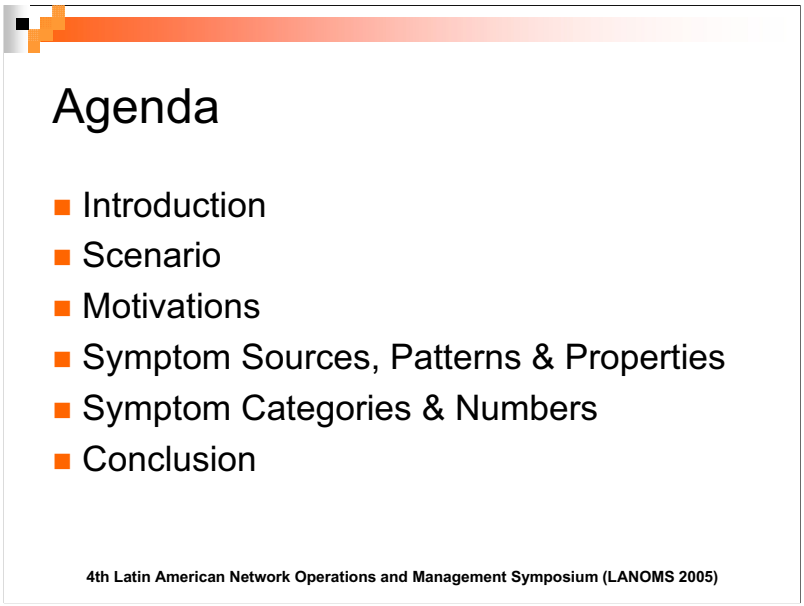
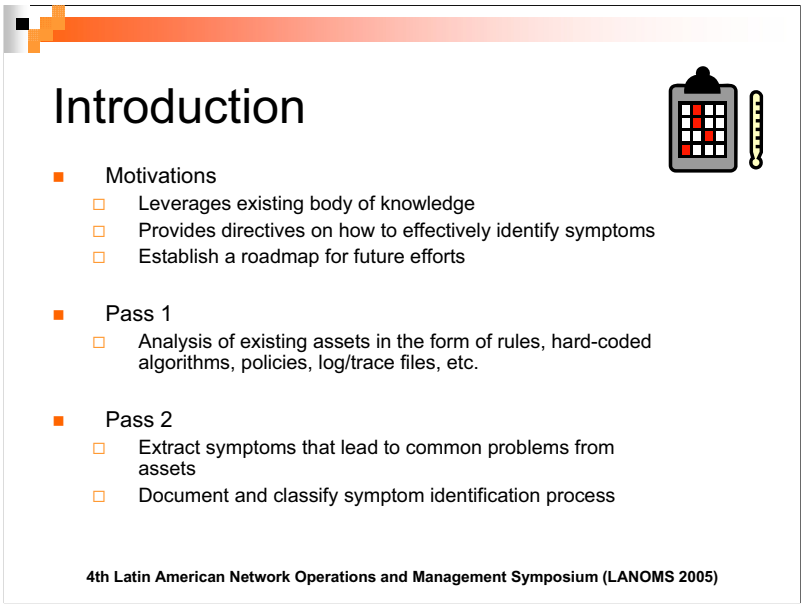
Keywords: Autonomic Computing, Self-Management, Self-Healing, Symptoms, Symptom Taxonomy

Abstract: IT professionals often seek ways to effectively detect and prevent problems associated to the resources they manage. Their solution so far has been to rely on past experiences and established processes. Novel industry initiatives, like the Autonomic Computing paradigm provide concepts such as Events and Symptoms to facilitate automatic processing and consequently identification and treatment of problems and incidents. But the lack of an established method for identifying and classifying such symptoms still presents itself as a complex problem. Answers would be more readily available if processes and practices were to merge into information resources comprising a generic taxonomy of symptoms and their associated knowledge types (situations and recommendations). The purpose of this study is to determine whether the essence of IT problem determination and prediction can be captured by a limited number of generic symptom types. Symptoms with nearly identical structures, such as "application connection error" and "server unreachable error" are to be classified in generic types, e.g. "network condition X". The implication of having a generic taxonomy is directly related to how easily situations can be automated and how knowledge can be identified. In this study we analyze common solutions deployed by IT shops around the world when they integrate their infrastructure with existing event management and problem determination products. The end results of this analysis and a proposal for a generic taxonomy of symptoms for IT problem determination and prediction is presented.



