Using Semantic Data Models for enhancing Process Mining in Issue Tracking Systems

[Extended Abstract]

David Mendes
Instituto Superior Técnico
david.mendes@tecnico.ulisboa.pt

ABSTRACT

In any software project lifecycle, there are unexpected problems that occur during the development.

To deal with this, there is an information system, called Issue Tracking System (ITS) that keeps track of software issues as they occur. Although the information about those tickets is stored, the respective internal processes are seen as black-boxes, making it difficult to understand what is really happening.

To solve that, Process Mining (PM) uses existing information from the event logs recorded by ITS to automatically create the processes that represent its behaviour.

However, there are several fields that identify a certain issue (like issue type, priority, resolution) and to analyze the processes for all different issue variants, it's necessary to extract an event log for each particular case and load them into a PM tool, making it a daunting task to manually customize a log for each type of process available.

This thesis consists in proposing an additional architecture layer in the PM approach that stores all the information of the event logs in a semantic data model (ontology) and allows the creation of parameterized queries over the existing data model to enable a customized business process creation.

The implemented artifact is applied in practice with a project that uses an ITS: Open edX, a platform that serves as a open source course management system, used for hosting Massive Open Online Courses (MOOCs). The evaluation is done according with a defined set of metrics that allow us to compare the results between the different created processes.

1. INTRODUCTION

Any software project has unforseen issues during its implementation. This can be due to changing requirements by the customers, poor project planning, incorrect requirements gathering, bugs found in the software and a myriad of other reasons [6].

To solve this, there is a software package called ITS that provides a way of solving all kinds of issues that occur during the project in a collaborative manner, and tracks the progression of those issues during their entire lifecycle.

To understand the processes behind the issue resolution and to obtain an accurate picture of what is actually done in the organization, there is an research area that uses the information recorded in information systems (like ITS), to generate the corresponding processes in a automated way and making them transparent to the end user. This process management technique is called Process Mining (PM) [17].

Process Mining (PM) allows us to analyze the collected information in a process-centric point of view in order to obtain, measure and identify the different existing processes, test if the guidelines are being followed, detect possible deviations, fined bottlenecks and observe potential delays. [18].

To create automatically a process via PM, we need a event log with at least 3 fields: the case identifier (the process instance), the timestamp and the main activity that we want to analyze.

However, a lot of information systems have other variables that are relevant for the creation of a automated process.

Since the PM tools only deal with the chosen activity (the field that changes over the course of the process), it would be necessary to obtain an event log for every specific model analysis that has the desired characteristics and load that log into a PM tool. Therefore, obtaining every single process in the scope of that particular information system, would be a daunting task.

In order to optimize the creation of the automated business processes, our proposal consists in using a data model to collect, store and relate the existing data in a logical way, allowing the user to query that data in order to obtain a parametrized process according to the fields chosen by him.

The main objective of this proposal is the creation of a semantic data model for an ITS that gathers information from the event logs, and allows the execution of queries in order to personalize the creation of process models, at a finer level of granularity.

This approach allows a more detailed creation of automated process models, supporting the creation of dozens of process models with a single event log.

We also choose several metrics for comparing the different generated processes. The chosen approach will be based upon the Design Science Research method and applied in practice with projects from ITS.

2. RESEARCH AREA

In this section, we will talk about ITS systems, and the benefits of those systems in software projects. Afterwards, we will present an concrete example of a ITS: Jira.

2.1 Issue Tracking Systems

In a large-sized project, responding efficiently to the bug reports per part of the users involved in the project (the testers, the clients that use the software, etc.) and develop

feature requests per order of importance, is considerably more complex than it seems.

To deal with this, usually it is necessary to have a central repository that collects and handles all kinds of requests within the context of a project development.

This kind of system, called ITS, manages issues, categorizes them, and keeps track of the changes in a structured way.

An ITS provides a way to store issue related information, and track properly any kind of issues that might occur before, during, and after the development of a project. By categorizing an issue, the system provides the user with a way to track progression of the issue from the beginning until its resolution [7].

In essence, issue tracking is nothing more that a "todo" list. By tracking the issues, we can categorize them properly, attribute a person responsible to solve the issue and categorize the entire lifecycle consequently, in order to deal with the different types of requests in a efficient manner.

Nowadays, there are several available issue tracking systems. In this subsection, we will give an overview of one of those tools: Jira.

Jira provides bug tracking and issue tracking and it's used by agile teams to help them in project management by storing issues in a ordered way.

3. RELATED WORK

The following sections present the inclusion of previous work done in the PM area, that is relevant for the topic in question.

3.1 Process Mining

Process mining (PM) is a process management technique that consists in the analysis of event logs in order to extract valuable information related with business processes of an enterprise [18]. The PM techniques allow the automatic discovery of process models that belong to a organization or business entity. Those process models are based on event logs.

The goal of PM is to discover, monitor and improve the internal business processes of the company, allowing the organization to have a deeper insight to their end-to-end processes [18]. However, by focusing on the organizational processes but using the actual recorded business data, PM extracts real, practical value from existing data events, showing what is actually happening in an organization [1].

Therefore, PM aims to gain knowledge from the existing business processes by exploring the data available from the organization's information systems in order to understand and consequentially improve the behavior the performance of those processes.

Regarding the relation of process models and event data, there are three types of PM techniques as we can see in Figure 1 : process discovery, conformance checking and enhancement [17].

1. *Process discovery*: A process discovery approach uses an existing event log to infer a business process model directly from that event log; i.e. from the behavior registered in the data, a representation of a process model is built automatically.

2. *Conformance Checking*: The conformance checking technique takes an existing process model and a event log and compares them in order to find differences between both [18].

   By "replaying" reality based on event data on top of the existing process model, we can confirm if the process works as expected or not, locate eventual deviations and detect potential inconsistencies.

3. *Enhancement*: This approach tries to improve the existing process model, modifying or extending it with another perspectives like a organizational perspective (based on roles inside a organization) or another perspectives, like resources, cost, or risks.

