ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

Cyber Physical System based Proactive Collaborative Maintenance

D1.2 Consolidated State-of-the-Art of Sensor-based Proactive Maintenance Appendix 1: Current ways to integrate different information sources

Work Package	WP1 - Service platform architecture requirement definition. Scenarios and use cases descriptions
Version	1.0
Contractual Date of Delivery	30/04/2016
Actual Date of Delivery	03/06/2015
Dissemination Level	Public
Responsible	Erkki Jantunen
Contributors	P□l Varga (AITIA), Csaba Hegedus (AITIA), Tibor Benke (BME), Istv□n Moldov□n (BME), Juha Valtonen(LUAS)
Reviewers	[Insert here list of reviewers separated by commas]

The MANTIS consortium consists of:

Num.	Short Name	Legal Name	Role	Country
1	MGEP	Mondragon Goi Eskola Politeknikoa J.M.A. S.Coop.	CO	ES
2	MONDRAGON	Mondragon Corporacion Cooperativa S.Coop.	BEN	ES
3	IKERLAN	Ikerlan S.Coop.	BEN	ES
4	TEKNIKER	Fundacion Tekniker	BEN	ES
5	FARR	Fagor Arrasate S.Coop.	BEN	ES
5.1	KONIKER	Koniker S.Coop.	TP	ES
6	GOIZPER	Goizper S.Coop.	BEN	ES
7	ACCIONA	Acciona Infraestructuras S.A.	BEN	ES
8	MSI	Mondragon Sistemas De Informacion S.Coop.	BEN	ES
9	VTT	Teknologian Tutkimuskeskus VTT Oy	BEN	FI
10	LUAS	Lapin Ammattikorkeakoulu Oy	BEN	FI
11	NOME	Nome Oy	BEN	FI
12	FORTUM	Fortum Power And Heat Oy	BEN	FI
13	SQ	Solteq Oyj	BEN	FI
14	WAPICE	Wapice Oy	BEN	FI
15	AAU	Aalborg Universitet	BEN	DK
16	DANFOSS	Danfoss A/S	BEN	DK
17	UNIV	Universal Foundation A/S	BEN	DK
18	HGE	Hg Electric A/S	BEN	DK
19	VESTAS	Vestas Wind Systems A/S	BEN	DK
20	SIRRIS	Sirris Het Collectief Centrum Van De Technologische Industrie	BEN	BE
21	ILIAS	Ilias Solutions Nv	BEN	BE
22	ATLAS	Atlas Copco Airpower Nv	BEN	BE
23	3E	3e Nv	BEN	BE
24	PCL	Philips Consumer Lifestyle B.V.	BEN	NL
25	PHC	Philips Medical Systems Nederland B.V.	BEN	NL
26	PHILIPS	Philips Electronics Nederland B.V.	BEN	NL
27	S&T	Science and Technology B.V.	BEN	NL
28	TU/E	Technische Universiteit Eindhoven	BEN	NL
29	RUG	Rijksuniversiteit Groningen	BEN	NL
30	UNINOVA	UNINOVA - Instituto de Desenvolvimento de Novas Tecnologias	BEN	PT
31	ISEP	Instituto Superior de Engenharia do Porto	BEN	PT
32	INESC	Instituto de Engenharia de Sistemas e Computadores do Porto	BEN	PT
33	ADIRA	ADIRA - Metal Forming Solutions S.A.	BEN	PT
34	ASTS	Ansaldo STS S.p.A.	BEN	IT
35	CINI	Consorzio Interuniversitario Nazionale per l'Informatica	BEN	IT
36	AIT	Austrial Institute of Technology GmbH	BEN	AT
37	HBM	Hottinger Baldwni Messtechnik GmbH	BEN	AT
38	INNOTEC	Innovative Technology and Science Limited	BEN	UK
39	AITIA	AITIA International Inc.	BEN	HU
40	BME	Budaperst University of Technology and Economics	BEN	HU
41	JSI	Josef Stefan Institute	BEN	SI
42	XLAB	XLAB d.o.o.	BEN	SI
43	FHG	Fraunhofer Institute for Experimental Software Engineering IESE	BEN	DE
44	M2X	M2Xpert GmbH & Co KG	BEN	DE
45	STILL	STILL GMBH	BEN	DE
46	BOSCH	Robert Bosch GMbH	BEN	DE
47	LIEBHERR	Liebherr-Hydraulikbagger GmbH	BEN	DE