Introduction – approach used by this method
Scenario – problem and concepts
Motivations – what can be expected when applying this methodology?
Identification – sources, identification patterns and properties
Results – categories identified and statistics
Conclusion – remarks and future work



No taxonomy so far was focused entirely on symptoms.

Taxonomies exist for raw events and for incidents, but not for in between (symptoms).

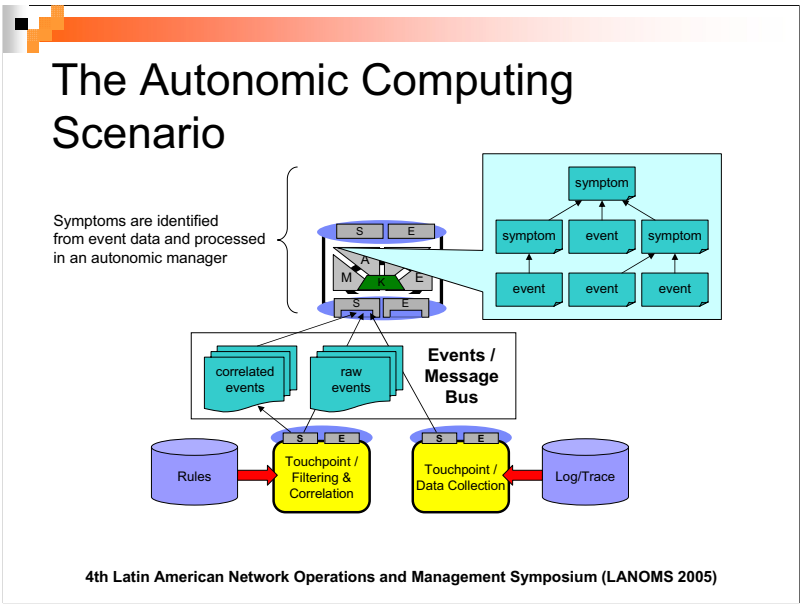
This study leverages an enormous body of knowledge comprised by rules, source code, policies, and log files used in the deployment of real field solutions.

The first step was to identify and categorize existing data.

The second step was to adopt a common methodology and extract specific symptoms by using data mining tools and manual analysis.

Identification of symptoms consists in enumerating which events are used to compose the symptom, what are their relationship (how they are identified), and also provide textual content used to describe and explain the symptom, as well as the recommended actions (if any) used for the treatment of the condition associated to the symptom.

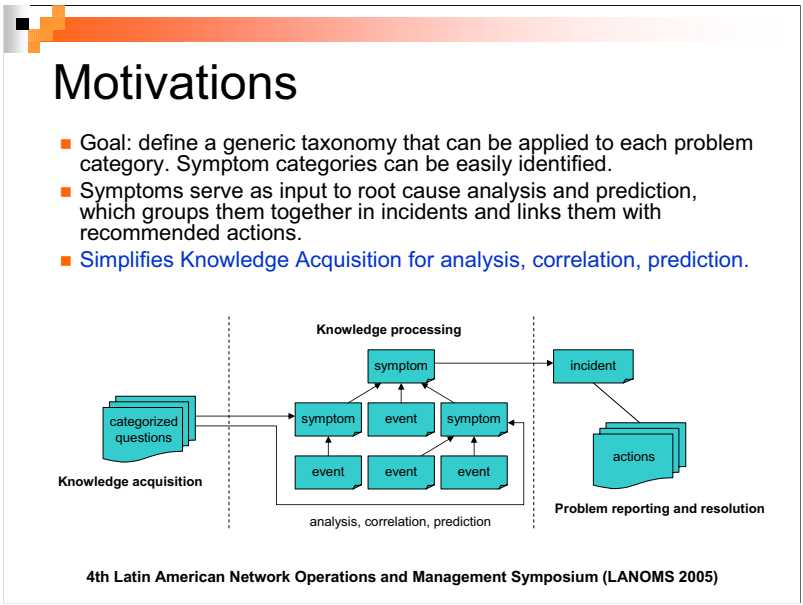
Finally, once the specific symptoms are enumerated, classify them in generic types using a consistent and reproducible approach.



