, and a similarity measure, indexing could help to efficiently extract the set of the most similar time series from the database. This similarity search in time series database has been introduced in 1993 by Agrawal et al. [1], and since then it has been an area of very active interest [85, 189, 188, 28, 157, 58]. Indexing on its own could be excluded from the time series data mining list, but what we intend to convey by indexing is what is called *query by content*.

2. **Clustering:** Generally, clustering is a technique where data are positioned into homogeneous groups, and no explicit definitions nor meta-information on the groups are known [155]. In the realm of time series, and given a database of this data model, clustering could help to find natural groupings of these time series. Specifically, and with a similarity measure at hand, clusters are constructed by grouping time series that have maximum similarity with other time series within the same group, and minimum similarity with time series from the other groups.

3. **Classification:** To simply put it, with a set of predefined classes at hand, a database of classified time series, and with the help of a specific similarity measure, an unlabeled time series, when going through the classification procedure, will be assigned a known class at the end. The early classification that we are interested in makes a subpart of this more complete data mining technique.

Before giving more details about the three broad categories above, one could find other researches, notions, and projects in the world of time series data mining, such as: subsequence matching [156, 89], motif discovery [37, 186, 144, 111], identifying patterns, trend analysis [32, 21], summarization [110], and forecasting [21, 31]. Depending on authors' point of views, these concepts could just mean other nomenclatures of the three tasks highlighted above, or they exist to serve the same purposes. Therefore, and in their cores, they could be traced back to indexing, clustering, or classification.

2.1.1.1 Representation Methods

Dimensionality reduction is one of the main goals of time series representation methods. By reducing the dimension, the number of data points that make up a time series will decrease. A myriad of representation methods exist that it is impossible to count them all. In general, the proposed methods reduce the data points within a time series as much as they could, but at the same time, they try to preserve the character, the shape, and the semantic that the unmodified time series hold.

One of the earliest followed methods is to merely sample the original data [7]. This transforms a time series of n data points (length) into a time series of m data points where $m < n$. In other words, m data points from the original time series are selected. The selection process could be run periodically, randomly, or following different criteria. Even though this method is extremely simple, it neglects the global shape and semantics of a given time series, and therefore the distortion may be high if the sampling rate is not specified carefully. To contend against the problem of distortion, other enhanced sampling variations were proposed in the literature [209, 94, 85]. Generally speaking, and instead of taking the data points directly from the original time series, other metrics could be computed for each sampled segment. For instance, in [209] the average is preserved. This method of calculating and keeping the means of segments is called Piecewise Aggregate Approximation (PAA) [94]. This technique is extended to allow for adaptive varying-length segments in [85]. Other than averages, in [106] authors represented the sampled segments with the Segmented Sum of Variation (SSV), and other researchers substituted data points with bit level representations [159, 11].

Perhaps one of the most frequently used representation methods is the Piecewise Linear Representation (PLR) [86, 87, 168, 147, 49, 197, 64, 81, 96]. In general, and among the data mining community, when time series segmentation is mentioned, what is meant is the PLR method. Briefly, PLR boils down a given time series of length n into k segments (segments are basically straight lines). This method, and in its purest form, is iterative. It starts by approximating an n-length time series with $n/2$ segments, and in the process, it iteratively merges each segment

with its right or left neighbor depending on a specific cost function. The algorithm is usually given stopping conditions, such as the envisioned number of segments. Various enhancements were later added to the PLR method, for example, in [91] authors introduced weighted segments, and users' feedback were integrated into PLR in [92]. Beside PLR, a way to approximate time series with straight lines is to employ linear regressions, and represent each segment with its best fitting line [177].

In 2001, Chung et al. [38] introduced the Perceptually Important Points (PIPs) identification process. This is a promising method that is currently followed to reduce the dimension of time series [191, 152]. It preserves the most important and descriptive points within a time series and omit all other points in between. To initiate the algorithm, the first and the last data points in the original time series are considered to be the first and the last PIPs. Then the next PIPs will be the ones with the maximum distance to the other PIPs that are already preserved [60].

Another common method to represent time series is by transforming them into symbolic strings. In essence, the time series is first segmented following one of the segmentation methods discussed above, then each segment is converted into a character [6, 132]. One of the most popular algorithms in this family is SAX [110, 112], which stands for Symbolic Aggregate approXimation. After sampling a time series with the PAA method discussed above, SAX transforms the resulted segments into symbols picked from a predefined set of alphabet characters. The length of the segments and the alphabet size are two obligatory parameters that need to be provided to SAX. This algorithms divide the y-axis into regions with the same size, where each region is represented by one of the alphabet characters. Then, each segment will be mapped to a symbol corresponding to the region in which it resides.

All aforementioned methods keeps the resulted transformation in the time domains. Other methods convert time series into different representations in various domains. This is the category where the famous Discrete Fourier Transform (DFT) resides [1, 154], and other transformation strategies such as the Discrete Wavelet Transform (DWT) [30, 149] and the Haar transform [183, 192].

2.1.1.2 Similarity Measures

When we briefly mentioned the three broad time series data mining tasks, we always highlighted the fact that these tasks are dependent on a specific similarity measure. In other words, with no similarity measure techniques, all data mining methods would become useless. Therefore, a way to measure similarities between time series is a fundamental step.

In the realm of time series, there is no best way to measure similarities, and different measures can perform better depending on the setups and the domains. In addition, and differently from traditional databases, similarity is not exact. Given the numerical and continuous nature of time series data, similarity is approximate [62]. Two main ways of measuring similarities could be considered. The first one is whole sequence matching, and the second one is subsequence matching.

Whole Sequence Matching: At this level, the whole length of all time series is considered in order to find similarities. The most popular approach is to employ a transformation method such as DFT or DWT, and then employ the Euclidean distance on the transformed coefficients. This practice started in the nineties with [1, 30]. However as it was later shown [153, 105], this approach is not always suitable in specific domains, such as in stock time series.

Hundreds of similarity measures could be found in the literature [67, 22, 103, 130, 164, 63, 198], and one of the most popular measure is the time warping. The first Dynamic Time Warping (DTW) technique to extract patterns from time series was proposed in 1994 [18]. In order to match two time series, $T^1 = \{t_1^1, t_2^1, ..., t_n^1\}$ and $T^2 = \{t_1^2, t_2^2, ..., t_m^2\}$, using DTW an n by m matrix M is constructed. The distance between t_i^1 and t_j^2 constitutes the element m_{ij} of the matrix M. If the Euclidean distance is used to compute the similarity, then $m_{ij} = (t_i^1 - t_j^2)^2$. This is called the alignment between the points $t_i^1 and t_j^2$. Then a mapping between T^1 and T^2 is extracted. This mapping is called the warping path, $W = \{w_1, w_2, ..., w_k\}$ with $max(m,n) \leq k < m+n-1$. W is a contiguous set of matrix elements and it should satisfy three main conditions. Foremost, the first and the last elements in W need to be the very first and last elements in the matrix M respectively, $w_1 = (1,1)$ and $w_k = (n,m)$. Given $w_k = (i,j)$ and $w_{k-1} = (i',j')$, the next two constraints are: $0 \leq i - i' \leq 1$ and $0 \leq j - j' \leq 1$. There is an undefined number of warping paths that could satisfy these conditions, the selected path is the one that minimize a given warping cost and it could be efficiently extracted by using dynamic programming as discussed in [17].

Finding the warping path could be computationally expensive, and therefore the literature is full of propositions to speedup the DTW matching process [99, 93, 84, 180, 167, 166, 56]. In [199, 160] authors have pruned the matrix to reduce the space that the warping path is allowed to visit. Others have proposed a sequential indexing structure [163] to balance the overhead of I/O access and the efficiency of indexing.

Other than speed concerns, DTW may not be the most accurate solution as Keogh and Pazzani pointed out [95]. One single point from one time series could map onto a very large number of points from another time series, the thing that could lead to unintuitive alignments. In addition, in the existence of abnormalities, DTW may fail to detect natural alignments. Other similarity measures [189, 131, 35] were proposed to answer the shortcomings of DTW while dealing with outliers and data imperfection.

Subsequence Matching: This mechanism matches two time series of different lengths. The query time series T_q has a length that is smaller than the time series T to match with. The main task is to find subsequences from T that have the same length as T_q and are similar to it. This search strategy requires that T_q be slid at every possible offset within T. The first approach to search for subsequences dates back to 1994 [57]. Afterward, the sliding window mechanism to match subsequences was introduced in [129], and then it was enhanced in [128]. As discussed in [98], subsequence matching is computationally expensive, and it suffers from bottlenecks regarding the processing time and the disk access time. Moreover, when matching subsequences redundant access to disk and CPU is required in order to account for the post-processing step

that further enhance the results [109].

The literature is full of innovative approaches that combine some kind of transformation before searching for subsequences [101, 184, 142, 174, 208]. The similarity in these approaches is usually computed using the Euclidean distance. For instance, SAX was used in [89] to detect abnormal subsequences. PLR was also employed to match subsequences in an online fashion [196] or counting on weighted segments [195]. The correlation and relationship between the extracted subsequences were studied by some authors as well [50, 73].

One of the most recent and fastest approaches to perform subsequence matching is proposed in [208]. It is a method that is adopted in signal processing, and it was recently brought into time series data mining. The method depends on the Fourier transform and on some statistical measures. To give the whole picture, we provide algorithm 1. The inputs of the algorithm are the full length time series T, and the shorter time series T_q. The output is an array of distances that contain all the distances between T_q and all subsequences within T that have the same length of T_q. More information about this algorithm and this method in general could be found in [208].

2.1.1.3 Indexing

Indexing is an efficient technique for organizing time series, the thing that helps in storing, querying, and processing them while reducing overheads. However, the effectiveness of indexing mainly depends on the representation method. Some methods do not allow for intuitive indexing, whereas others could be indexed naturally. For example, when a time series is transformed into other domains using DFT or DWT, each resulted coefficient could map onto one dimension of an index tree.

Having mentioned indexing structures, R-tree and its variations are among the most popular [182, 114, 57, 30, 54, 210]. Many indexing schemes were proposed to suit the different dimensionality reduction methods. For example, In [36], authors proposed an indexing method for PLR representations. On the other hand, for SAX-encoded series, iSAX was proposed [178] to index massive time series. In general a good indexing mechanism is expected to fulfill the following requirements:

1. It should be much faster than sequential scanning
2. It should require little space overhead
3. It should be able to handle queries of various lengths
4. It should allow insertions and deletions without rebuilding the index
5. It should be correct with no false dismissals

Indexing is the fundamental step that makes query by content feasible. Query by content is one of the most active area of research in time-series analysis. Given a query time series entered by

Algorithm 1: The MASS Algorithm

Input: T, T_q
Output: $dist$

1 $T \leftarrow T$.append$(0 \times |T|)$ // Append $0 \times |T|$ to T
2 $y \leftarrow y$.reverse()
3 $T_q \leftarrow T_q$..append$(0 \times |T| - |T_q|)$
4 $TF \leftarrow$ fastFourierTransform(T)
5 $TF_q \leftarrow$ fastFourierTransform(T_q)
6 $Z \leftarrow TF \times TF_q$ // The dot product
7 $Z \leftarrow$ inverseFastFourierTransform(Z)
8 $sumT_q \leftarrow sum(T_q)$ // Sum the elements of T_q
9 $sum2T_q \leftarrow sum(T_q^2)$ // Sum the elements of T_q squared
10 $cSumT \leftarrow cumulativeSum(T)$
11 $cSumT2 \leftarrow cumulativeSum(T^2)$
12 $sub1 \leftarrow cSumT.subarray(|T_q|, |T|)$
13 $sub2 \leftarrow cSumT.subarray(0, |T| - |T_q|)$
14 $sub3 \leftarrow cSumT2.subarray(|T_q|, |T|)$
15 $sub4 \leftarrow cSumT2.subarray(0, |T| - |T_q|)$
16 $sumX \leftarrow sub1 - sub2$
17 $sumX2 \leftarrow sub3 - sub4$
18 $meanX \leftarrow \frac{sumX}{|T_q|}$
19 $sigmaX2 \leftarrow \frac{sumX2}{|T_q|} - meanX^2$
20 $sigmaX \leftarrow \sqrt{sigmax2}$
21 $subZ \leftarrow z.subarray(|T_q|, |T|)$
22 $dist \leftarrow \frac{sumX2 - 2 \times sumX \times meanX + |T_q| \times meanX^2}{sigmaX2} - 2 \times \frac{subZ - sumT_q * meanX}{sigmaX} + sum2T_q$
23 $dist \leftarrow \sqrt{dist}$ **return** $dist$

the user, a set of the most similar time series are retrieved and returned. The queries could be broadly divided into two types: the ϵ-range, and k-nearest neighbors. The former retrieves all time series where the distances between them and the query time series are less or equal to ϵ. The latter returns to nearest K neighbors to the query time series depending on a specific similarity measure. Later approaches enhanced query by content techniques to deal with scaling, gaps, and faster rejection of false candidates [100, 204, 108].

2.1.1.4 Clustering

The main goal of clustering is to collect similar time series in natural groups, or what is called clusters. The distribution of time series over clusters should be carried out in a way that maximizes inter-cluster variance, and minimizes intra-cluster variance. Such grouping mechanisms could help to understand and analyze what knowledge is conveyed in data. In fact, distance-based clustering is extensively used for motif discovery [176], anomaly detection [83, 104], and

finding discords [206]. One of the main challenges when it comes to clustering is to specify a (near) correct number of clusters in order to capture the (dis)similarity of time series. Time series clustering could be further divided into two parts, whole series and subsequence.

Whole Series Clustering: At this point, the complete time series is used, and the goal is to form clusters of entire time series. Therefore, time series in one cluster are supposed to be similar in their entireties. Generally, five categories of clustering could be found in the literature: partitioning, hierarchical, model-based, density-based, and grid-based. In the realm of time series, only the first three are adopted. Beside pattern discovery as we mentioned earlier, time series clustering have proved to be very efficient for data streams [40, 39, 34, 126].

Subsequence Clustering: In the technique, clusters are constructed from subsequences instead of the complete time series. Therefore, time series may not be similar in their entireties, yet subsequences extracted from them may belong to the same cluster. One approach [74] is to employ DFT in order to analyze the periodicity of time series, and therefore to slice them into non-overlapping chunks. These chunks could be later tested for similarities and grouped into clusters. Another approach from the other side of the spectrum would be to extract overlapping subsequences [88]. However, the same authors proved that overlapping subsequence clustering may produce meaningless results, and that non-overlapping counterpart may miss important structures. Solutions to these inconsistencies started to emerge when not all subsequences were forced into the clustering procedure, but some of them were ignored all together. This is the followed method in [80, 48].

2.1.1.5 Classification

The main difference between clustering and classification is that in the latter classes are known in advance. Therefore, the primary goal of a classification model is boiled down to the task of learning distinctive features that characterize each class. With these distinguishing features at hand, any new and unlabeled time series could be given a class. Optimally, these features should be non-overlapping, distinguishing, and discriminative, therefore whenever they appear in an unlabeled time series, it could be easily classified.

Since this work heavily counts on time series classification, section 2.2 will focus on classification algorithms that are forged to only deal with time series. Other model-based generative classification methods have been proposed in the literature, like fitting auto-regressive models [39, 10], Hidden Markov Models (HMM) [181], and kernel models [33]. These generative approaches are omitted because they are not competitive against other algorithms that are completely dedicated for time series classification [12].

2.1.2 Real-Time Processing Systems

Traditional Database Management Systems (DBMSs) store and index data so they can be explicitly queried by users (Query on demand). Even though this is desirable, but from the point of

view of distributed real-time applications, this is not necessary neither efficient. These applications seek real-time responses in order to efficiently carry out their missions. For example, a fire detection application that should detect the existence of fire as fast as possible so timely countermeasures could be taken. It makes no sense to store and index the fire sensor data, then query them on demand to detect if a fire has happened or not. Such examples and many more constitute the main motivation behind Real-Time Processing Systems (RTPSs). RTPSs store queries instead of data, and then data are processed against these queries as they arrive. In other words, in application domains where users are interested in collecting information from multiple sources to process them in a timely manner, and to extract knowledge from them as soon as possible, then RTPSs are needed.

RTPSs are being used consistently in real life applications, some scenarios include: Environmental monitoring and predicting disasters as early as possible [24]. Identifying trends in stocks [46]. Fraud detection and prevention [172]. Intrusion detection systems [45]. RFID-based inventory management [194]. And anomaly detection in smart manufacturing [150].

The database community were the first to notice the need for implicit queries over data in addition to the common query-on-demand mechanism. Therefore, the notion of *active database systems* was introduced in 1989 [123] where actions are allowed to be automatically executed on stored data. Not until a decade later, real and more matured RTPSs emerged. The dominant two systems nowadays are Data Stream Management Systems [8] (DSMSs) and Complex Event Processing (CEP) systems [118]. The former processes multiple streams by performing some transformation operations, and then produces new continuous streams as output. The latter sees the flowing information as notifications of events that happen in the outside world, and by filtering, combining, processing, and analyzing these events, particular complex patterns and situations of interest could be detected.

2.1.2.1 Active Database Systems

Traditional DBMSs are completely passive, i.e., they present data only on the request of users. Therefore, there is no way to automatically present data after the occurrence of a specific situation or the fulfillment of a specific condition. Active database systems [43] were created to address this exact limitation.

In order to be considered as an active database, the system should support the active rules knowledge model. By active rules, we mean rules that adhere to the ECA (Event-Condition-Action) principle. In more details:

- **Event:** This is the trigger that stimulates the automatic action. At this level, different event generators could be supported. E.g., some active databases only consider internal operators such as insertion, update, deletion, etc. Others extend this functionality to even support external event sources such as sensors and clocks.
- **Condition:** After the events to be supported are specified, and on the detection of such events, actions are taken only if specific conditions are met. For example, on the insertion

of a tuple that contains a temperature value greater than a fixed threshold.

- **Action:** If an event is being detected, and the conditions are being fulfilled, then the action could be automatically executed. Actions could be categorized as internal and external. Internal actions modify the state of the internal database (e.g., insertion, update, deletion, etc.). On the counterpart, external actions could alert and trigger functionalities within external applications.

2.1.2.2 Data Stream Management Systems

Active databases are built with persistent storage in mind. Therefore, even if ECA rules are supported, persistently storing all coming data may not be desirable for RTPSs. For instance, if the number of coming events is too high and many updates are to be performed, then the limitation of such store-all systems may become more apparent.

In this context, DSMSs [43] come to the rescue. These systems can process large streams and perform transformations in a timely manner. The differences between DSMSs and DBMSs are numbered, the most obvious ones are:

- DBMSs store all data by default whereas the expected behavior of DSMSs is to discard all data by default
- DSMSs can efficiently handle unbounded streams and not just bounded data (tables)
- DSMSs could account for out-of-order events. That is, no assumption are made on data arrival order
- Since data are discarded, one-time processing is the best mechanism to deal with data streams
- Queries are standing and active in DSMSs, and therefore they continuously produce results as new stream events arrive. Users are not supposed to explicitly ask for updated information.

2.1.2.3 Complex Event Processing Systems

Rather than just calling them stream of events or flow of information as in DSMSs, incoming events in CEP systems have a specific semantic, they represent notifications of something that happened in the real world. CEP engines then are responsible of filtering these events, combining them, and deduce higher knowledge from them. This higher knowledge is commonly referred to as composite events or situations of interest. The deduction of such complex situations and patterns are made possible because of the expressiveness of CEP operators and the chaining nature of CEP agents, where each agent could process the output of other agents and adds to it.

Historically, the emergence of CEP started with the popularity of publish/subscribe systems in 1997 [162]. These system decoupled event generators from event consumers, where a message-oriented middleware was in charge of delivering the right events to the right consumers. Users express their interest in specific topics by subscribing to the relevant event classes at the middleware level, then sources publish the information to these middleware, where they will be filtered, processed, and outputted to the relevant subscribers. Two dominant types of publish/subscribe systems are available nowadays:

1. The first one is topic-based, where publishers classify events with a given topic, and every consumer who is subscribed to the specific topic will receive the event. This is the simplest form of the publish/subscribe pattern

2. The second one is content-based, in a way that complex filters could be specified to filter events based on their contents. Filters are specified by using different languages such as XML schema, simple attribute/value pairs, etc. Middleware engines are in charge of running the incoming events through the filtering requirements in order to dispatch these events to the right subscribers.

CEP systems adopted the publish/subscribe patterns, but they went way beyond that as well. Typical publish/subscribe systems, topic- or content-based, process events independently, each one as it arrives, and they do not take into account the history or summaries of already received events. Therefore, interesting queries such as: *trigger an alert if the same credit card was used to withdraw money from two different countries within 10 hours*, or *suspend a credit card if three wrong PIN codes were attempted within 1 minute one from the other*, are impossible to achieve. To overcome this limitation, CEP systems were devised, and they could be seen as powerful extensions of traditional publish/subscribe systems. CEP allows for users to express their interest no matter how complex and time-dependent it is.

In order to achieve this great expressiveness, CEP operates on complex patterns that could be achieved by specifying complete and accurate CEP rules. Since CEP rules are of paramount importance for use in our research, an in-depth discussion about them will be presented in a dedicated section later on.

2.2 Time Series Classification

As mentioned in the prologue of this chapter, tons of time series classification algorithms have been proposed so far, and they could be categorized in different ways. In this section, we will group them based on the type of features that they attempt to learn, and at the end, we will discuss the early classification style.

2.2.1 Whole Series Classification

With the exclusion of the class, whole series classification is almost indistinguishable from whole series similarity matching discussed in 2.1.1.2. Usually, one nearest neighbor classifiers

(1-NN) are used, i.e., the unlabeled query is classified with the same class as the most similar (the smallest distance) labeled time series. This technique could be very useful if there may be discriminatory features that characterize the whole time series. In its simplest form, Euclidean distance could be used, either on raw time series or on transformed representations, like PLR, SAX, and others. However, this assumes that time series are perfectly aligned, and this is not the case in real life situations. Therefore, alternative elastic distance measures are usually employed, with DTW (discussed in sec. 2.1.1.2) as the standard benchmark.

As discussed earlier, DTW suffers from some drawbacks when it comes to outliers and data imperfection, therefore to boost classification accuracy, different approaches have enhanced this raw method. In [82], authors added a multiplicative weight penalty in order to reduce the warping. The idea is to add weights to the element of the matrix M before searching for the warping path. Therefore, and if the Euclidean distance is used, the matrix elements would be $m_{ij} = w_{|i-j|}(t_i^1 - t_j^2)^2$ instead of $m_{ij} = (t_i^1 - t_j^2)^2$ that is figured in section 2.1.1.2. If the two compared time series differ in their complexity, and in order to avoid the unintuitive mapping that the DTW may create in such circumstances, the Complexity Invariant Distance (CID) proposed in [13] could prove really handy. This distance is weighted in a way that compensates the differences in the complexity. Another interesting approach that uses NN classifiers but with a derivative DTW distance measure is proposed in [69]. In this approach, two distances are computed, the first one is the distance between the raw series, and the second one is the distance between the first-order differences of these raw series. At the end, the returned distance is a combination of the two aforementioned distances, with the help of a parameter α that is learned during training. The same authors also proposed another derivative that works on transformed time series [70], specifically the sin, cosine and Hillbert transform.

2.2.2 Interval Based Classification

In case of noise and redundant shapes, whole series classifiers may get confused and deliver inaccurate results. This is where extracting features from intervals rather than the whole series could be desirable. The challenge to conduct such technique is finding the best interval. Specifically, there is an undefined number of possible intervals, so how to select the best one? Next, what to extract from each interval once selected? The first approach to answer these questions was proposed in [161], where intervals' lengths were equal to powers of two, and binary features were extracted from each interval. Afterward, a Support Vector Machine (SVM) was trained on the extracted features.

Another popular interval based classification approach is the Time Series Forest (TSF) [47]. It is a random forest approach. After specifying the number of desired decision trees, each tree is trained by dividing the time series into random $\sqrt{n}$ intervals where n denotes the length of the time series. Effectively, the training is done on 3 features extracted from each interval, they are the mean, the standard deviation, and the slop. The final classification results is done by a majority voting.

One of the most popular interval based classification methods is the Bag of Features (BoF) approach proposed in [16]. BoF could be thought of as an extension of TSF that incorporate multiple levels.

- The first level generates a new training data set with additional features. This is done by generating subsequences from the original existing training set. The number of generated subsequences depends directly on the length of the desired intervals (which is inputted as a parameter). Subsequently, for each subsequence and just as in TSF, the mean, the standard deviation, and the slop are computed. At the end, these features are combined with other global stats of the full series.
- After the first level, a new training set that contains the number of instances multiplied by the number of subsequences will be available. Then decision trees will be trained on this newly created data set. The number of decision trees will be increased incrementally until the out of bag error stops decreasing. After building the random forest, class probabilities for each subsequence are computed.
- This third level creates a histogram of class probability estimates for each subsequence, and then it combines these estimates for each original time series. This is how a bag of patterns for each series is created.
- The final level is to build the definitive random forest classifier using the bag of features of each time series in the training set.

2.2.3 Dictionary Based Classification

If motifs, or frequent patterns are what characterizes a given class, then dictionary based classification methods are the most suitable approach. This technique counts the frequency of patterns. They usually operate on symbolic representations of time series. The main idea is to slide a window of a given length, and then compute the distribution of words over the different training instances. Following this mechanism, the correlation between the frequency of specific patterns and the occurrence of particular classes could be established.

The Bag Of Patterns (BOP) approach proposed in [113] computes a histogram of words for time series that are transformed with SAX. Then the same method is followed to build a histogram for unlabeled time series, and finally the new series is given the class of the most similar histogram. In [175] authors discussed another approach that is very related to BOP, as it works on SAX-transformed time series. The key difference is that it adds techniques used in Information Retrieval such as the vector space model. BOSS [170] is yet another recent approach that works on DFT representations of time series, and that could efficiently count frequent patterns in the presence of noisy data.

2.2.4 Shapelet Based Classification

In scenarios where one (or a limited number of) discriminative pattern(s) could characterize the class of the time series, shapelets are the most suitable technique. They are shapes (subse-

quences) that occur within the time series, independently from the phase (the placement at the beginning, middle, end), and they are very distinguishing and discriminatory. For example, one could think of abnormality detection like fatal heartbeats in ECG data. Shapelets were devised in 2009 [207], and in their work, authors discover shapelets through testing all possible candidates between two given lengths. This approach is called the brute force algorithm, it yields very accurate results but on the expense of high computation time. The time complexity is $O(n^2m^4)$ where n is the number of time series and m is the length.

When they first emerged, shapelets were incorporated into decision trees as the best split points to discriminate between classes [207, 143]. A more recent approach [158] suggested the Fast Shapelet (FS) algorithm. This algorithm drastically improved the time complexity to find shapelets. It was reduced from $O(n^2m^4)$ to $O(nm^2)$, however it is heuristic, with random projections, and it is not guaranteed to have the same results as the brute force algorithm. FS contains different steps:

1. The first step is to change the representation of the data form raw real-valued time series into discrete and low dimensional. This is done by transforming them into symbolic representations with SAX.

2. After the transformation, the brute force algorithm could be applied. To contend against the problem of false dismissals — i.e., two close time series may produce different SAX words — FS uses random projections. It randomly masks and further transforms SAX words of high dimensionality into lower dimensions.

3. After multiple random projections are performed, a frequency count histogram is learned for each class. A discrimination score for each SAX word is attributed depending on how much it distinguish between classes.

4. At the end the k-best SAX words are selected, and they are mapped back to their original form, where they could be used to build the decision tree just like in [207].

Other shapelet based approaches [79, 20] devised what is called Shapelet Transform (ST). These approaches are more concerned with the discovery of discriminative shapelets rather than building a classifier to use them. Following the ST, the original time series are transformed into a vector of distances where each element represent the distance between a given time series and a specific shapelet. ST balances the trade-off between the number and quality of shapelets, i.e., it evaluate shapelets on how well they discriminate just one class. Assessing shapelets is done by counting on the information gain metric. At the end, the top k shapelets for each class are returned.

Another interesting approach that learns shapelets is discussed in [71], it is called LS (Learned Shapelets). This proposition learns shapelets that characterize the available classes, however the learning procedure is different than other algorithms like FS and ST. Originally, shapelets are defined as subsequences from the original time series, yet in LS they are not limited to that. LS uses k-means clustering of candidates in the training data set to initialize k shapelets. Then these shapelets are adjusted based on a regression model for each class.

2.2.5 Early Classification

So far traditional classification approaches were discussed where earliness has no significance. However in some scenarios, for example when detecting abnormal heart beats, earliness could play a major role. Early classification usually deals with online stream classification where the major interest is to classify as early as possible the currently streamed time series, without waiting for the complete time series to be read. In other words, given the smallest possible subsequence of a time series, the main goal is to accurately predict its class.

Among the different classification techniques that we have discussed, we argue that shapelets are the best fit for early classification. As mentioned earlier, shapelets are specific shapes that are characteristics of a given class, and no matter where they appear in a given stream, the class could be immediately predicted. Furthermore, if shapelets' lengths are taken into consideration when attributing utility scores, earliness could be greatly boosted. Approaches such as [135, 138] prove that shapelets perfectly fit for early classification and rule discovery.

By definition, shapelets only work on univariate time series, yet few recent research initiatives have proposed different methods to extend this limitation and to favor multivariate time series [115, 65, 66, 151].

In [65], authors proposed MSD by projecting the logic of univariate shapelet discovery algorithms into the multivariate world. The idea is to have a sliding window — between the minimum and the maximum lengths of shapelets — but instead of sliding it over one dimension, the window is moved along all dimensions at the same time. Therefore, what the authors call *multivariate shapelets* could be extracted. After extracting these multivariate subsequences using this windowing technique, authors continued normally as in the univariate shapelet extraction algorithm, i.e., calculating distances, information gains, and utility scores, etc. but they proposed multivariate algorithms and methods that replace their univariate counterparts. Typically, methods to deal with matrices instead of vectors. Even though, this approach attempts to employ shapelets in multivariate environments, it still suffers from some drawbacks: First, sliding a global window imposes the restriction that all shapelets from the different dimensions need to start and end at the same time, and they are limited to have the same lengths as well. Second, the algorithm always tries to seek shapelets from all dimensions, regardless of the fact that in some cases some dimensions may be meaningless. Third, and since shapelets from the different dimensions start and end at the same time, no correlations, like sequence and time constraints, could be learned.

To overcome the limitations in [65], the same authors proposed a new shapelet-based approach for early diagnostics [66]. The newer approach builds a binary matrix where rows are instances and columns are all the different extracted subsequences within the given lengths. Each cell contains a binary indicator stating if a subsequence appears in a given instance or not. Authors then followed a convex-concave optimization problem to select one and only one subsequence from each dimension. Finally, they reduced the number of outputted shapelets by using mixed integer programming, and so disregarding unimportant subsequences. Authors overcame some

limitations from the approach proposed in [65], specifically no restrictions on the start, end, and length of multivariate shapelets are imposed, and more importantly irrelevant dimensions are also disregarded. However, getting one and only one representative from each dimension is somehow restrictive, as sometimes one shapelet from a given dimension may perform differently when combined with various shapelets from another dimension. On the other hand, and as in [65], the approach stopped at the point of extracting different shapelets from the various dimensions, with no regards to any correlation that may exist among them.

An approach that generalized univariate shapelet discovery to the multivariate setting is discussed in [115]. It is called REACT and this approach differs from our work by dealing with numerical and categorical time series as well. Authors proposed a strategy, inspired from equivalence class mining [117], to reduce the number of generated features. Therefore they devised different formulas to learn categorical and numerical minimal frequent itemsets. Moreover, the paper presents a GPU implementation that significantly reduced the learning time. Nonetheless, all dimensions are used to construct patterns, and finally they did not achieve complete rules with both time and sequence constraints.