Document Revisions & Quality Assurance

Revisions:

Version	Date	By	Overview
0.1	10/08/2015	Csaba Hegedus	First draft
0.2	19/08/2015	Csaba Hegedus	Inserted in document template
0.3	26/08/2015	Csaba Hegedus	Minor changes
0.4	27/08/2015	P□l Varga	Further, minor changes
0.5	31/08/2015	Juha Valtonen	Document template inserted
0.6	07/09/2015	Juha Valtonen	Added one chapter
0.7	30/09/2015	Juha Valtonen	Combined contributions
0.8	30/09/2015	Istv□n Moldov□n	Reference checked, conclusion added
0.9	05/10/2015	Riku Salokangas	Added contractual date of delivery etc.
1.0	02/06/2016	Mikel Muxika (MGEP)	Format correction Deliverable info update

Abstract

The MANTIS projects aims to consolidate and aggregate data from several data sources, e.g. textual logs. These logs are mainly generated for human reading (structured sentences) and they are sometimes hard to process into query-able forms. This document aims to describe the State of the Art approaches of log parsing and processing technologies. Later in this document four different integration techniques are introduced. Some of these techniques are mainly used in business solutions. Different integrations related to maintenance work are usually achieved by manual integration or by some middleware solution.

Table of Contents

1 Introduction 2
2 Log Parsing 3
2.1 Logstash 4
2.2 Hierarchical Event Log Analyzer 5
2.3 Sequence 6
3 Log Processing 7
3.1 Message Correlation 8
4 Integration techniques 9
4.1 Manual integration 10
4.2 Middleware based integration 11
4.3 Virtual Integration 12
4.4 Physical Data Integration 13
5 Conclusion 14
References 15

1 Introduction

In this document we aim to provide three approaches to log parsing and a methodology for event detection using statistical correlation between log entries. Later section introduces four common data integration methods explaining briefly how they work and their status in maintenance related work.

2 Log Parsing

The aim here is to parse the log entries that were originally written for human reading machine-readable. In this Section we will introduce three different implementations and provide some overall benchmarks how these approaches perform.

2.1 Logstash

Logstash [1] is a complex open-source, Java based log parsing, processing and visualization tool by Elastic. It might be horizontally scalable, but does not provide any operational resiliency yet. It has a log parsing tool named Grok, which is a **regular expression-based parser** (see Example 1).

Example 1. :Processing HTML request log lines [2]

55.3.244.1 GET / index .html 15824 0.043

%{ IP: client } %{ WORD : method } %{ URIPATHPARAM : request } %{ NUMBER : bytes } %{ NUMBER : duration }

It has gained popularity with its easy-to-use interface and that it uses regular expressions. Users can make patterns and therefore these patterns are easily transferable but they only provide recognition for predetermined entry formats. The algorithm is not really fast (couple 100 lines/sec) [2] as it tries every single pattern to match on the current entry. It uses sub-patterns with the following format: *%–TYPE:name"* (see Example 1.).

2.2 Hierarchical Event Log Analyzer

The Hierarchical Event Log Analyzer (HELO) [3] is used at NCSA (National Center for Supercomputing Applications, Illinois, USA) for processing the logging events generated by supercomputers. It provides online learning and offline teaching phases as well. One log entry consists of constants (that identify the entry type) and variables.

The **offline clustering component** aims to find the differentiating column between message header and the variable part (clusters). It divides the log file into clusters based on entry similarity using □cluster goodness□ evaluation that describes the similarity between messages in the clusters. This process is helped by the recognition that words containing numbers or symbols are more likely to be variables than constants. Therefore, defining the likelihood of a word (symbols, numbers, English words or a combination of those) to be a constant helps finding the right columns for establishing the baseline of the clusters.

$$Cluster\ Goodness = \frac{\#\ of\ common\ words\ of\ cluster\ messages}{average\ message\ length}$$

After this partitioning (□clustering□) is finished, the templates for the groups are created, see Example 2. HELO uses 3 type of wildcards: **d+** for numbers, ***** for any word, **n+** for values that are not present in every message. The best positions for partitioning is when the beginning of the cluster contains the most amount of constant words (e.g. the headers of the log entries). The clustering stops when the cluster goodness factor reaches 40%.