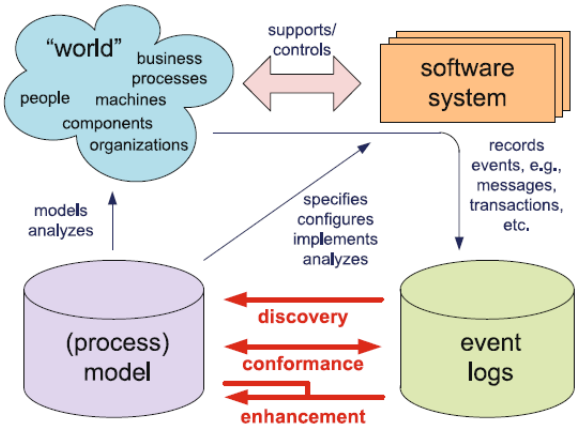

Figure 1: Process Mining Techniques [18]

Considering that one of the problems is the lack of visibility about business processes inITS, the main PM approach used in this thesis will be the process discovery. In order to use this approach, we will use event logs, which are explained in the next section.

3.2 Event logs

As mentioned before, the information systems used by organizations (like ERP or CRM systems) store events correspondent to the business activities of a certain process.

Event logs are events that take place during the execution of a process, from a particular user, given a certain timestamp. In essence, a event log is a collection of traces, and each trace is a sequence of events. Each event contains those four core elements of a process log: the identifier of the process (ususally mentioned as the case identifier), the activity done on that process instance, the author that executed the activity, and the exact moment when the activity occurred (timestamp).

Figure 2 illustrates the previously mentioned elements on a event log; the identifier of the process is represented as "Key", the timestamp is homonymous to the same column in the Figure and the person responsible for the execution of that activity is represented by "authorName".

In this particular log, we wanted to analyze the status of a certain issue (if a certain issue is open, closed, etc.) in a ITS. Therefore, that field is represented by the name "StatusHistory".

However, the data stored by the information systems tends to lack any structure or organization and its usually available in different sources: transaction logs, spreadsheets, several

	Key	authorName	timestamp	changedField	statusHistory
1	XRAY-1	Pedro Gonçalves	2014-04-21 15:09:12.065+0100	Created issue	Open
2	XRAY-1	Bruno Conde	2014-04-21 17:50:36.641+0100	status	Resolved
3	XRAY-1	Bruno Conde	2014-04-28 16:32:20.783+0100	status	Closed
4	XRAY-2	Pedro Gonçalves	2014-04-21 15:09:12.151+0100	Created issue	Open
5	XRAY-2	Bruno Conde	2014-04-21 17:50:36.490+0100	status	Resolved
6	XRAY-2	Bruno Conde	2014-04-28 16:32:20.366+0100	status	Closed
7	XRAY-3	Pedro Gonçalves	2014-04-21 15:09:12.193+0100	Created issue	Open
8	XRAY-3	Bruno Conde	2014-04-21 17:50:36.710+0100	status	Resolved
9	XRAY-3	Bruno Conde	2014-04-28 16:32:20.487+0100	status	Closed
10	XRAY-4	Pedro Gonçalves	2014-04-21 15:09:12.230+0100	Created issue	Open
11	XRAY-4	Bruno Conde	2014-04-21 17:52:22.083+0100	status	In Progress
12	XRAY-4	Bruno Conde	2014-04-21 18:27:48.631+0100	status	Waiting for Testing
13	XRAY-4	Pedro Rodrigues	2014-04-24 15:45:24.003+0100	status	Testing
14	XRAY-4	Pedro Rodrigues	2014-04-24 17:28:29.011+0100	status	Resolved
15	XRAY-4	Bruno Conde	2014-04-28 16:32:19.880+0100	status	Closed
16	XRAY-6	Pedro Gonçalves	2014-04-21 15:09:12.326+0100	Created issue	Open
17	XRAY-6	Bruno Conde	2014-05-13 17:33:31.112+0100	status	In Progress
18	XRAY-6	Bruno Conde	2014-05-13 17:34:10.040+0100	status	Waiting for Testing
19	XRAY-6	Pedro Rodrigues	2014-05-19 14:08:04.211+0100	status	Testing
20	XRAY-6	Pedro Rodrigues	2014-05-19 14:08:21.577+0100	status	Resolved

Figure 2: Event data log

tables, etc. For that reason, before applying PM techniques to an event log, a pre-processing phase is necessary to ensure that the relevant data from those sources is collected and mapped to event logs that can be used in PM. [18].

This data treatment phase, existant in most PM projects, is essential to guarantee the correct analysis of the existing business processes created by the data provided by information systems.

3.3 Tools about process mining

The two main tools available for PM are ProM and Disco.

ProM is an Open Source framework that supports many types of PM analysis and process models, providing hundreds of PM plug-ins. Those plug-ins are not only related with the three types of PM mentioned in the section 3.1 (discovery, conformance and enhancement) but also include conversion plug-ins, which convert models between different formats or semantic-related plug-ins like "Semantic LTL Checker" [19].

Disco is a PM tool created by Anne Rozinat and Christian W. Gunther. Alongside with a general perspective, Disco also offers different views that allow the user to have a more detailed outlook about the event logs. Those views include: the "Statistics" view that offers several performance metrics about the discovered process, their activities and resources; the "Cases" view that allow the user to analyze in detail each individual process instance; and the "Filters" view, which filters a certain path or activity of the process model, in order to make it easier to drill down the data, according to the specified parameters [4].

The main disadvantage compared with ProM is the limited variety of analysis, since it cannot provide conformance checking and enhancement techniques, unlike ProM that has hundreds of plug-ins that cover all the areas of PM.

3.4 Semantic Process Mining

As mentioned in section 3.1, PM allows the extraction of knowledge about business processes with the creation of automatic process models, based on event logs.

However, the analysis of those logs is performed at a syntactic level; the fields of the event logs have strings, which do not provide any information about the actual meaning within the business context, being the business analyst responsible to interpret those fields [2].

Therefore, the PM can be expanded with a semantic approach. By attributing meanings to the fields in the event logs and linking them to ontologies, semantic PM allows us to define a automatic process structure that can be reasoned upon with proper queries, increasing the business value of the process as a whole and potentially identify patterns about the business, reusing and adapting the process more easily for other solutions [10].