Shapelet Forests [151] and Shapelet Ensembles [29] are other interesting approaches to address the problem of shapelet extraction from multivariate temporal observations. Even thought they follow different strategies to get univariate shapelets from the various dimensions first, at the end both approaches, and instead of building one single decision tree like in the univariate settings [158, 207], they built an ensemble of decision trees in order to classify new multivariate instances. In [151], feature selection techniques were used to assign weights to the existing dimensions, finally, the decisions given by the trees are ranked by their weights, and the one with the highest weight is accounted for. On the other hand, the authors in [29] employed a majority-based voting to assess the classes of new instances. Exploiting such ensemble techniques could make it rather difficult for domain experts to understand what is the exact pattern that leaded to a specific situation. Moreover, both approaches failed to consider time and sequence constraints among shapelets on the different dimensions.

The detailed discussion in this section particularly touched the numbered approaches that exploited shapelets in the multivariate settings. However, a vast majority of other proposals exists to address different data mining problems for multivariate time series, such as [124] to find motifs, and [190] for multidimensional indexing.

2.3 CEP Rules Specification

Complex Event Processing is a very active area of research, and in the same time it is adopted by business enterprises as the standard course for real-time analysis and situation detection [119]. CEP engines are scalable regarding the number of data sources, input events, and registered CEP rules. Normally, they are concerned with the processing of large and fast streams of input events in order to recognize on-the-fly some situation of interests or composite events. Due to the increasing interest in the CEP technology, various languages have been proposed to define

events and rules [23, 3, 41, 172].

CEP rules are basically expressed through a combination of both declarative and pattern-based languages.

- **Declarative Languages:** This type of languages allows users to express what they want, rather than how they want it to be done. They are mainly derived from languages such as relational algebra and SQL [52]. CEP rules extend these relational languages by adding ad-hoc operators to support stream-specific operations, such as time windows and sequencing constraints.

- **Pattern-based Languages:** If we return to the abstract ECA model, pattern-based languages define the conditions of rules through complex patterns. These patterns could incorporate logical, sequencing, content, and timing operators among others. Moreover, actions could be also specified as patterns to create composite events that may be further processed by other CEP operators.

Successful CEP systems such as Oracle[1] and Esper[2] support the expression of CEP rules in both these languages.

2.3.1 CEP Operators

CEP rules are built on top of CEP operators. This set of operators defines the expressiveness that could be achieved and the complexity of patterns that could be detected. CEP rules definition languages such as CQL [5] and EPL [118] include a rich set of operators, we will discuss the most important ones at this level. This discussion will help to evaluate the CEP rules learned by our approach, as we will show in later chapters.

Single-Item Operators Most operators that fall under this category are imported from relational algebra, and they are enhanced to deal with streams and flowing events. They operate on events one by one. The most known ones are selection, mapping, and projection operators.

Selection and just as in the SQL language, keeps events if their contents satisfy a given constraint, and discard events if their contents do not. For instance selecting temperature events that have values greater that a specific threshold.

Mapping could modify the attributes of events and perform some kind of transformations, for example capitalize the name of persons. Projection refers to extracting a subset of attributes from a given event, for instance project only the timestamps of sensor readings.

[1] http://www.oracle.com/technetwork/middleware/complex-event-processing/overview/index.html
[2] http://espertech.com/esper/

Logic Operators These operators help to define patterns that detect the existence (or absence) of events coming from one or more streams. They are usually order independent, i.e., they are just concerned with the detection of non detection of specific information. Logic operators are typically found in pattern-based languages

Among the most popular operators are: Conjunction and it means that all events that are specified for this operator need to be detected, it is like a logical AND operator. On the contrary, the logical OR operator is represented by the CEP disjunction operator, and this means that at least one of the specified events must be detected. The third known operator is the repetition, and it allows patterns to be fired if an event is repeated at least (and optionally at most) a specified number of times.

Sequencing Operators Sequencing operators resemble to some extent the logic operators but order is of utmost importance. In other words, they detect the occurrence of the specified events but in the specified order. For example, stating a pattern where the event e_1 must be followed by e_2 will not be fired if e_2 arrived into the system before e_1.

Pattern-based languages provide sequencing operators out-of-the-box. This kind of operators is highly desirable in CEP systems and in building CEP rules. Almost all CEP rule learning approaches that we will discuss next support the learning of this operator. In some part, this is due to its importance, and in another part, it could be one of the easiest operators to learn. This simplicity turns into complexity when sequence constraints among events coming from multiple sources need to be learned.

Windowing Operators Arguably, windows are the most important CEP operators to consider, and it is the only mechanism that allows to transform unbounded streams into bounded set of events. They could be used in combination with any other CEP operator to introduce time constraints. For example, a sequence must be detected within this specif time window. Due to the importance of windows, they are naturally supported in any CEP rule definition language that could be found nowadays. Some authors [43] —rightfully— consider windows to be language constructs rather than just operators, because they are mainly used to scope the actions of other CEP operators.

Two types of windows could be found. The first one is time-based, that is, it scopes operators based on time constraints. The other one is count-based, that is, it scopes the operators based on the number of events that are currently inside the window. Regarding their bounds move, time- and count-based windows could be mainly classified as fixed (tumbling) or sliding windows.

Fixed windows are non-overlapping and they do not flush unless the constraints are satisfied. For example, a count-based window with a length equals to ten events will wait until ten events are effectively inside the window before sending them to processing and emptying itself, in preparation to receive other new ten events. Sliding windows may be considered the most com-

mon type of used windows. They have fixed sizes but they are also attributed a step. This step allows these windows to overlap over each other.

From the point of view of CEP rule specification, windows may be the trickiest to define. This adds an extra layer of complexity while attempting to automatically learn the size of such windows.

Flow Management Operators CEP typically includes a set of operators to support the management of flowing events. For example, inserting simple or composite events into named windows (more on named windows will be discussed in chapter 3), updating, duplicating, deleting, and joining multiple streams.

Perhaps, one of the most popular flow management operator may be the grouping mechanism (or the group-by operator). It allows to divide event streams into different partitions internally, and then carry out the processing on each partition independently. Such operators are very similar to the grouping operators in relational languages such as SQL.

Aggregation Operators It is natural for CEP engine to detect and perform some aggregation operations over events. For example computing the average of temperature values within the last hour. Aggregation functions could be used as conditions in rules, or as actions to output results as well.

All existing languages include a large set of predefined and known aggregates such as the average, maximum, minimum to count a few. Moreover, almost all systems offer the possibility to define and integrate customized aggregate functions.

2.3.2 Automatic CEP Rule Generation

Little work exists on the learning of CEP rules, and to the best of our knowledge, only five related approaches [121, 122, 146, 185, 173] have suggested to take the path of integrating data mining to this end. However, none of them is capable of dealing with periodic and numeric readings of events (time series) or with trend recognition. In addition, all these approaches focus only on the detection of situations of interest, and not on the prediction. So as far as we can tell, this is the only work that is done on: first integrating trend mining techniques with CEP, and second automatically learning **predictive** CEP rules.

Authors in [121, 122] proposed the iCEP framework for the automatic learning of CEP rules. The two approaches count on different techniques but they follow the same logic. The problem of learning CEP rules in their work is boiled down to the learning of the operators of these rules. In the recent version [122] a remarkable amount of CEP operators are supported. Specifically: selection, conjunction, parametrization, sequence, window, aggregation, and negation. Authors followed a flexible modular architecture, where they associated each operator with a module. Therefore in each module, ad-hoc algorithms could be used to learn one specific operator, and

build one part of the rule.

Although the followed methodology has its strong points regarding the rule expressiveness that it tries to achieve, but it still has some limitations. The approach in [121] uses data mining techniques and metrics, and it heavily relies on thresholds while learning the relevant events, the window length, the negative events, etc. The thresholds in this approach are not learned, but they need to be specified by experts. We argue that this is a non-evident task, and if the user is left to do it, then imprecise thresholds are very probable, leading to inefficient rule generation. One of the limiting factors of these two approaches [121, 122] is that they count on the strict intersection theory that leads to one and only one rule. In other words, the proposals will only work under the assumption that for each situation of interest there is just **one** rule that leads to it. We argue that in real life different rules may indeed lead to the same situation. To clarify with a simple illustration, let's suppose that we have a situation of interest SoI that will occur if the event E_1 is followed by the event E_2, **or** the event E_2 is followed by the event E_3.

$E_1 \; and \; E_2 \mapsto SoI$
$E_2 \; and \; E_3 \mapsto SoI$

Using an approach that strictly intersects the available records and outputs only one rule —no matter how much history data we have— the learned rule will be $E_2 \mapsto SoI$ (i.e., the intersection), which will obviously yield false results. Therefore we deem the assumption that *each situation has one and only one rule* as unrealistic.

Other work to integrate data mining techniques is proposed in [185] and [173]. From an abstract point of view, both approaches are based on reinforcement learning styles. The main idea is to let the user defines a CEP rule with arbitrary parameters, then following a prediction-correction paradigm these parameters could be tuned depending on the feedback of human experts. The drawback of such approaches is that users are in charge of defining the structure, the template, and the operators of the rule, only parameters could be tuned.

In the work discussed in [146], authors proposed an extension for the hidden Markov models. These extended models could learn sequences of events but they could also discard the noise. More specifically, when a noisy event is received, the Markov model stays on the same state, and does not proceed to the next one. In general the work is more concerned with the exclusion of noisy events rather than learning a complete rule. In addition the approach is demonstrated to work just on sequence patterns, but it cannot take windowing or other constraints into account.

2.4 The CEP/BPM Integration

From a research perspective, predictive monitoring and proactive adaptation are storming the domain of workflow and process management [134, 133]. Related approaches with different scopes, advantages, and limitations are arising continuously. Generally, these approaches try to efficiently implement the concept of the MAPE *(Monitor-Analyze-Plan-Execute)* loop, introduced by Kephart and Chess in their vision on autonomic computing [97]. In respect to the

phases, the loop consists in monitoring the process run-time execution to get some strings of events *(Monitor)*, analyzing them in an attempt to detect the need of modification *(Analyze)*, generating a series of adaptation actions *(Plan)*, and finally applying them to the ongoing process *(Execute)*. The loop is considered a cornerstone to achieve proactive computations, as it drew theoretically-feasible solutions that could anticipate upcoming violations, and address them in advance. However when it comes to its realization, researchers are still suffering from numbered limitations when drawing timely predictions, and serious flaws when practicing adaptations. As in most of the cases, and despite their efforts, they are still reactive rather than proactive, and more importantly, they lack dynamic adaptation policies, where each exception needs to be addressed at design-time.

As hinted in the introduction, different research efforts [14, 76, 78, 26, 25, 120, 77, 4, 102, 127, 187] are devised to integrate CEP within BPM in an easy and generic way. All these proposals, in a way or another, promoted design-time integration of events over the process model. Typically, they assign what we call design-time monitoring points over the process model, so that the monitoring, subsequently the prediction (if supported) are executed upon reaching these points. This practice will not allow for a fine-grained processing inside the monitorable task itself.

One of the earliest frameworks that started this method is the PREvent framework [107]. It defines a set of prediction checkpoints over the flow of the process. Every time one of these points is reached, a prediction is enacted to check for probable future violations depending on the current state of the flow. Then some predefined adaptation actions are executed if needed. The monitoring and predictions are triggered only upon checkpoints, and adaptation plans are statically assigned regarding the placement of each point. In other words, adaptation actions at a specified position will be the same despite the process context. Both the definition of good checkpoints and the assignment of good adaptation actions for all undesired situations that may arise at run-time, are tedious and human-oriented tasks, susceptible to erroneous configurations.

Other approaches employ predictions by statically defining preferable attribute values at design-time, then monitor the same attributes at run-time, and finally deduce the consequences of any mismatch on the state of the process. (i.e., if a violation will occur). For instance, They annotate all services in a composition with preferable execution times to suit the agreement regarding the global completion time. These times are assumed like in [171], or learned from past executions like in [193]. Afterward, during execution, the real response time for each invoked service is monitored and compared to the preferable one. In case, the first exceeds the latter, a summation operation takes place to check if the global completion time will be violated. If so, the process is suspended and the adaptation is triggered. This kind of approaches operates on the detection of already violated local constraints, so regarding local violations, it remains reactive rather than proactive.

More recently, the Green European Transportation Service or the GET Service[3] was launched as a European project in the domain of logistics. It incorporates efforts to enhance business processes in order to efficiently support transportation missions. The ultimate goal is to reduce cost, time, and CO2 emission. The project embraces different publications with heterogeneous research problems, ranging from boosting design languages such as BPMN to better support process monitoring [15], going through the semantic enrichment of transportation-related events, tackling their geographical proximity to the planned transportation routes [125], and finally to employ predictive monitoring in order to detect violations prior to their occurrences in an online logistics process [27]. The latter is one of the most interesting approaches. Authors inspect the same question we are trying to address in our work. They distinguished monitorable tasks that need to be continuously monitored in a business process. The focus is mainly on flight shipping activities. They defined constrained and monitored attributes for this activity at design time. During execution, they monitor these attributes in an interval-based scheme. Then by counting on a trained support vector machine classifier, they detect divergence if any. They demonstrated their idea with a transportation mission and a support vector classifier, however it is really not obvious how it could be propagated to other application fields. The intricate integration of data mining and event processing is not abstracted for BPM users.

2.5 Summary

This chapter provided in-depth discussions of the most related fields to our work. We started by showing the global picture, introducing the different domains that we are working with, and showing where they stand in the overall computer science field. Later sections showed with fine-grained details the exact subjects that we are dealing with. Specifically, classification over time series, CEP rules specification, and the CEP/BPM integration.

The propositions discussed in this chapter made it clear that there is still some shortcomings to address. On the time series classification level, shapelets are still independently extracted from multiple time series dimensions. In other words, correlations such as sequence and time constraints are not learned from historical observations. Rules in the CEP domain are still specified manually, and the few approaches that tried to change this did not completely succeed. On one hand, they cannot learn complete rules, and on other hands they fail to integrate prediction within the CEP domain. Lastly, some approaches to integrate BPM and CEP were discussed, but it is shown that this integration is not evident and the great potentials of CEP are waster on uncomplicated analysis and the detection of straightforward situations.

[3] http://getservice-project.eu/

CHAPTER 3

BACKGROUND

> Our scientific age demands that we provide definitions in order to be taken seriously.
>
> *Dennis Prager*

This chapter provides an elaborated list of definitions that will be referenced in the upcoming chapters. Understanding the concepts discussed and defined in this chapter is important to smoothly go over the more technical chapters to come. Better yet is to reference this chapter now and then from any other part of the dissertation, in order to shatter any arisen ambiguity.

The definitions are divided into two sections. The first one will tackle the early classification on time series domain. Afterward, some notations and standards from the complex event processing field will be discussed. At the end, performance measures of our approach will be identified, and a more formal problem statement will be included.

3.1 Early Classification on Time Series

Early classification in our work is achieved through shapelets [207]. Briefly, shapelets are special shapes in a time series that are discriminative and have high significance in determining a time series class.

Definition 3.1.1. A time series $T = \{t_1, t_2, ..., t_{n-1}, t_n\}$ is a sequence of observations that is made chronologically at discrete times. Each time series is characterized by its length, $|T| = n$.

Definition 3.1.2. To measure the similarity between two time series T_1 and T_2 of the same length $|T_1| = |T_2| = n$, the Euclidean distance is used. This distance is denoted as $||T_1, T_2||$.

$$||T_1, T_2|| = \sqrt{\sum_{i=1}^{n} (T_1[i] - T_2[i])^2} \tag{3.1}$$

Definition 3.1.3. A subsequence s of a given time series T is defined as a set of continuous real values that are sampled from T, we say $s \subset T$, and more formally $s = \{t_i, t_{i+1}, ..., t_{i+n-1}\}$ where $|s| = n < |T| = N$.

Definition 3.1.4. In order to calculate the similarity between two time series s and T of different lengths $|s| = n < |T| = N$, a window of length equals to n is slid over the time series T, then on each stop the distance between the subsequence q_i within that window and s is calculated. Finally the best matching distance is extracted as:

$$||s, T||_{bmd} = \min_{\forall i \in \{1,2,...,N-n+1\}} ||s, q_i|| \tag{3.2}$$

Definition 3.1.5. For the sake of this work, a shapelet $\hat{s}$ is defined as $\hat{s} = (s, \delta, c_s, score)$, where s is a time series, δ is a distance threshold, c_s is the class of the shapelet, and $score$ is a utility score.

Definition 3.1.6. A shapelet $\hat{s} = (s, \delta, c_s, score)$ is said to be contained in a time series T, if the similarity between them is less or equal to the distance threshold δ, $\hat{s} \in T \Leftrightarrow ||\hat{s}, T||_{bmd} \leq \delta$.

$$\hat{s} \in T \Leftrightarrow ||\hat{s}, T||_{bmd} \leq \delta \tag{3.3}$$

Definition 3.1.7. A d-dimensional time series is represented as $dT = \{T^1, T^2, ..., T^{d-1}, T^d\}$. For brevity, we may refer to Multivariate Time Series as MTS, and Univariate Time Series as UTS. The length of an MTS is defined as $|dT| = \max_{\forall i \in \{1,...,d\}} |T^i|$, so it is equal to the longest UTS. Whereas $|dT|_{min} = \min_{\forall i \in \{1,...,d\}} |T^i|$. Each multivariate time series is labeled with a class $c \in C$ where C is a finite set of class labels.

Definition 3.1.8. A dataset of d-dimensional time series is designated as a collection of pairs, i.e, each time series with its class, $D^d = \{(dT_i, c_i)\}$.

As noted before, shapelets are extracted from training sets with univariate time series—we will refer to a univariate training set simply as $1D$. In a multivariate environment, d univariate training sets could be constructed from a dataset of d-dimensional time series if grouped by dimensions. $1D^i$ symbolizes a univariate training set containing time series for the dimension i.

Definition 3.1.9. Given a multivariate training set D^d, $\hat{S}^i$ with $i = 1...d$ is a set containing the shapelets extracted from $1D^i$. For clarity, shapelets in $\hat{S}^i$ may also have i as a superscript to state that they are extracted from the dimension i. The set of all shapelets $\hat{S}_D$ of a given D^d contains all the shapelets extracted from all dimensions, therefore, $\hat{S}_D = \bigcup_{i=1}^{d} \hat{S}^i$.

Definition 3.1.10. A Time-Annotated Sequence (TAS) of shapelets and as its name implies, is a list of shapelets that are ordered and with time annotations between them. Formally, $TAS = \{\hat{Sh}, W, c\}$ where $|W| = |\hat{Sh}| - 1$. $\hat{Sh}$ is a list of shapelets that constitutes the sequence TAS (the order of the list matters). W is a set that contains real values representing the time windows between shapelets, its cardinality is always equal to the cardinality of $\hat{Sh}$ minus 1. Finally, c is the class of the sequence.

Example 3.1.1. For a given dataset D^2, if we have $TAS = \{(\hat{s_1}^2, \hat{s_2}^1), (3), c_i\}$, this sequence means that the shapelet named $\hat{s_1}$ from the second dimension needs to be followed by the shapelet named $\hat{s_2}$ from the first dimension within a window of 3 time steps in order for the pattern to be fulfilled. And a sequence of the form $TAS = \{(\hat{s}^2), \emptyset, c_i\}$ simply indicates that if the shapelet $\hat{s}$ is found on the second dimension, then this is sufficient for the pattern to be fired.

Definition 3.1.11. A given sequence $TAS = \{\hat{Sh}, W, c\}$ is said to be contained in an MTS dT, if for all $\hat{s}^i \in \hat{Sh}$, $\hat{s}^i$ needs to appear in T^i, in addition the order of the shapelets and the time windows that exist in W need to be respected. In such case, the predicted class of dT will be c.

Definition 3.1.12. Given a dataset $D^d = \{(dT_i, c_i) \mid i = 1..N\}$, the set of all sequences that could be extracted from D^d and will lead to all classes is denoted as TAS_D.

3.2 Complex Event Processing

CEP rules are defined using different CEP operators like windowing, selection, sequence, etc. These operators are considered the main enabler to define complex patterns. In this work we will demonstrate rules using the Event Processing Language (EPL). EPL is an SQL-like language, that is understandable by different known CEP engines such as Esper[1]. In EPL, queries operate on streams of events instead of relational tables. Note that we will skip a detailed description of the vast features of EPL, and we will focus only on the necessary operators in this section. Interested readers may refer to the Esper documentation for a comprehensive description of EPL.

Example 3.2.1. This example demonstrates the selection operator. Given a weather event WE that has two attributes $temperature$ and $humidity$. An EPL statement to select all attributes would be **SELECT * FROM WE**. In order to select only temperature values, the statement would change to **SELECT temperature FROM WE**. On the other hand and to select events with temperature values that are larger than 10, then the statement will be **SELECT WE(temperature** $>$ 10**) FROM WE**, etc.

Definition 3.2.1. To define a sliding window over an event stream **.win:length(**X**)** is used. This creates a window of length X events. If $X = 4$, and four events already exist in the window, when a fifth event arrives the oldest event in the window will be thrown out

Definition 3.2.2. To select all events that are captured within a given window, the following syntax is used: **window(*)**. To count the number of these events **count(*)** is employed instead.

[1] http://espertech.com/esper/

Definition 3.2.3. To impose constraints on events, three different methods could be used in EPL. We already demonstrated the first one with the selection operator. Another method is to employ the **WHERE** construct. The latest method is to utilize **HAVING**. The **HAVING** clause, and differently from the first two methods, operates on what is called *aggregation methods*. Briefly, the selection and **WHERE** impose constraints on single events, whereas **HAVING** could impose constraints on the collection of events within a given window, like the count of these events, the average value of temperature, etc.

Example 3.2.2. To select weather events that have an average temperature value greater than 10 degrees from a window of 10 events, one could write: **SELECT * FROM *WE*.win:length(10) HAVING avg(temperature)** > 10

Definition 3.2.4. A *Named Window NW* is a table-structured window with column headers and data types (just like a table in SQL) that could be created explicitly in EPL. A *Named Window* could be used for many ends, e.g., when a given CEP rule matches it could insert data into NW, and another could listen for any inserted data, etc. It is exactly like a stream of events but created explicitly, in other words, anything that could be done on a stream of events could also be performed on NW. These windows need to have a storing policy, because by default any event that is inserted into the window, is dispatched to the engine and discarded right away (just like a stream). To instruct the window to keep all inserted events, the syntax **.win:keepall()** could be used. The main utility of named windows is to insert *composite events* (see next) into them, and with the existence of CEP queries to process these composite events even further, a complex chain of event processing that is famous, fast, and advantageous in CEP systems could be devised.

Definition 3.2.5. *Raw events* are events that come directly from sensors or other simulation and event-emitting sources like CSV files. On the other hand, *composite events* are more complex events that are the results of CEP rules. That is, when a CEP rule matches, it could emit a composite event of any form. Typically, this composite event is stored in a named window in order to be processed by other more complex CEP rules.

Definition 3.2.6. For the sake of this paper, simple CEP rules R are rules that operate on single stream (dimension) of raw events. Complex CEP rules CR on the other hand, are rules that operate on multiple streams (dimensions). These complex rules combine the results (*composite events*) of the different simple rules (typically through *Named Windows*), and impose sequence and time constraints on them by employing CEP operators to create complex patterns. Just as TAS_D, the set of all complex rules that could be extracted from a given data set D^d is denoted as CR_D.

3.3 Performance Metrics

Other than execution and classification times, we are interested in several measures in this paper to assess the performance of our algorithms. More specifically we are interested in the average f-score, accuracy, applicability and the earliness of the approach.

The accuracy (acc) is computed from the number of true and false classifications as:

$$acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.4}$$

TP stands for the number of True Positives, TN is that of True Negatives, whereas FP and FN are False Positives and Negatives respectively. By definition, $0 \leq acc \leq 1$, and the higher the better.

The avg. f-score is computed using equation 3.5, with c as a class in the set of all classes C, $precision(c) = \frac{TP}{TP+FP}$ and $recall(c) = \frac{TP}{TP+FN}$.

$$avg.f\text{-}score = \frac{1}{|C|} \sum_{c \in C} \frac{2 * precision(c) * recall(c)}{precision(c) + recall(c)} \tag{3.5}$$

By definition, $0 \leq avg.f\text{-}score \leq 1$, and the higher the better.

The earliness on the other hand is computed from the average number of observations needed before classifying a full-length MTS. Therefore for a given MTS $dT \in D^d$, supposing that TAS was used to classify dT and $||TAS||_{dt}$ denotes how much observations are read before assessing the classification, one can compute the earliness as:

$$earliness = \frac{1}{|D^d|} \sum_{dT \in D^d} \frac{||TAS||_{dT}}{|dT|} \tag{3.6}$$

By definition, $0 \leq earliness \leq 1$, and the smaller the better.

Finally, the applicability (app) simply signifies the percentage of testing instances that could be classified by our approach. By definition, $0 \leq app \leq 1$, and the higher the better.

3.4 The objective revisited

Time series represent a popular data model, and in real-life applications, they often are multivariate. Typically in these application fields where the different events are recorded as multivariate time series, particular situations take place when specific temporal patterns over the different dimensions are observed. In real-life scientific domains, the patterns are rather complex and they may not be known, but regardless of this, domain experts will be able to recognize the situation of interest whenever it happens. Therefore they could annotate or classify the history observations with the proper situation or class.

Our proposed algorithm takes advantage of these classified multivariate time series in order to perform two novel tasks. The first one is to learn predictive complex patterns that could classify a given scenario **as early as possible**. By *complex* we specifically mean, patterns that capture the various existing correlations (synchronous, sequential, and time gaps) that might exist between the different dimensions of the multivariate time series. The second task is to seamlessly and

algorithmically transform the learned patterns into predictive CEP rules with no human intervention.

More formally, given a dataset D^d as input, the first fold of the problem is how to deliver the set of all time-annotated sequences TAS_D as output? The second fold is how to go from the learned time-annotated sequences into the CEP world in a seamless manner? Given that the earliness of the approach must be kept to minimum in order to achieve predictions, but in the same time high accuracy values must be preserved.

A post-goal then will be to let the BPM world benefit from the results of such approach. That is, within any business process, and whenever a monitorable task could be found, the approach should be employed without complexity. We stress on the fact that monitorable tasks are monitored by using different sensors, and these sensors usually emit their data as time series. Historical observations (the recorded time series) of these tasks should be classified with proper classes and situations.

Therefore the most important requirement to exploit our proposal is met for these kind of processes. That is, the existence of classified multivariate time series. What is left is the integration part, which we did swiftly as we will later discuss.

CHAPTER 4

KNOWLEDGE-DRIVEN BUSINESS PROCESS MANAGEMENT

> Success in management requires learning as fast as the world is changing.
>
> *Warren Bennis*

This chapter holds our contributions in the field of business process management. It is on the application level, and it is the least technical part compared to other chapters.

Following the path of most research efforts in the domain we consider complex event processing as one of the main enabler to achieve the goal of creating knowledge-driven processes, but we argue that data mining is the main source of knowledge. Therefore we further contribute by combining data mining techniques with CEP to devise a full approach that could be integrated easily to predict challenging violations.

Before starting to detail our added-values in this domain, we will mention the concept of *instance-based process management*, and then we will answer the question: to what extent the benefits of CEP are being exploited in the world of BPM?

1. As shown earlier, related approaches have been limited to design-time methodologies and overlooked the contexts and environments of the instantiated processes during execution. They assign (explicitly or implicitly) *model-based monitoring points* over the process model in order to track its progress, detect coarse-grained deviations, and predict end-to-end time violations. This practice helped to correlate events stemming from different sensors (like GPS) to a task of an ongoing process instance (like Trucking) [26, 25]. It also assisted to create a finer level of visibility over continuous tasks, which in turn facilitated end-to-end predictions concerning the remaining time for the whole process to

finish. Nonetheless, only considering the process at design-time is not sufficient for continuous tasks that are executed in dynamic environments, i.e., tasks where the dynamicity of their execution environments will potentially affect them. In such cases, in order to predict for challenging violations and force the agreements, each instance of the model needs to be handled differently and according to its context. Generally, current approaches fail to consider an **instance-based** management of processes and their continuous and monitorable tasks.

2. Since CEP rules are written manually, then the benefit of using CEP is limited, and this is clearly shown in current approaches. To some extent, they have been restricted to straightforward rules that monitor typical and all-purpose events (such as delays based on the current GPS location) without further analysis and advanced inferences. However if instance-based management is to be applied, then CEP needs to be used for prediction and beyond just all-purpose events.

To answer the shortcomings of other approaches, we have devised the concept of contextual templates [141, 139], and we have created a template-driven framework that is called **the Butterfly** [137]. **The Butterfly** is mainly a conceptual framework. Its contributions could be attributed to both design-time and run-time levels. The conceptual part of our proposals will be proven through a real running scenario from the domain of artwork transportation. The run-time potential of the approach is already realized, integrated with **autoCEP**, and will be demonstrated through a real scenario from the manufacturing domain.

4.1 Scenarios

As stated, two main real-world scenarios are going to be used. The artwork transportation one is going to serve as a proof of concept. Then the manufacturing scenario is going to be fully demonstrated with the help of **autoCEP**.

4.1.1 Artworks Transportation

This domain serves as a good proof for why instance-based process management is critical. Each transported piece of art is different, and it has its own capacity to resist environmental factors. For example, one object could handle temperature fluctuations, whereas another cannot. Therefore and in this domain, each instance of the transportation process needs to be managed differently and according to its context, such as the transported objects, the routes, locations, weather, etc. This is exactly what current business process management systems [14, 51] fail to consider, because they are mainly activity-based, and they support the model of a process, but not the instances.

It is important to note that almost 80% of the overall real scenarios have recorded contract violations in a way or another, as our partner in the project have indicated[1]. Therefore in this section,

[1]The C2RMF center in Paris

we show a real scenario that will point out some of the frequently encountered problems while transporting artworks.

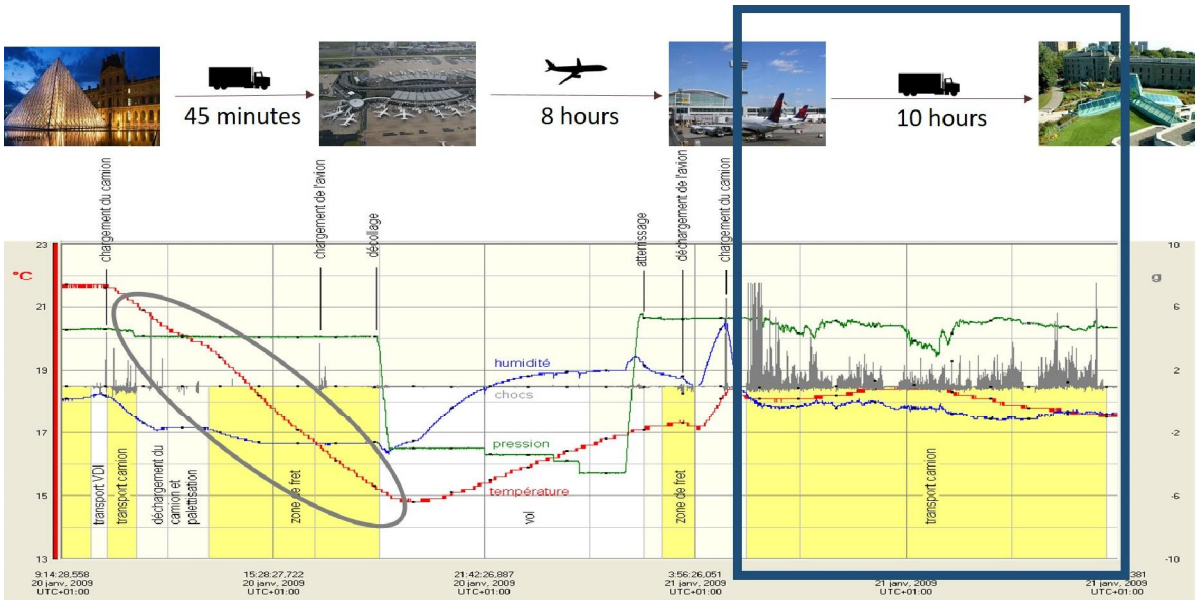

Figure 4.1: Artwork Transportation Scenario

Figure. 4.1 depicts a multi-modal transport process for an artwork that needs to be transported from the Louvre museum in Paris to the national museum in Quebec City. The plan was to go from Paris airport to New York airport and from there almost ten hours of trucking to reach the final destination in Canada. The C2RMF center usually attaches sensors on the transported piece of arts, so the conservation experts could check what happened during transportation and how much the pieces were affected by their dynamic surroundings. The lower part of figure. 4.1 illustrates these sensor readings as *time series*. On that trip, shocks, temperature, humidity, and pressure were monitored, and they were represented by the gray, red, blue, and green lines respectively.