Example 2. : Pattern recognized by HELO

Added d+ subnets and d+ addresses to DB

The **online learning** capability□s purpose is to adapt the recognition to the log streams generated real-time by the system. Rare messages, that only come up in practice (that might not be present in teaching materials) have very high importance (e.g. warning entries). Therefore, catching them is essential: HELO can classify the messages as an *existing template*, slightly *alter a template* to fit the new message type or create a completely *new template* for the new message type (see Figure 1.).

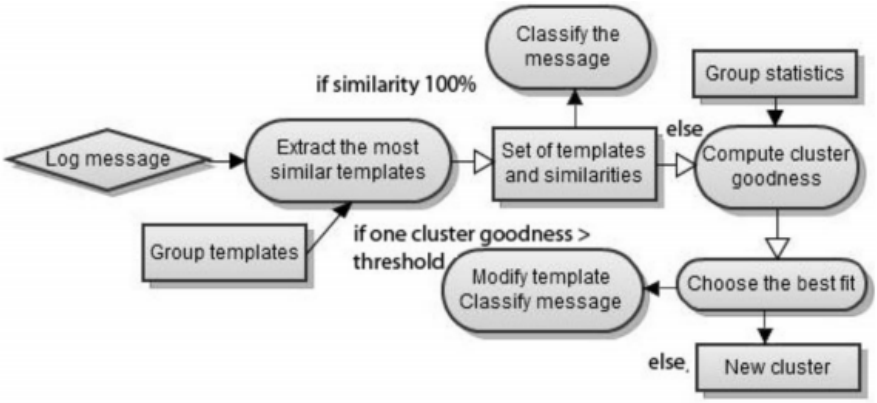

Figure 1. The HELO online component algorithm

2.3 Sequence

Sequence [4] uses a completely different approach to log parsing in comparison to the previous systems. It uses **practical heuristics** to recognize 75% of the messages at a 100 000 messages per second rate. This program has 3 elements:

- Tokenizer: processes messages and slices them into □tokens□ (detecting words and recognizes word types, see Example 3.)
- Analyzer: produces a pattern for a message type using the tokens
- Parser: uses the tokens of the current message and compares them with the pattern database

The tokenizer uses state machines to recognize elements like Time (timestamps), HexStrings (IPv6 or MAC addresses, hashes and signatures), and General (URL, IPv4, texts). All three state machines process the text simultaneously and have a strict return hierarchy. It does not use regular expressions.

The Analyzer builds a tree from the token sequences it has received from the Tokenizer: two tokens that have the same parents and children will definitely be a variable part of the message (see Example 3.).

Example 3. SSH Logs recognized by Sequencer Analyzer [4]

Mar 14 20:36:45 server sshd [29235]: Accepted password for fred from …

Nov 23 19:35:48 server sshd [3096]: Failed password for root from …

Word between "**:**" (parent) and "**password**" (child) are variables (e.g.fred or root).

The Parser gives semantical meaning to tokens using empirical observations for example:

- E-mail and host names: matches top level domain addresses from a database
- Syslog headers (RFC 3164 [5] and RFC 5424 [6])
- □Key=value□ structures with string like □src□ for sources, □dst□ for destinations, etc.

To conclude, using manual log processing best practices, we can create heuristics (like URLs in logs start with http) that are very fast and scale favourably. Still, this methodology might backfire, as it merely fits œ of the log entries.

3 Log Processing

In the previous section we dealt with recognizing individual log entries and processing them into queryable form (basically understanding the log format). In this section we will provide an approach to build on the logical connections between log entries (e.g. a warning for something might come before a failure message of the same thing).