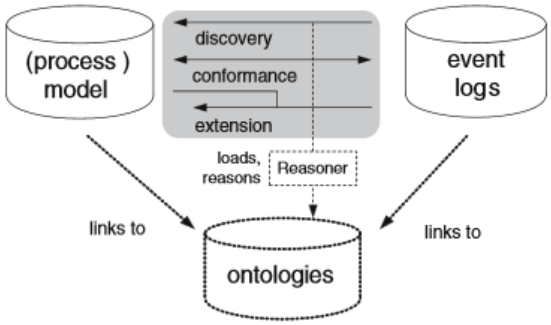

Figure 3: Core building blocks for semantic process mining [10]

The core building blocks are present in Figure 3 for semantic process mining are [10]:

1. Ontologies
2. References from elements in logs to concepts in ontologies
3. Ontology reasoners

3.5 Ontologies

An ontology is responsible for the formalization of concepts and their relationships regarding a certain domain knowledge. In essence, an ontology tries to build a domain model about a certain area of expertise.

To define an ontology, there are a number of steps recommended, that can help us to build an ontology [11]:

1) *Determining the domain and the scope of the ontology*: In this first step, we try to answer some basic questions: What is going to be the use of this particular ontology, for what type of questions we want to know the answer using this ontology, what is the domain of this ontology, etc.

2) *Enumerate important terms in the ontology*: According with the chosen domain, we should establish a list of terms which we want to define and are the core of our ontology.

3) *Define the classes and the class hierarchy*: For this step there are several different approaches: [11, 16]

- a top-down development process that starts with the definition of the broader concepts and breaks down those concepts to more specialized ones.
- a bottom-up development process, starts with the definition of the more specific concepts of the system and groups them into more general concepts.
- A combination of both processes mentioned before: bottom-up and top-down.

4) *Defining the properties and the types of those classes*: Each one of those classes has some properties that are used to characterize the class in question and define its internal structure (for example a Dog has race, color, origin, etc.).

Also, each property has to have a proper type (a string, number, Boolean, etc.) and correspondent cardinality.

5) *Creating instances of classes in the hierarchy*: After the structure of the ontology is done, we create individual instances of certain classes, in order to test the created ontology.

3.6 Ontology languages

To implement a representation of an ontology, there are several languages available. Some of those languages are Web standards adopted by the World Wide Web Consortium (W3C) such as RDF and OWL [9].

3.6.1 RDF

The Resource Description Framework (RDF) is a standard framework used for representing information in the Web. The RDF is characterized by a graph data model composed by a set of triples (subject, predicate, object) in which the subject and the object are entities and the predicate is a relation between those entities [8].

Each RDF triple is represented by two nodes and an arc that points from subject to the object just like in Figure 4.

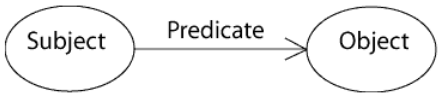

Figure 4: Representation of an RDF triple

An example of a RDF expression is "Eça de Queiróz is the author of 'Os Maias'.", where "Eça de Queiróz" is the subject of the statement, "author" is a predicate, and "Os Maias" denotes the object. It's similar to the classical notation of an entity–attribute–value model; in this example, we have the entity (Eça de Queiróz), the attribute (author), and the value (Os Maias).

Therefore, the RDF graph is composed by a collection of RDF triples. One of the most common serialization formats for RDF is XML-based syntax [21].

3.6.2 RDFS

RDF refers to a data model that indicates the data is structured. To organize RDF data, we need to go one step above in the Semantic Web layer and define a schema that can represent several RDF groups, their classes and relations between them; that is called RDF Schema (RDFS) [3].

Using the example provided in the RDF, where we have "Eça de Queiróz" as the subject, "author" as the predicate, and "Os Maias" as the object, RDFS can specify that "Eça de Queiróz" is an instance of the class "Person" and "Os Maias" is an instance of the class "Book".

Those RDFS elements and properties (subclass, range, etc.) are responsible for the proper hierarchy of the concepts presented before, allowing their classification. RDFS elements and properties compose the basic vocabulary that can be used to construct an ontology.

3.6.3 OWL

The Web Ontology Language (OWL) is a ontology language for the Semantic Web, used for the creation of ontologies. [9]

Web Ontology Language (OWL) can be used to describe in a flexible and formal manner, the meaning of terms used in the ontology. OWL is more expressive than RDFS because has a vaster vocabulary than RDFS, including combinations of expressions (union, intersection), equalities between elements, (sameAs), restriction of values (cardinality) just to name a few [15].

Also, unlike RDFS, OWL has a more rigid structure, restricting what you can actually do with the provided vocabulary. By extending the semantics of the RDF Schema, and increasing the degree of complexity of we can write with RDF, OWL can be used for automatic reasoning and to infer more information about the existent data models.

3.7 Semantic annotations

To make the event logs capable of supporting semantic process mining, it is necessary to extend those logs with semantic annotations [1].

A semantic annotation is responsible for identifying a specific concept in a model, with the objective of enriching the data and give it a proper meaning in a certain context.

To validate the semantically annotated data and to reduce the ambiguity of the given concept, the data should be mapped to a proper ontology, in order to support the formal representation that is given to the elements present in a log [2].

Hence, the semantic annotations can serve as a "bridge" between the models and the ontologies that give meaning to those models. There are several formats that can be used for this annotation process: MXML, SA-MXML and XES.

The MXML (Mining XML format) is a XML-based language used to store event logs from information systems. This data format was created in order to design a standard for event logs that could be shared between different PM tools [10].

The SA-MXML (Semantically Annotated MXML) format is an extension of MXML format that allows for semantic annotations. In addition to the elements mentioned in MXML, those elements have an extra attribute (modelReference) that links the references mentioned in the logs, to concepts represented in ontologies [10].

XES is the standard used for able to capture event logs from different sources , simplifying to way information is represented. XES ended up the successor of MXML and is adopted by the IEEE Task Force on Process Mining, becoming the new standard for event logs [20].

3.8 SPARQL

The Simple Protocol and RDF Query Language (SPARQL) is a W3C recommended semantic query language, used for querying RDF data models [13]. SPARQL is somewhat analogous to SQL; while SQL is designed to query relational data, SPARQL is used to query any data source that can be mapped to RDF. Therefore, SPARQL is used to query ontologies and obtains the result in the form of a RDF graph.