In the common scenario associated with the Self-Healing discipline of the Autonomic Computing paradigm we often encounter two distinct types of monitored information: raw events and correlated events.

A raw event is merely an observation of a certain condition in a monitored resource, and is associated with status and reporting conditions. In contrast, correlated events aim to include a certain level of intelligence to the data being reported. A correlated event is eventually something very similar to a symptom, but it's identified in lower layers that are near to the monitored resource itself. Frequently we can find a 1-to-1 mapping between correlated events and specific symptoms associated to a particular resource type.

Continuing in the scenario, these events enter the autonomic manager process, where they are transformed into symptoms in the "monitoring" block. These symptoms can then be further correlated together or with other events to form root-cause symptoms, or incidents.



The goal of this study is to present hard results derived from real data. The symptoms that were found as a result of this study where then analyzed by following a common classification method that consists in retrofitting the data into categorized questions that will then associate in a 1-to-1 mapping to first level symptoms (or specific symptoms, so called because they often refer to a specific technology and can be further extrapolated into general categories).

The study continues on to find two kinds of relationships between these first level symptoms: inheritance and correlation. When an inheritance relationship exists, a symptom category is created to accommodate the symptoms that were used in its identification. These generic symptoms are also called second level symptoms or a first level symptom category.

The other kind of relationship, called correlation, is also very desirable because it will ultimately lead to root-cases or incidents in the causality chain associated to symptoms and their respective monitored resources.

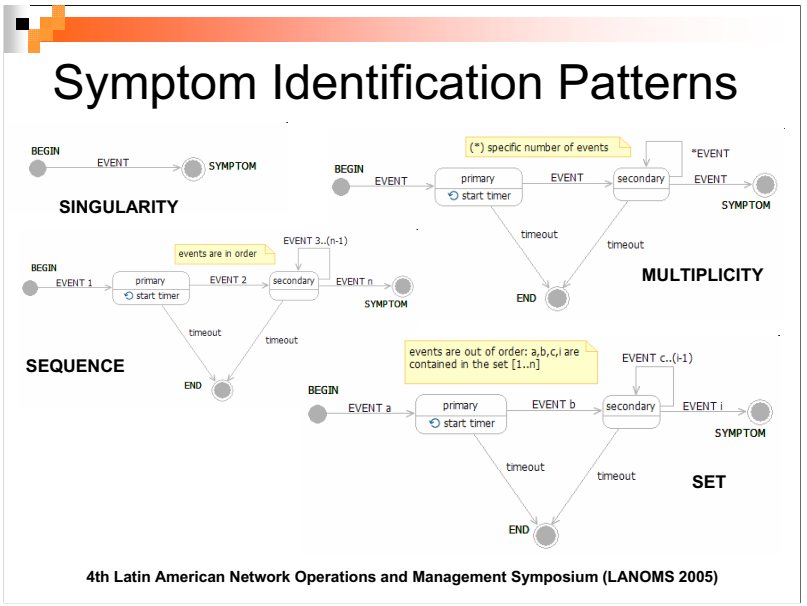
The process is iterative until symptoms can be further classified into the identified main categories (four main categories were identified in this study, please refer to the symptom taxonomy section of this presentation).