Temperature violations were identified during the trip, i.e., going from high-temperature values to low values and bypassing a minimum threshold. This variation in temperature is highlighted on the figure and it may have long-term negative effects on the life of the transported piece. Moreover, and during the ten hours of trucking, high and continuous shocks were recorded, causing more direct and obvious damages. At the end, the painting reached the museum with some cracks and frictions on its borders, leading to contract violations and dissatisfaction.

4.1.2 Smart Manufacturing

In the manufacturing of semiconductor microelectronics, several silicon wafers need to be etched. Different sensors to capture important measurements are deployed on the machines that etch these wafers. After the creation of one silicon wafer, it is manually checked if it is normal or ab-

normal, and then the sensor data related to this specific wafer are classified accordingly.

Such manufacturing processes have rather complex models, however for the sake of demonstration we will just include the relevant parts. Figure 4.2 shows a portion of a manufacturing process. The *manufacture wafer* task is long-running and beyond the reach of the BPM engine.

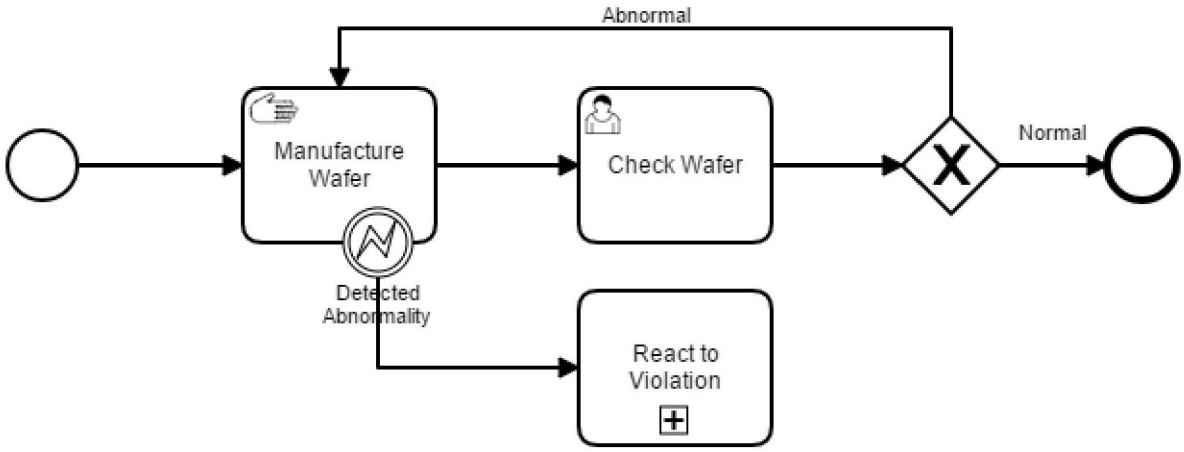

Figure 4.2: Manufacturing Business Process

In fact, typical engines in this case, just wait for a manual signal that the task has started, and another signal when it finishes. After the manufacture, an expert checks the produced wafer, if it is normal then the process reaches its end, else the *manufacture wafer* task is repeated.

Theoretically, following state-of-the-art BPM/CEP integration approaches and creating an ad-hoc CEP support, one could have the interrupting error event that is attached to the manufacture task. Thanks to this, reactive procedures could be later executed (React to Violation sub-process in figure 4.2).

In contrast what could be achieved using our approach is shown in figure 4.3. The *manufacture wafer* task could have another interrupting event, however this time for a predicted situation. Therefore, the management of the process could be carried out proactively (Proact to Violation sub-process in figure 4.3). In this example, the current batch could be dropped thus saving time and resources, or other measures could be taken to prevent the violation if possible.

The strong point to emphasize is that this is not going to be achieved through an ad-hoc support, i.e., specifically for manufacturing processes, but it could be done easily in many domains and for any number of situations. The CEP engine will work automatically and signal any violation that it is trained to predict. The only key requirement is a history data set that is classified with the different situations of interest.

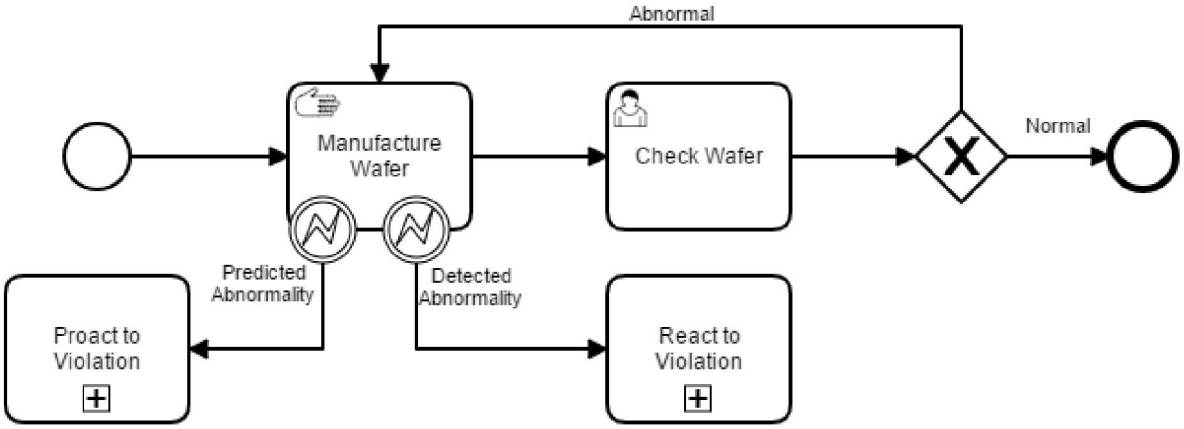

Figure 4.3: Manufacturing Business Process with Proactivity

4.2 Contextual Templates

A main concept that paves the way for a better management of monitorable and long-running activities is contextual templates. Templates are supposed to wrap such activities in order for the CEP engine to infer what should be monitored and predicted for this specific instance (it is a way of specializing the management of processes on the instance level).

Templates contain two sections, an attribute section and a prediction one.

1. **Attribute Section:** This section contains three types of attributes. The first type is *instance-based attributes*, these are static data that are useful to configure the engine, and to specialize the management of a specific instance. They are filled before the execution of the process, and they assist to provide context-aware and design-time support. The second type is *run-time attributes*, and they constitute the attributes that the CEP engine should monitor in real-time, i.e., provide real-time monitoring capabilities for BPM users. The final type contains constrained *attributes*. This group contains the Run-Time Attributes, which values may reflect signs of violations. They need to be constrained by the user. In other words, they may constitute some agreement clauses, and the framework needs to continuously predict and estimate the values of these attributes in order to give insight about violations if any.

   Figure 4.4 presents a concrete example of the attribute section of a template that could be applied to a trucking task

2. **Prediction Section:** The other part of a template is *the prediction section* where check-boxes for each situation of interest (classes) need to be provided. Then the user can check (select) what situations she is interested in predicting for this specific instance, and later on the CEP engine will make sure to predict and signal them. Figure 4.5 shows an example template for manufacturing activities. It is obvious that there are six sensors deployed on the machine, the CEP engine will see to provide real-time values from each sensor during

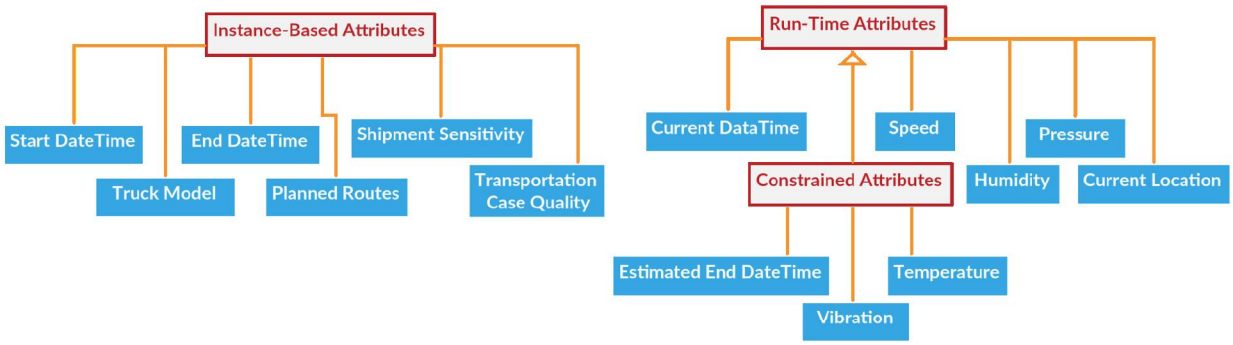

Figure 4.4: Attribute Section of a Template for a Trucking Task

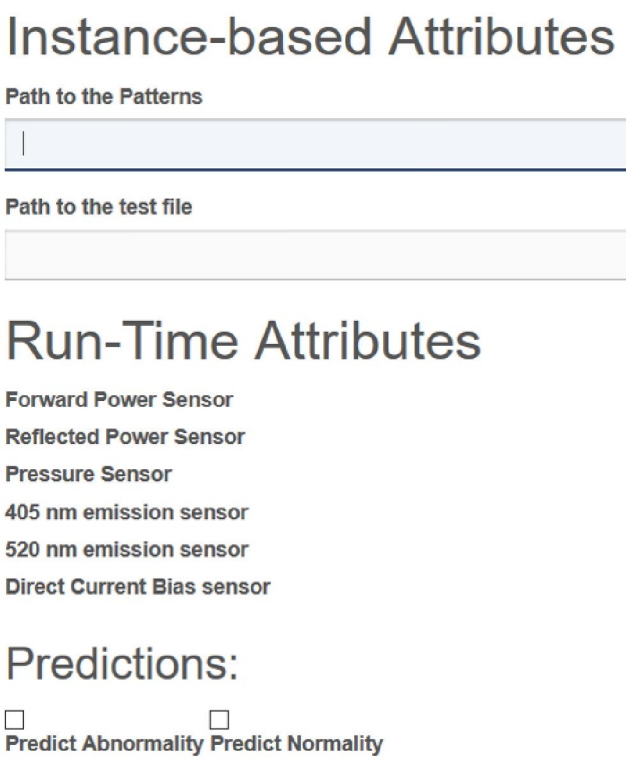

Figure 4.5: An example of a Template for the Manufacturing Task

the execution of the manufacture task. Regarding the predictions, there are two situations or classes in this case because historical wafer manufacturing scenarios are classified as normal or abnormal. Historical scenarios from different application domains may contain multiple classes rather than binary, and they all need to be written down in the prediction section of the template. Selecting some of these classes will notify the CEP engine that users are interested in predicting them, and then the CEP engine will work its magic to predict these classes in an automatic way and without any manually written CEP code.

Templates are extensible, flexible, and they can be easily created using current BPM engines. All typical engines provide an easy way (usually HTML forms) in order to create templates to assign values to process variables. A template could be associated to the start event of a process, and so it can be filled (checkboxes checked) before the long-running task is reached.
We suggest that all monitorable and long-running tasks should be wrapped with contextual templates. We envision some extensions to current business process management systems, so they could identify these tasks and trigger some template-driven frameworks to handle them whenever they are encountered in a business process model. This is going to be shown in the evaluation section for the manufacturing process.

4.3 The Butterfly

AutoCEP, as we will later discuss, will automatically take care of the prediction section of the template.

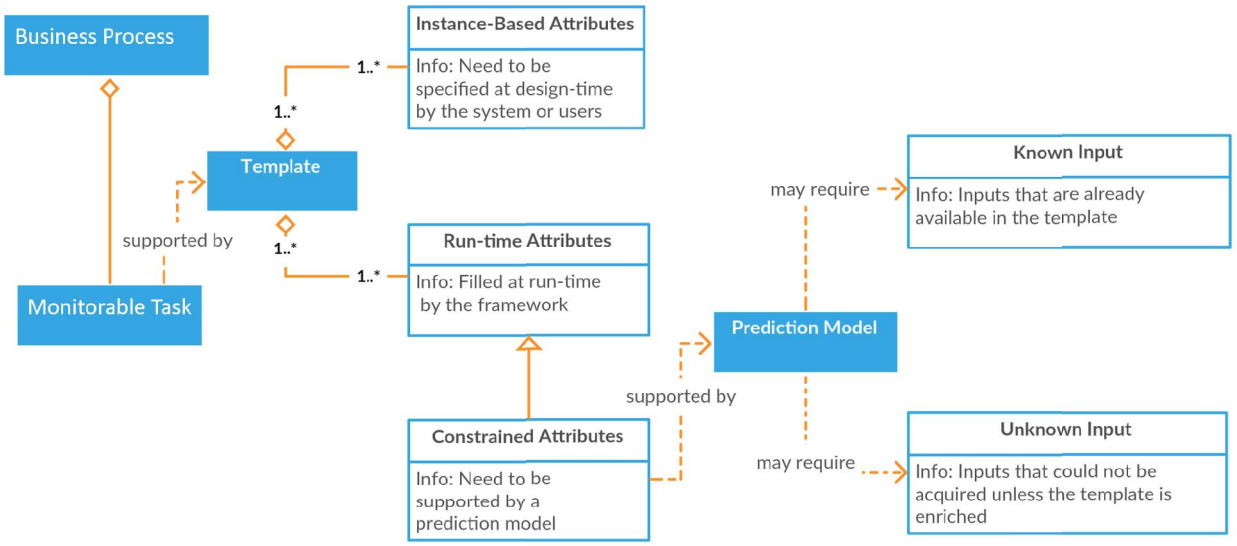

Figure 4.6: Class Diagram for the Main Concepts used in the Butterfly

For the attribute section, any kind of prediction model (given by experts or learned) could be employed. To provide a bigger picture, Figure 4.6 presents a class diagram for all important concepts related to the attribute section of the template, and it depicts the logical relationships between them as well. Obviously, a business process is composed of monitorable tasks, these tasks need to be supported by templates, and we already covered the different types of attributes. The remaining concepts are going to be explained in the remainder of the chapter.

Figure 4.7 explains the name of the framework, i.e., **the butterfly** and it also shows that the template is the cornerstone of the whole approach, because it is central and available for all the

modules to use.

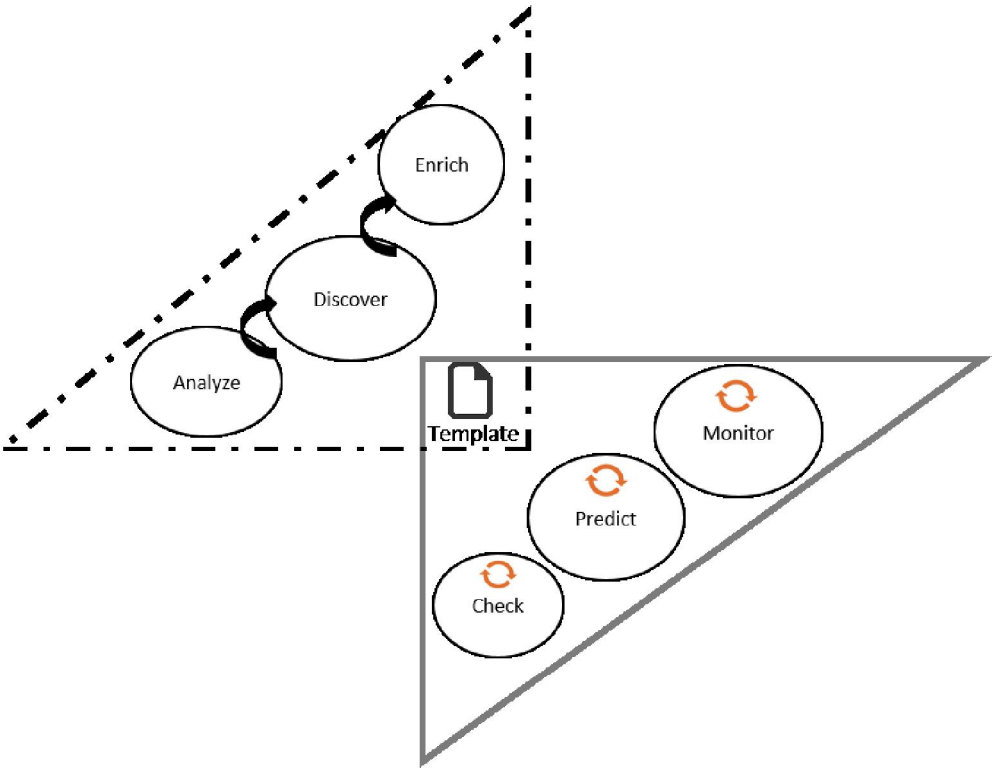

Figure 4.7: A Conceptual Representation of the Butterfly

As figure 4.7 proposes, the framework is clearly divided into two parts, the Analyze Discover Enrich (ADE) modules within the dashed wing of the butterfly, and the Monitor Predict Check (MPC) modules within the other wing.

4.3.1 The ADE Principle

The ADE are design-time modules, they operate in a sequential manner starting with the Analyze, then the Discover, and finally the Enrich.

4.3.1.1 Analyze

This module is the first to interact with the template, it analyzes its attributes in order to achieve a sole goal, which is to assess if an enrichment is needed or not. To this end, it examines the attributes that were constrained by the user for this specific instance (i.e., the constrained attributes). Each one of these attributes needs to be supported by a prediction model (i.e., to predict its values). Typically, a prediction model can be thought of as a data mining algorithm (e.g., a regression function, a support vector...) that requires a defined set of inputs, the **Butterfly** framework classifies these inputs into two categories: *known* and *unknown*. The former type

can be extracted directly from the template, however, the latter cannot and it can be only acquired through a specific type of enrichment. Figure. 4.6 describes this relationship between the constrained attributes, prediction models, and the inputs. Putting it all together and by examining the constrained attributes, the Analyze module presents our dynamic enrichment strategy, as it stresses the need for enrichment if it finds one or more required *unknown inputs*.

4.3.1.2 Discover

If an enrichment is necessary, then the Analyze module will transfer the needed unknown inputs to the Discover module where they are going to be examined. According to the examination, external services and sources - from where the needed inputs can be acquired - will be discovered. Such discovery mechanisms are already proposed in the literature [75]. At the most, a set of predefined sources can be made known and accessible to the framework, then depending on the needs of the currently running instance, the most suitable services will be invoked for the enrichment. It is important to distinguish between two types of external services that could be discovered by this module, the first one will be exploited by the Enrich at design-time (see next), and the second will be preserved for run-time usage whenever needed (e.g., real-time road status from services offered by Google maps).

4.3.1.3 Enrich

After the discovery and identification of external sources, at this stage, the actual enrichment takes place. This module will query the different necessary sources and services in order to enrich the attributes of the template. Typically, it will transform raw data in the instance-based attributes into higher knowledge that can address the needs of the prediction models (e.g., going from raw coordinates in the template into full street and city names). After this module is executed, the template will be transformed into an enriched one, carrying more meanings and semantics.

4.3.1.4 ADE in Action

To value the benefits of our proposed Analyze-Discover-Enrich technique, we will run a small demonstrative example. Taking a trucking activity with its template shown in figure. 4.4. Considering that the end-user constrained the vibration attribute, so the **Butterfly** is supposed to estimate vibration values along the planned routes. The prediction model that foresees such vibration values is a trained regression function that takes four parameters as input: the road smoothness, the road surface, the transport case quality, and the permitted road speed. Among these inputs, only the transport case quality could be known from the template, but the other three inputs are unknown. Therefore the Analyze task triggers the need to enrich. Afterward, during the Discover task, the framework examines the needed missing inputs and it concludes what external services to trigger and what attributes in the template to enrich. The planned routes attribute in figure 4.4 needs to be enriched from raw coordinates to more valuable knowledge in order to give the regression function what it expects as input (see figure. 4.8). To this end,

Figure 4.8: Transforming Raw Data into Valuable Knowledge

external sources and services[2] are queried and the attribute is enriched, to finally estimate vibration values over roads even before the execution of the process, which is considered as a very beneficial move for the risk assessment phase. Figure. 4.9 shows an example of the output, the red section that is highlighted on the road alerts about expected high vibration values on this specific route.

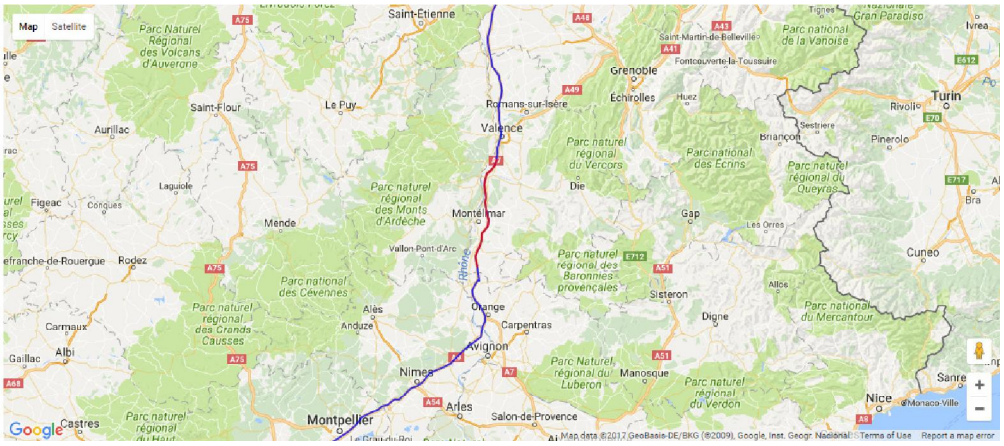

Figure 4.9: Expected Vibration Values

4.3.2 The MPC Principle

Differently from the ADE, the modules in this wing are dedicated for run-time usage, and they are not executed in a sequential manner but they are required to continuously and simultaneously perform their operations as long as the process is still running. This is why the small looping arrows in the MPC wing (fig. 4.7) exist. The MPC modules have access to the enriched template, they can make use of it and update it as needed.

[2] http://www.openstreetmap.org/

4.3.2.1 Monitor

The Monitor module incarnates the functionality of the CEP engine. It receives different real-time events from internal sensors and external services, correlates them, and finally delivers them to the corresponding listeners to further deduce inference and knowledge. One of the main duty of this module is to continuously update the run-time attributes (including the constrained ones) in the template with their proper values.

4.3.2.2 Predict

This module needs to continuously predict future values for the constrained attributes. This is done by employing the various prediction models that are associated to this kind of attributes in the first place (fig. 4.6), the ADE modules made sure that all the unkonwn inputs will be available at this stage. The method of prediction is not limited to a specific set of models, and it depends on the target attribute itself, for example if it is numerical or categorical, periodic or not... In general, various data mining algorithms exist to suit the different requirements. Examples of prediction models are regression functions, shapelets [207], CEP rules, etc.

When it comes to the prediction section of the template, **autoCEP** is in charge of predicting all global classes before they occur. This part is going to be discusses at the end of the **autoCEP** chapter (chapter 6)

4.3.2.3 Check

As the name suggests, this module will check the constrained attributes for violations in a continuous fashion. It is necessary to distinguish between two types of violation detection. In case the Monitor module updated a constrained attribute with a violated value (i.e., out of the accepted range), then the detection in this case is reactive and the violation is caught as it happens in real-time. Whereas if the Predict module estimated an unaccepted value for one of the constrained attributes, then the detection is completely predictive, and a violation is signaled before it happens.

4.4 Summary

In this chapter, we introduced a novel framework that could address the gap between processes with monitorable tasks and information systems. Its main goal is to extend current model-based methodologies, in order to reach a finer level of processing, and specialize the handling of each case according to its context. We proposed the usage of contextual templates that go beyond design-time monitoring points to exercise more advanced support for continuous and monitorable tasks.

The templates are broadly composed of two sections. The first contains the attributes. These attributes could be used for different purposes as we explained. They are exploited for design-time supports and enrichment, for real-time monitoring, and for customized prediction, in case

one of the real-time attributes is to be anticipated. The second is the prediction section that contains global classes to be predicted.

The overall **Butterfly** framework is discussed on the conceptual level with the artwork transportation as a running scenario. The integration with **autoCEP** is going to be presented later.

CHAPTER 5

USE & SEE

Change is the end result of all true learning

Leo Buscaglia

As hinted in the introduction, we proposed a two-step approach to answer the problem of learning time-annotated sequences from multivariate time series. The first step is an algorithm called **USE**, it stands for **U**nivariate **S**hapelet **E**xtractor. At **USE**, the algorithm takes D^d as input and delivers the set of all shapelets $\hat{S}_D$ as output, therefore **USE**$(D^d) = \hat{S}_D$. Afterward, $\hat{S}_D$ is fed to the second algorithm, **SEE** that symbolizes **SE**quence **E**xtractor, and the set of all time-annotated sequences TAS_D is returned as the final output, thus **SEE**$(\hat{S}_D) = TAS_D$. Note that the two algorithms may take more that one parameter as we will shortly discuss. For clarity reasons, the logic behind **USE** and **SEE** are respectively depicted in figure 5.1 and figure 5.2 in a demonstrative manner. The details will be discussed in the upcoming two sections.

It is important to note, that the learned sequences may have different lengths. So if our proposal is confronted with a multivariate training set D^d, it is not obliged to produce d shapelets, one from each dimension, as current state-of-the-art multivariate shapelet algorithms do [115, 65, 66, 151]. In fact, some authors proved that even in multivariate environments, extracting shapelets from one dimension could be enough to deliver good results [158, 208]. While other researchers insisted on the fact that in real world applications, patterns are complex mixtures of shapelets over more than one dimension [66, 151]. We argue that we do not know in advance and we let the algorithm decides on the number of shapelets. Therefore, our approach breaks the ice by learning rules with the minimum sufficient number of shapelets, whether it is only one or more.

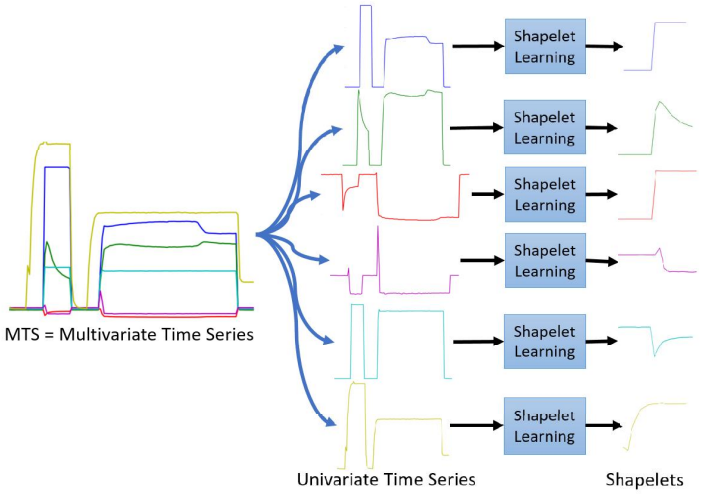

Figure 5.1: The USE Algorithm

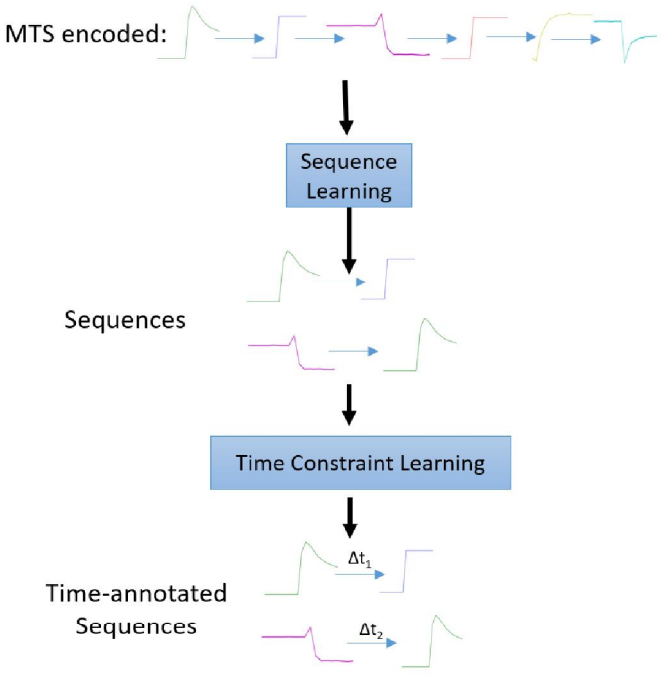

Figure 5.2: The SEE Algorithm

5.1 USE

This stage contains our algorithm that is in charge of extracting highly useful shapelets by learning them from historical multivariate time series. The algorithm, written in listing 2, takes only the training data set D^d as a mandatory parameter and all other parameters are optional. Basically, the **USE** could be deemed as parameter-free, which is an advantage over other learning algorithms especially for non-expert users. On the other hand, the algorithm is parametrized, i.e., all optional parameters have values by default, yet they could be changed. The thing that could be a plus for experts in order to integrate their knowledge in the learning process.

Algorithm 2: The USE Algorithm

Input: D^d, $minLength = 10$, $maxLength$ = None, $distanceMeasure$ = brute, $pruning$ = cover, $k = 10$
Output: $\hat{S}_D$

$\hat{S}_D \leftarrow \emptyset$
if $maxLength = None$ **then**
    $dT \leftarrow D^d[1]$
    $maxLength = |dT|_{min}$
$1D^1, 1D^2, ..., 1D^d \leftarrow$ divide(D^d)
for *i in range(d)* **do**
    $\hat{S}' \leftarrow$ shapeletExtractionAlgorithm($1D^i$, $minLength$, $maxLength$, $distanceMeasure$)
    $\hat{S}_D \leftarrow \hat{S}_D \cup \hat{S}'$
$\hat{S}_D \leftarrow$ pruneShapelets($\hat{S}_D$, D^d, $pruning$, k)
return $\hat{S}_D$

5.1.1 Parameters Explained

The $minLength$ and $maxLength$ parameters could be specified by domain experts in case there are any preferences or prior knowledge about the desired length of shapelets (i.e., they desire to learn shapelets between these lengths). Specifying the minimum and the maximum boundaries may guide the learning process and dramatically decrease the learning time. However if no such preferences or knowledge exist, then users may leave the default values shown in listing 2, and the algorithm will learn the shapelets of the best length by its own.

The $distanceMeasure$ specifies the strategy to follow while calculating distances between time series. By default, it is the brute force strategy that searches all time series subsequences (within $minLength$ and $maxLength$) for similarities using the Euclidean distance. This technique is highly adopted to search for time series shapelets [135, 207, 115, 65, 143] as it yields the most accurate results. Another value that could be taken by this parameter is 'MASS', which is a fast technique to calculate similarities. It is adopted in signal processing domains, and it was

recently exported into the time series data mining fields. It is based on the Fast Fourier Transform. 'MASS' is implemented into our algorithm, it was briefly discussed in the background chapter 2.1.1.2, but to obtain more details, readers are invited to check [208]. The final value that could be provided to this argument is 'DTW', which stands for the technique of dynamic time warping.