3.9 Jena

Jena is a open source Java RDF framework that offers several application programming interfaces (API) for the creation of data models (like RDF) and Ontologies (like OWL). Jena is fundamentally an RDF platform that can navigate, manipulate and search upon RDF models and uses those RDF data models to build an ontology formalisms on top of RDF.

4. DESIGN SCIENCE RESEARCH

In the field of Information Systems, there is a research paradigm based on the creation of artifacts with the objective of solving real-life problems. That paradigm is called Design Science Research (DSR) [5].

In contrast with other research paradigms, DSR has a more pratical, problem-oriented approach: "Whereas natural sciences and social sciences try to understand reality, design science attempts to create things that serve human purposes" [14].

For a proper approach to the DSR methodology, [12] offers a specific process model, a strict set of activities in order to produce the accomplished result. Those activities are:

1. First step: *Problem identification and motivation.*

   In this first step, we search for a specific problem in the industry that has not been solved. This is done by doing a background research in the areas related to the problem, discover what has been done regarding the identified problem and the advantages and limitations of those approaches, building upon prior research and in order to contextualize "the state of the problem and the importance of its solution" [12].

   The motivation about a potential solution for the identified problem is done to drive the entire process behind the creation of a new artifact. The value of a possible solution is justified by focusing on the benefits the proposed artifact can bring to the areas of research.

2. Second step: *Define the objectives for a solution.*

   After defining the problem and support it with the proper motivation to solve it, it is necessary to specify the main objectives of the proposed solution; what our proposal can bring regarding the body of research already available. This is done by specifying the main objectives that we want to solve with our potential artifact solution.

3. Third step: *Design and development.*

   After the problem is correctly identified, and the main objectives for a solution are defined, we can design an conceptual model of our solution. In this step, we specify in detail the main steps used for the creation of the artifact, the techniques and technologies that will be used in this process and detail each of the activities done for the actual creation of the artifact. After designing the solution, we use that as a blueprint for the actual construction of the artifact. In order to get a consistent solution, the development should be alligned with the objectives established before.

4. Fourth step: *Demonstration of the artifact.*

   After creating the solution, we have to showcase the created artifact in a practical setting to solve the identified problem. This is done to show that the proposed idea works in a realistic setting. Some of the practical examples can include simulations or case studies.

5. Fifth step: *Evaluation.*

   After creating the solution and testing that artifact in a practical application, we need to evaluate it in order to measure how well is supports the objectives defined for the problem.

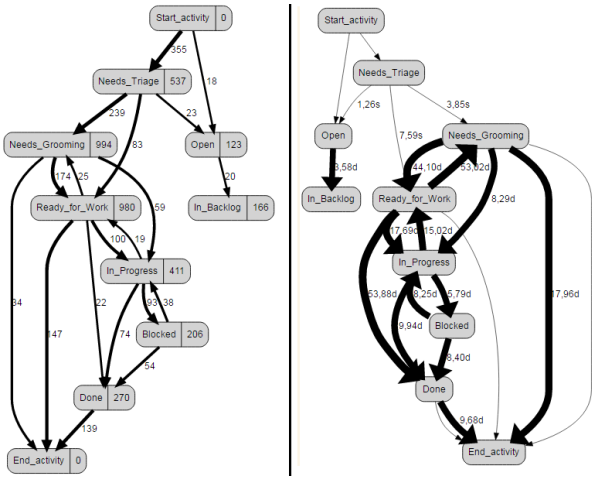

Figure 5: Business process that represents the progression of the status during an issue - frequency and average time between activities

In the following sections, each of the steps mentioned on the DSR methodology will be presented in the context of this work.

5. PROBLEM IDENTIFICATION

As mentioned in Chapter 2, managing issues of a software project is a complicated task.

For that reason, there are ITS systems that are responsible for gathering all types of issues regarding a software project, providing a way to handle the issues in a structured and centralized form.

In order to obtain tangible information about the internal business processes related to those issues, we can use PM tools to discover the real processes since the creation of the process is based on the real data stored in the system.

To do PM, we need an event log with four elements: the case identifier, the timestamp, the resource and the activity that is executed. This activity is a certain field that changes over time. However, for certain information systems (like ITS), this approach is limited since a single activity does not provide a full scope of the business process.

For a practical example, let's use the status of an issue as an example. A status represents the present situation of the issue during its lifecycle. If we use ProM or Disco and analyse the status progession of a list of issues, we create an automated business process that represents the real status progression during the execution of an issue, as seen in Figure 5.

However, we have a narrow scope of the process that is being created, since it only takes only one activity ("Status" in this case) into account, and fields like the assignee, the priority, the issue type and so on, are ignored in the creation of the generated process.

To deal with this, we should obtain a subset of the event log that is relevant for our analysis, (with a certain priority or a certain type or a certain resolution), and load only the relevant issues into the PM tools.

Since there are several categories for several of those types, it may be a daunting task to select manually each log and

potentially having dozens or hundreds of logs for all the PM iterations.

In order to deal with the diversity of processes, an additional architecture layer is proposed: a data model that contains all the event log information and establishes the relations between the different fields.

With a model that stores all the information related about issue tracking, we can create parametrized queries to obtain only the relevant information in a form of an event log and load that event log into a PM algorithm.

This way, we can obtain a more detailed and accurate perspective of the process in question, since we consider the entire scope of an ITS system and all its fields into account, instead of getting only the activity field that creates the business process.

The main motivation of this problem consists in creating a metamodel that represents a ITS and the relations between the different fields, convert that metamodel in a data model that is responsible for all the event log information, and creating parametrized queries responsible for filtering automatically the relevant part of the event log that supports the process creation.

5.1 Goals

The main goal of this thesis is to create a more sophisticated approach in the generation of process models, allowing the end user to parametrize the input for the process model creation.

To do this, instead of using directly an event log for the creation of business processes, this thesis proposes a new architecture layer, based on the existance of a semantic data model.

For a practical approach of the proposed solution, we want an artifact that allows the user to select the relevant fields available in the data model, generating a customized process model based on user input.

In the next section, the proposal that allows us to achieve our goals, is explained in detail.