3.1 Message Correlation

The following techniques are elaborated deeply in [7]. For analyzing correlation, a sample database is required with each message and their classifications (in this case it is done by HELO). Every event type is examined with every other type how they correlate, building a directed acyclic graph (DAG). This graph can represent the cause and effect relationships between messages and can be used for root cause analysis (RCA) purposes later.

For examining the correlation between every message pairs in a given time window, a huge, scalable data clustering algorithm is required. In this case, Density-based spatial clustering of applications with noise (DBSCAN) is used, which a data clustering algorithm proposed by Martin Ester et al in [8].

Of course, a sufficiently big data sample is required to build a statistically representative graph. Therefore statistical hypothesis tests (like Student□s t-test and Pearson correlation) might also come in handy to determine the strength of the connection detected between elements.

This graph contains now the cause-effect relationships of different message types with the following information:

- average delay between message types, deviation of events
- statistical parameters: chosen significance level, confidence level (correlation strength),

Event detection requires following this graph. In the NCSA solution [3], not all events (message types) are tracked: Event of Interests (EoI) are marked and when those are triggered, the graph is followed backwards searching for the cause messages within a timeframe. This path can be used for forecasting as well.

4 Integration techniques

Today companies have many different systems for different types of information. These systems are usually made by different companies, so communication between these systems is mostly non-existent. But it is often required or beneficial that these systems share information between each other. In many cases this "communication" is manual work done by the user. But for efficiency, it would be preferred that this integration would be automatic and would not require any work by the user. There are several different methods to achieve this type of integration.

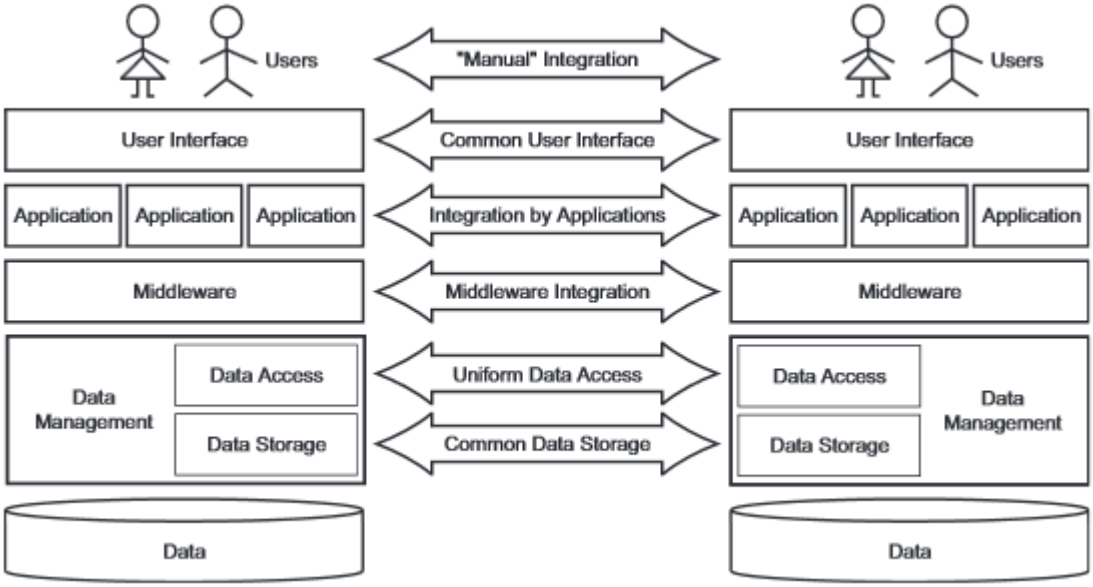

Figure 2. General Integration Approaches on Different Architectural Levels [9]

4.1 Manual integration

Manual integration is an integration type where user has to do the work. In many cases data is first exported to a common file type like csv/text file or excel spreadsheet. Then data is imported into another system. Sometimes it might even need some manual processing before being imported to target system. In some cases user may need to input the data directly between sources. Still today this is a really common method and causes extra workload.