Lastly the $pruning$, as its name suggests, identifies the pruning strategy to follow. It could take two values, the default value is 'cover', which is an algorithm that we have proposed and will discuss next. The other value is 'top-k' to select the typical top-k pruning algorithm, and the remaining k parameter exists to serve this matter. The evaluation chapter will shed some light on the impact of these parameters on the performance of the approach 7.

5.1.2 Algorithm Explained

The explanation of algorithm 2 is straightforward. First if $maxLength$ is not defined then it is set to equal the shortest time series in the training set. At line 5, the D^d is divided into d training sets with univariate time series. Therefore, all time series related to one dimension are grouped into one data set. It is important to note that a UTS gets the class of its parent MTS. In other words, given a $dT = \{T^1, T^2, ..., T^{d-1}, T^d\}$ and $class(dT) = c$, then $\forall i \in \{0...d\}\ class(T^i) = c$. Afterward, the learning is executed on all the obtained data sets (more details in section 5.1.2.1), and the resulted shapelets are gathered in $\hat{S}_D$. The final step is to prune the large set of extracted shapelets, and select only the ones that have high utility scores (detailed information are included in section 5.1.2.2).

5.1.2.1 Shapelet Extraction Phase

Algorithm 3 gives some insight on how the shapelet extraction process is performed. The parameters are already explained. As for the logic: First, it loops over the training data set $1D$, then from each time series T and for the specified minimum and maximum boundaries, it extracts a subsequence using the *getSubsequence* function. This function takes a time series T, a starting position k, a given length l, and it simply returns a subsequence s of the time series that starts at k, so $s[1] = T[k]$ and $|s| = l$. Basically, $s = \{T[k], T[k+1], ..., T[K+l-1]\}$.

Afterward, an array of similarities sim is calculated. This array represents the distances between the extracted s and all time series in the training data set. If the distance measure is set to 'brute', then the similarity array is calculated using the Euclidean distance as shown in equation 5.1.

$$\forall i \in \{1, ..., |1D|\},\ sim[i] = ||s, T_i||_{bmd} \tag{5.1}$$

Else if the value is set to 'MASS', then the technique of applying the Fast Fourier Transform, proposed by the authors in [208], is executed to calculate the similarities.

With sim at hand, δ could be calculated by following different methods like the Chebyshev's inequality [203] or the information gain split [143]. Our *calculateDelta* function adopts the latter

Algorithm 3: The Shapelet Extraction Algorithm

Input: $1D$, $minLength$, $maxLength$, $distanceMeasure$
Output: A set of shapelets $\hat{S}$

```
1  Ŝ ← ∅;
2  for T ∈ 1D do
3      for l in range(minLength, maxLength) do
4          for k in range(1, |T| − l + 1) do
5              s ← getSubsequence(T,k,l);
6              sim ← calculateSimilarities(1D, s, distanceMeasure);
7              δ, score ← calculateDelta(1D, sim, s);
8              ŝ ← (s, δ, c_s = class(T), score);
9              Ŝ.add(ŝ);
10 return Ŝ;
```

approach. It takes the training data set, the array of similarities, and the sequence s as parameters. Afterward, it searches for the best distance value that could split the distances in sim in a way that assorts time series belonging to the same class as s in one group, and other time series in another group. More technical details about this technique exist in the appendix A, but basically, the algorithm tries different candidate thresholds, then it takes the one that maximizes the information gain as the selected distance threshold. As shown at line 7, the algorithm also returns the maximum gain as a utility score (we favor shorter subsequences to break ties). In other words, the $score$ variable refers to how much information we gained from using this specific s. Finally, a candidate shapelet is constructed at line 8 and it is added to $\hat{S}$. These steps are repeated for every subsequence within $minLength$ and $maxLength$ in $1D$, before $\hat{S}$ could be eventually returned.
The information gain formula is discussed in appendix A, but for completeness purposes we will discuss it here. The gain is dependent on the entropy of the dataset. If D is a dataset, and its instances are classified with labels from C, d_c refers to the number of instances classified with the class c, then the entropy is calculated as:

$$Entropy = -\sum_{c \in C} \frac{d_c}{|D|} log(\frac{d_c}{M}) \tag{5.2}$$

We calculate the information gain of splits as we discussed earlier, so let us suppose that the dataset D will be divided into two sets, D_l to the left of the split and D_r to the right. With these notions, the information gain is calculated as:

$$InformationGain = Entropy(D) - \frac{|D_l|}{|D|} Entropy(D_l) - \frac{|D_r|}{|D|} Entropy(D_r) \tag{5.3}$$

5.1.2.2 Pruning Phase

At this stage, we will have an enormous amount of candidate shapelets, of which only few may be highly important. To this end, the pruning algorithm (algo 4) is essential. The first step of this

algorithm is to divide the data set and the big list of shapelets into different chunks grouped by the class and the dimension altogether. In other words, for each combination of (class, dimension) there exist a data set of time series and a list of related shapelets. E.g., for (class = $c1$, 1^{st} dimension) there exists one, for (class = $c2$, 1^{st} dimension) there is another, the same could be said for (class = $c1$, 2^{nd} dimension), and so on...

Algorithm 4: The Prune Shapelets Algorithm

Input: A set of candidate shapelets $\hat{S}'$, D^d, $pruning$, k
Output: A set of pruned shapelets $\hat{S}$

```
1  Ŝ ← ∅;
2  list_D ← groupByDimensionAndClass(D^d);
3  list_Ŝ' ← groupByDimensionAndClass(Ŝ');
4  for D ∈ list_D and Ŝ'' ∈ list_Ŝ' do
5      Ŝ'' ← sortByScore(Ŝ'');
       // sort in decreasing order
6      if pruning = top-k then
7          Ŝ ← Ŝ ∪ Ŝ''[1..k];
8      if pruning = cover then
9          for ŝ ∈ Ŝ'' do
10             for T ∈ D do
11                 if all D is covered then
12                     Continue from line 5;
13                 if ŝ ∈ T then
                       // definition 3.1.6
14                     Check T as covered;
15                     if ŝ ∉ Ŝ then
16                         Ŝ.add(ŝ);
17 return Ŝ;
```

The algorithm then loops over these groups, and it sorts the set of shapelets $\hat{S}''$ depending on the utility score, in decreasing order. If the pruning strategy is set to top-k, then the first k elements from each group are simply added to $\hat{S}$.

On the other hand, if 'cover' is specified as a pruning strategy, then the algorithm performs nested loops. The outer loop is over the shapelets in $\hat{S}''$, and then for each shapelet $\hat{s}$, it loops again over the time series in D. At line 11, it checks if all time series in D are already covered, if so then it continues the outermost loop at line 5. Else, the algorithm checks if $\hat{s}$ is contained in T, if so T is highlighted as covered and $\hat{s}$ is added to the final set of shapelets. Basically, in the cover strategy, the algorithm starts to test shapelets with high scores first, and all covered time series by the selected shapelet are ruled out when testing the other shapelets in $\hat{S}''$. The

logic continues until no time series are left to test (i.e., they are all covered) or there are no more shapelets to try.

It is important to highlight the grouping mechanism that we did at the beginning of the pruning algorithm. We intentionally did not search for false positives and negatives when testing shapelets, as we plan to obtain conclusive discrimination (accuracy) in the **SEE** algorithm, when various shapelets from different dimensions are combined to build accurate rules.

5.2 SEE

After executing the algorithms described in section 5.1, a set of useful shapelets $\hat{S}_D$ will be available at this level. This set of shapelets in addition to the original data set are the only mandatory inputs for the **SEE** algorithm, shown in listing 5. The $minAcc$ and the $maxEarliness$ parameters are the accuracy and the earliness thresholds, and they are set to 80% and 50% respectively. They are calculated using the formulas discussed in section 3.3. These settings could be changed by the user, and by default the algorithm searches for sequences with a minimum of 80% accuracy and a maximum of 50% earliness. The last parameter n specifies the number of shapelets from each dimension that are allowed to be found in a sequence. More details on this parameter can be found next. The output of the **SEE** algorithm is the set of all time-annotated sequences TAS_D.

5.2.1 The Algorithm Explained

The algorithm first starts by creating an empty set of rules. Then it calls the *encodeWithShapelets* method. Briefly, this method transforms a given multivariate time series dT into a sequence of shapelets, listed as they appear in dT. Figure 5.3 shows a demonstrative example. For a given time series $3T$ and three shapelets $\hat{s}^1$, $\hat{s}^2$, and $\hat{s}^3$, $3T$ is encoded as ($\hat{s}^2$, $\hat{s}^1$, $\hat{s}^1$, $\hat{s}^3$, $\hat{s}^2$). It is the order of appearance of the three shapelets in $3T$. For clarity, in algorithm 5 and for a given MTS, dT represents the unchanged format of a multivariate time series, and $enc(dT)$ refers to the encoded format. It is important to note that the time series and the encoding shapelets need to belong to the same class, $\forall \hat{s} \in enc(dT),\ class(\hat{s}) = class(dT)$. In other words, shapelets with different classes than the time series, can not be part of the encoding for this specific time series.

In addition to the list of shapelets and the data set, the *encodeWithShapelets* method takes a third parameter, which is the n. As hinted earlier, it limits the number of shapelets from each dimension that are allowed to appear in a given encoding. By default, it is equal to one, so only one shapelet from each dimension is permitted to appear. To clarify with the same example in figure 5.3, without the parameter, it is evident that $3T$ = ($\hat{s}^2$, $\hat{s}^1$, $\hat{s}^1$, $\hat{s}^3$, $\hat{s}^2$), but with $n = 1$, then the algorithm neglects multiple appearances of shapelets from the same dimension and it keeps only the earlier occurrence, therefore $3T$ becomes encoded as ($\hat{s}^2$, $\hat{s}^1$, $\hat{s}^3$), i.e., keeping one shapelet from each dimension. n can take any integer, if it is less than one, then the encoding of the multivariate time series is left as is, without disregarding any shapelet. If it is larger or equal

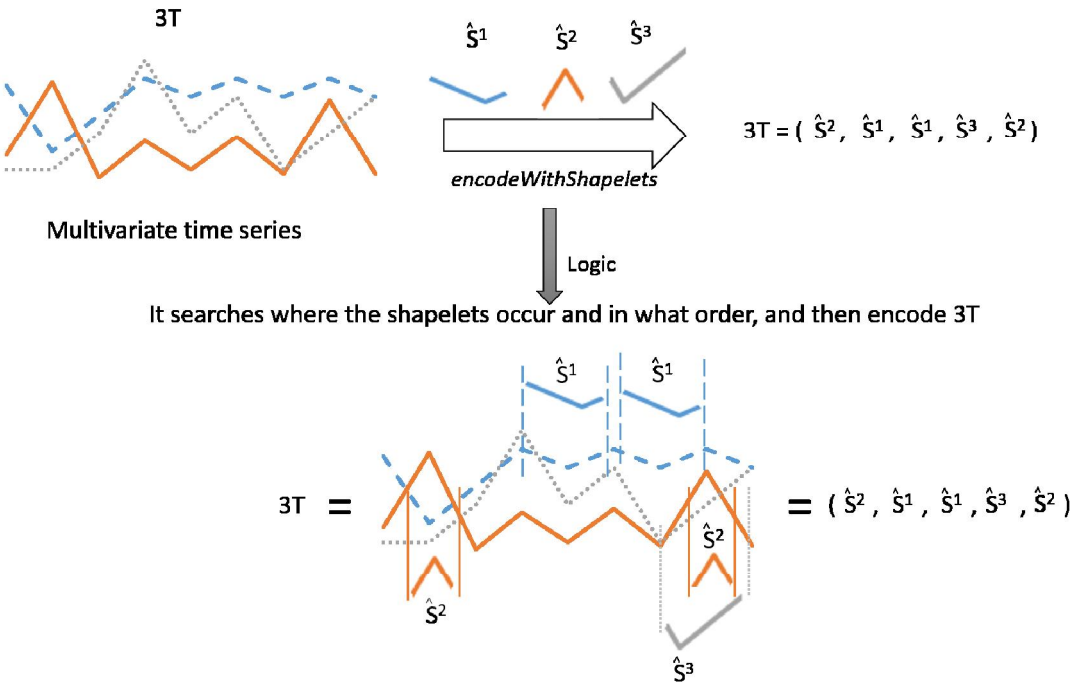

Figure 5.3: The Encode with Shapelets Method

to one then the first n shapelets are kept and the others are dropped out. From a performance perspective, and in order to learn rules with the best earliness percentages (as shown in the evaluation section), we recommend that this parameter is kept to the default. Unless and for exceptional reasons, advanced users are searching for sequences with more than one shapelet from each dimension.

The algorithm then loops over the time series in the training data set, and at line 4 it gets all possible permutations (of all lengths) for a given encoded time series. E.g., for dT encoded as $(\hat{s}^1, \hat{s}^2)$, permutations$(enc(dT)) = \{(\hat{s}^1); (\hat{s}^2); (\hat{s}^1, \hat{s}^2); (\hat{s}^2, \hat{s}^1)\}$. The set of all permutations is stored in the variable $perms$, afterward, the algorithm loops over this variable, and for each sequence seq in it, it performs the following:

1. First it checks if seq is already tested, if it is, then it is skipped, and the code continues to test the next sequence in $perms$.

2. Else if seq is new, then it is tested for being prefixed with another accepted sequence, if so, then seq is dropped, and the algorithm jumps to the next sequence. Since we are searching for earliness, if a sequence with only $\hat{s}^1$ is accepted, then it makes no sense to take a sequence like $(\hat{s}^1, \hat{s}^2)$ (i.e., since the occurrence of only $\hat{s}^1$ is enough). This helps us to learn rules with the minimum needed number of shapelets.

3. If seq surpassed the aforementioned two tests, then it is tested for accuracy. It is important to note that at this stage the algorithm does not use time constraints, but tests only

Algorithm 5: The SEE Algorithm

Input: $\hat{S}$, D^d, $minAcc = 80$, $maxEarliness = 50$, $n = 1$
Output: The set of all time-annotated sequences TAS_D

```
1  TAS_D ← ∅
2  D^d = encodeWithShapelets(D^d, Ŝ, n)
   /* After this line of code, each time series in D^d is encoded with a
      sequence of shapelets. And all shapelets that encode a given time
      series dT, they belong to the same class as dT by definition.     */
3  for dT ∈ D^d do
4      perms ← permutations(enc(dT))
5      for seq ∈ perms do
6          if seq already tested then
7              Continue
8          if seq already prefixed then
9              Continue
10         if acc(seq) ≥ minAcc then
11             W ← getTimeWindows(seq, D^d)
12             TAS ← (seq, W, class(seq))
13             if earliness(TAS) ≤ maxEarliness then
14                 TAS_D.add(TAS)
15 return TAS_D
```

the ordering. If the accuracy test succeeds, then the code learns time constraints as discussed in section 5.2.2, and builds complete time-annotated sequences. Afterward, it tests the earliness. Finally, accepted rules are added to TAS_D, which is returned at the end. The accuracy and earliness are computed as shown in section 3.3.

Even thought some minor codes are not shown in algorithm 5 for clarity reasons, they deserve to be explained in order for the reader to obtain a complete picture. First, it is important to recall that shapelets that constitute a single sequence (seq at line 5), they all belong to the same class, because $enc(dT)$ contains shapelets from the same class by definition. Second, it is necessary to highlight that the list of already tested and accepted sequences, which are used for checking at line 6 and 8, are different for each class. E.g., at line 6, the algorithm does not check a sequence that belongs to class c_1 with the already tested sequences that belong to class c_2.

5.2.2 Learning the Time Gaps

Before this method is being called at line 11 of listing 5, the algorithm has at hand a set of shapelets that constitute accurate sequences but with no time constraints. Therefore, this method learns the time windows for each sequence, in order to complete it. To find the different time windows between shapelets, the algorithm searches for multivariate time series dT where the

sequence seq is found, given that $class(dT) = class(seq)$, then it calculates the time steps between the different shapelets of the sequence. At the end, the maximum time steps that could be found are considered to be the selected time window. For example, given two time series dT_1 and dT_2 and a sequence $seq = (\hat{s}_1,\ \hat{s}_2)$, with $class(dT_1) = class(dT_2) = class(seq)$: For dT_1, $\hat{s}_1$ appears at time point x and $\hat{s}_2$ appears at $x+10$, therefore the time window is equal to 10. For dT_2, $\hat{s}_1$ appears at time point y and $\hat{s}_2$ appears at $y+15$, thus the detected time window is 15. As a result, $w = \max(10,15) = 15$ is selected. This logic is formally detailed in listing 6.

Algorithm 6: The Get Time Windows Method

Input: A sequence seq, D^d
Output: W

```
1  W ← ∅
2  if |seq| = 1 then
       // If there is only one shapelet in the sequence
3      return W
   // Initialize W to 0
4  for dT ∈ D^d do
5      if seq ∈ dT and class(dT) = class(seq) then
6          for (ŝ, ŝ') ∈ seq do
               // Take two consecutive shapelets from seq
7              candWindow ← timePoint(ŝ') − timePoint(ŝ)
8              if candWindow > W_(ŝ,ŝ') then
                   /* W_(ŝ,ŝ') means the time window that is specific for these
                      two shapelets (between them)                              */
9                  W_(ŝ,ŝ') ← candWindow
10 return W
```

5.3 Summary

In this chapter, we introduced **USE & SEE**, a novel two-step algorithm to tackle the problem of rule discovery and early classification over multivariate time series. The presented approach exploits time series shapelets to build accurate rules and to address the challenging problem of rule-based prediction. Therefore, we crossed the barrier where many researchers have stopped in the realm of multivariate shapelets discovery, and we built complete rules that support a greater number of application domains where sequence and time constraints are important. Basically, our approach is generic to be used in univariate and multivariate settings, and more importantly in normal and strict domains where time and sequence matter.

CHAPTER 6

AUTOCEP

Life is really simple, but we insist on making it complicated.

Confucius

The overall goal of our work is to realize a complete framework that would allow for the CEP technology to be easily employed in predictive applications, and yet the solution needs to be generic and used in as many domains as possible. Therefore we seek a solution with two requirements: Generic and predictive. By automatically learning and deploying accurate and predictive CEP rules, a major step towards the solution could be taken, and in order for this automatic learning to be carried out, we bridge a gap between data mining and complex event processing.

After some research in the area of predictive analytics, we found that Early Classification on Time Series (ECTS) techniques are of utmost importance to address our needs. First, on the architectural level, learning patterns in ECTS is a design-time phase and the CEP is a run-time technology so they complement each other. Second, time series are a popular data model that exists in almost every application domain, and by exploiting the fact that time series are the same as timestamped events, the combination of the two fields would support the fulfillment of the generic solution requirement. Lastly, following an **early** classification style will eventually yield rules with predictive capabilities, since by definition the approach learns (mini) patterns that classify a situation given just the minimum set of observed events.

Figure 6.1 hints at the overall architecture till this point. After **USE & SEE** we will have a set of time-annotated sequences at hand. The next step, and at run-time, these learned sequences (TAS_D) will be transformed on-the-fly into CEP rules which in turn will be enrolled into a CEP engine.

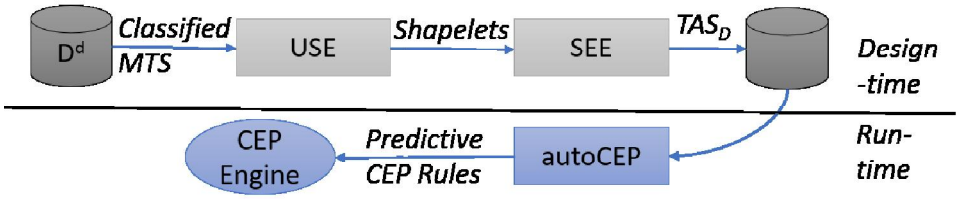

Figure 6.1: The Overall Architecture

6.1 autoCEP with univariate time series

Before getting into the final version of our CEP. It is important to note that this project supported only univariate time series [135, 136, 138] when it was first developed, and it didn't count on **USE & SEE**. However, and as we project to deprecate this early version, we decided to move it out of the flow of this dissertation and put it in appendix A.

AutoCEP with univariate time series used shapelet techniques as well, and it was a generic approach that could be used in different domain. However, the shortcoming is that it deals with univariate time series, and we argue that this is not completely suitable for real life applications. It was tested with artwork transportation processes and it showed good results. Interested readers are encouraged to take a look at the mentioned appendix where all details are presented.

6.2 autoCEP with multivariate time series

The learning process is concluded at this stage and the output is a set of accurate time-annotated sequences (TAS_D). The main task of **autoCEP** [140] is to transform these sequences into deploy-ready CEP rules. In fact, **autoCEP**—behind the scenes—performs the following: creation of a CEP engine, configuration of the engine, transformation of TAS_D into complex CEP rules CR_D, and enrollment of the rules into the engine. The result will be an engine that is ready for real-time predictions. The only required input of **autoCEP** is the learned time-annotated sequences, therefore **autoCEP**$(TAS_D) = CR_D$.

Algorithm 7 sheds some light on the reasoning behind **autoCEP**, and how it transitions from pure data mining into the CEP world. In this listing we kept away the details of EPL just to demonstrate the logic first, and then we will show each EPL query afterwards. We also omitted the details of configuring and creating the CEP engine as shown in the first comment. The only significant configuration that needs to be known is that events always have a timestamp attribute called *time* and the stream of events has a global name, i.e., *stream*.

What is important to realize from algorithm 7 is that **autoCEP** creates a named window and a complex CEP rule for each time-annotated sequence (The outer for loop line 3 and line 6), whereas it creates a simple CEP rule for each shapelet within any given TAS (the inner for loop

line 5).

The logic goes like this: For any given time-annotated sequence (e.g., TAS_1), the shapelets that are related to this sequence are transformed into simple CEP rules. These created rules fire their results (composite events), whenever their patterns match, into the unique named window (e.g., NW_1) that belongs to this specific TAS_1 (created at line 3). In the meanwhile, a complex query CR that is related to TAS_1 (created at line 6) will be listening on the appointed named window (NW_1). This complex query captures the patterns—in EPL jargons—that are required by any given TAS (the sequence and time constraints). Therefore, CR processes the composite events emitted by the different simple CEP rules that are hooked to NW_1, and whenever the pattern occurs **autoCEP** could trigger a prediction—stating that the stream is predicted to have this specific class (or this situation is about to happen).

Algorithm 7: The autoCEP Algorithm

Input: TAS_D
Output: The Complex CEP Rules CR_D
`// Engine and Events Configuration`
1 $CR_D \leftarrow \emptyset$
2 **for** $TAS = (\hat{Sh}, W, c) \in TAS_D$ **do**
3 $NW \leftarrow$ createNamedWindow($id(TAS)$)
`// Create a unique named window for` TAS
4 **for** $\hat{s}^i = (s, \delta, c_s, score) \in \hat{Sh}$ **do**
`/* i refers to the dimension from where` $\hat{s}$ `is extracted */`
5 createSimpleCEPRule(i, s, δ, NW)
`/* The Created CEP Rule emits composite events to the specific` NW `*/`
`/* What remains is to create the query on the` NW `in order to construct the complex CEP rule */`
6 $CR \leftarrow$ createComplexCEPRule($\hat{Sh}$, W, c, NW)
7 $CR_D.add(CR)$
8 **return** CR_D

6.2.1 Named Window Creation

The EPL code behind the *createNamedWindow* method in algorithm 7 line 3 is listed below. Algorithm 8 shows the EPL syntax to create a named window. As mentioned before, a unique window is created for each time-annotated sequence. The name of the window is *rule* + *id of the sequence* (e.g., if $id = 1$ then the name of NW will be *rule1*). As shown in the code, we create three columns as the structure of our named windows. The first column is named *startTime* and it refers to the starting time of a pattern. The second *endTime* is just the opposite, i.e., the

Algorithm 8: The createNamedWindow method

Input: id
Output: A named window NW
// In EPL Syntax
1 query ← "**CREATE WINDOW** rule" + id + "**.win:keepall()**
2 (startTime int, endTime int, dim String)"
3 NW ← engine.registerQuery(query)
4 **return** NW

end time of a pattern. Finally, the *dim* is a string that represents the dimension. Recalling from the explanation of the **autoCEP** algorithm (algo. 7) that each simple CEP rule created for the shapelets will be hooked to a named window just like the one shown in algorithm 8. Therefore, each simple CEP rule and whenever its pattern matches, is expected to register its starting time, ending time, and the dimension over which it operates into the named window. This is done through the EPL syntax discussed next.

6.2.2 Simple CEP Rule Creation

Algorithm 7 makes it clear that for each shapelet in a given time-annotated sequence, a simple CEP rule is created. To do that, the following EPL syntax is called (algo. 9). Some attributes and defined names used in this listing are configured during the initialization of the engine in algorithm 7, they are *time* (line 2 and 3) and *stream* (line 4). The input to algorithm 9 are self-

Algorithm 9: The createSimpleCEPRule method

Input: $dimension$, s, δ, NW
// In EPL Syntax
1 query ← "**INSERT INTO** NW
2 **SELECT prior**($|s|-1$, time) **AS** startTime,
3 time **AS** endTime, $dimension$ **AS** dim
4 **FROM** stream**.win:length**($|s|$)
5 **HAVING count(*)** = $|s|$ **AND**
6 $||$**window(*)**, $s|| \leq \delta$"
7 engine.registerQuery(query)

explanatory if they are tracked back to algorithm 7. *dimension* is the name of the dimension that the shapelet belongs to. s is the subsequence (UTS) that constitutes the shapelet. δ is the distance threshold and NW is the named window.
The EPL statement in algorithm 9 is a simple CEP rule with the three blocks discussed in the definitions: **SELECT**, **FROM**, and **HAVING**. The addition is the **INSERT INTO** at the beginning that inserts the selected information (within the **SELECT**) into the named window NW. Basically, NW expects three inputs per row (*startTime, endTime*, and *dim*) as discussed before, and this is exactly what the **SELECT** construct extracts from the stream of events. First the *startTime* is

extracted by using yet-another EPL function called **prior**, this function helps to yield the timestamp of the first event in the window. Second, the *endTime* is obtained by simply getting the timestamp of the last event in the window (for *startTime* and *endTime* the *time* attribute of the *stream* is used). The final input expected by NW is the dimension and it is transferred directly from the parameter called *dimension*. The **FROM** simply specifies the stream to process (it is always called *stream*), and a sliding window that is equal to the length of the concerned shapelet is created over that stream (line 4). Finally, the conditions that need to be met on the captured events within the window are expressed in the **HAVING** clause. The first condition (line 5) is that we want for the number of events in the window to be the same as the number of observations in the shapelet, this condition prevents unnecessary calls to the function that calculates the distance (i.e., only when the lengths are the same, then calculate the distance). The second condition is the most important, it effectively calculates the Euclidean distance between the events within the window and the shapelet, in case the distance is smaller or equal to δ then a record (*composite event* of the form *startTime*, *endTime*, *dim* as instructed by **SELECT**) is inserted into NW.

It is important to highlight that through object-oriented reflective programming and based on the dimension names, when the events captured within the window (**window(*)**) are transmitted to the distance function, the events themselves are multivariate time series however only the distance between the valid dimension and the shapelet is calculated.

6.2.3 Complex CEP Rule Creation

At this point, whenever a simple CEP rule matches, it will output a *composite event* into the appointed named window. Therefore, a more complex CEP rule is needed to listen on each named window, to search for patterns among the various *composite events* emitted by the different simple CEP rules, and to draw predictions when patterns are fulfilled. The code listed in algorithm 10 demonstrates how this could be done in EPL. The inputs to this algorithm are: the set of shapelets alongside the set of time windows that constitute the TAS in question (the outer for loop in algo. 7), the class of the sequence, and the named window NW. This function creates the complex rule, registers it into the engine and then returns it.
In order to present the whole picture in a clear way, we will explain the code of algorithm 10 through an example.

Example 6.2.1. Supposing that we have the following $TAS_1 = \{\hat{Sh}, W, c_1\}$ with $\hat{Sh} = (\hat{s_1}^2, \hat{s_2}^3, \hat{s_3}^1)$ and $W = (10, 15)$. This basically means that the shapelet named $\hat{s_1}$ from the second dimension needs to be followed by $\hat{s_2}$ from the third dimension within 10 time steps, and finally $\hat{s_3}$ from the first dimension has to appear within 15 time steps after the start of $\hat{s_2}$, in order for the whole pattern to be fired. This time annotated sequence could be presented as:

$$\hat{s_1}^2 \xrightarrow{10} \hat{s_2}^3 \xrightarrow{15} \hat{s_3}^1 \tag{6.1}$$

Supposing that the simple CEP rules for each shapelet are already created, and they are outputting (whenever their patterns matches) the start time (*startTime*), end time (*endTime*), and the dimension names (*dim*) to a named window called NW_1 that belongs to TAS_1. Given $\hat{Sh}$, W

Algorithm 10: The createComplexCEPRule method

Input: $\hat{Sh}$, W, c, NW
Output: CR_D

```
1  define alphabet
   /* alphabet is a dummy array of characters just to give names (such as
      'a', 'b', etc.) to the events                                        */
   // In EPL Syntax
2  query ← "SELECT * FROM PATTERN [EVERY"
3  for index in |Ŝh| do
4      ŝ ← Ŝh[index] // get shapelet at position index
5      dimension ← getDimensionOf(ŝ)
6      if index = 0 then
           // First shapelet in the sequence
           // + ← means append to the query
7          query + ← "alphabet[index]=NW(dim=dimension)"
8      else
9          query + ← "⇒" // add a CEP sequence operator to query
10         query + ← "alphabet[index]=NW(dim=dimension, startTime - alphabet[index
             -1].startTime ≤ W[index - 1])"
11 CR ← engine.registerQuery(query)
12 CR.registerListener(warn about situation c)
   /* The listener could be any piece of code that one would like to put.
      The listener is executed whenever the complex pattern is fulfilled.
      The listener in our implementation simply warns about the predicted
      situation.                                                           */
13 return CR
```

and NW_1 to the *createComplexCEPRule* method shown in algorithm 10, the following query will be created: ***SELECT * FROM PATTERN[EVERY*** *a=NW_1 (dim=2) ⇒ b=NW_1 (dim=3, startTime - a.startTime ≤ 10) ⇒ c=NW_1 (dim=1, startTime - b.startTime ≤ 15)]*. Above we explained TAS_1 in plain English, and this is its exact translation into the EPL language. **PATTERN** means that we are about to define a complex pattern within the two brackets []. Then **EVERY** instructs the engine that for **every** composite event that it detects where $dim = 2$, it should stay fully prepared and search for the described pattern. Without the $EVERY$ keyword, the engine will try to match the pattern only on the first occurrence of an event that has *dim=2*. Then sequence constraints are employed through the CEP sequence operator ⇒, and the time constraints are accounted for when the events are being selected (condition inside the parentheses as shown in the definitions section).

6.3 Integration into BPM

After going through contextual templates and our novel techniques to go from classified scenarios into CEP rules, the integration within BPM could now be easily worked out.

Obviously the learning should be performed way before running the process and **USE & SEE** should be executed on the available historical scenarios, so they can yield and save advanced temporal patterns. Afterward when the process starts executing, users can check from the template which classes they are interested to predict (e.g. in the manufacturing scenario, it makes sense to predict abnormality). When the execution flow reaches the monitorable activity (e.g. manufacture) whose behavior we aim to predict, the BPM engine will trigger **autoCEP** while pointing it to the location where the temporal patterns are saved, i.e., the output of **USE & SEE** (this pointing could be done through the instance-based attributes of the template see fig. 4.5. or easily through the **autoCEP** API). **AutoCEP** then transforms on-the-fly the different patterns into predictive CEP rules, it configures the CEP engine to predict for the checked classes in the template, it runs the engine, and starts its real-time monitoring and analysis. Subsequently, whenever one of the checked classes (in the template) is predicted, **autoCEP** will signal an event to the BPM engine (e.g., throw a BPM error with the code equals to the name of the class). Finally, attached events to the monitorable tasks (as shown in the scenario section, fig. 4.3) with the same code as the class will catch the error event dispatched by **autoCEP**, and the management could be carried out proactively.