6. PROPOSAL

As a proposal to the problem identified above, this section presents the main elements of the suggested solution, and specifies its design and development steps.

Afterwards, the activity sequence with all the detailed implementations is explained in practice, in the context of the created artifact.

6.1 Defining the objectives for a solution

As mentioned in chapter 5, instead of having the usual PM approach that consists in obtaining an event log and converting that log automatically in a business process, our proposal consists in storing the data in a model that may be queried, in order to obtain more customizable and accurate business processes that take into account the several variables that occur during the execution of a process.

In order to implement our proposal, we need three building blocks that are the crux of the proposed solution. The elements that represent our proposal are:

1. Creation of a metamodel that represents the data of a ITS: we will analyse the existing data in ITS in order to design a model that will store all the relevant issue data. This metamodel is an explicit representation of the main fields that exist in this kind of information systems and establishes the relations and structure between them.

2. Implementation of an ontology based upon the created issue tracking meta-model: After creating the metamodel, we need a data model to store the information that represents the issues. So, we will use the metamodel to create an ontology in order to store the actual data and give context and structure to the model of a ITS.

3. Creation of SPARQL queries in order to obtain the correct subset of data from the model, for a more accurate process model creation.

By allowing parameters chosen by the user in the creation of those models, the proposed artifact will create more detailed processes based on the different data registered by the event logs, allowing for business processes tailored for each specific case.

6.2 Design and development

In order to implement our solution, we need the first two steps commonly used in PM projects: those steps are the obtainment of an event log and loading that log into a PM tool [18].

As mentioned before, between the first two steps present in the figure, we propose the creation and implementation of a extra layer that stores the relevant data in a structured way, through the implementation of an ontology that stores all the issue related information. Afterwards, the ontology will be queried for obtaining a specific data subset that is relevant for the model.

We will present in theory every step of the proposed method for the creation of the artifact:

1. *Obtain the event log (extraction, pre-processing, data treatment, ETL)*

   The first step of the project consists in obtaining the event log and processing it to get the main elements necessary for the creation of a process model. In our proposal we will store all the main data that changes over the execution of an issue, since one of the main strengths of the proposed approach consists precisely in combining several data elements for a more accurate process model creation.

2. *Create the base model that represents for issue tracking systems*

   To store all the information that is related to the issues, we will need a model that is capable of representing all the data related to an issue execution. Therefore, after obtaining the event log and analyzing thoroughly the elements that are present in the history of an issue, we will select the relevant fields to create a metamodel that represents a ITS.

   Therefore, a model with those fields will be constructed.

3. *Build the core ontology for issue tracking systems*

   After creating the metamodel, the next step consists in a creation of an ontology that contextualizes the logs and relates the different fields, giving them meaning in

the overall scope of the existing processes in ITS, providing different perspectives (including organizational or task-based) for the same data.

4. *Annotate the different fields with an semantic meaning*

   One of the main objectives of using an ontology is to get a common understanding of the designated field, it makes sense to collect and describe every single possible field and add a proper description about them.

   Therefore, we need to collect the different fields available in ITS, and respective definitions and store them in the ontology.

5. *Load the issue data into the existing model ontology*

   After creating the ontology that represents the model of a ITS, we need to fill the base model with data. However, instead of containing information related to only one activity (just like its done in the regular PM tools), it contains all the existing information relevant for an ITS.

6. *Structure the queries in order to obtain an parametrized event log*

   As mentioned before, the main PM tools simply receive an event log as an input, and create directly the process model. Since we used an ontology as the model that contains all the data related to a ITS project, SPARQL queries are used to select the data we are interested in to create the generated process.

7. *Implementation of a process mining algorithm from the produced event log*

   Since the main tools already use a proper algorithm that converts the data into the business process, its necessary a implementation of a PM algorithm that receives a subset of the event log and converts them into the tailored process we want to create.

   The alternative consists in obtaining the event log and loading that log into a tool like ProM or Disco, but that would negate the benefit of this artifact, since loading the logs one by one would be counterproducent to the fact of having a system that parametrizes the fields of a data model based on a bigger event log, and from there, generate the process automatically.

8. *Select a set of metrics to analyze the systems in a process-centric point of view*

   As mentioned before, PM tools can provide the process-centric point of view but are limited to a single activity each time. Therefore, we chose a set of metrics that helped the software team to analyze with detail the efficiency of the issue tracking processes and permit the comparison between different process models.

6.3 Activity sequence

The major components that support our solution were implemented in a prototype that contains three main features: data extraction, data loading and the process mining creation.

In the next subsections, we will explain in practice each one of the steps present in three components and necessary to build the artifact.

6.3.1 Data extraction

The first module of the proposed prototype deals with information extracted from ITS. We decided to use Jira as a source of project issues since there are hundreds of open-source projects that use this kind of tool, providing issues from the most varied origins.

To obtain all the issue information regarding a project, we use the REST API provided by Jira that uses JSON as its communication format. The first step consists in obtaining the URL for the project in question.

After inserting the URL, the artifact will obtain a JSON file with all the available projects and offer the end user the possibility to choose the project that they may want to analyse.

After choosing the respective project, the prototype will gather all the relevant history for each issue and store it in a XML file. Afterwards, the artifact will convert that XML file into a proper CSV file, with the help of a XSLT template. This CSV file, equivalent to those that are used by PM programs like ProM and Disco, will be loaded into the ontology.

6.3.2 Data loading

To load the CSV file, we need a data model capable of storing all the information related to the event logs in order to parametrize them. Based on the data extraction, we created a model that is representative of an ITS.

There are two main sets of fields in the ontology. The first set of fields represent the information of an issue at the time that the data was extracted (the "current" fields). Each issue has only one field of this set at a time. For example, each issue can only have one current status at a time.

The second set of fields represent the several changes that occured during the issue lifecycle. For every change in a field, there is an issue id that is linked to the main fields that we want to analyse. This issue id stores every single change in the context of that field. As an example, one issue may be in the "Closed" status but, by analysing its history, we can see that the status execution was "Open ->Blocked ->Closed".

We opted to use Protégé for the creation of an ontology, due to its flexibility and simplicity in the ontology creation.