An example scenario is where service personnel does maintenance route with a handheld device which assists making the inspections. After completing the route, he/she needs to make a report to ERP-system. But results from the handheld device cannot be imported straight to the ERP system. Service personnel has to input all the necessary information directly in to the ERP system by hand. It is also possible that same information needs to be inputted in to multiple different systems in slightly different manner. In one case, information about one fault is stored to five a different information system which are not linked to each other in any way. This makes monitoring of the chain of events difficult. [10]

4.2 Middleware based integration

In this method there is a middleware layer which handles the necessary integration. It is often referred as glue between different systems. All data is still stored in the original sources. Middleware layer has the logic which makes the link between different systems for required information. Data is queried from one system or database and passed to the other by the middleware in required format.

Legacy systems are still big part of the information especially in industry. Middleware is often only way to integrate these inflexible legacy systems to another systems without the use of manual integration. [11]

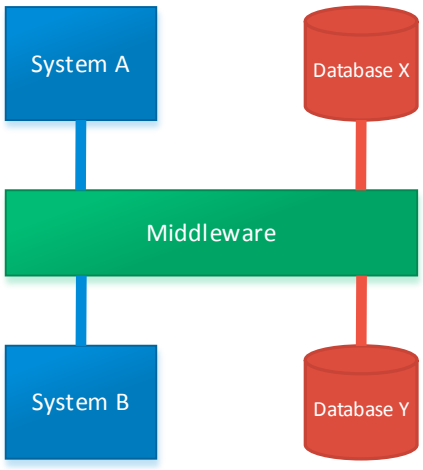

Figure 3. Middleware-based integration

4.3 Virtual Integration

This integration type is also known as Uniform data access. In this method data is left in to the source systems and integration is done at the data access level. Virtual databases are created which will provide unified views of the data from all different sources. From user standpoint it feels like they are accessing one large database instead of multiple ones.

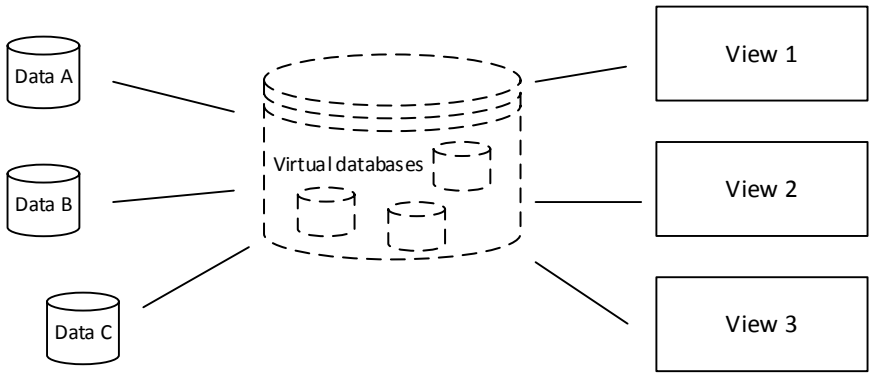

Figure 4. Virtual integration

Data federation is one subset of this integration type. Data federation allows to bring data together from heterogeneous sources. This means that data can have different storage structures, different access languages and different APIs. [12] There are multiple data federation tools available on the market.

Data Federation tools: [13]

- SAP BusinessObjects Data Federator
- Sybase Data Federation
- IBM InfoSphere Federation Server
- Oracle Data Service Integrator
- SAS Enterprise Data Integration Server
- JBoss Enterprise Data Services Platform

4.4 Physical Data Integration

This integration type is also known as common data storage. In this method all data is stored in one system. This method enables fast access to required information. To build common data storage, all previous data from other systems needs to be transferred to the new storage. Old data sources can remain operational, but this means that the common data storage needs to be periodically refreshed.

One of the better known examples of this method is Data Warehouse. Data from different sources is transferred with ETL tool. ETL is a three step process which stands for:

- Extract. Extracts data from the different data sources.
- Transform. Transforms the data into a common format so its compatible with the warehouse
- Load. Loads the data in to the warehouse.