Nothing CEP-related at all is required from BPM users, on the contrary they are left to design their business processes as they accustomed to. The only requirement is to give the process variables of the checkboxes (the prediction section of the template) and the codes of the attached events the same names as the existing classes from the historical scenarios. For example, the manufacturing scenarios are classified as *normal* and *abnormal*, accordingly the checkbox *predict abnormality* (resp. *predict normality*) from figure 4.5 should represent a process variable with the name *abnormal* (resp. *normal*). The same is true for attached events, if users seek to catch when **autoCEP** predicts abnormality, an event with the code *abnormal* should be attached to the activity. Given this simple setup, the interaction between the BPM and CEP engine will be worked out magically behind the scenes.

6.4 Summary

As shown in this chapter, going from pure data mining into CEP rules could be very complex, however it is all done swiftly and automatically with the help of **autoCEP**. No human intervention is required at this stage, and no tedious manual rule specification is needed anymore. Moreover, our approach creates a steady portal to easily employ CEP techniques in predictive domains, the thing that was only a vision.

The CEP world is very known for its expressiveness, where a massive amount of CEP operators

could be employed. Our automatic rule learning method is able to generate rules with the most relevant CEP operators as discussed in [43]. More specifically, our data-driven rules account for selection, conjunction, parametrization, sequence, window, and aggregation.

At the end, we concluded this chapter by discussing the link to the BPM world. With automatically learned CEP rules at hand, it turns out that most of the BPM/CEP integration complexity is waived away.

CHAPTER 7

VALIDATION

All life is an experiment. The more experiments you make the better.

Ralph Waldo Emerson

The few existing approaches on CEP rule generation are discussed in the related work chapter, however it is important to note that they are not capable of learning rules in the manner we are doing it. They either make unrealistic assumptions, are very user-dependent, or deal with symbolic sequences and not numeric time series, and all of them do not account for earliness and prediction, therefore a direct comparison is not feasible.

On the other hand, and as far as we can tell, our approach is the first to create a generic framework that exploits CEP for prediction inside monitorable and long-running tasks. Other proposals are focused on design-time integration over the flow of the process, and they usually are ad-hoc and require efforts to employ. Therefore a direct comparison does not seem sensible on this level as well.

Therefore for the evaluation and comparison, we have selected different real-world data sets as benchmarks to evaluate our approach and to compare it with different state-of-the-art proposals on multivariate shapelet learning. Our approach is the only one that is capable of learning sequence and time constraints among the different dimensions, other work tends to learn independent shapelets from each dimension. The goal of the tests is to prove the performance and applicability of our approach while tackling various real analytic tasks such as anticipating robot failure [9], predicting if ECG data are normal or abnormal [148], foreseeing failures in wafer manufacturing [148]. All experiments were conducted on a PC with an Intel i7 2.8GHz CPU and 32 GB of main memory. The algorithms of **USE** & **SEE** were written in Python, whereas

autoCEP is implemented in Java, Esper[1] is used as a CEP engine, and the integration is done with the open-source Camunda BPM engine[2].

7.1 Wafer Manufacturing Scenario

This section will demonstrate the integration of CEP and BPM for the wafer manufacturing scenario. The complete demonstration and its videos could be found on the website of the project[3].

Historical classified scenarios of real wafer manufacturing activities can be obtained from [148]. After deploying and configuring the process (in BPMN format), executing it will trigger **autoCEP**. Then **autoCEP** will go side by side with Camunda to provide real-time analysis and prediction for the manufacturing task.

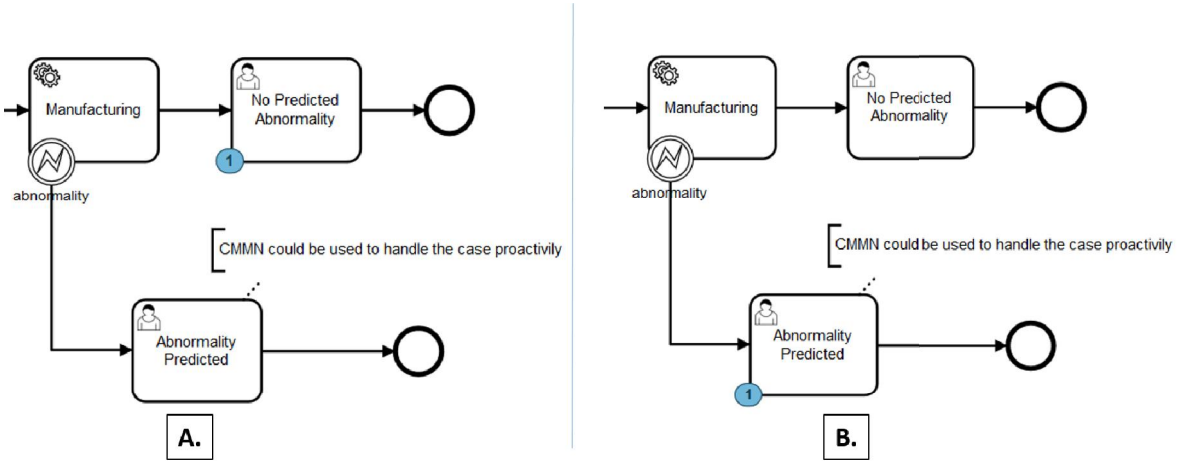

Figure 7.1: A. Normal Instance B. Abnormal Instance

Figure 7.1 presents two portions of two running instances of the manufacturing process. On the left side of the figure, a normal test instance was streamed through **autoCEP**, correctly no abnormality was predicted, and the flow of the process continued normally — this is shown with the execution token (this is inside the Camunda engine cockpit) on the *No Predicted Abnormality* task. On the right side of figure 7.1, another abnormal test instance was streamed, and as shown the execution of the process was interrupted by the attached error event — the execution token on the *Abnormality Predicted* task. Thanks to this easy integration with CEP, proactive measures could now be employed. Even if these measures are not known in advance (the structure is not evident), existing standards such as CMMN[4] could help to efficiently manage the case.

[1] http://espertech.com/esper/
[2] https://camunda.org/
[3] https://goo.gl/2aDHyu
[4] http://www.omg.org/spec/CMMN/1.1/

7.2 Evaluation

Since we are dealing with real-time prediction over a stream of events, **autoCEP** records all the predicted classes over time (classification variation). To examine the behavior of our algorithm, we utilized five different methods, and for each we present the accuracy and earliness. *Closest classification*: the stream is classified with the most similar pattern so far. *First classification*: The stream is attributed the class of the first detected pattern. *Abnormality Detection*: Normally in proactive applications, users are interested in predicting abnormality (or specific situations) rather than normal streams, therefore this test focuses on specific classes and disregard unimportant situations. *True class exists:* This method tests if the true class exists among the classification variation over time. Finally *majority voting*: For this method the class of the majority is accounted for, earliness is always 100% because the full-length time series is used at the end.

Table 7.1: The Data Sets

Dataset	# Dimensions	Classes	# Instances	Max. Length	Min. Length	Imbalance
Wafer	6	Binary	1194	198	104	x
ECG	2	Binary	200	152	39	
Robots	6	Multiple	93	15	15	

Table. 7.1 presents different characteristics of the multivariate data sets that are used for the experimentation[5]. For the comparison with state-of-the-art approaches all parameters are kept to the default values, and the *closest classification* method is used. Tables 7.2, 7.3, 7.4 illustrate the results that we have achieved on the wafer, ECG, and robot data sets respectively.

Table 7.2: Results on the Wafer Data Set

Approach	Avg. f-score	Earliness	App.	Acc.
autoCEP	90.5%	28.7%	100%	92.6%
REACT	91.9%	32.8%	100%	–
Full 1NN	87.2%	100%	100%	89.9

Table 7.3: Results on the ECG Data Set

Approach	Avg. f-score	Earliness	App.	Acc.
autoCEP	81%	21.2%	100%	82.7%
REACT	76.7%	10.5%	100%	–
Full 1NN	87.7%	100%	100%	88.7
MSD	58.8%	12.8%	100%	–

[5]The robot dataset is a collection of five different datasets, so each dataset is handled separately and the reported results are the average

Table 7.4: Results on the Robot Data Set

Approach	Avg. f-score	Earliness	App.	Acc.
autoCEP	76.6%	50%	100%	80.8%
REACT	72.7%	40.7%	94.7%	–
Full 1NN	71.9%	100%	100%	79.3
MSD	39.6%	27.4%	96.3%	–

For each experiment and when applicable, we have compared our method with the 1NN classifier as a baseline, and with two state-of-the-art approaches to learn shapelets from multivariate time series. In general, the different compared methods for these experiments were: **autoCEP**, REACT [115], 1NN classifier, and MSD [65]. Other than **autoCEP**, all approaches use ad-hoc algorithms in order to implement real-time classification. In our work, we algorithmically shifted toward the world of CEP, and thus taking advantage of its speed and performance.
As depicted in the aforementioned tables, our approach competes with current multivariate shapelets learning methods, as it reported better or close results. In general, autoCEP showed a good trade-off between earliness and average f-score. On the tests, where it didn't score the best earliness, it scored the best average f-score. Of course, the 1NN classifier always scored 100% earliness because it consumed the whole time series before deciding to which class it belongs. In terms of applicability, **autoCEP** always reported 100% scores.

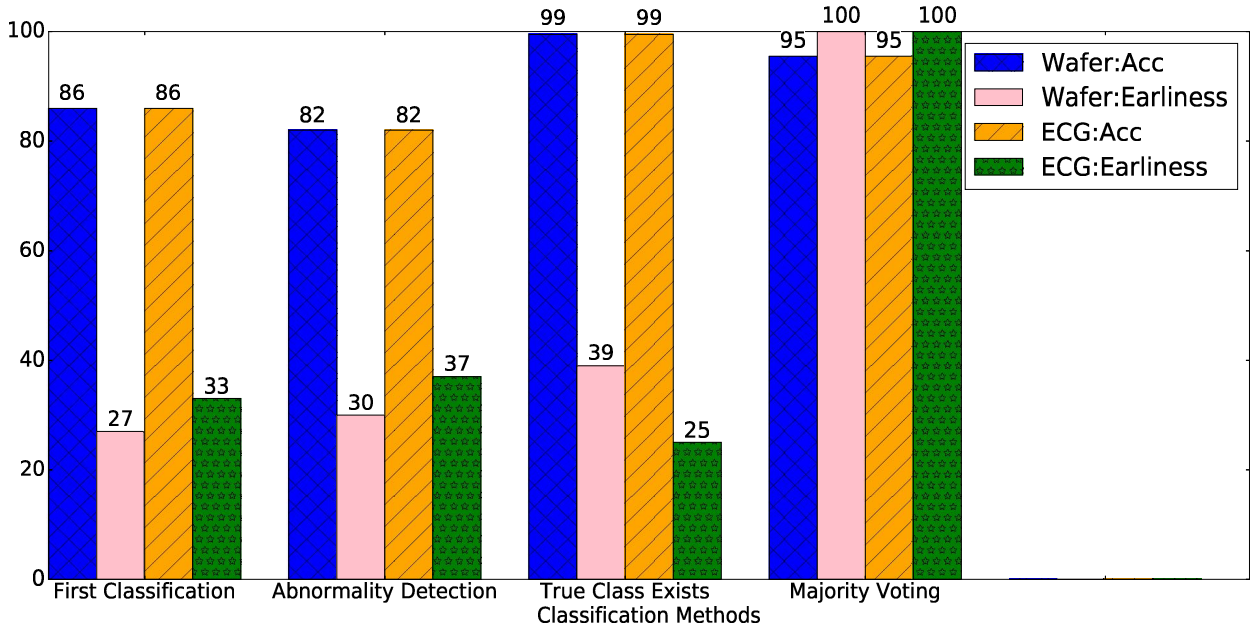

Figure 7.2: Classification Results

On the other hand, and to get the global picture, Figure 7.2 depicts the results on the Wafer and ECG data sets while employing the remaining four classification methods (the closest classification method is already reported in the above tables when comparing to the state-of-the-art).

The majority voting always scored a 100% earliness score because the approach waited to read the whole time series before deciding to which class it belongs.

7.2.1 Efficiency

7.2.1.1 Memory-Aware Execution

Searching and calculating distances between time series subsequences is known as a very complex, time-, and memory-consuming problem, even in the univariate settings [208]. Typically, super machines are dedicated for this kind of problems with more than 96 GB of main memory like in [115].

However, **USE** & **SEE** is an approach that is agnostic to the size of memory, and it could be executed on normal machines without having memory errors. The trick behind this is in our smart pruning strategy. More specifically, a thread that is responsible on monitoring the memory usage could interrupt the code shown in algorithm 2 to force a pruning phase. It saves the states of the algorithm and the progress that is made so far, it forces the pruning, it cleans the memory from unwanted candidates, and then it signals the main algorithm to continue the learning from where it stopped.

7.2.1.2 Learning Time

In the USE algorithm, and since we are testing every subsequence, a trade-off exists between accurate results and learning time. To this end, we held different experiments with different variations. More specifically we tried a straightforward implementation and an optimized one (**USE** & **SEE**-*O*) with the help of multi-threading. In the two implementations, we attempted both *brute* and *mass* as values to the distance measure parameter of the USE algorithm (see algo. 2). It is important to note that we have implemented *mass* as the authors directed on their page [145], yet their implementation omits the calculation of the first distance (i.e., the first element in the *sim* array of algo. 3). Therefore, we slightly changed the code to account for the missing distance as well.

Table. 7.5 compares all the variations of **USE** & **SEE** that we have tried, in addition it shows results reported by two state-of-the-art approaches [115, 65]. Default values are used for our algorithm, and the same amount of training instances were employed in all experiments (For brevity we will refer to our algorithm as U&S). The first row of table 7.5 hints about the testing environments.

As shown in table. 7.5, learning time is quite expensive on big datasets, however it is still comparable to other recent approaches. Nonetheless, the training is a design-time phase and by exploiting concurrent methods it could be optimized as shown in the table. Another point that is important to highlight is that our approach is in its default settings regarding the minimum and maximum lengths. So it tried to find the optimal lengths without any guidance, but if $minLingth$ and $maxLength$ are to be provided then this will drastically affect the learning

Table 7.5: Learning Time

	CPU @ 2.80GHZ, 32 GB of RAM				CPU @ 2.40GHZ, 96GB of RAM		
Dataset	U&S (brute)	U&S (mass)	U&S-*O* (brute)	U&S-*O* (mass)	MSD	REACT	REACT-GPU
Wafer	52.2 hours	31.8 hours	29.4 hours	17.1 hours	> 2 weeks	41.8 hours	18.9 hours
ECG	8 hours	3.5 hours	2.2 hours	44 minutes	3.6 hours	4.25 hours	1.7 minutes
Robots	14.3 minutes	6.8 minutes	2 minutes	1.1 minutes	2.9 minutes	4.25 minutes	5.64 seconds

time. For example, table. 7.6 shows the results after re-running the experiments with $minLength$ and $maxLength$ manually specified. For the wafer dataset, the settings were $minLength = 40$, and $maxLength = 50$. For the ECG, the minimum was 35 and the maximum 39. Finally for the robots dataset, $minLength = maxLength = 10$.

Table 7.6: Learning Time with Lengths Specified

Dataset	U&S (brute)	U&S (mass)	U&S-*O* (brute)	U&S-*O* (mass)
Wafer	21.8 hours	11.5 hours	7.3 hours	2.9 hours
ECG	1.2 hours	46 minutes	20.8 minutes	2.3 minutes
Robots	1.4 minutes	55 seconds	4 seconds	0.89 seconds

7.2.2 Sensitivity of Parameters

Instead of running **USE** & **SEE** with its default settings, we will run different experiments while varying the various parameters, in order to assess their sensitivities and their effects on the results. To this end, figure A.3 reports the accuracy and earliness scores on the ECG dataset.

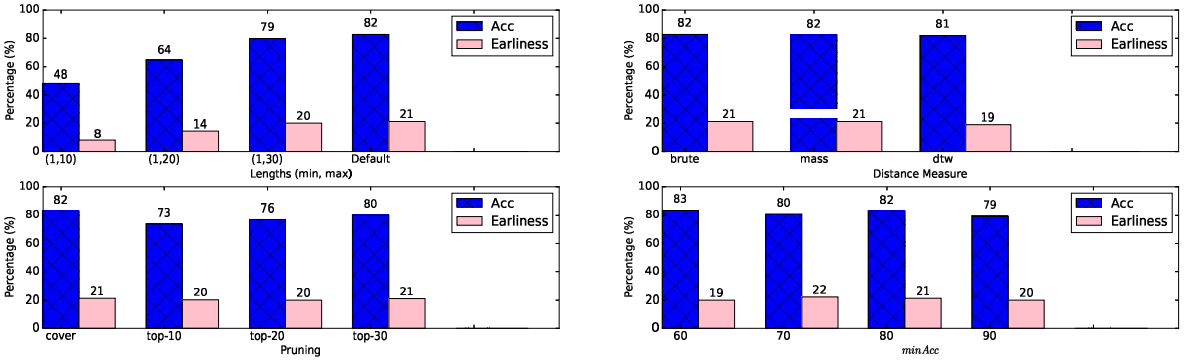

Figure 7.3: Results on the ECG dataset

7.2.2.1 Discussion

The first graph (upper left) of figure A.3 depicts the results when the minimum and maximum lengths are manipulated. As shown in table. 7.1, the shortest time series in the ECG data set is equal to 39, so we changed the values of the lengths accordingly. As expected given the algorithm a small space to search for patterns like (min=1, max=10) or (min=1, max=20) will affect the accuracy, as very short shapelets may not be representative, and will yield false positives or negatives. However with longer patterns, the approach was able to keep high accuracy. Intuitively, the earliness is directly affected by the minimum and maximum lengths as the graph demonstrates. Providing these lengths is a good way for experts to get involved in the learning, yet the default settings could also yield useful results.

The second graph (upper right) sketches the outcomes with the three distinct values that the *distance measure* parameter of algorithm 2 could take. All other settings are kept to the default. On the ECG dataset, the results were identical, so this parameter is not sensitive for the results. On the other hand, table 7.6 demonstrates the advantage of using *mass* from time-consumption perspective.

The lower graph to the left proves the effect of the pruning algorithm. It shows that the *cover* strategy is the most beneficial one. The intuition behind this is that the top-k algorithm will yield shapelets with elevated scores, but there is a high probability that these shapelets are so close in their shapes (almost identical). The cover strategy allows for more diversified shapelets to exist in the results. The earliness was not affected by the pruning algorithm, however the same could not be said for the accuracy.

The last graph presents the results when the $minAcc$ parameter of algorithm 5 is modified. This parameter is tricky to calibrate, however even when given low values like 60, it yielded very good results. In general, we consider the default value (80%) as a logical threshold, however user may substitute it with another preferable one. Higher thresholds (> 80) and lower ones (< 50) may probably lead to over and underfitting respectively.

7.2.3 Interpretability of Rules

In data mining, it will make more sense if users could understand how algorithms classify instances. In other words, patterns that could be digested and interpreted by users are really important, as they may help domain experts in understanding the causes and the reasons of a specific situation of interest. Then these understandings could help experts to carry new studies, in order to shed some lights on otherwise ambiguous situations.

Since shapelets are interpretable temporal patterns [207], a strong point to emphasize in our approach is the interpretability of the outputted rules. Specifically, our implementation is equipped with visualization features that could assist experts on interpreting the results.

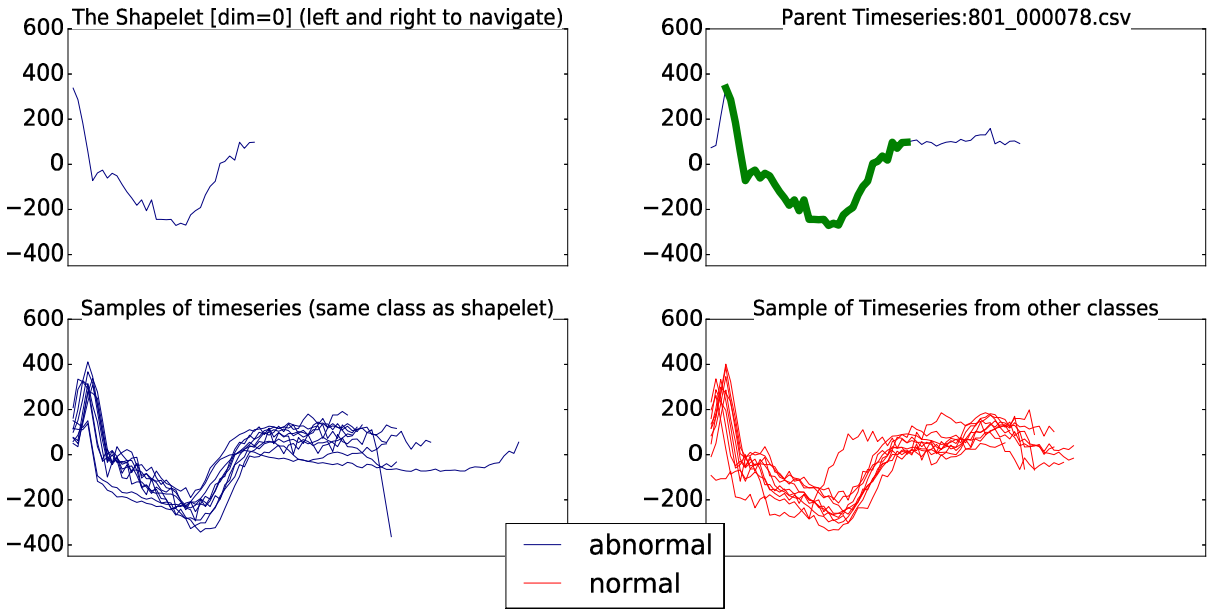

Figure 7.4: Shapelet Explorer

Figure 7.4 hints at the shapelet explorer feature. With this equipment, experts could interpret shapelets one by one to see what patterns and shapes exist over the different dimensions. The shapelet explorer provides information about the temporal shape, the dimension and the class of the selected shapelet (upper left graph). It also draws the time series from which the shapelet was extracted, i.e., the parent time series. On the lower part, the explorer gives a clue on the profiles of the time series that belong to the same class as the shapelet, and others that belong to different classes. This visualization feature deals with univariate shapelets, so only the concerned dimension of the multivariate time series is depicted. Figure 7.4 illustrates a shapelet on the first dimension of the ECG data that is concerned with the abnormal class. As pictured on the lower part of the figure, and for the ECG data, it is not evident to use just one dimension to classify the whole multivariate time series, as the profiles for the normal (right part) and abnormal (left part) time series seem almost the same at the first glimpse.

Furthermore, our approach comes equipped with a more advantageous visualization feature, that could prove really handy when it comes to interpreting multivariate time series. Figure 7.5 sheds some light on the rule explorer.

The explorer sketches the rule on the upper part, showing the used dimensions, the class, the order of the used shapelets, and the time windows between them. On the lower part, the explorer presents example multidimensional time series where the above rule could be found, shapelets are depicted in bold on these instances. The figure draws an abnormal case from the wafer dataset. Users can easily navigate the different rules, and the different instances that are covered by each rule.

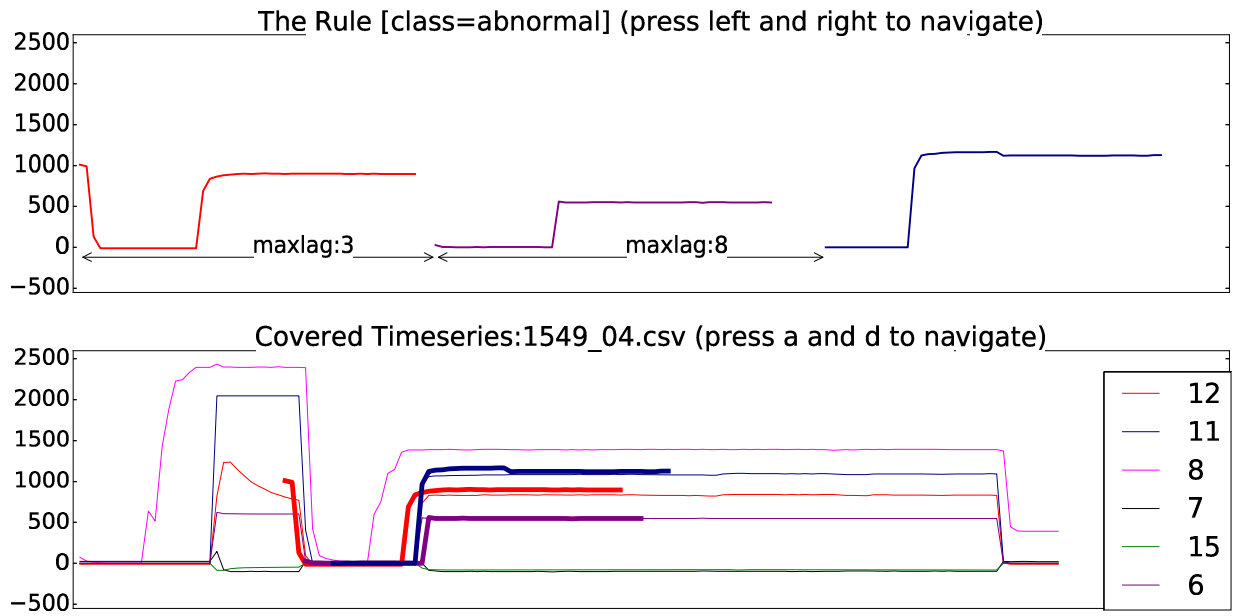

Figure 7.5: Rule Explorer

In this regard, our approach has the upper hand over different multivariate shapelet algorithms, by learning complete rules and drawing them to the users. Following this way, domain experts could explore their data while noticing and learning new and otherwise hidden sequences, time constraints, and correlations. It is obvious from figure 7.5 that only a subset of dimensions are used for the outputted rules (in the figure a rule from 3 out of 6 dimensions is pictured). Figure 7.6 yet shows another pattern from the same data and for the same class, but with only one shapelet from the 5^{th} dimension.

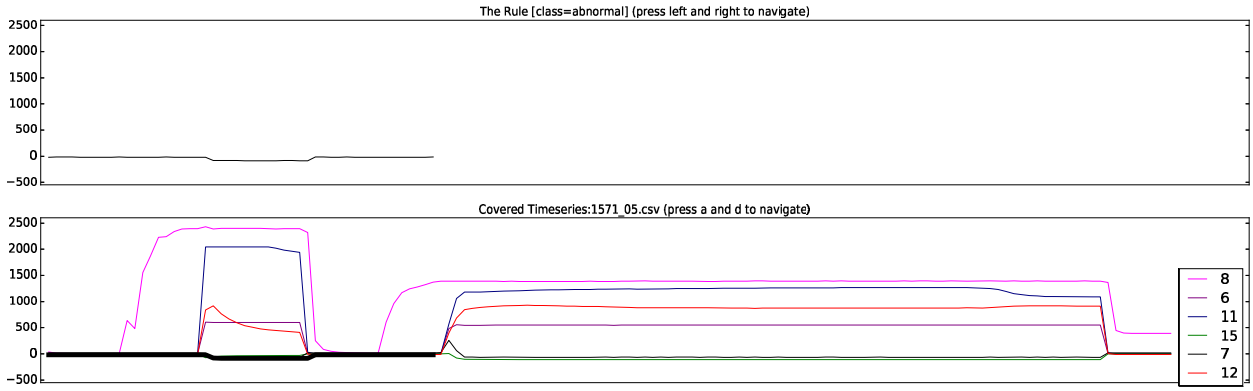

Figure 7.6: Rule Explorer: Rule 2

7.2.4 Complexity

7.2.4.1 USE

In this section, different tests will assess the time complexity of the **USE** algorithm. To this end, we will variate the number of instances n, the length of time series m, and the number of dimensions d. Two values of the distance measure will be tested: **brute** and **mass**.

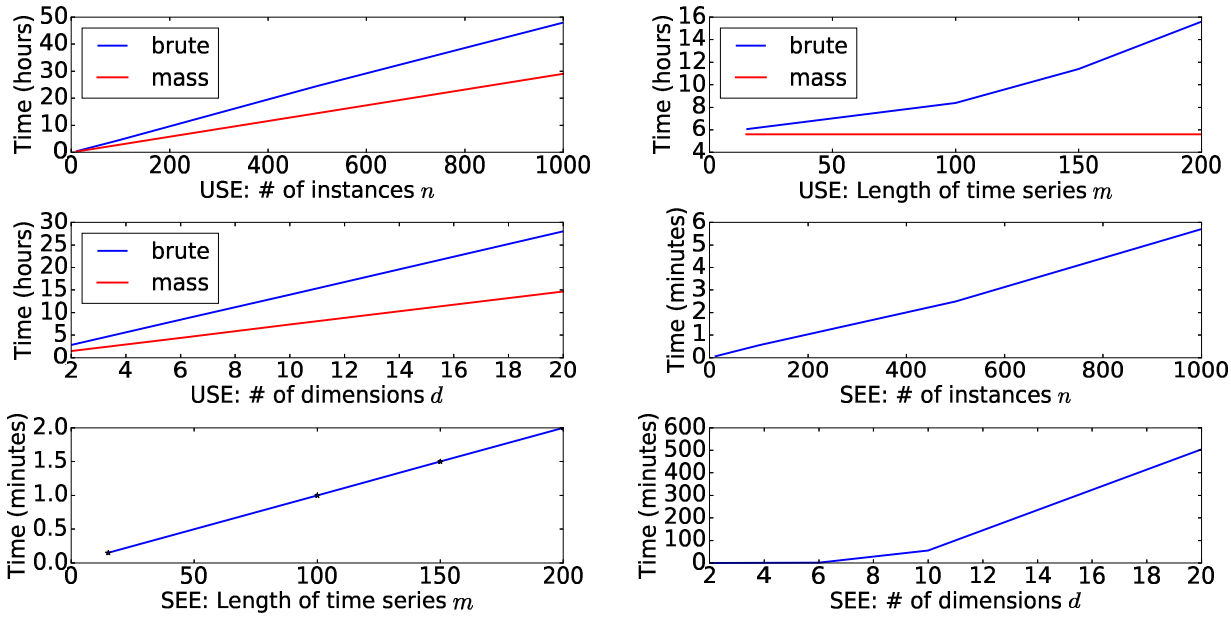

Figure 7.7: Performance of **USE** & **SEE**

The first three graphs of figure 7.7 are related to the **USE** algorithm (they have the word USE in the title of the x axes). These graphs depict the results of the tests, the y axis represents the time complexity in hours. For the first experiment, reported in the upper left graph, four tests were carried with $n \in \{10, 100, 500, 1000\}$, while $m = 150$ and d is set to 6. As shown our algorithm in both variations is linear in regard to the number of instances in the dataset, given that with *mass*, it is less complex.

In the second experiment (upper right graph), n and d are fixed to equal 200 and 6, while $m \in \{15, 100, 150, 200\}$. The *brute* strategy proved to be quadratic to the length of time series, while the *mass* showed no inclination at all because it is in fact independent from this factor [208].

The third experiment (second row left graph) proved that the **USE** algorithm, again in both variations, is linear to the number of dimensions. The setups were as follow, $n = 200$, $m = 100$ and $d \in \{2, 6, 10, 20\}$.

In general, the complexity of the *brute* and *mass* are $O(nm^2)$ and $O(n\log n)$ respectively. As for the *cover* pruning strategy, it takes another $O(n\log n)$. Therefore adding the number of dimensions, the complexity of **USE**(brute) will be approximated as $O(d * n(m^2 \log n))$, as for the **USE**(mass) it is $O(d * n\log n)$.