After getting our model data in the CSV format, the data will be loaded into the ontology, conforming to the relations previously established in the ontology design.

Issue	hasKeyID	currentType	currentResolution	currentPriority	currentStatus	currentAssignee	hasTimestamp	hasAuthorHistory	hasChangedField	hasStatus	hasAssignee	hasPriority
AC-551	AC-551-1	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:14:0	Chris_Rodriguez	Created_issue	Needs_Triage	Chris_Rodriguez	Unset
AC-551	AC-551-2	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:14:0	Chris_Rodriguez	Link	Needs_Triage	Chris_Rodriguez	Unset
AC-551	AC-551-3	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:14:0	Chris_Rodriguez	Link	Needs_Triage	Chris_Rodriguez	Unset
AC-551	AC-551-4	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:14:0	Chris_Rodriguez	Link	Needs_Triage	Chris_Rodriguez	Unset
AC-551	AC-551-5	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:14:0	Chris_Rodriguez	Link	Needs_Triage	Chris_Rodriguez	Unset
AC-551	AC-551-6	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:15:0	Chris_Rodriguez	status	In_Progress	Chris_Rodriguez	Unset
AC-551	AC-551-7	Task	Unresolved	Unset	Ready_for_Work	Chris_Rodriguez	2016-08-02T10:33:0	Chris_Rodriguez	status	Ready_for_Work	Chris_Rodriguez	Unset
AC-550	AC-550-1	Bug	Unresolved	Unset	Needs_Triage	Unassigned	2016-08-01T12:02:0	Brian_Guertin	Created_issue	Needs_Triage	Unassigned	Unset
AC-549	AC-549-1	Task	Unresolved	CAT-2	Needs_Review	Chris_Rodriguez	2016-07-29T13:50:0	Chris_Rodriguez	Created_issue	Needs_Triage	Chris_Rodriguez	CAT-2
AC-549	AC-549-2	Task	Unresolved	CAT-2	Needs_Review	Chris_Rodriguez	2016-07-29T13:50:0	Chris_Rodriguez	status	In_Progress	Chris_Rodriguez	CAT-2
AC-549	AC-549-3	Task	Unresolved	CAT-2	Needs_Review	Chris_Rodriguez	2016-07-29T14:40:0	Chris_Rodriguez	status	Blocked	Chris_Rodriguez	CAT-2
AC-549	AC-549-4	Task	Unresolved	CAT-2	Needs_Review	Chris_Rodriguez	2016-07-29T14:40:0	Chris_Rodriguez	status	Needs_Review	Chris_Rodriguez	CAT-2

Figure 6: Example of a CSV log that is loaded into the ontology

As we can see in the Figure 6, the event log has 14 fields, compared to the usual 4 that are used in PM tools (identifier, author, activity, timestamp).

To fill the data with the semantic perspective, we use the ontology as a repository of terms, sustaining a knowledge base in an area where those terms have generally a common description but there is not any place where those terms are explicitly and clearly defined.

Between the data loading part and the process creation, there is a manual process regarding the collection of semantic annotations. Therefore, for all the loaded terms into the ontology, we go to the source project in order to identify the meaning behind them and write them into the base ontology, enriching the original terms.

Also, the fact that a ontology is a shared conceptualization of a model, allows for a easier filling for those terms; we only need to add the meaning of those terms once, and the fact that everyone can contribute to the knowledge base makes it easier to obtain a repository of relevant terms in issue tracking.

6.3.3 *Process mining creation*

The third and final part of the artifact consists in querying the ontology model filled with information to obatin to a subset of an event data, that will be loaded into the PM algorithm to obtain automatically the process model defined by the user.

The different terms that correspond to the different fields, are presented to the end user in a drop-down list, in order for him to choose and parametrize the created process model.

The queries are done with SPARQL, a query language for RDF . The base query is selected and according to the input of the user, taking into account the only fields that are always present in a ITS. Therefore, the base query is:

```
SELECT ?issue ?issue_ID ?timestamp
WHERE { ?issue a ticket:Issue .
?issue_ID a ticket:Issue_ID .
?issue ticket:hasKeyID ?issue_ID .
ticket:hasTimestamp ?timestamp }
ORDER BY ?issue ?timestamp ?issue_ID
```

which selects all issues that have an issue id (a field that serves as an identifier for all the changes that occur during the progression of the project) and the respective timestamp.

To each field selected by the user, a following line is concatenated:

"?issue ticket:current" + chosen Field + *" ticket:"* + fieldValue + *" . "*);"

where the chosen field corresponds to the field that the user wants to parametrize, and the field value is the concrete value that corresponds to the field in question.

For example, if the process chosen has "Blocker" priority and is from "Bug" type, the two following lines are added to the query:

- `?issue ticket:currentType ticket:Bug`

  - means that we want all the issues from the Bug type that exist in the ontology.

- `?issue_ID ticket:hasPriority ticket:Blocker`

  - means that we will select all the issues with the "Blocker" priority.

This query returns an event log, that will be loaded into the PM algorithm that returns the corresponding process model.

To conclude is the implementation of a PM algorithm. Since our proposal consists in the creation of an artifact capable of generating dozens of business processes over a single data model, it would be counterproductive to query the data model and, for each attempt, use the resulting event log in an existing PM tool in order to generate each process individually.

Therefore, a PM algorithm similar to heuristic miner [22] was implemented to streamline the creation of business processes based on selected data from the data model.

With the implementation of the PM algorithm, the artifact receives the user input, translates the input into a subset of the event log, and that event log will be redirected to the process mining algorithm, generating the corresponding process model as an output.

In the next section, we will showcase the artifact in a practical setting with a issue tracking project from Jira.

7. DEMONSTRATION

As mentioned before, our approach allows the process discovery tailored for several situations according to the user input. For this demonstration, it was used a project from Open edX: a platform that serves as a open source course management system, used for hosting Massive Open Online Courses.

In this example, we used PLAT project: a Jira project that has over 1000 issues about the instalation and configuration of the Open Edx platform.

Considering that the regular analysis combines all the different kinds of issues, we will use our artifact to explore the process model in depth, according to four different perspectives, and show that there are more processes at a finer level of granularity that do not appear when considering the existing tools.