Hundreds of different resource types were considered for this study.

Approximately fifty resource types were selected for symptom collection.

The criteria for selection were mainly originality (which resources types would lead to different, non-common symptoms) and quality of derived rules, descriptions and recommendations.

This study utilizes the main resource types from where rules, code, policies and log files were used to perform symptom identification.



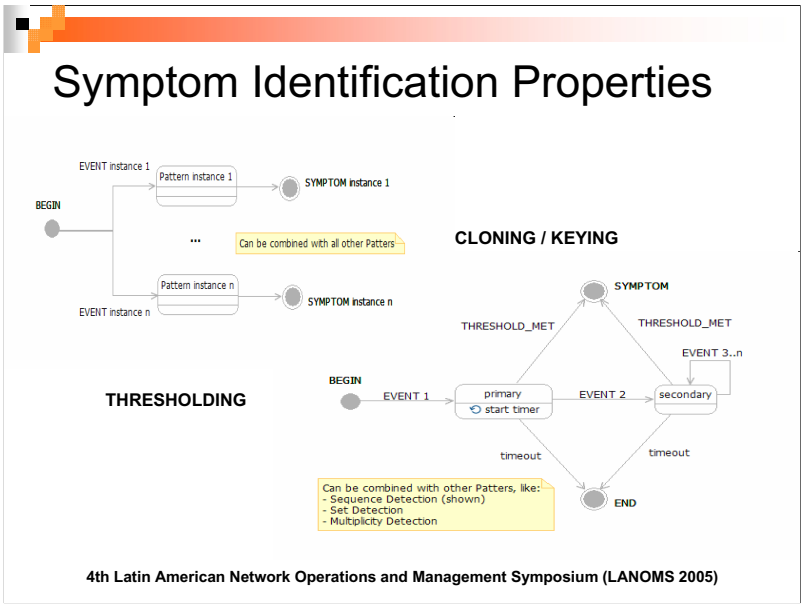
As part of the methodology to identify how a symptom is formed, this study considered four main patterns, which are also fundamental building blocks of composition, commonly found in diverse application, such as collection classes for code libraries, for example. The basic patterns used in the composition of symptoms are:

Singularity – specifies a 1-to-1 mapping relationship between an event and a symptom.

Multiplicity – specifies an n-to-1 mapping relationship between multiple instances of the same event and a symptom.

Sequence – specifies an n-to-1 and ordered relationship between multiple instances of different events and a symptom.

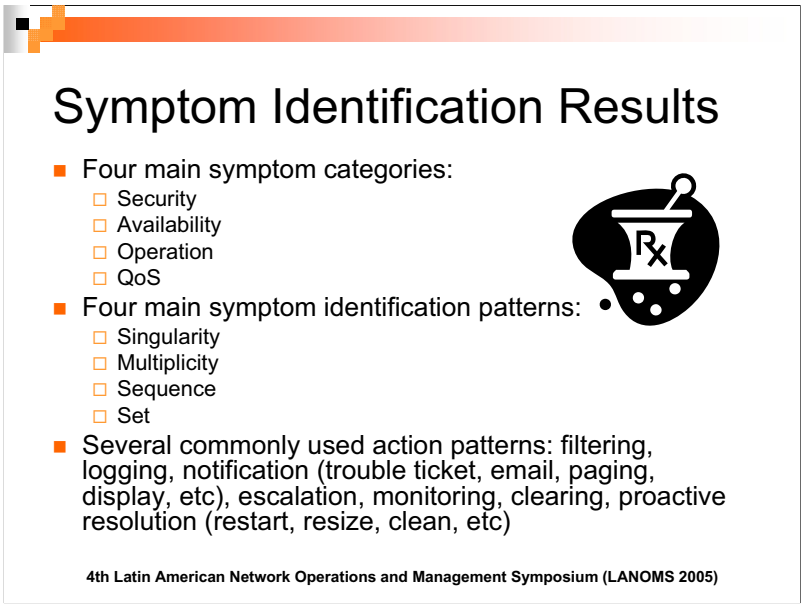
Set – specifies an n-to-1 and unordered relationship between multiple instances of different events and a symptom.