7.2.4.2 SEE

The results of the **SEE** algorithm are also reported in figure 7.7 (the remaining three graphs) for the same experiments as the above section.

The **SEE** algorithm was relatively fast compared to **USE** (the y axis represents the complexity in minutes), however it is mostly affected by d because it searches for the different permutations among the different dimensions. Even thought, we are not using a naive approach where all combinations are tested, but we are encoding and departing from the training data set (see algo. 5), and pruning many sequences, so converging rapidly towards solutions. In fact, we devised this permutation strategy to account for all the important patterns from all dimensions. This is completely different from state-of-the-art approaches like [151, 66] that label a whole dimension as relevant or irrelevant, and so disregarding it without taking into consideration some specific combinations that could be formed using this dimension with others.

When varying n (second row right graph) and m (last row left graph) **SEE** was able to finish within 6 and 2 minutes respectively for the highest values. When we fixed n and m at 200 and 100 respectively, with up to 6 dimensions, **SEE** took only 1 minute to execute. However with 10 dimensions it spanned over 50 minutes, and then with d set to 20, the learning dramatically increased to score 500 minutes.

In general, **SEE** proved to be linear with the number of instances and the length of time series, yet at its worse it might be factorial to the number of dimensions. Therefore the complexity of **SEE** could be estimated (in extreme situations) as $O(n * m * d!)$. An important note to highlight is that the complexity never reached the factorial of d, because and as mentioned before, we are departing from the time series and not all time series have shapelets on all dimensions.

7.2.4.3 autoCEP

Differently from the previous two algorithms, **autoCEP** has no time complexity as it transforms rules on the fly. As a matter of fact, we transformed 40, 60 and 100 rules, and they all finished within some fractions of a second.

CHAPTER 8

CONCLUSION

Enough research will tend to support your conclusions.

Arthur Bloch

Change will always be the main theme that will dominate our technological world, and the pace of this change is continuously accelerating. Every new day is bringing new concepts, visions, and advancements that are shaping our lives. The realms of hardware, software, and network are rapidly evolving. IoT is transmuting, from being a mere vision and fiction, into tangible objects in the real world. All these changes are forcing technologies to modify the way they handle data. Therefore, technologies who seek to survive this paradigm shift are compelled to adapt and integrate within the ever-evolving world.

The main driver of this change is data, its abundance is reforming the way all applications and businesses are done. To highlight the importance of data, it is said that good data beats better algorithms. To put it in another form, no matter how excellent an algorithm is, when more data will be available, the algorithm will become obsolete. Therefore, data is one of the big influencer when it comes to changes and paradigm shifts, and currently data is being acquired more than ever before. As a matter of fact, our age is called the information age, and many things have changed — and continuously being changed — from the prior industrial age.

In any application field, the followed techniques and practices have been reexamined and reevaluated to assess if they are exploiting data in a sufficient way. Medicine, diagnostics, healthcare, construction, marketing, optimization, sports, security, finance, and any other domain that one could think of, are striving to be more and more driven by data. Data mining techniques are some of the best solutions that are able to achieve exactly this goal. This is the reason why data science is currently taking the initiative, and data mining is being employed in almost every do-

main.

Data mining and analysis of data could bring out precious hidden knowledge, and with the abundance of data throughout the ages, analysis techniques evolved and matured. At the earliest stages, descriptive analytics was the trend. Typically, monitoring and reporting was the fundamental job of analysis, and it was sufficient and beneficial for businesses. However, this is no longer the case, currently other advanced types of analysis emerged and all industrial and research practitioners are seeking to efficiently apply them. Presently, predictive, prescriptive, and proactive analytics are the trends. The first one predicts situations and signals them to interested parties, then these parties are in charge of handling the anticipated situations. The second predicts as the first type of analysis, but it also generates the best course of actions to deal with the prediction, then involved parties are in charge of taking the recommended countermeasures. Lastly, proactive analytics predicts situations, searches for the best responses to the prediction, and autonomously takes action without the intervention of humans. This is where the CEP technology is still lagging behind, as it is still used for monitoring and detecting situations. It still didn't make the jump toward the more advanced types of analysis.

This is where our work comes in handy, as we tweaked a fundamental principle in the CEP world to introduce data mining to the mix. The core value that we changed is the way CEP rules are defined. By definition, they should be specified manually, but with our proposition, the process was automatized. Integrating data mining at this deep level of CEP, helped us to go from reactive into predictive. At the end, the efforts were culminated when the integration between BPM and CEP became easy to achieve.

8.1 Research Results

The answers to the research questions that we asked in the introduction were laid out throughout the dissertation. Some of the answers were explicitly highlighted, and others were concealed within the sentences. In this section we will go over the main questions again but in an outright manner, we will try to summarize the answers and to exhibit the significance of our work.

Why time series is the data model that we targeted in our whole approach?

Answer: Devising a completely generic solution where all application domains and data types are supported is somehow very difficult. In our work, and to fulfill the generic constraint, we tried to support time series, which is a popular data model that is adopted in a myriad of real-life domains. The secret to this ubiquity is IoT devices. As discussed before, sensors are becoming more pervasive, and they emit their data in the form of time series. They are continuously monitoring the surrounding environments to produce time series with high dimensionality. Add to this the seamless transition that we could do from pure data mining to the CEP world. In fact, some would argue that time series are exactly the same as timestamped events, but they just happen to have two names in two research communities.

Why shapelets are chosen as the data mining technique to be adopted for CEP?

Answer: Indeed different models could be adopted. Some researchers tried to integrate SVM, Markov chains, and others. However, all these integrations were complex. In some approaches, the prediction models require additional external inputs to function [61], in others, users are in charge of specifying the structure and operators of CEP rules [146]. In our work, we showed that shapelets are simpler to adopt, as the overall approach learns CEP rules from scratch and it supports multiple CEP operators. Shapelets are chosen because: First, their inputs are time series which could be easily represented as events. Second, shapelets are suitable for early classification, the thing that supports our prediction requirement.

What are the latest advancements on shapelets? Do these state-of-the-art approaches fit to be integrated in the CEP core?

Answer: Shapelets are a stable data model that is mainly used for time series classification. Different authors in various applications adopted shapelet-based techniques [165, 116, 79, 68]. Typically, all approaches that adopt shapelets function in univariate environments, because and by definition, shapelets are basically extracted from these environments. In our approach, we refrained from deeply discussing related work on shapelets extraction in univariate settings, because we have built our proposition on top of this work and we did not substitute it. Our main focus was on multivariate environments. Therefore, we referenced the shapelet extraction algorithms in this dissertation, we included light details, and we have covered more specifics in appendix A.

The direct answer to the second fold of the question is yes, these approaches could be integrated. In fact, this is exactly what we did when we started with **autoCEP**. We built on top of univariate shapelet extraction algorithms, and **autoCEP** only worked on a single stream of events (appendix A yields more details). The approach that we have created was generic and easy-to-use, but it was limited to simple application domains where patterns are found only on one dimension. We argue that in real life, patterns are more complex, and they account for shapelets from more than one dimension. Therefore, as we mentioned, the direct answer is yes but with limitations that we still need to address.

What extensions into the multivariate world exist? Could these extensions take advantage of the expressiveness of CEP, should we decided to adopt them?

Answer: The above answer brings up the limitation of using univariate time series with **autoCEP**, and therefore answers for the current questions must be addressed. The related work chapter (chapter 2) explicitly answered the first fold of the question. The different multivariate shapelet learning approaches were discussed, their strong and weak points were included as well. From their shortcomings comes the answer to the second part of the question. Since the authors of such papers did not take the CEP technology into account when they first developed their approach, they could not take advantage of the expressiveness of CEP. In other words, current approaches learns independent shapelets from different dimensions, but when it comes to CEP, this technology detects correlated patterns from different dimensions. The output of current methods are not correlated,

and therefore it makes no sense to adopt them for a rich pattern-detection technology such as CEP. From this point, the motive for a new learning approach arises.

The above responses could also be referenced to answer other literature questions that are related to the complex event processing field, such as: **how shapelet-based solutions could help CEP to adapt to the paradigm shift?**. However we will complete the picture with some brief answers.

How to go from manually specifying CEP rules into automatically defining them?
Answer: The obvious answer to this question is to go through data mining techniques. In fact, as we discussed in the introduction, this is why domains such as data mining are created in the first place.

What work existed before on the integration of data mining and CEP? Is there some predictive approaches on this matter?
Answer: The research on the CEP domain is very lively. However the main goals of involved researchers are computing efficiency, scalability, latency, etc. When it comes to specifying CEP rules, only few approaches are concerned. These approaches were discussed in chapter 2. Among them, there is no one that accounts for prediction and earliness. They are all interested in learning rules that only detect situations.

What is the state-of-the-art on integrating CEP with BPM?
Answers: Chapter2 also discussed a myriad of approaches that integrated CEP and BPM. The integration is highly targeted, so monitorable tasks could be better managed. The main theme that describe all these approaches is the design-time methodology. They focus on the model of the process, they assign monitoring points, and they monitor the same events for every instances. Of course this has its advantages but still it suffers from some limitations as the related work chapter presents.

To what extent CEP capabilities are exploited in BPM systems?
Answers: We would confidently say that CEP capabilities are not sufficiently exploited in BPM systems. First, CEP rules are written manually, and there is a limit to how advanced a user's inference could be. Second, since monitoring points are allocated on the model, the same all-purpose events are monitored, and the management is not customizable for different instances.

Is there real predictive practices when it comes to managing business process instances? Are instances being managed in a context-aware fashion?
Answer: Prediction in business process management is centered on the overall workflow. For instance, the global completion time of the process could be predicted regarding the registered completion times of tasks that have already happened. However inside the executing tasks, and since there is no prediction in CEP, there is also no prediction in BPM. On the other hand, we argue that instances are not being managed in a context-aware fashion because monitored events are general and not specific for each instance. The root problem is that monitoring points are being allocated on the model of the process as we discussed.

How to go directly from pure data mining into the CEP world?
Answer: To exploit the CEP capabilities for prediction we need to start searching for patterns that predict situations rather than detect situations. The CEP would detect these predictive patterns and then afterward the overall situation would be anticipated. Predictive patterns could be learned using data mining techniques, but CEP rules need to be written in CEP jargon. **AutoCEP** is devised to bridge this gap by using shapelet learning algorithms. In the heart of dissertation, we discussed with fine details, how to transform the learned sequences from data mining into CEP. We explicitly explained, in a technical manner, how all CEP operators such as sequencing, windowing, parametrization, etc. could be extracted and transformed. The algorithms showed the exact EPL syntax that is responsible for the transformation. In addition, how advanced concepts such as complex processing chains and named windows are exploited to achieve our goal.

How to extract shapelets with advanced knowledge from multivariate time series?
Answer: We have discussed the shortcomings of approaches related to the learning form multivariate settings. To overcome these limitations, we have proposed new learning algorithms that we have discussed in details throughout the dissertation. The approach extracted independent shapelets from each dimension, however we did not stop at this point like most related approaches, but we additionally learned sequence and time correlations that may exist among the different dimensions.

How to go into the BPM field in an easy way?
Answer: After working out the bridge between data mining and CEP, the integration within the BPM world was a lot easier. In fact, just little setups are required, like attaching interrupting events and using process variables to communicate with **autoCEP** through its API. All these concepts are BPM-related, therefore the hassle of writing CEP rules is completely negligent.

8.2 Contributions Revisited

In this section, we will revisit the three domains that we have touched in our work. Specifically, we will re-mention what have been missing and how our contributions addressed them.

8.2.1 Time Series Data Mining

As shown in the second chapter, the number of approaches that have adopted shapelets for time series classification are large relative to the freshness of this data mining primitive. However, this number boils down to five approaches when it comes to consider shapelets with multivariate time series. All these approaches have focused on extending shapelets learning algorithms to the multidimensional settings but did not give extra attention to the correlation that may exist between shapelets on the different dimensions. In some domains, this may not be necessary, but in others it may be critical, especially for the sake of learning complex event processing rules which is the main motivation behind our work.

This is exactly where **USE & SEE** lend a helping hand, as these algorithms go beyond the point where current approaches have stopped, to deduce advanced knowledge and correlation among the different shapelets. In particular, **USE** supports independent shapelet extraction from multivariate time series. This algorithm divides the original data set into different smaller data sets grouped by dimension. In other words, each one of the smaller data set contains univariate time series that belongs to the same dimension. This division then would allow shapelets to be extracted by following typical shapelet learning algorithms. The learned shapelets are filtered so only the most useful ones are kept, which in turn will be emitted to the second phase **SEE**. At this phase, complete time-annotated sequences are learned in a sense that sequence and time constraints are also deduced from historical observations. Moreover, these time-annotated sequences are constructed from the minimum needed number of shapelets to accurately assess the classification. This strategy boosts the earliness and online classification time of the approach.

8.2.2 Complex Event Processing

Predictive and proactive complex event processing was always been discussed on the conceptual level and as a future vision, and surveying state-of-the-art approaches in this field has absolutely confirmed that claim. It has always been very difficult to employ CEP in predictive contexts and applications, and this limited the CEP to merely reactive ends. All research efforts in this domain have always discussed reactive scenarios like the detection of fraud, the detection of traffic jams, and the detection of abnormal behavior. It has always been the detection but not the prediction. We argued in this dissertation that the main reason for this is the manual specification of CEP rules. Beside all the powerful real-time performance of CEP engines, users are left without any support to define CEP rules themselves. This assumption is very limiting, because the required knowledge may not be present, especially when we are dealing with predictions.

Some related approaches already introduced different ways to support users in the rule specification process. They proposed different learning techniques to go from historical data into CEP rules, however they all are reactive in nature. Instead of predicting situation of interests, they all try to detect these situations after they occur. Even though, these approaches helped users to better define their rules, yet in the context of employing CEP for prediction, they did not add any value.

AutoCEP is proposed to exactly bridge this gap. Thanks to our data mining algorithms (**USE & SEE**), sequence and temporal patterns could be extracted from history, then thanks to **autoCEP** these patterns are swiftly transformed into predictive CEP rules. In summary, we have proposed the first approach to learn predictive rules for complex event processing. With that being said, **autoCEP** continues further by creating a CEP engine and configuring it, so it could be ready for prediction.

8.2.3 Business Process Management

Business process management systems are activity-based workflow managers. That is, they focus on the business process as a model, and they monitor compliance and detect divergence on the flow level. For example, task A should not be executed in parallel with task B. Such management systems are beneficial to better understand and optimize business operations, however with the availability of sensors and data this is no longer sufficient. Fine-grained processing became essential and it is currently provided by the complex event processing technology. So BPM and CEP have always gone hand in hand.

Integrating CEP to monitor all purpose events and assigning monitoring points over the model of the process have already been addressed by state-of-the-art approaches. However, only few and timid attempts tried to go yet into a finer level of details and manage in-activities behaviors. In fact, it is shown to be far from simple to integrate CEP for custom and complex analysis, because BPM users are not accustomed to writing complex CEP rules, especially if prediction is to be considered.

With our approach that automatically learns predictive CEP rules now at hand, this problem could be easily addressed. More specifically, in this dissertation, we have showed that with an easy setups in BPM engines, a complete communication between BPM and CEP engines could be established. Moreover, prediction could be now supported, which in turn will help to manage activities in a proactive fashion. BPM users are not required to write CEP rules, in contrast, and for the most of it, they are left to model their processes and work in the environment that they are accustomed to.

8.3 Limitations

Even though our propositions have provided innovative solutions to some intricate problems in the related domains, but still some limitations exist and they should be mentioned.

First, the learning algorithms, and on big data sets, may have high time complexity. Indeed the learning is a design-time phase, but there is always room for improvements. **USE** employs a brute force learning strategy that consumes a lot of resources and that spans over a long interval of time. Even with the *mass* strategy as a distance measure, the learning time is still considerable. On the other hand, **SEE** could stall if it is confronted with a lot of dimensions. The main reason for this is the permutation strategy that it is employed.

In addition, the learning algorithm works only with numerical time series, whereas a complete proposal that could handle categorical and numerical variables would be much beneficial and applicable in more domains. Especially that the literature is full with proposals that could be exploited to fulfill this specific objective.

We argue that on the complex event processing level, our approach is new and innovative, and it

could be considered as a first actual step toward proactive business process management. However, the language of CEP is very expressive, and we claim that one learning approach could not capture this broad expressiveness. For example, a CEP operator that may be frequently used in some domains is the negation, it means the absence of a specific event, and it is not supported by **autoCEP** in its current version. Still, our approach is generic in a sense that it works in application fields where numeric time series exist (usually sensor readings), which is a wide range of domains, but we argue that it is suitable for applications that seek abnormality prediction more than other goals.

8.4 Future Work

Presenting the limitations in the previous section could give some hints about the directions that could be followed for future improvements and updates.

Basically, more research should be carried out to improve the performance of the learning algorithms. Shapelet learning approaches are becoming more and more popular and fast algorithms may surface in the upcoming years. Exploiting faster and more efficient algorithms in the **USE** part would be a priority. Then and since at the **SEE** phase, time series are being encoded with the shapelets that appear in them, this could be considered as a representing continuous numerical time series with discrete representations, and then sequence mining algorithms could be exploited to speed up the learning. Specifically, state-of-the-art approaches about sequence mining from strings should be surveyed, and approach that could deal with high dimensions should be integrated.

Also from an optimization perspective, big data technologies and frameworks should be exploited. Only using these frameworks would be enough to speed up the processing and the learning time. For example, a framework such as Apache Flink[1] would distribute the learning and make the most out of the available resources. Additionally, Flink supports complex event processing for real-time analysis, and more studies should be performed to assess its advantages over other CEP engines such as Esper.

Another potential extension is to support the learning from categorical variables as well. Some application fields and a considerable amount of sensors produce categorical and boolean data. Without supporting this data type, our approach could not be propagated to such domains, and it could not handle such sensor readings. As previously mentioned, the literature is full of approaches that support this type of learning, and they even consider earliness. Therefore, more studies and surveys about this aspect should be carried out in the future.

[1] https://flink.apache.org/

APPENDIX A

AUTOCEP WITH UNIVARIATE TIME SERIES

AutoCEP started as a project that worked only with univariate time series. In the main flow of the dissertation we discussed the recent version that works with **USE & SEE**. This appendix is dedicated to discuss the early version of **autoCEP**. Therefore, and unless otherwise noted, any mention of **autoCEP** in this appendix is intended to reference the early univariate version

Even in its earlier versions, **autoCEP** exploited data mining and CEP to devise an easy-to-use solution that would allow for the CEP technology to be employed with ease in the business process management domains. Differently from current usage of CEP, **autoCEP** also make its predictive capabilities available.

At this stage, **autoCEP** operated through two consecutive phases. Figure A.1 sheds some light on the framework from a high-level perspective. The first phase consumes the classified history records in order to produce the most useful shapelets. These shapelets are inputted into the second phase in order to build the proper deploy-ready CEP rules from them.

A.1 CEP Rules: Reintroduced

This section is going to reintroduce CEP rules from the perspective of this version of **autoCEP**.

CEP rules are defined using different CEP operators like windowing, selection, sequence, etc. These operators are considered the main enabler to define complex patterns. Regardless of the various concrete models, we will keep an abstract representation for rules that could be expressed in any description language, and that captures the necessary operators. In this work, a CEP rule is divided into three blocks. First the timeframe (or window) of the rule, which is defined using the **within** construct, it indicates the maximum borders of a pattern. Second the

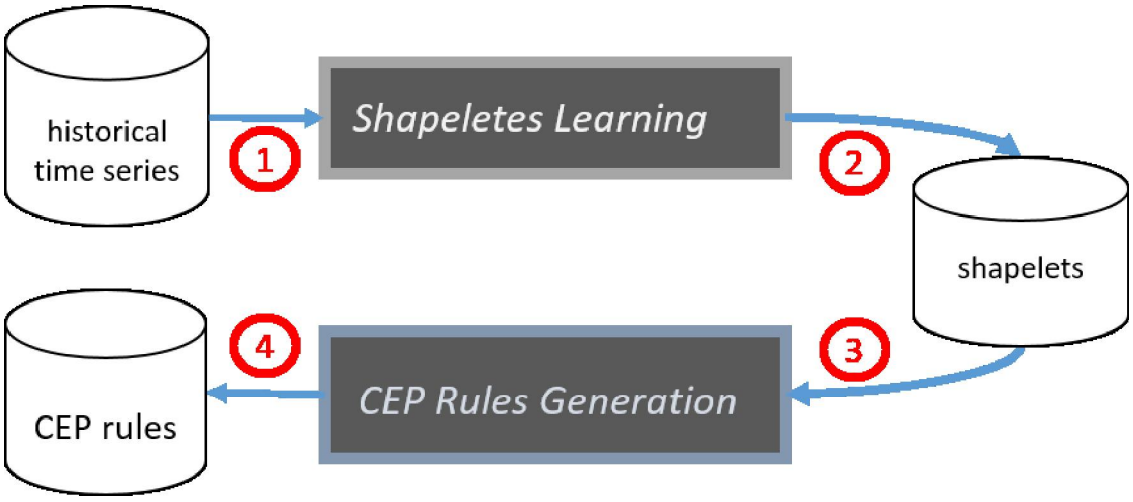

Figure A.1: Two Phase High Level Framework of AutoCEP

filter block, which contains the events that are relevant for the rule (events within the time window), they will be written between two curly brackets {}. Finally the conditions that need to be met on the captured sequence of events in order for the rule to be fired, this block is defined using the **where** construct. In general:

(A.1) **within**[*window*] {*relevant events*} **where**[*conditions*]

Supposing that we have a temperature event E, the following pattern shows an example of a rule:

P_1 :
within[30min] {E} **where**[$avg(E.temp) > 20$]

Pattern P_1 is fired only if the average of temperature values within 30 minutes is greater than 20°C.

A.2 First Phase: Shapelets Learning

This stage contains our learning algorithm that can extract shapelets with the highest utility scores by learning them from historical time series. This algorithm is the outcome of surveying recent state-of-the-art approaches [207, 200, 115, 65, 203] regarding this kind of classification problems.

Algorithm 11 takes as inputs the training data set D, which is a pair of time series along side their classes. It also takes two other optional parameters ($minLength$ and $maxLength$). These are specified by domain experts in case there are any preferences or prior knowledge about the shapelets' lengths (i.e., they want to learn shapelets between these lengths). This may guide the learning process and decrease the learning time (later on this in the evaluation section).

However, if no such preferences or knowledge exist, then users may leave the default values that are $minLength = 1$ and $maxLength = \min_{T \in D} |T|$, and then the algorithm will learn by its own the shapelets of the best lengths. On the other side, the algorithm outputs a set of trained shapelets $\hat{S}$ that have the highest utility scores.

Algorithm 11: The Learning Algorithm

Input: Training dataset D, $minLength$, $maxLength$
Output: A set of shapelets $\hat{S}$ with the highest utility scores

```
1  Ŝ ← ∅;
2  for each time series T ∈ D do
3      for l ← minLength to maxLength do
           /* for each shapelet length                          */
4          for k ← 1 to |T| − l + 1 do
               /* create an empty candidate shapelet ŝ          */
5              ŝ.s ← buildShapelet(T, k, l);
6              ŝ.s_c ← class(T);
7              similarities ← calculateSimilarities(D, ŝ);
8              ŝ.δ ← calculateDelta(D, similarities, ŝ);
9              calculateUtilityScore(D, ŝ);
10             Ŝ.add(ŝ);
11 Ŝ ← pruneShapelets(Ŝ);
12 return Ŝ;
```

First the algorithm (algo 11) loops over the time series T in the training dataset D, then for the specified minimum and maximum lengths of the shapelets, and for each starting position k, it creates an empty shapelet $\hat{s}$ and continues to define its attributes (i.e., s, δ, c_s).
s that constitutes $\hat{s}$ is defined by using the $buildShapelet(T, k, l)$ function. This function simply outputs a subsequence $\hat{s}.s$ from a given time series T. The first element in $\hat{s}.s$ is the k^{th} element of T, and $|\hat{s}.s| = l$. So the output of $buildShapelet(T, k, l)$ is $\hat{s}.s = \{T[k], T[k+1], ..., T[k+l-1]\}$. This is called brute force building of the shapelets, and basically the algorithm extracts every subsequence from every training time series within the specified lengths.
Afterward, the algorithm specifies the class of the shapelet c_s as the class of the current time series T.
The next step is to calculate the similarities between the created shapelet and the whole training data set. The similarities simply stand for the distances between $\hat{s}$ and all $T \in D$ by applying equation 3.2. They are represented as an array of real values:

$$\forall i \in \{1, ..., |D|\}, similarities[i] = ||\hat{s}, T_i||_{bmd} \tag{A.2}$$

A.2.1 Learning the Distance Threshold

As explained in the definition about shapelets, distance thresholds are used for classifying new instances. To this end, different methods have been proposed in the literature to best learn and estimate this kind of thresholds. Some of the most popular and accurate methods are the Chebyshev's inequality [203] and the information gain split [143]. In this work we adopt the information gain-based technique, since it has been successfully adopted in data mining.
After attributing the sequence of real values s and the class c_s to the created shapelet $\hat{s}$, and after calculating the similarities, algorithm 11 can learn now the best distance threshold δ for this specific shapelet by calling the $calculateDelta$ function. This function is an algorithm by its own and before explaining it we will highlight some needed concepts and definitions.

Given a specific shapelet $\hat{s}$, a time series T can be either positive or negative regarding $\hat{s}$. T belongs to the positive instances, denoted as I^+, if its associated class is the same as the class of the concerned shapelet. On the contrary, T is an element in the set of negative instances, I^-, if it is labeled with a different class than the shapelet. Therefore, given a shapelet $\hat{s} = (s, \delta, c_s)$ and a dataset of time series D, positive and negative instances are defined as (these are different for each shapelet):

$$I^+ = \{T \in D \mid class(T) = c_s\} \tag{A.3}$$

$$I^- = \{T \in D \mid class(T) \neq c_s\} \tag{A.4}$$

Second, given a dataset D, a set of finite classes C, and L_c designates the number of time series in D that belong to the class $c \in C$, the entropy E_D of the dataset is defined as:

$$E_D = -\sum_{c \in C} \frac{L_c}{|D|} \, log(\frac{L_c}{|D|}) \tag{A.5}$$

The best δ is the best distance that can separate positive instances I^+ from the negative counterparts I^-. To clarify the mechanism of getting this split figure A.2 presents an example, it draws the values of the similarity array (see eq. A.2) on an x-axis. The positive instances (shaped as + on the figure) will be gathered on the left side of the axis due to their relatively small distances from - high similarities with - the shapelet, and the negative ones (shaped as x on the figure) will be grouped on the other side, but of course this does not need to be a pure way as sometimes they may blend (this is shown in the figure as well). To find the best split, the algorithm takes the mid-point between two consecutive distances as a candidate distance threshold $\hat{\delta}$. This $\hat{\delta}$ splits the available instances into two datasets, D_L and D_R for the left and the right sides respectively. Then the Information Gain (IG) is computed using equation A.6, where D is the whole dataset, E_L and E_R are the entropies of D_L and D_R respectively (calculated using eq. A.5). The calculated IG will show the gain in information that results from the split by using this specific $\hat{\delta}$. Finally we select the $\hat{\delta}$ that maximizes IG to be the real δ of the shapelet. This logic is listed in algorithm 12.

Figure A.2 also shows two candidate distance thresholds δ_1 and δ_2. As sketched, δ_1 has one false positive and one false negative (1 "+" on the right side and 1 "x" on the left side), whereas δ_2

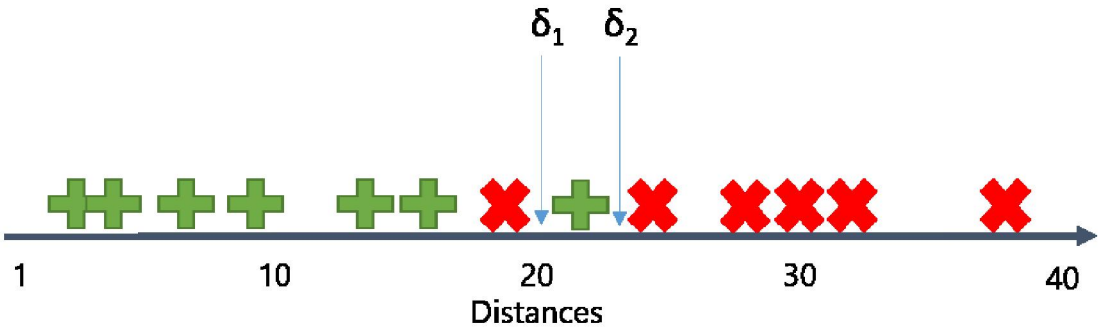

Figure A.2: Example to Clarify The Distance Threshold

has only one false positive (1 "x" on the left side). The obtained IG by splitting using δ_2 will be higher, and δ_2 will eventually be selected.

$$IG = E_D - \frac{|D_L|}{|D|}E_L - \frac{|D_R|}{|D|}E_R \tag{A.6}$$

A.2.2 Utility Score

Continuing algorithm 11, after learning the distance threshold, every shapelet needs to have a utility score. This score verifies the usefulness of the shapelet. As with the distance thresholds, many utility score functions had been proposed in the literature, like the weighted F_1 score [203] and the weighted information gain [65]. However this work uses another costume utility score function that takes into account the frequency, the discrimination, and the earliness of a shapelet.

To list the formula that computes the utility score, we need to first define a predicate that given a shapelet $\hat{s}$ and a time series T, it can state if $\hat{s}$ covers T. So we will define the Boolean operator $\cong$ as:

$$\hat{s} \cong T = \begin{cases} true & \text{if } ||\hat{s}, T||_{bmd} \leq \delta \\ false & \text{otherwise} \end{cases} \tag{A.7}$$

$\hat{s} \cong T$ is read as the shapelet $\hat{s}$ covers the time series T. This signals that $\hat{s}$ and T are similar (the distance between them is less or equal to the distance threshold δ of $\hat{s}$).
Given a dataset D and a shapelet $\hat{s}$, the subset of D that contains time series covered by $\hat{s}$ is denoted as $\hat{D}$, and is defined as:

$$\hat{D} = \{T \in D \mid \hat{s} \cong T\} \tag{A.8}$$

Given a shapelet $\hat{s}$ and a time series T, we define the Earliest Matching Time ($EMT(\hat{s}, T)$) as the point in time when $\hat{s}$ covers T, i.e., how many values we have read from T before assessing that

Algorithm 12: Learning the Distance Threshold

Input: Training dataset D that contains the time series T, the array of $similarities$ that we have computed before, the shapelet $\hat{s}$ that we are building

Output: Assign the best distance threshold δ to $\hat{s}$

```
1  maxIG ← 0;
   /* Sort the similarities array in an increasing order                    */
2  sortedSim ← sort(similarities);
3  L_sim ← length(similarities);
4  for i ← 0 to L_sim − 1 do
       /* Get the candidate distance threshold as the mid-point between two
          consecutive distances                                             */
5      δ̂ ← (sortedSim[i] + sortedSim[i+1]) / 2;
       /* Initialize the true positive t+, false positive f+, true negative
          t−, and false negative f− counters                                */
6      t+ ← f+ ← t− ← f− ← 0;
7      for j ← 0 to L_sim do
8          if similarities[j] ≤ δ̂ then
9              if class(T_j) = ŝ.c_s then
10                 t+ ← t+ + 1;
11             else
12                 f+ ← f+ + 1;
13         else
14             if class(T_j) ≠ ŝ.c_s then
15                 t− ← t− + 1;
16             else
17                 f− ← f− + 1
18                 ;
19     E_L ← −(t+/(t+ + f+)) log(t+/(t+ + f+)) − (f+/(t+ + f+)) log(f+/(t+ + f+));
20     E_R ← −(t−/(t− + f−)) log(t−/(t− + f−)) − (f−/(t− + f−)) log(f−/(t− + f−));
21     IG calculated using eq. A.6;
22     if IG > maxIG then
23         maxIG ← IG;
24         ŝ.δ ← δ̂;
```

it is similar to $\hat{s}$.
Taking all the above definitions into consideration, and using equation A.5 to calculate the entropy of a dataset, the utility score U of a given shapelet $\hat{s}$ and a given training data set D is defined as:

$$U = w * (E_D - E_{\hat{D}}) \times \frac{1}{|D|} \sum_{T \in \hat{D}} \frac{1}{EMT(\hat{s}, T)} \tag{A.9}$$

In this equation, w is a parameter to weight frequency against earliness, i.e., if they matter the same then w should be set to equal 1.