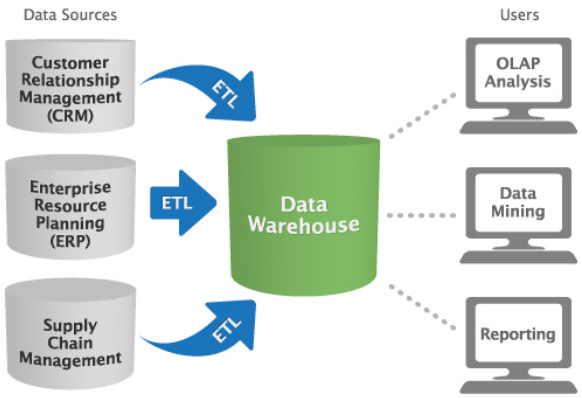

Figure 5. Data Warehouse Overview

There are lots of different commercial ETL tools available and also some open source options.

Commercial tools: [13]

- Microsoft SQL Server Integration Services (SSIS)
- IBM Infosphere DataStage
- SAP Business Objects Data Integrator (BODI)
- Oracle Data Integrator (ODI)
- Oracle Warehouse Builder (OWB)

Open source tools:

- Pentaho Data Integration (Kettle)
- Jasper ETL
- CloverETL

5 Conclusion

Most maintenance systems today use the manual and middleware type methods. The latter methods mentioned in this appendix are more common in the business solutions. It would be beneficial to a maintenance personnel that the data they are working would be accessible through only one application where they could handle all the necessary data. This would reduce the extra work they currently need to go through when working with daily maintenance reports and work orders.

In many cases the information source produces output intended for human reading, and when different sources provide such human-readable output, integration becomes challenging. Several, most commonly used log processing approaches used in telecommunications have been presented.

References

[1] Elastic, "Logstash," Elastic, [Online]. Available: http://www.elastic.co/products/logstash. [Accessed 10 08 2015].

[2] C. Moultrie, "Logstash Grok Speeds," [Online]. Available: http://ghost.frodux.in/logstash-grok-speeds/. [Accessed 10 08 2015].

[3] J. Fullop, A. Gainaru and J. Plutchak, "Real Time Analysis and Event Prediction Engine," National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign.

[4] JianZhen, □Sequencer-High Performance Log Analyzer and Parser,, [En l□nea]. Available: http://sequencer.io. [□ltimo acceso: 10 8 2015].

[5] "RFC 3164 (2001): The BSD Syslog Protocol.," [Online]. Available: https://www.ietf.org/rfc/rfc3164.txt. [Accessed 19 8 2015].

[6] "RFC 5424 (2009): The Syslog Protocol.," [Online]. Available: https://tools.ietf.org/html/rfc5424. [Accessed 19 8 2015].

[7] A. Gainaru, F. Cappello, S. Trausan-Matu and B. Kramer, "Event Log Mining Tool for Large Scale HPC Systems," *Proceedings of the 17th International Conference on Parallel Processing,* vol. I, pp. 52-64, 2011.

[8] M. Ester, H.-P. Kriegel, J. Sander y X. Xu, □A density-based algorithm for discovering clusters in large spatial databases with noise,, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* pp. 226-231, 1996.

[9] P. Ziegler and K. R. Dittrich, "Three Decades of Data Integration," [Online]. Available: http://www.cin.ufpe.br/~if694/artigos/Three%20Decades%20of%20Data%20Integration.pdf.

[10] V. Rauhala, □K□ynniss□pidon tiedonkeruun tehostaminen, K¨YNTI □ K□ynniss□pidon tiedonhallinta,□ 2013. [Online]. Available: http://www.theseus.fi/bitstream/handle/10024/56016/rauhala%20B%201%202013.pdf?sequence=1.

[11] "Middleware Technology for Integration," [Online]. Available: https://www.mulesoft.com/resources/esb/integration-middleware-technology.

[12] R. F. v. d. Lans, "Data Virtualization in Business Intelligence Architectures," 2012. [Online]. Available: https://books.google.fi/books?id=7eMd-46wXdMC&lpg=PP1&dq=data%20virtualization%20in%20business%20intelligence&pg=PP1#v=onepage&q&f=false.

[13] "Data Integration Techniques (ETL and Data Federation)," 2011. [Online]. Available: http://bi-insider.com/portfolio/data-integration-techniques-etl-and-data-federation/.

[14] A.Gainaru, □Evenrt log mining,, de *pROCEedings*, Berlin, 2011.