The core status process was analysed according to four different perspectives: priority, resolution, assignee and issue type.

From the core process, our artifact generated 5 processes according to priority (CAT-1, CAT-2, CAT-3, CAT-4, Unset), 3 processes according to resolution ("Done", "Fixed" and "Unresolved"), 3 processes according to type ("Bug", "Story" and "Sub-task") and 2 processes according to assignee ("Adam" and "Calen")

Considering the number of existing processses, instead of showcasing one or two, we present a table with a comparison between the main paths according to each perspective.

All Status	CAT-1	CAT-2	CAT-3	CAT-4	Unset
Start Activity (756)	Start Activity (3)	Start Activity (43)	Start Activity (58)	Start Activity (5)	Start Activity (746)
Needs Triage (718)					**Needs Triage (711)**
Open (283)	**Open (3)**	**Open (31)**	**Open (50)**	**Open (5)**	**Open (241)**
In Backlog (155)		**In Backlog (31)**	**In Backlog (17)**	**In Backlog (4)**	
In Progress (139)	**In Progress (4)**	**In Progress (30)**		**Open (3)**	
In Code Review (114)	**In Code Review (3)**	**In Code Review (22)**			
		Merged (18)			
Closed (376)	**Closed (6)**		**Closed (29)**		
				Ready for Grooming (2)	
End Activity	End Activity	End Activity	End Activity	End Activity	End Activity

Table 1: Table that compares the most common paths generated with different priorities)

In Table 1, we compare the different processes with the core process present in the first column of Table 1, according to the different priorities.

Each column in the table represents the main path for the chosen process. As an example, the column referent to the process with CAT-3 should be read in the following way: the issue starts with "Start Activity" (A universal activity that indicates the beginning of the process). From that point, the process goes from "Start Activity" to "Open" 58 times. Then the process goes from "Open" to "In Backlog", 50

times, (the number inside parentheses right after "Open"). After that, the issue goes from "In Backlog" to "Closed" 17 times, ending the process right after and concluding the main path of the issue.

The semantic annotaions are important in this case since they provide a proper description for those labels, and consequently, allow us to define the priority order between them (CAT-1 has a bigger priority than CAT-2 and so on).

All Status	Bug	Epic (Path 1)	Epic (Path 2)	Sub-task
Start Activity (756)	Start Activity (175)	Start Activity (69)	Start Activity (69)	Start Activity (119)
Needs Triage (718)	Needs Triage (155)	Needs Triage (68)	Needs Triage (68)	
Open (283)	Open (95)	Open (34)	Open (23)	
			In Discovery (19)	Open Task (44)
In Backlog (155)	In Backlog (49)		Ready For Grooming (14)	
In Progress (139)	In Progress (43)		In Backlog (10)	In Progress (23)
In Code Review (114)	In Code Review (28)		Ready For Grooming (6)	In Code Review (22)
	Merged (33)		IceBox (18)	
Closed (376)				Closed (62)
End Activity	End Activity	End Activity	End Activity	End Activity

Table 2: Table that compares the most common paths generated with different issue types)

Table 1 compares the generated processes according to their type.

By generating different processes according to different perspectives, we understood that the most common paths are distinct, the time between activities is different, the number of paths varies between activities and there are activities that appear in certain processes and not in others.

Therefore, we showed that our artifact added new layers of depth in the created process that did not exist before since the PM tools analyse the process as a whole.

However, we can go one step further and combine the different perspectives to obtain a customized version of any process we want, taking into account the existing data present in the ontology. In this section, its shown one process that combines the four different perspectives to generate a comprehensive process that meets the chosen requirements.

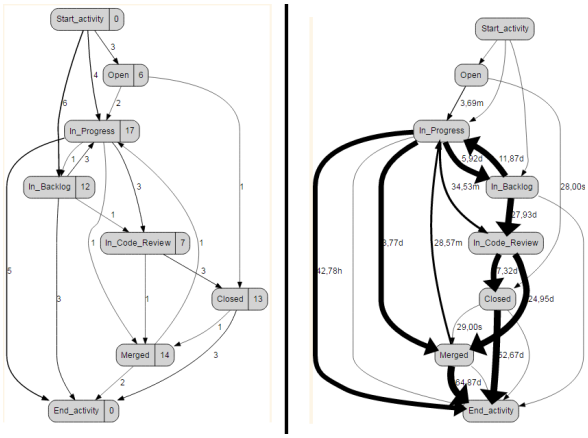

Figure 7: Business process that combines the four different perspectives

As an example, we want to generate a process for issues of type "Bug", with very high priority (CAT-2), that were fixed (resolution Fixed), and were worked on by Adam (the assignee). The resulting process is shown in Figure 7, showcasing that several fields can be combined in the business process generation.

With this approach, we can have any permutation of available fields, increasing considerably the amount of generated processes with a single data model, compared to the single process created by a PM tool.

8. EVALUATION

In this section, we measure how well the artifact supports the proposed objectives. To do that, we compare the different processes available considering the metrics used to evaluate the efficiency of the processes and were presented in chapter 6, defined in the proposal.

The metrics we used to evaluate our proposal are:

1. See if the more urgent tickets are those who are solved first

   The perspectives that were mentioned before (priority, resolution, issue type and assignee), can also be used to obtain the average, median e maximum process time of the issues we choose and consequently, compare the times of different process perspectives.

   The table 3 compares the process according to their priorities.

	All Priorities			
Priority	Number of Issues	Average Process Time	Median Process Time	Max Process Time
All issues	856	77 Days, 8H, 11M, 55S	17 Days, 11H, 41M, 11S	700 Days, 31M, 38S
CAT-1	6	42 Days, 16H, 13M, 47S	10 Days, 22H, 22M, 32S	150 Days, 17 H, 31 M, 48 S
CAT-2	59	67 Days, 7H, 40M, 34S	37 Days, 21H, 35M, 10S	349 Days, 22H, 8M, 36S
CAT-3	71	114 Days, 16H, 57M, 53S	47 Days, 21H, 52M, 51S	587 Days, 3H, 43M, 29S
CAT-4	7	365 Days, 21H, 55M, 6S	363 Days, 4H, 33M, 26S	572 Days, 17H, 12M, 31S
Unset	713	72 Days, 11H, 52M, 2S	12 Days, 21H, 35M, 45S	700 Days, 31M, 38S

Table 3: Table with the process times according with different priorities

   Delving onto the processes with different priorities, and knowing the hierarchy of importance between them (CAT-1 priority is more important than CAT-2 and so on), we can make a proper comparison between priorities and see if, in fact, the more prioritary issues take less time to be solved. Looking again at the Table 3, we can confirm that the priority order is maintained and the most important issues, are in fact, solved in less time.