A.2.3 Pruning the Shapelets

At this stage, the algorithm will have a very large set of shapelets in which only few are really important. So we need to prune this set, and keep only the most useful shapelets. To this end, different pruning algorithms could be used, ranging from top-k to more complex ones.
In our work we implemented different pruning algorithms that can efficiently select the best patterns, we discuss one of them that showed good results in the evaluation.
The algorithm sorts the set of shapelets in a decreasing order regarding their utility scores. It loops over the sorted set, it selects the next shapelet, and then it checks which other shapelets are covered by the selected one. All the covered shapelets are considered similar, and they are pruned. The algorithm continues to perform the aforementioned steps for each shapelet as long as the input set is not empty yet. This is explained in algorithm 13.

Algorithm 13: Pruning the Set of Shapelets

Input: The set of all the extracted shapelets $\hat{S}$
Output: A set of filtered shapelets $\hat{FS}$

```
Ŝ ← sort(Ŝ);
F̂S ← ∅;
for each shapelet ŝ in Ŝ do
    Ŝ.remove(ŝ);
    F̂S.add(ŝ);
    for each other shapelet ôs in Ŝ do
        if ŝ ≅ ôs then
            Ŝ.remove(ôs);
return F̂S
```

A.3 Second Phase: CEP Rules Generation

At this point, the learned shapelets serve as inputs, where they will be automatically transformed into CEP rules to be used later for predictions. Typically, in data mining approaches,

ad-hoc classification algorithms are devised to test and use the different learned shapelets, and these algorithms do not fit for real-time usage [203, 115, 65]. Therefore in our work, autoCEP transforms the shapelets into CEP rules, creating a general approach that could be applied to any numeric periodic attributes, and that benefits from the speed and the scalability of CEP engines, without the need to implement ad-hoc classification algorithms.

The proposed algorithm in this phase extracts the three building blocks of the rule from the input shapelets and their parameters.
For each shapelet a CEP rule is created, and thus we overcome the limitation of assuming just one rule for each composite event, which is the assumption that is made by other approaches [121, 122].
Given a shapelet $\hat{s} = (s, \delta, c_s)$, the window parameter win for the **within** block is derived directly from the length of the shapelet. The algorithm calculates the size of the steps (e.g., each 1 second) between the elements that constitute the shapelet, and subsequently the window is deduced from the whole length of the shapelet. Then the relevant stream of events are of the same type as the elements that constitute the sequence s of the shapelet. Finally, the condition to be met in order to predict if a stream of incoming events correspond to c_s (the same class as the shapelet) is that $\hat{s}$ needs to cover (eq. A.7) the stream within the window win. This is listed in the following algorithm (algo. 14).

Algorithm 14: Transforming Shapelets into CEP Rules

Input: A set of shapelets $\hat{S}$
Output: A set of CEP rules $rules$

```
1  rules ← ∅;
2  for each shapelet ŝ in Ŝ do
       /* create an empty cep rule cep                                  */
3      win ← |ŝ.s|;
4      cep.setWindowBlock(win);
5      E ← Extract event types from s;
6      cep.setEventTypes(E);
7      cep.setConditionBlock(ŝ ≅ E);
       /* The above line means: cep.setConditionBlock(||ŝ,E|| ≤ δ). If we go
          back to eq.A.7 and eq.3.1                                     */
8      cep.setListener( this stream is predicted to belong to the class c_s);
       /* Listeners of the rules are set by experts, they are called in
          case the rule is fired (the conditions are met), and they are
          different depending on the domain. The default listener that we
          provide in our algorithm is to alert about the prediction and
          about the time when it was done                               */
9      rules.add(cep);
10 return rules;
```

Algorithm 14 outputs a rule for each shapelet $\hat{s} = (s, \delta, c_s)$ of the form:

(A.10) **within**[$|s|$] $\{E\}$ **where**[$||\hat{s}, E|| \leq \delta$]

A.4 Experiments

The approach had been tested with different data sets from various domains. More precisely, we have selected 4 of the most used data sets from the UCR time series archive [90] as a benchmark to evaluate our approach and to compare it with different state-of-the-art proposals. Afterward, we tested the approach on logs of real artworks transportation processes to predict challenging in-activities SLO violations such as temperature variations. All experiments were conducted on a PC with an Intel i7 2.8GHz CPU and 32 GB main memory. The algorithms were written in Java, with a JVM heap size of 4 GB.

Interested readers are encouraged to download the complete solution that we have implemented from GitHub[1]. The Git repository contains a short demonstrative video as well.

A.4.1 UCR Time Series Archive

The results of the four data sets are listed in the shown table A.4.1. More precisely we have used 4 UCR data sets (ECG, CBF, Gun-Point as Gun, and Synthetic Control as Syn) that are highlighted in bold in addition to some information that relates to each one of them. These information include the number of classes (cl), number of training (train) and testing (test) instances, and finally the length of the time series (l).

On each data set, we have compared our approach with 8 different state-of-the-art methods. The different compared methods are (as shown in the tables): autoCEP, EDSC-CHE [203], EDSC-KDE [203], ECTS [201], RECTS [202], the 1NN classifier with its three variations (information about these 1NN classifiers could be found in [90, 201, 202]), and finally SCR [200]. The shown results include the accuracy (Acc.) and the average length (Avg. Len.) for each approach. The average length is a factor showing the average lengths of the patterns that were used to classify the given time series, so it could hint about the earliness, and therefore the smaller the better – hence, a short pattern could classify a time series earlier than longer ones. The shapelets learning algorithm of autoCEP was always executed with the default parameters for $minLength$ and $maxLength$ (refer to section A.2).

As depicted in the tables, autoCEP competes with current state-of-art approaches from an accuracy perspective, as it always scored close results. However, regarding the earliness, our method outperformed the others. Therefore autoCEP was able to find the shortest possible length to classify a time series without deteriorating the accuracy.

[1] https://github.com/rmgitting/autoCEP

Table A.1: Accuracy and Earliness Comparison

ECG: cl=2; train=100; test=100; l=96		
Approach	Acc.	Avg. Len.
autoCEP	86%	20%
EDSC-CHE	82%	25%
EDSC-KDE	88%	32%
ECTS	89%	77%
RECTS	89%	60%
Early 1NN	89%	86%
Fixed 1NN	89%	92%
Full 1NN	88%	100%
SCR	73%	39%

CBF: cl=3; train=50; test=900; l=128		
Approach	Acc.	Avg. Len.
autoCEP	82%	22%
EDSC-CHE	87%	35%
EDSC-KDE	85%	36%
ECTS	85%	71%
RECTS	85%	71%
Early 1NN	86%	80%
Fixed 1NN	83%	42%
Full 1NN	85%	100%
SCR	55%	35%

Syn: cl=6; train=300; test=300; l=60		
Approach	Acc.	Avg. Len.
autoCEP	88%	33%
EDSC-CHE	87%	55%
EDSC-KDE	90%	55%
ECTS	89%	89%
RECTS	88%	87%
Early 1NN	88%	91%
Fixed 1NN	88%	100%
Full 1NN	88%	100%
SCR	58%	50%

Gun: cl=2; train=50; test=150; l=150		
Approach	Acc.	Avg. Len.
autoCEP	90%	28%
EDSC-CHE	94%	46%
EDSC-KDE	94%	46%
ECTS	86%	46%
RECTS	86%	46%
Early 1NN	87%	71%
Fixed 1NN	91%	92%
Full 1NN	91%	100%
SCR	62%	77%

A.4.1.1 Sensitivity of Parameters

The aforementioned experiments were done using the default settings of autoCEP, i.e., regarding the lengths of the patterns to learn – $minLength$ and $maxLength$ (refer to section A.2) – and also the w parameter is set to 1 in equation A.9. In this section we use the challenging CBF data set (50 training and 900 testing instances) to test how changes to these parameters could affect the result of autoCEP.
Figure A.3 illustrates the results of the tests.

The upper part depicts the accuracy and average length scores when w of equation A.9 is changed. As expected, favoring one aspect over the other could potentially change the results of autoCEP, yet the result changes are not big. With $w < 1$, then earliness is favored over frequency and autoCEP will try to search for shorter patterns on the expense of slightly smaller accuracy. On the other side, favoring accuracy will direct autoCEP to give less importance for shorter patterns while pruning the shapelets.

The bottom part of the figure demonstrates the effect of changing the minimum and maximum

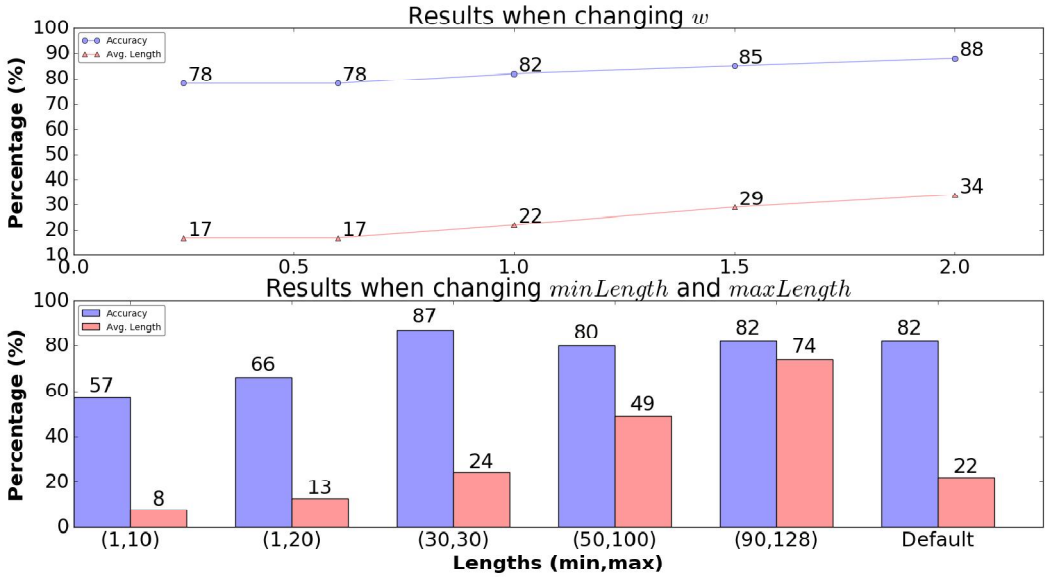

Figure A.3: Testing the Sensitivity of Parameters

lengths of the shapelets, i.e., autoCEP will try to find shapelets only within these lengths. As expected, given the algorithm a small space with short lengths (min=1,max=10) or (min=1,max=20) will drastically affect the accuracy as very short shapelets may not be discriminative and will yield false positives. However with longer patterns autoCEP always performed better by keeping the accuracy higher than 80%, but of course with different average lengths as this factor depends directly on the shapelet lengths. Entering the shapelet lengths is a good way for experts to get involved in the learning process and to guide the algorithm. However as the experiments show, the default values for these length could be used by non-experts, and the algorithm will still score high results.

A.4.1.2 Efficiency

The algorithm that learns the shapelets is a brute force learning algorithm, and therefore it has a tradeoff between good results and learning time. For the experiments that we held, we tried two versions of our algorithm: a straightforward implementation and an optimized one (autoCEP-*O*) with the help of concurrency and parallel computing.

Table A.2 compares the two implementations of autoCEP with other two approaches [203]. As this table shows, a major drawback of our approach is the learning time, but still the training is a design-time phase and the optimized version of autoCEP can significantly improve its time. Another point that is important to highlight is that autoCEP in its default settings searches for the optimal patterns without any guidance, but if $minLength$ and $maxLength$ are to be provided then this will drastically affect the learning time. For example, running the optimized version on the CBF data set with $minLength = 30$ and $maxLength = 40$ reduced the learning time

Table A.2: Learning Time Comparison

Data Set	Sample	Length	autoCEP	autoCEP-*O*	EDSC-CHE	EDSC-KDE
CBF	30	128	312 sec	106 sec	37 sec	41 sec
ECG	100	96	823 sec	341 sec	123 sec	137 sec
Syn	300	60	867 sec	344 sec	252 sec	332 sec
Gun	50	150	1149 sec	670 sec	165 sec	170 sec

from 106 seconds to only 17 seconds.

Classification time on the other hand, is negligible, always smaller than 0.005 seconds. This fast classification is due to the employment of the CEP technology that can process events and match them against CEP rules faster than other ad-hoc algorithms used by different approaches.

A.4.2 Artworks Transportation Data Set

In this kind of transport processes, the involved parties are interested in analyzing temperature readings and predict them in advance to prevent violations whenever possible (i.e., trespassing a minimum or a maximum threshold). These violations will eventually affect the qualities of the transported piece of arts, and violate SLO agreements. A dedicated framework such as the Butterfly and autoCEP could handle such cases in a running process, in case they are integrated into BPMSs.

Table A.3: Training and Testing Data Sets

	Violated Scenarios	Normal Scenarios	Longest Series	Shortest Series
Training	16	17	451	51
Testing	17	17	460	39

Table A.3 presents information about the training and the evaluation data sets[2] used for this experiment. As depicted in the table, these data sets have time series with different length, and therefore we will measure the earliness of the approach using the formula in equation A.11 (EMT is discussed in section A.2.2) instead of the average length used above. This factor describes how much in advance we predicted the violation, i.e., how much time remains before the violation occur, hence, the greater the better.

$$\frac{1}{|D|}\sum_{T\epsilon D}\frac{|T|-EMT(\hat{s},T)}{|T|} \tag{A.11}$$

[2]These data sets are provided by our partner in the project: C2RMF http://c2rmf.fr/

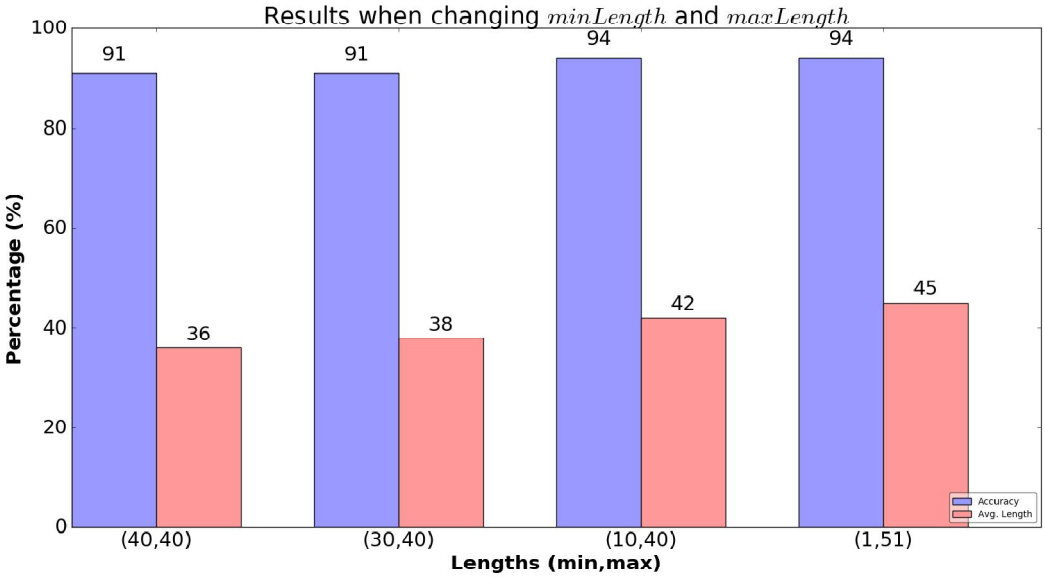

Figure A.4: SLO Violation Prediction

Figure A.4 depicts the accuracy and the earliness of the framework when given different spaces to build shapelets (different min and max).

The experiments on predicting temperature violation prove the benefits of using such approach for anticipating challenging SLO violations in manual process environments. They also show that the earliness and the accuracy are slightly affected by the values of the maximum and minimum lengths (fig. A.4), as giving the framework more space to search for patterns will eventually help it to detect more useful and smaller shapelets. Nonetheless, the autoCEP framework proved to maintain high accuracies regardless of the shapelet lengths but with different earliness percentages, i.e., it was always able to successfully classify the evaluation instances.

A.5 Summary

This appendix discussed with details the early version of **autoCEP**. The main two phases are thoroughly explained. Finally, different experiments on a real life scenario and the UCR data sets are presented.

APPENDIX B

RESEARCH METHODOLOGY

Achieving optimal answers to the problems and research questions that we have asked throughout this dissertation seem to be very beneficial for CEP and BPM users. Indeed, a complete framework that could allow for the CEP technology to be used easily for predictive purposes, and at the same time, that could seamlessly integrate these predictive capabilities within BPM systems, is practical and very advantageous in real life. However, and before the answers, reaching the point of defining these questions and finding these problems at the first place, was the fruit of a profound research process. In this section, the story of how this work has culminated will be highlighted.

1. **Problem Definition:** Like many other research efforts, our work typically started with many attempts to define and properly formulate the problems that we are targeting. We approached this subject from a practical point of view with the help of our partner[1]. Specifically, we aimed at addressing the challenges of transporting artworks (i.e., transporting sensitive objects in general). First, we modeled transportation processes through typical BPM systems, the thing has helped us to gain some visibility, but it did not provide much support on how to proactively manage monitorable tasks —the trucking activity itself for example, as most of the violations happened during such activities. Afterward, we sought the help of CEP, but the integration was not obvious, and given that the CEP is a reactive technology, then it was impossible to handle situations proactively. Subsequently, we searched for predictive CEP solutions, but again we encountered visions and conceptions that were away from being realized. The conclusion of this step was the problems and research questions that we previously noted.

2. **State-of-the-art:** During our research, the first step will always systematically carries us to this step. Whenever we wanted to utilize, understand, and inspect a technology, we would thoroughly go over the literature. For example, when we wanted to check the support of

[1] C2RMF: http://c2rmf.fr/

BPM for transport processes, the integration of CEP and BPM, and prediction in the CEP domain, all these topics drove us to fully inspect state-of-the-art approaches. Additionally, a point worth mentioning is that we looked at innovative ways of exploiting a specific technology. For instance, CEP is mainly used for the detection of situations of interest, however we sought its beneficial real-time capabilities for prediction, and this compelled us to keep an up-to-date view on all existing and emerging work from research perspectives. As mentioned, the prior step shaped the research questions, and this step helped us to answer most of them.

3. **Solution:** Studying the literature has answered most of our questions, however it also left others that are waiting to be answered. Having inspected the different state-of-the-art approaches, stating their strengths and weaknesses, gave us clear ideas about what contributions could be held in the domain. Specifically, it turned out that it is up to our solution to address some fundamental issues. Such as promoting shapelet learning algorithms over multivariate time series, changing the notion of predictive CEP from mere visions into a practical approach, the real and effective jump from data mining into CEP, and the seamless integration within the BPM world.

4. **Validation & Evaluation:** The validation of our work was performed through two main scenarios. The first is from the domain of smart manufacturing, where a complete practical example from data mining to BPM is discussed and validated. The second serves as a support and use case for our proof-of-concept when it comes to the context-aware management of business process instances. It is from the domain of artwork transportation. Moreover, the overall approach is evaluated on publicly available benchmarks, and then it is compared to other state-of-the-art proposals. At the end, the contributions are evaluated and well-positioned among other work in the literature.

These steps summarize our research approach. It is important to note that they are not sequential, but we revisited each step many times whenever needed.

APPENDIX C

LIST OF PUBLICATIONS

This appendix contains the list of accepted publications[1] throughout the thesis.

1. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Predictive Analysis of BPM tasks with autoCEP. To appear in BPM 2017.
2. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Automatic Learning of Predictive CEP Rules: Bridging the Gap between Data Mining and Complex Event Processing. DEBS 2017.
3. Raef Mousheimish, Yehia Taher, Karine Zeitouni, Michel Dubus: Smart Preserving of Cultural Heritage with PACT-ART. Journal: Multimedia Tools and Applications 2017.
4. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Apprentissage automatique de règles CEP prédictives: combler le gap entre fouille de données et traitement des événements complexes. To appear in BDA 2017.
5. Raef Mousheimish, Yehia Taher, Karine Zeitouni: autoCEP: Automatic Learning of Predictive Rules for Complex Event Processing. ICSOC 2016: 586-593
6. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Complex event processing for the non-expert with autoCEP: demo. DEBS 2016: 340-343
7. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Automatic learning of predictive rules for complex event processing: doctoral symposium. DEBS 2016: 414-417
8. Raef Mousheimish, Yehia Taher, Karine Zeitouni: The Butterfly: An Intelligent Framework for Violation Prediction within Business Processes. IDEAS 2016: 302-307

[1] Submissions that are under review are not listed

9. Raef Mousheimish, Yehia Taher, Karine Zeitouni, Michel Dubus: PACT-ART: Enrichment, Data Mining, and Complex Event Processing in the Internet of Cultural Things. SITIS 2016: 476-483

10. Raef Mousheimish, Yehia Taher, Karine Zeitouni: Toward the Support of Challenging Service Level Agreements (SLAs) in Manual and Context-Dependent Activities. COMPSAC Workshops 2016: 38-43

11. Raef Mousheimish, Yehia Taher, Karine Zeitouni, Michel Dubus: PACT-ART: Adaptive and context-aware processes for the transportation of artworks. Digital Heritage 2015: 347-350

12. Raef Mousheimish, Yehia Taher, Béatrice Finance: Towards Smart Logistics Process Management. EC-Web 2015: 127-138

13. Raef Mousheimish, Yehia Taher, Béatrice Finance: Towards smart logistics processes: a predictive monitoring and proactive adaptation approach. ICSSP 2015: 169-170

APPENDIX D

LIST OF ACRONYMS

This appendix contains the list of all acronyms used throughout the thesis.

- **CEP:** Complex Event Processing
- **BPM:** Business Process Management
- **IoT:** Internet of Things
- **USE:** Univariate Shapelet Extractor
- **SEE:** SEquence Extractor
- **TAS:** Time-Annotated Sequence
- **ARIMA:** Autoregressive Integrated Moving Average models
- **RFID:** Radio Frequency Identification
- **ECTS:** Early Classification on Time Series
- **API:** Application Programming Interface
- **PAA:** Piecewise Aggregate Approximation
- **APCA:** Adaptive Piecewise Constant Approximation
- **IBM:** International Business Machines
- **SSV:** Segmented Sum of Variations

- **PLR:** Piecewise Linear Representation
- **PIP:** Perceptually Important Point
- **SAX:** Symbolic Aggregate Approximation
- **DFT:** Discrete Fourier Transform
- **DWT:** Discrete Wavelet Transform
- **DTW:** Dynamic Time Warping
- **HMM:** Hidden Markov Model
- **SVM:** Support Vector Machine
- **BoF:** Bag of Features
- **BOP:** Bag of Patterns
- **FS:** Fast Shapelet
- **ST:** Shapelet Transform
- **LS:** Learned Shapelets
- **DBMS:** Database Management System
- **RTPS:** Real-Time Processing System
- **DSMS:** Data Stream Management System
- **SQL:** Structured Query Language
- **CQL:** Continuous Query Language
- **XML:** eXtensible Markup Language

BIBLIOGRAPHY

[1] R. AGRAWAL, C. FALOUTSOS, AND A. SWAMI, *Efficient similarity search in sequence databases*, Foundations of data organization and algorithms, (1993), pp. 69–84.

[2] H. C. ANDRADE, B. GEDIK, AND D. S. TURAGA, *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*, Cambridge University Press, 2014.

[3] D. ANICIC, P. FODOR, S. RUDOLPH, R. STÜHMER, N. STOJANOVIC, AND R. STUDER, *Etalis: Rule-based reasoning in event processing*, in Reasoning in event-based distributed systems, Springer, 2011, pp. 99–124.

[4] S. APPEL, S. FRISCHBIER, T. FREUDENREICH, AND A. BUCHMANN, *Event stream processing units in business processes*, in Business Process Management, Springer, 2013, pp. 187–202.

[5] A. ARASU, S. BABU, AND J. WIDOM, *The cql continuous query language: semantic foundations and query execution*, The VLDB Journal-The International Journal on Very Large Data Bases, 15 (2006), pp. 121–142.

[6] W. G. AREF, M. G. ELFEKY, AND A. K. ELMAGARMID, *Incremental, online, and merge mining of partial periodic patterns in time-series databases*, IEEE Transactions on Knowledge and Data Engineering, 16 (2004), pp. 332–342.

[7] K. J. ÅSTRÖM, *On the choice of sampling rates in parametric identification of time series*, Information Sciences, 1 (1969), pp. 273–278.

[8] B. BABCOCK, S. BABU, M. DATAR, R. MOTWANI, AND J. WIDOM, *Models and issues in data stream systems*, in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2002, pp. 1–16.

[9] K. BACHE AND M. LICHMAN, *Uci machine learning repository. university of california, irvine*, 2013.

[10] A. BAGNALL AND G. JANACEK, *A run length transformation for discriminating between auto regressive time series*, Journal of classification, 31 (2014), pp. 154–178.

[11] A. BAGNALL, E. KEOGH, S. LONARDI, G. JANACEK, ET AL., *A bit level representation for time series data mining with shape based similarity*, Data Mining and Knowledge Discovery, 13 (2006), pp. 11–40.

[12] A. BAGNALL, J. LINES, A. BOSTROM, J. LARGE, AND E. KEOGH, *The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances*, Data Mining and Knowledge Discovery, (2016), pp. 1–55.

[13] G. E. BATISTA, E. J. KEOGH, O. M. TATAW, AND V. M. DE SOUZA, *Cid: an efficient complexity-invariant distance for time series*, Data Mining and Knowledge Discovery, 28 (2014), pp. 634–669.

[14] A. BAUMGRASS, D. CICCIO, C. CLAUDIO, R. DIJKMAN, M. HEWELT, J. J. MENDLING, A. A. MEYER, S. S. POURMIRZA, M. M. WESKE, AND T. WONG, *Get controller and unicorn: Event-driven process execution and monitoring in logistics*, CEUR Workshop Proceedings, 2015.

[15] A. BAUMGRASS, N. HERZBERG, A. MEYER, AND M. WESKE, *Bpmn extension for business process monitoring.*, in EMISA, 2014, pp. 85–98.

[16] M. G. BAYDOGAN, G. RUNGER, AND E. TUV, *A bag-of-features framework to classify time series*, IEEE transactions on pattern analysis and machine intelligence, 35 (2013), pp. 2796–2802.

[17] D. BERNDT AND J. CLIFFORD, *Finding patterns in time series in advances*, U. Fayyad et al.(eds.) Knowledge Discovery and Data Mining, (1996).

[18] D. J. BERNDT AND J. CLIFFORD, *Using dynamic time warping to find patterns in time series.*, in KDD workshop, vol. 10, Seattle, WA, 1994, pp. 359–370.

[19] C. M. BISHOP, *Pattern recognition*, Machine Learning, 128 (2006), pp. 1–58.

[20] A. BOSTROM AND A. BAGNALL, *Binary shapelet transform for multiclass time series classification*, in International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2015, pp. 257–269.

[21] G. E. BOX, G. M. JENKINS, G. C. REINSEL, AND G. M. LJUNG, *Time series analysis: forecasting and control*, John Wiley & Sons, 2015.

[22] T. BOZKAYA AND M. OZSOYOGLU, *Distance-based indexing for high-dimensional metric spaces*, in ACM SIGMOD Record, vol. 26, ACM, 1997, pp. 357–368.

[23] L. BRENNA, A. DEMERS, J. GEHRKE, M. HONG, J. OSSHER, B. PANDA, M. RIEDEWALD, M. THATTE, AND W. WHITE, *Cayuga: a high-performance event processing engine*, in Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, 2007, pp. 1100–1102.

[24] K. BRODA, K. CLARK, R. MILLER, AND A. RUSSO, *Sage: a logical agent-based environment monitoring and control system*, in European Conference on Ambient Intelligence, Springer, 2009, pp. 112–117.

[25] S. BÜLOW, M. BACKMANN, N. HERZBERG, T. HILLE, A. MEYER, B. ULM, T. Y. WONG, AND M. WESKE, *Monitoring of business processes with complex event processing*, in Business Process Management Workshops, Springer, 2013, pp. 277–290.

[26] C. CABANILLAS, A. BAUMGRASS, J. MENDLING, P. ROGETZER, AND B. BELLOVODA, *Towards the enhancement of business process monitoring for complex logistics chains*, in Business Process Management Workshops, Springer, 2013, pp. 305–317.

[27] C. CABANILLAS, C. DI CICCIO, J. MENDLING, AND A. BAUMGRASS, *Predictive task monitoring for business processes*, in Business Process Management, Springer, 2014, pp. 424–432.

[28] A. CAMERRA, J. SHIEH, T. PALPANAS, T. RAKTHANMANON, AND E. KEOGH, *Beyond one billion time series: indexing and mining very large time series collections with isax2+*, Knowledge and information systems, 39 (2014), pp. 123–151.

[29] M. S. CETIN, A. MUEEN, AND V. D. CALHOUN, *Shapelet ensemble for multi-dimensional time series*, in Proceedings of the 2015 SIAM International Conference on Data Mining, SIAM, 2015, pp. 307–315.

[30] K.-P. CHAN AND A. W.-C. FU, *Efficient time series matching by wavelets*, in Data Engineering, 1999. Proceedings., 15th International Conference on, IEEE, 1999, pp. 126–133.

[31] M. CHAOUCH, *Clustering-based improvement of nonparametric functional time series forecasting: Application to intra-day household-level load curves*, IEEE Transactions on Smart Grid, 5 (2014), pp. 411–419.

[32] C. CHATFIELD, *The analysis of time series: an introduction*, CRC press, 2016.

[33] H. CHEN, F. TANG, P. TINO, AND X. YAO, *Model-based kernel for efficient time series analysis*, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2013, pp. 392–400.

[34] J. CHEN, H. SUN, D. WOODRUFF, AND Q. ZHANG, *Communication-optimal distributed clustering*, in Advances in Neural Information Processing Systems, 2016, pp. 3720–3728.

[35] L. CHEN, M. T. ÖZSU, AND V. ORIA, *Robust and fast similarity search for moving object trajectories*, in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 491–502.

[36] Q. CHEN, L. CHEN, X. LIAN, Y. LIU, AND J. X. YU, *Indexable pla for efficient similarity search*, in Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, 2007, pp. 435–446.

[37] B. CHIU, E. KEOGH, AND S. LONARDI, *Probabilistic discovery of time series motifs*, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 493–498.

[38] F.-L. CHUNG, T.-C. FU, R. LUK, AND V. NG, *Flexible time series pattern matching based on perceptually important points*, (2001).

[39] M. CORDUAS AND D. PICCOLO, *Time series clustering and classification by the autoregressive metric*, Computational Statistics & Data Analysis, 52 (2008), pp. 1860–1872.

[40] G. CORMODE, S. MUTHUKRISHNAN, AND W. ZHUANG, *Conquering the divide: Continuous clustering of distributed data streams*, in Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, IEEE, 2007, pp. 1036–1045.