Two main properties were considered in the symptom identification exercise:

Cloning/Keying – this property parameterizes a certain event by a certain attribute value – example: we want to associate multiple instances of the same symptom type when events arrive for different host machines (event parameter = hostname).

Thresholding – this property parameterizes a collection of events by a threshold calculated from a certain attribute of these events – example: we want to identify a symptom only when the sum of the attribute “wrong password attempt” (where 1 means wrong password and 0 means the password was correct) reaches the value 3.



Result numbers and statistics.

From the whole body of data utilized in this study, only four general symptom categories were found.

Likewise, there was the need of only four main patterns (that can be combined if necessary) for their identification.

A plethora of different recommended actions could be extrapolated from the sample data in the deployed solutions. These actions were performed when a symptom was deemed to be a root-cause or an incident. In these cases the following actions were often performed:

-Filtering of the events associated with the symptom

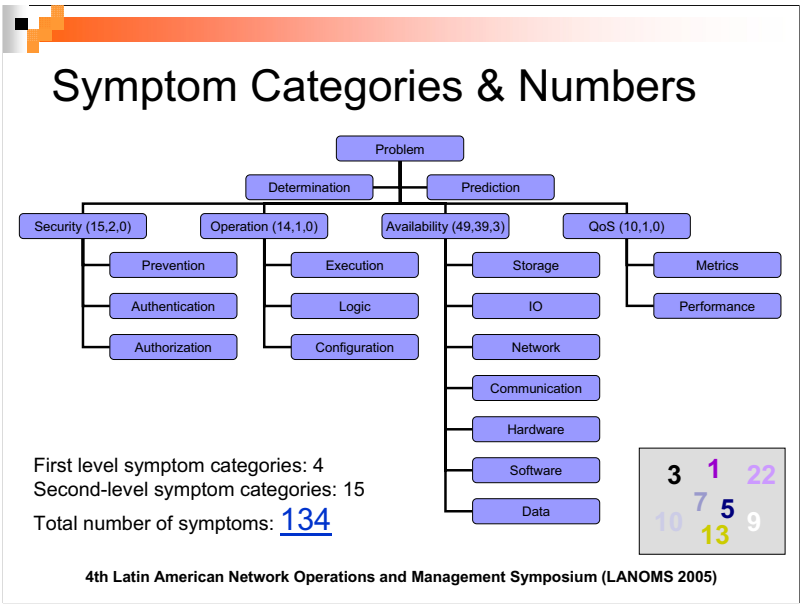
-Logging of the events associated with the symptom

-Notification of the incident to the end user (management operator), which assumed varied forms like opening a trouble ticket, sending an e-mail, sending a page, or just displaying it in the screen

-Escalation control of the incident, usually if the incident is not resolved in a certain amount of time, it's severity is increased and other actions (such as notification) can also be executed

-Monitoring of the symptom, i.e., it's only displayed and waits for manual confirmation and resolution

-Proactive resolution performed automatically by the system – this depends on each individual incident but often involved remediation in a failed resource (restart of a machine, resize of a file system, cleaning of a database, etc). It should be noted that this class of actions accounts for a very low percentage compared to others, which means there's a LOT to grow in terms of automatic resolution of problems in the context of autonomic computing for current products and solutions.