2. Compare minimum, average and maximum times for a certain issue regarding other kinds of fields (resolution, status, issue types)

   The first metric that was presented, set side by side the issues according to their priority. But other fields can also be compared. In this particular case, we will compare the issues according to the issue type, and their resolution.

   Once there is no distinction between issues that are completed, and those who are still in development, we will make a proper comparison with issues with different resolutions (like resolution "Done", "Fixed" and so on).

   This allows us to have a legitimate comparison between issues that are finished and those who are incomplete (issues that have resolution "Unresolved").

	All Resolutions			
Resolution	Number of Issues	Average Process Time	Median Process Time	Max Process Time
All issues	856	77 Days, 8H, 11M, 55S	17 Days, 11H, 41M, 11S	700 Days, 31M, 38S
Done	287	54 Days,14H, 12M, 39S	24 Days, 4H, 43M, 59S	670 Days, 20H, 48 M, 56 S
Duplicate	24	49 Days, 16H, 46M, 26S	15 Days, 10H, 21M, 35S	346 Days, 5H, 22M, 53S
Fixed	118	71 Days, 28M, 20S	15 Days, 3H, 54M, 35S	572 Days, 17H, 12M, 31S
No Action Needed	9	38 Days, 23H, 22M, 56S	9 Days, 3H, 45M, 53S	167 Days, 17H, 10M, 5S
Not Reproducible	7	89 Days, 11H, 41M, 32S	42 Days, 19H, 7M, 51S	325 Days, 23H, 8M, 49S
Not a Bug	10	11 Days, 16H, 38M, 58S	8 Days, 23H, 49M, 54S	32 Days, 21H, 16M, 12S
Unresolved	401	99 Days, 1H, 13M, 14S	12 Days, 20H, 20M, 37S	700 Days, 31M, 38S

Table 4: Table with the process times according with different resolutions

The Table 4 shows a big contrast between the average time of "Unresolved" issues (99 days) and the median time (12 days). This is explained by the existence of several issues that were just created, and a lot of issues that are still unresolved, and have durations superior to a year. Similar comparison tables can also be done for issues with different types.

9. CONCLUSION

The main objective of this thesis was the creation of a knowledge base for an ITS that gathers information from the event logs, and allows the execution of queries in order to parametrize the creation of process models at a finer level of granularity.

Our approach offered a additional architecture layer between the event log obtainment and the automatic process model creation that deals with the different fields by storing the existing data in a repository, and queries them to obtain a specific subset of an event log that can be loaded into a PM algorithm. By doing that, the proposed artifact can create dozens of process models tailored for the end user, according to the chosen input.

Our artifact was demonstrated thorugh the analysis of a process model according to four different fields in the ontology; four different perspectives that generated more than a dozen of different processes (five from "Priority" analysis, four from "Resolution", four from issue "Type" and two from "Assignee"), all from one single process, showcasing that the different fields available allow for a more detailed process creation.

For future work, we will continue to collect more information about different issue tracking projects in order to generate a richer ontlogy model with more different fields with proper descriptions added to them.

10. REFERENCES

[1] A. K. Alves de Medeiros and W. M. P. van der Aalst. Process mining towards semantics. In *Advances in Web Semantics I - Ontologies, Web Services and Applied Semantic Web*, volume 4891 LNCS, pages 35–80. 2008.

[2] A. K. et al. Alves de Medeiros. An Outlook on Semantic Business Process Mining and Monitoring. *Proceedings of the On the Move to Meaningful Internet Systems 2007 Workshops*, 4806:1244–1255, 2007.

[3] Dan Brickley and R.V. Guha. RDF Schema 1.1 - W3C Recommendation, 2008.

[4] Christian W. Günther and Anne Rozinat. Disco: Discover your processes. In *CEUR Workshop Proceedings*, volume 936, pages 40–44, 2012.

[5] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[6] Ws Humphrey. Why big software projects fail: the 12 key questions. *The Software Engineering Institute*, 18(3):25–29, 2005.

[7] J. Janák. Issue tracking systems. pages 9–13, 2009.

[8] Graham Klyne and Jeremy J Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C Recommendation*, 10:1—-20, 2004.

[9] Deborah L McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Technical report, 2004.

[10] Alves De Medeiros, Ana Karla, and Van Der Aalst. Semantic process mining tools: Core building blocks. In *In 16th European Conference on Information Systems*, 2008.

[11] Nf Noy and Dl McGuinness. Ontology development 101: A guide to creating your first ontology. *Development*, 32:1–25, 2001.

[12] Ken Peffers, Tuure Tuunanen, Marcus a. Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, 2008.

[13] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and Complexity of SPARQL. In *The Semantic Web - ISWC 2006*, volume 4273, pages 30–43. 2006.

[14] Herbert A. Simon. *The Sciences of the Artificial*, volume 1. 1969.

[15] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide, W3C Recommendation. Technical report, 2004.

[16] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–136, 1996.

[17] Wil et al. Van Der Aalst. Process mining manifesto. *Lecture Notes in Business Information Processing*, 99 LNBIP:169–194, 2012.

[18] W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 136. 2011.

[19] B F van Dongen, A K A de Medeiros, H M W Verbeek, AJMM Weijters, and W M P van der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and theory of Petri Nets*, volume 3536, pages 444–454, 2005.

[20] Eric H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. XES, XESame, and ProM 6. *Lecture Notes in Business Information Processing*, 72 LNBIP:60–75, 2011.

[21] W3C Working Group. RDF 1.1 XML Syntax, 2014.

[22] A. J M M Weijters and J. T S Ribeiro. Flexible heuristics miner (FHM). In *IEEE SSCI 2011: Symposium Series on Computational Intelligence - CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining*, pages 310–317, 2011.