[41] G. CUGOLA AND A. MARGARA, *Complex event processing with t-rex*, Journal of Systems and Software, 85 (2012), pp. 1709–1728.

[42] ——, *Low latency complex event processing on parallel hardware*, Journal of Parallel and Distributed Computing, 72 (2012), pp. 205–218.

[43] ——, *Processing flows of information: From data stream to complex event processing*, ACM Computing Surveys (CSUR), 44 (2012), p. 15.

[44] G. DAS, K.-I. LIN, H. MANNILA, G. RENGANATHAN, AND P. SMYTH, *Rule discovery from time series.*, in KDD, vol. 98, 1998, pp. 16–22.

[45] H. DEBAR AND A. WESPI, *Aggregation and correlation of intrusion-detection alerts*, in International Workshop on Recent Advances in Intrusion Detection, Springer, 2001, pp. 85–103.

[46] A. DEMERS, J. GEHRKE, M. HONG, M. RIEDEWALD, AND W. WHITE, *Towards expressive publish/subscribe systems*, in International Conference on Extending Database Technology, Springer, 2006, pp. 627–644.

[47] H. DENG, G. RUNGER, E. TUV, AND M. VLADIMIR, *A time series forest for classification and feature extraction*, Information Sciences, 239 (2013), pp. 142–153.

[48] A. DENTON, *Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model*, in Data Mining, Fifth IEEE International Conference on, IEEE, 2005, pp. 8–pp.

[49] Y. DING, X. YANG, A. J. KAVS, AND J. LI, *A novel piecewise linear segmentation for time series*, in Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on, vol. 4, IEEE, 2010, pp. 52–55.

[50] D. H. DORR AND A. M. DENTON, *Establishing relationships among patterns in stock market data*, Data & Knowledge Engineering, 68 (2009), pp. 318–337.

[51] M. DUMAS, M. LA ROSA, J. MENDLING, AND H. A. REIJERS, *Fundamentals of business process management*, Springer, 2013.

[52] A. EISENBERG AND J. MELTON, *Sql: 1999, formerly known as sql3*, ACM Sigmod record, 28 (1999), pp. 131–138.

[53] Y. ENGEL AND O. ETZION, *Towards proactive event-driven computing*, in Proceedings of the 5th ACM international conference on Distributed event-based system, ACM, 2011, pp. 125–136.

[54] P. ESLING AND C. AGON, *Time-series data mining*, ACM Computing Surveys (CSUR), 45 (2012), p. 12.

[55] O. ETZION AND P. NIBLETT, *Event processing in action*, Manning Publications Co., 2010.

[56] W. EUACHONGPRASIT AND C. RATANAMAHATANA, *Accurate and efficient retrieval of multimedia time series data under uniform scaling and time warping*, Advances in Knowledge Discovery and Data Mining, (2008), pp. 100–111.

[57] C. FALOUTSOS, M. RANGANATHAN, AND Y. MANOLOPOULOS, *Fast subsequence matching in time-series databases*, vol. 23, ACM, 1994.

[58] E. FINK AND K. B. PRATT, *Indexing of compressed time series*, Data mining in time series databases, 57 (2003), pp. 43–65.

[59] T.-C. FU, *A review on time series data mining*, Engineering Applications of Artificial Intelligence, 24 (2011), pp. 164–181.

[60] T.-C. FU, F.-L. CHUNG, R. LUK, AND C.-M. NG, *Representing financial time series based on data point importance*, Engineering Applications of Artificial Intelligence, 21 (2008), pp. 277–300.

[61] L. J. FÜLÖP, Á. BESZÉDES, G. TÓTH, H. DEMETER, L. VIDÁCS, AND L. FARKAS, *Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics*, in Proceedings of the Fifth Balkan Conference in Informatics, ACM, 2012, pp. 26–31.

[62] M. GAVRILOV, D. ANGUELOV, P. INDYK, AND R. MOTWANI, *Mining the stock market (extended abstract): which measure is best?*, in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 487–496.

[63] X. GE AND P. SMYTH, *Deformable markov model templates for time-series pattern matching*, in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 81–90.

[64] ——, *Segmental semi-markov models for endpoint detection in plasma etching*, IEEE Transactions on Semiconductor Engineering, 259 (2001), pp. 201–209.

[65] M. F. GHALWASH AND Z. OBRADOVIC, *Early classification of multivariate temporal observations by extraction of interpretable shapelets*, BMC bioinformatics, 13 (2012), p. 1.

[66] M. F. GHALWASH, V. RADOSAVLJEVIC, AND Z. OBRADOVIC, *Extraction of interpretable multivariate patterns for early diagnostics*, in Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 201–210.

[67] D. Q. GOLDIN AND P. C. KANELLAKIS, *On similarity queries for time-series data: constraint specification and implementation*, in International Conference on Principles and Practice of Constraint Programming, Springer, 1995, pp. 137–153.

[68] D. GORDON, D. HENDLER, AND L. ROKACH, *Fast randomized model generation for shapelet-based time series classification*, arXiv preprint arXiv:1209.5038, (2012).

[69] T. GÓRECKI AND M. ŁUCZAK, *Using derivatives in time series classification*, Data Mining and Knowledge Discovery, (2013), pp. 1–22.

[70] ——, *Non-isometric transforms in time series classification using dtw*, Knowledge-Based Systems, 61 (2014), pp. 98–108.

[71] J. GRABOCKA, N. SCHILLING, M. WISTUBA, AND L. SCHMIDT-THIEME, *Learning time-series shapelets*, in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 392–401.

[72] J. HAN, J. PEI, AND M. KAMBER, *Data mining: concepts and techniques*, Elsevier, 2011.

[73] W.-S. HAN, J. LEE, Y.-S. MOON, AND H. JIANG, *Ranked subsequence matching in time-series databases*, in Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, 2007, pp. 423–434.

[74] G. HEBRAIL AND B. HUGUENEY, *Symbolic representation of long time-series*, in Proceedings of the Applied Stochastic Models and Data Analysis Conference, 2001, pp. 537–542.

[75] N. HERZBERG, O. KHOVALKO, A. BAUMGRASS, AND M. WESKE, *Towards automating the detection of event sources*, in International Conference on Service-Oriented Computing, Springer, 2013, pp. 111–122.

[76] N. HERZBERG AND A. MEYER, *Improving process monitoring and progress prediction with data state transition events.*, in ZEUS, 2013, pp. 20–23.

[77] N. HERZBERG, A. MEYER, AND M. WESKE, *An event processing platform for business process management*, in Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International, IEEE, 2013, pp. 107–116.

[78] N. HERZBERG AND M. WESKE, *Enriching raw events to enable process intelligence: research challenges*, no. 73, Universitätsverlag Potsdam, 2013.

[79] J. HILLS, J. LINES, E. BARANAUSKAS, J. MAPP, AND A. BAGNALL, *Classification of time series by shapelet transformation*, Data Mining and Knowledge Discovery, 28 (2014), pp. 851–881.

[80] X. HUANG, Y. YE, L. XIONG, R. Y. LAU, N. JIANG, AND S. WANG, *Time series k-means: A new k-means type smooth subspace clustering for time series data*, Information Sciences, 367 (2016), pp. 1–13.

[81] J. HUNTER AND N. MCINTOSH, *Knowledge-based event detection in complex time series data*, in Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making, Springer, 1999, pp. 271–280.

[82] Y.-S. JEONG, M. K. JEONG, AND O. A. OMITAOMU, *Weighted dynamic time warping for time series classification*, Pattern Recognition, 44 (2011), pp. 2231–2240.

[83] M. JONES, D. NIKOVSKI, M. IMAMURA, AND T. HIRATA, *Exemplar learning for extremely efficient anomaly detection in real-valued time series*, Data Mining and Knowledge Discovery, 30 (2016), pp. 1427–1454.

[84] E. KEOGH, *Exact indexing of dynamic time warping*, in Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 406–417.

[85] E. KEOGH, K. CHAKRABARTI, M. PAZZANI, AND S. MEHROTRA, *Locally adaptive dimensionality reduction for indexing large time series databases*, ACM SIGMOD Record, 30 (2001), pp. 151–162.

[86] E. KEOGH, S. CHU, D. HART, AND M. PAZZANI, *An online algorithm for segmenting time series*, in Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, IEEE, 2001, pp. 289–296.

[87] ——, *Segmenting time series: A survey and novel approach*, Data mining in time series databases, 57 (2004), pp. 1–22.

[88] E. KEOGH AND J. LIN, *Clustering of time-series subsequences is meaningless: implications for previous and future research*, Knowledge and information systems, 8 (2005), pp. 154–177.

[89] E. KEOGH, J. LIN, AND A. FU, *Hot sax: Efficiently finding the most unusual time series subsequence*, in Data mining, fifth IEEE international conference on, Ieee, 2005, pp. 8–pp.

[90] E. KEOGH, X. XI, L. WEI, AND C. A. RATANAMAHATANA, *The ucr time series classification/clustering homepage*, URL= http://www. cs. ucr. edu/eamonn/time series data, (2006).

[91] E. J. KEOGH AND M. J. PAZZANI, *An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback.*, in KDD, vol. 98, 1998, pp. 239–243.

[92] ——, *Relevance feedback retrieval of time series data*, in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, pp. 183–190.

[93] ——, *Scaling up dynamic time warping for datamining applications*, in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 285–289.

[94] ——, *A simple dimensionality reduction technique for fast similarity search in large time series databases*, in Pacific-Asia conference on knowledge discovery and data mining, Springer, 2000, pp. 122–133.

[95] ——, *Derivative dynamic time warping*, in Proceedings of the 2001 SIAM International Conference on Data Mining, SIAM, 2001, pp. 1–11.

[96] E. J. KEOGH AND P. SMYTH, *A probabilistic approach to fast pattern matching in time series databases.*, in KDD, vol. 1997, 1997, pp. 24–30.

[97] J. O. KEPHART AND D. M. CHESS, *The vision of autonomic computing*, Computer, 36 (2003), pp. 41–50.

[98] S.-W. KIM AND B.-S. JEONG, *Performance bottleneck of subsequence matching in time-series databases: observation, solution, and performance evaluation*, Information Sciences, 177 (2007), pp. 4841–4858.

[99] S.-W. KIM, S. PARK, AND W. W. CHU, *An index-based approach for similarity search supporting time warping in large sequence databases*, in Data Engineering, 2001. Proceedings. 17th International Conference on, IEEE, 2001, pp. 607–614.

[100] M. KONTAKI, A. N. PAPADOPOULOS, AND Y. MANOLOPOULOS, *Similarity search in time series*, in Handbook of Research on Innovations in Database Technologies and Applications: Current and Future Trends, IGI Global, 2009, pp. 288–299.

[101] A. KOTSIFAKOS, I. KARLSSON, P. PAPAPETROU, V. ATHITSOS, AND D. GUNOPULOS, *Embedding-based subsequence matching with gaps–range–tolerances: a query-by-humming application*, The VLDB Journal, 24 (2015), pp. 519–536.

[102] S. KUNZ, T. FICKINGER, J. PRESCHER, AND K. SPENGLER, *Managing complex event processes with business process modeling notation*, in International Workshop on Business Process Modeling Notation, Springer, 2010, pp. 78–90.

[103] S. K. LAM AND M. H. WONG, *A fast projection algorithm for sequence data searching*, Data & Knowledge Engineering, 28 (1998), pp. 321–339.

[104] H. LAMBA, T. J. GLAZIER, J. CÁMARA, B. SCHMERL, D. GARLAN, AND J. PFEFFER, *Model-based cluster analysis for identifying suspicious activity sequences in software*, (2017).

[105] L. J. LATECKI, V. MEGALOOIKONOMOU, Q. WANG, R. LAKAEMPER, C. A. RATANAMAHATANA, AND E. KEOGH, *Elastic partial matching of time series*, in European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2005, pp. 577–584.

[106] S. LEE, D. KWON, AND S. LEE, *Dimensionality reduction for indexing time series based on the minimum distance*, Journal of Information Science and Engineering, 19 (2003), pp. 697–711.

[107] P. LEITNER, A. MICHLMAYR, F. ROSENBERG, AND S. DUSTDAR, *Monitoring, prediction and prevention of sla violations in composite services*, in Web Services (ICWS), 2010 IEEE International Conference on, IEEE, 2010, pp. 369–376.

[108] X. LIAN, L. CHEN, AND B. WANG, *Approximate similarity search over multiple stream time series*, Advances in Databases: Concepts, Systems and Applications, (2007), pp. 962–968.

[109] S.-H. LIM, H. PARK, AND S.-W. KIM, *Using multiple indexes for efficient subsequence matching in time-series databases*, Information Sciences, 177 (2007), pp. 5691–5706.

[110] J. LIN, E. KEOGH, S. LONARDI, AND B. CHIU, *A symbolic representation of time series, with implications for streaming algorithms*, in Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, ACM, 2003, pp. 2–11.

[111] J. LIN, E. KEOGH, S. LONARDI, J. P. LANKFORD, AND D. M. NYSTROM, *Visually mining and monitoring massive time series*, in Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 460–469.

[112] J. LIN, E. KEOGH, L. WEI, AND S. LONARDI, *Experiencing sax: a novel symbolic representation of time series*, Data Mining and knowledge discovery, 15 (2007), pp. 107–144.

[113] J. LIN, R. KHADE, AND Y. LI, *Rotation-invariant similarity in time series using bag-of-patterns representation*, Journal of Intelligent Information Systems, 39 (2012), pp. 287–315.

[114] R. A. K.-L. LIN AND H. S. S. K. SHIM, *Fast similarity search in the presence of noise, scaling, and translation in time-series databases*, in Proceeding of the 21th International Conference on Very Large Data Bases, Citeseer, 1995, pp. 490–501.

[115] Y.-F. LIN, H.-H. CHEN, V. S. TSENG, AND J. PEI, *Reliable early classification on multivariate time series with numerical and categorical attributes*, in Advances in Knowledge Discovery and Data Mining, Springer, 2015, pp. 199–211.

[116] J. LINES, L. M. DAVIS, J. HILLS, AND A. BAGNALL, *A shapelet transform for time series classification*, in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, pp. 289–297.

[117] D. LO, S.-C. KHOO, AND J. LI, *Mining and ranking generators of sequential patterns*, in Proceedings of the 2008 SIAM International Conference on Data Mining, SIAM, 2008, pp. 553–564.

[118] D. LUCKHAM, *The power of events*, vol. 204, Addison-Wesley Reading, 2002.

[119] D. C. LUCKHAM, *Event processing for business: organizing the real-time enterprise*, John Wiley & Sons, 2011.

[120] F. M. MAGGI, C. DI FRANCESCOMARINO, M. DUMAS, AND C. GHIDINI, *Predictive monitoring of business processes*, in Advanced Information Systems Engineering, Springer, 2014, pp. 457–472.

[121] A. MARGARA, G. CUGOLA, AND G. TAMBURRELLI, *Towards automated rule learning for complex event processing*, tech. rep., Technical Report, 2013.

[122] ——, *Learning from the past: automated rule generation for complex event processing*, in Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, ACM, 2014, pp. 47–58.

[123] D. MCCARTHY AND U. DAYAL, *The architecture of an active database management system*, in ACM Sigmod Record, vol. 18, ACM, 1989, pp. 215–224.

[124] A. MCGOVERN, D. H. ROSENDAHL, R. A. BROWN, AND K. K. DROEGEMEIER, *Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction*, Data Mining and Knowledge Discovery, 22 (2011), pp. 232–258.

[125] T. METZKE, A. ROGGE-SOLTI, A. BAUMGRASS, J. MENDLING, AND M. WESKE, *Enabling semantic complex event processing in the domain of logistics*, in International Conference on Service-Oriented Computing, Springer, 2013, pp. 419–431.

[126] K. Ø. MIKALSEN, F. M. BIANCHI, C. SOGUERO-RUIZ, AND R. JENSSEN, *Time series cluster kernel for learning similarities between multivariate time series with missing data*, arXiv preprint arXiv:1704.00794, (2017).

[127] M. MONTALI, F. M. MAGGI, F. CHESANI, P. MELLO, AND W. M. VAN DER AALST, *Monitoring business constraints with the event calculus*, ACM Transactions on Intelligent Systems and Technology (TIST), 5 (2013), p. 17.

[128] Y.-S. MOON, K.-Y. WHANG, AND W.-S. HAN, *General match: a subsequence matching method in time-series databases based on generalized windows*, in Proceedings of the 2002 ACM SIGMOD international conference on Management of data, ACM, 2002, pp. 382–393.

[129] Y.-S. MOON, K.-Y. WHANG, AND W.-K. LOH, *Duality-based subsequence matching in time-series databases*, in Data Engineering, 2001. Proceedings. 17th International Conference on, IEEE, 2001, pp. 263–272.

[130] J. P. MORRILL, *Distributed recognition of patterns in time series data*, Communications of the ACM, 41 (1998), pp. 45–51.

[131] M. D. MORSE AND J. M. PATEL, *An efficient and accurate method for evaluating time series similarity*, in Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, 2007, pp. 569–580.

[132] M. MOTOYOSHI, T. MIURA, AND K. WATANABE, *Mining temporal classes from time series data*, in Proceedings of the eleventh international conference on Information and knowledge management, ACM, 2002, pp. 493–498.

[133] R. MOUSHEIMISH, Y. TAHER, AND B. FINANCE, *Towards smart logistics process management*, in International Conference on Electronic Commerce and Web Technologies, Springer, 2015, pp. 127–138.

[134] ——, *Towards smart logistics processes: a predictive monitoring and proactive adaptation approach*, in Proceedings of the 2015 International Conference on Software and System Process, ACM, 2015, pp. 169–170.

[135] R. MOUSHEIMISH, Y. TAHER, AND K. ZEITOUNI, *autocep: Automatic learning of predictive rules for complex event processing*, in International Conference on Service-Oriented Computing, Springer, 2016, pp. 586–593.

[136] ——, *Automatic learning of predictive rules for complex event processing: doctoral symposium*, in Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, ACM, 2016, pp. 414–417.

[137] ——, *The butterfly: An intelligent framework for violation prediction within business processes*, in Proceedings of the 20th International Database Engineering & Applications Symposium, ACM, 2016, pp. 302–307.

[138] ——, *Complex event processing for the non-expert with autocep: demo*, in Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, ACM, 2016, pp. 340–343.

[139] ——, *Toward the support of challenging service level agreements (slas) in manual and context-dependent activities*, in Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual, vol. 2, IEEE, 2016, pp. 38–43.

[140] ——, *Automatic learning of predictive cep rules: Bridging the gap between data mining and complex event processing*, in Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, ACM, 2017.

[141] R. MOUSHEIMISH, Y. TAHER, K. ZEITOUNI, AND M. DUBUS, *Pact-art: Adaptive and context-aware processes for the transportation of artworks*, in Digital Heritage, 2015, vol. 2, IEEE, 2015, pp. 347–350.

[142] A. MOVCHAN AND M. ZYMBLER, *Time series subsequence similarity search under dynamic time warping distance on the intel many-core accelerators*, in International Conference on Similarity Search and Applications, Springer, 2015, pp. 295–306.

[143] A. MUEEN, E. KEOGH, AND N. YOUNG, *Logical-shapelets: an expressive primitive for time series classification*, in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 1154–1162.

[144] A. MUEEN, E. KEOGH, Q. ZHU, S. CASH, AND B. WESTOVER, *Exact discovery of time series motifs*, in Proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, 2009, pp. 473–484.

[145] A. MUEEN, K. VISWANATHAN, C. GUPTA, AND E. KEOGH, *The fastest similarity search algorithm for time series subsequences under euclidean distance*, August 2015. http://www.cs.unm.edu/ mueen/FastestSimilaritySearch.html.

[146] C. MUTSCHLER AND M. PHILIPPSEN, *Learning event detection rules with noise hidden markov models*, in Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on, IEEE, 2012, pp. 159–166.

[147] Z. M. NOPIAH, M. I. KHAIRIR, S. ABDULLAH, C. K. E. NIZWAN, AND M. N. BAHARIN, *Peak-valley segmentation algorithm for kurtosis analysis and classification of fatigue time series data*, European Journal of Scientific Research, 29 (2009), pp. 113–125.

[148] R. T. OLSZEWSKI, *Generalized feature extraction for structural pattern recognition in time-series data*, tech. rep., DTIC Document, 2001.

[149] J. PAPARRIZOS AND L. GRAVANO, *k-shape: Efficient and accurate clustering of time series*, in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 1855–1870.

[150] J. PARK, S. A. REVELIOTIS, D. A. BODNER, AND L. F. MCGINNIS, *A distributed, event-driven control architecture for flexibly automated manufacturing systems*, International Journal of Computer Integrated Manufacturing, 15 (2002), pp. 109–126.

[151] O. P. PATRI, A. B. SHARMA, H. CHEN, G. JIANG, A. V. PANANGADAN, AND V. K. PRASANNA, *Extracting discriminative shapelets from heterogeneous sensor data*, in Big Data (Big Data), 2014 IEEE International Conference on, IEEE, 2014, pp. 1095–1104.

[152] W. PEDRYCZ, W. LU, X. LIU, W. WANG, AND L. WANG, *Human-centric analysis and interpretation of time series: a perspective of granular computing*, Soft Computing, 18 (2014), pp. 2397–2411.

[153] C.-S. PERNG, H. WANG, S. R. ZHANG, AND D. S. PARKER, *Landmarks: a new model for similarity-based pattern querying in time series databases*, in Data Engineering, 2000. Proceedings. 16th International Conference on, IEEE, 2000, pp. 33–42.

[154] D. RAFIEI AND A. O. MENDELZON, *Querying time series data based on similarity*, IEEE Transactions on Knowledge and Data Engineering, 12 (2000), pp. 675–693.

[155] P. RAI AND S. SINGH, *A survey of clustering techniques*, International Journal of Computer Applications, 7 (2010), pp. 1–5.

[156] T. RAKTHANMANON, B. CAMPANA, A. MUEEN, G. BATISTA, B. WESTOVER, Q. ZHU, J. ZAKARIA, AND E. KEOGH, *Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping*, ACM Transactions on Knowledge Discovery from Data (TKDD), 7 (2013), p. 10.

[157] T. RAKTHANMANON AND E. KEOGH, *Data mining a trillion time series subsequences under dynamic time warping*, in Twenty-Third International Joint Conference on Artificial Intelligence, 2013.

[158] ——, *Fast shapelets: A scalable algorithm for discovering time series shapelets*, in Proceedings of the 13th SIAM international conference on data mining, SIAM, 2013, pp. 668–676.

[159] C. RATANAMAHATANA, E. KEOGH, A. J. BAGNALL, AND S. LONARDI, *A novel bit level time series representation with implication of similarity search and clustering*, in Pacific-

Asia Conference on Knowledge Discovery and Data Mining, Springer, 2005, pp. 771–777.

[160] C. A. RATANAMAHATANA AND E. KEOGH, *Three myths about dynamic time warping data mining*, in Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM, 2005, pp. 506–510.

[161] J. J. RODRÍGUEZ, C. J. ALONSO, AND J. A. MAESTRO, *Support vector machines of interval-based features for time series classification*, Knowledge-Based Systems, 18 (2005), pp. 171–178.

[162] D. S. ROSENBLUM AND A. L. WOLF, *A design framework for Internet-scale event observation and notification*, vol. 22, ACM, 1997.

[163] P. RUENGRONGHIRUNYA, V. NIENNATTRAKUL, AND C. RATANAMAHATANA, *Speeding up similarity search on a large time series dataset under time warping distance*, Advances in Knowledge Discovery and Data Mining, (2009), pp. 981–988.

[164] E. H. RUSPINI AND I. S. ZWIR, *Automated qualitative description of measurements*, in Instrumentation and Measurement Technology Conference, 1999. IMTC/99. Proceedings of the 16th IEEE, vol. 2, IEEE, 1999, pp. 1086–1091.

[165] P. SABZMEYDANI AND G. MORI, *Detecting pedestrians by learning shapelet features*, in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[166] Y. SAKURAI, M. YOSHIKAWA, AND C. FALOUTSOS, *Ftw: fast similarity search under the time warping distance*, in Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2005, pp. 326–337.

[167] S. SALVADOR AND P. CHAN, *Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms*, in Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on, IEEE, 2004, pp. 576–584.

[168] ——, *Learning states and rules for detecting anomalies in time series*, Applied Intelligence, 23 (2005), pp. 241–255.

[169] P. SANG HYUN, W. WESLEY, ET AL., *Discovering and matching elastic rules from sequence databases*, Fundamenta Informaticae, 47 (2001), pp. 75–90.

[170] P. SCHÄFER, *The boss is concerned with time series classification in the presence of noise*, Data Mining and Knowledge Discovery, 29 (2015), pp. 1505–1530.

[171] E. SCHMIEDERS AND A. METZGER, *Preventing performance violations of service compositions using assumption-based run-time verification*, in European Conference on a Service-Based Internet, Springer, 2011, pp. 194–205.

[172] N. P. SCHULTZ-MØLLER, M. MIGLIAVACCA, AND P. PIETZUCH, *Distributed complex event processing with query rewriting*, in Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, ACM, 2009, p. 4.

[173] S. SEN, N. STOJANOVIC, AND L. STOJANOVIC, *An approach for iterative event pattern recommendation*, in Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, ACM, 2010, pp. 196–205.

[174] P. SENIN, J. LIN, X. WANG, T. OATES, S. GANDHI, A. P. BOEDIHARDJO, C. CHEN, S. FRANKENSTEIN, AND M. LERNER, *Grammarviz 2.0: a tool for grammar-based pattern discovery in time series*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 468–472.

[175] P. SENIN AND S. MALINCHIK, *Sax-vsm: Interpretable time series classification using sax and vector space model*, in Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 1175–1180.

[176] J. SERRÀ AND J. L. ARCOS, *Particle swarm optimization for time series motif discovery*, Knowledge-Based Systems, 92 (2016), pp. 127–137.

[177] H. SHATKAY AND S. B. ZDONIK, *Approximate queries and representations for large data sequences*, in Data Engineering, 1996. Proceedings of the Twelfth International Conference on, IEEE, 1996, pp. 536–545.

[178] J. SHIEH AND E. KEOGH, *isax: disk-aware mining and indexing of massive time series datasets*, Data Mining and Knowledge Discovery, 19 (2009), pp. 24–57.

[179] M. SHOKOOHI-YEKTA, Y. CHEN, B. CAMPANA, B. HU, J. ZAKARIA, AND E. KEOGH, *Discovery of meaningful rules in time series*, in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1085–1094.

[180] Y. SHOU, N. MAMOULIS, AND D. W. CHEUNG, *Fast and exact warping of time series using adaptive segmental approximations*, Machine Learning, 58 (2005), pp. 231–267.

[181] P. SMYTH ET AL., *Clustering sequences with hidden markov models*, Advances in neural information processing systems, (1997), pp. 648–654.

[182] N. T. SON, *Time series discord discovery based on r*-tree*, Tá°¡p chi Khoa hoc, (2017), p. 133.

[183] Z. R. STRUZIK AND A. SIEBES, *The haar wavelet transform in the time series similarity paradigm*, in European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 1999, pp. 12–22.

[184] M. TOYODA, Y. SAKURAI, AND Y. ISHIKAWA, *Pattern discovery in data streams under the time warping distance*, The VLDB Journal, 22 (2013), pp. 295–318.

[185] Y. TURCHIN, A. GAL, AND S. WASSERKRUG, *Tuning complex event processing rules using the prediction-correction paradigm*, in Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, ACM, 2009, p. 10.

[186] A. VAHDATPOUR, N. AMINI, AND M. SARRAFZADEH, *Toward unsupervised activity discovery using multi-dimensional motif detection in time series.*, in IJCAI, vol. 9, 2009, pp. 1261–1266.

[187] W. M. VAN DER AALST, M. H. SCHONENBERG, AND M. SONG, *Time prediction based on process mining*, Information Systems, 36 (2011), pp. 450–475.

[188] M. VLACHOS, D. GUNOPULOS, AND G. DAS, *Indexing time-series under conditions of noise*, Data mining in time series databases, 57 (2004), pp. 67–100.

[189] M. VLACHOS, M. HADJIELEFTHERIOU, D. GUNOPULOS, AND E. KEOGH, *Indexing multidimensional time-series with support for multiple distance measures*, in Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 216–225.

[190] ——, *Indexing multidimensional time-series*, The VLDB Journal. The International Journal on Very Large Data Bases, 15 (2006), pp. 1–20.

[191] Y. WAN, X. GONG, AND Y.-W. SI, *Effect of segmentation on financial time series pattern matching*, Applied Soft Computing, 38 (2016), pp. 346–359.

[192] Y. WAN AND Y.-W. SI, *Adaptive neuro fuzzy inference system for chart pattern matching in financial time series*, Applied Soft Computing, 57 (2017), pp. 1–18.

[193] C. WANG AND J.-L. PAZAT, *A two-phase online prediction approach for accurate and timely adaptation decision*, in Services Computing (SCC), 2012 IEEE Ninth International Conference on, IEEE, 2012, pp. 218–225.

[194] F. WANG AND P. LIU, *Temporal management of rfid data*, in Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005, pp. 1128–1139.

[195] H. WU, B. SALZBERG, G. C. SHARP, S. B. JIANG, H. SHIRATO, AND D. KAELI, *Subsequence matching on structured time series data*, in Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 682–693.

[196] H. WU, B. SALZBERG, AND D. ZHANG, *Online event-driven subsequence matching over financial data streams*, in Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM, 2004, pp. 23–34.

[197] J.-L. WU AND P.-C. CHANG, *A trend-based segmentation method and the support vector regression for financial time series forecasting*, Mathematical Problems in Engineering, 2012 (2012).

[198] L. WU, C. FALOUTSOS, K. SYCARA, AND T. R. PAYNE, *Falcon: Feedback adaptive loop for content-based retrieval*, tech. rep., DTIC Document, 2000.

[199] X. XI, E. KEOGH, C. SHELTON, L. WEI, AND C. A. RATANAMAHATANA, *Fast time series classification using numerosity reduction*, in Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 1033–1040.

[200] Z. XING, J. PEI, G. DONG, AND S. Y. PHILIP, *Mining sequence classifiers for early prediction.*, in SDM, SIAM, 2008, pp. 644–655.

[201] Z. XING, J. PEI, AND S. Y. PHILIP, *Early prediction on time series: A nearest neighbor approach.*, in IJCAI, Citeseer, 2009, pp. 1297–1302.

[202] ——, *Early classification on time series*, Knowledge and information systems, 31 (2012), pp. 105–127.

[203] Z. XING, J. PEI, S. Y. PHILIP, AND K. WANG, *Extracting interpretable features for early classification on time series.*, in SDM, vol. 11, SIAM, 2011, pp. 247–258.

[204] X. XU, C. GAO, J. PEI, K. WANG, AND A. AL-BARAKATI, *Continuous similarity search for evolving queries*, Knowledge and Information Systems, 48 (2016), pp. 649–678.

[205] S. YAKOWITZ, *Nearest-neighbour methods for time series analysis*, Journal of time series analysis, 8 (1987), pp. 235–247.

[206] D. YANKOV, E. KEOGH, AND U. REBBAPRAGADA, *Disk aware discord discovery: finding unusual time series in terabyte sized datasets*, in Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, IEEE, 2007, pp. 381–390.

[207] L. YE AND E. KEOGH, *Time series shapelets: a new primitive for data mining*, in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 947–956.

[208] C.-C. M. YEH, Y. ZHU, L. ULANOVA, N. BEGUM, Y. DING, H. A. DAU, D. F. SILVA, A. MUEEN, AND E. KEOGH, *Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets*, in IEEE ICDM, 2016.

[209] B.-K. YI AND C. FALOUTSOS, *Fast time sequence indexing for arbitrary lp norms*, VLDB, 2000.

[210] K. ZOUMPATIANOS, S. IDREOS, AND T. PALPANAS, *Indexing for interactive exploration of big data series*, in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, ACM, 2014, pp. 1555–1566.