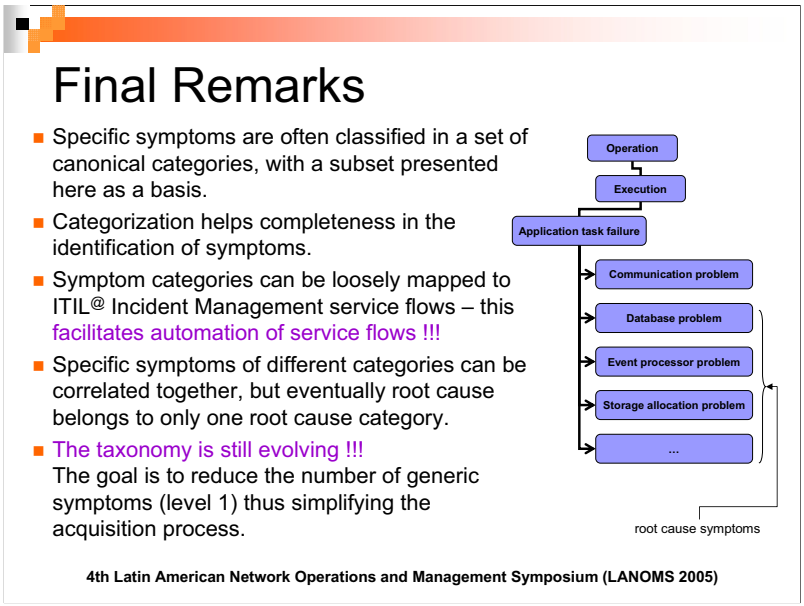
This is the identified taxonomy of symptoms extrapolated from the sample data.

These categories apply evenly to either “Problem Determination” or “Problem Prediction”.

There are four main categories: Security, Operation, Availability and QoS.

These categories are each composed by several lower level categories. For example, the Security category contains 15 second-level categories and 2 third-level categories (not shown) and no fourth-level categories (for now!!!). In contrast, the Operation category numbers were 14, 1, 0; the Availability category numbers were 49, 39, 3; and the QoS category numbers were 10, 1, 0;

The total number of symptoms identified was 134!!!



The existence of this taxonomy actually improved the task of identifying and classifying symptoms in a large data set. Often classifying a specific symptom under a generic category also gives an idea of how the symptom should be handled and to which other symptoms it could be correlated in order to produce incidents.

Generic symptom categories can be loosely coupled to incident categories as defined by standard Incident Management processes, like the ones defined by the ITIL@ group.

This taxonomy is still in evolution. It is not expected that it will work for all existing symptoms extracted from various sources of data. Also, further simplification of the taxonomy is a possible enhancement to be sought. The number of generic categories is large as it is now, and some low level categories can possibly be grouped together into higher level categories, this will help to further reduce the number of categories and simplify the classification method.

Also pointed out here is the correlation relationship between symptoms. This is a different but useful relationship where symptoms are associated together to determine the root-cause of problems – these root-cause symptoms are then promoted to be incidents, and this is what will drive higher layers of the autonomic control loop.

Future Work

- Process more symptom sources (i.e. more products, solutions, technologies, etc).
- Propose ways to align existing problem and incident taxonomies to the one described in this document.
- Produce tooling to enable mass consumption of a symptoms database and to facilitate incorporation and easy maintenance of symptoms and symptom categories.
- Apply discovered symptoms to ITIL® service flows in order to foster automation in the implementation of these flows.
- Establish a sharing model that would make the symptoms body of knowledge available to all adopters of the autonomic computing technology.

4th Latin American Network Operations and Management Symposium (LANOMS 2005)

As a follow up to this study, the following activities are planned:

-Augment the number of symptoms by processing more and original data collected from other solutions and products

-Align existing taxonomies related to events, incidents and problems to reconcile them into a meta-taxonomy that can be applied and extended as needed for different knowledge types associated to the problem determination and prediction disciplines.

-Produce tooling around the concepts of symptoms and the taxonomy of symptoms. Make the process of identification and classification easy for end users.

-Further explore standard IT service flows and identify which symptoms can be applied and extracted from the various parts of the processes.

-Share the identified content in the form of a symptoms database that could be reused by adopters of the Autonomic Computing architecture.