Use of Historian data for time balance analysis of EAF process

Federico J. Ahualli[1], Juan G. Sagasti[2]

[1]AustralTek LLC
800 Old Pond Rd St 706K, Bridgeville, PA
fahualli@australtek.com

[2]AustralTek LLC
800 Old Pond Rd St 706K, Bridgeville, PA
jsagasti@australtek.com

Keywords: EAF Time structure, Historian, Heat Tracking, Algorithms

INTRODUCTION

This paper presents a method for developing Finite-State-Machine (FSM) algorithms using historic data on a SQL (Structure Query Language) platform. As an example an FSM was implemented for a real EAF, and it generated accurate time balance data of more than 5000 heats. This approach might provide plant managers new valuable information using existing data sets that are ubiquitous in steel-making plants.

DISCUSSION

From Massive Time Series Databases to Valuable Wisdom

The amount of process data collected by computer systems has been increasing since the popularization of PLC-based industrial automation in the steel industry in the 1980s and 1990s. This growth is reflected, not only in the amount of data points (tags), but also in the amount of time the data is kept [1]. Most steel plants today have a historian database that collects a number of data points for auditing, process engineering, quality assurance and maintenance purposes and store those time-stamped values for several years in what is called a time series database (TSB). The amount of data stored is so big that custom strategies have to be used to extract valuable information, knowledge and eventually wisdom from the dataset.

The EAF Batch Process: The Heat

The purpose of this paper is to analyze a suitable strategy to extract information and knowledge from existing historian data related to **batch processes** using a simplified algorithm for a typical EAF (Electrical Arc Furnace) as an example. The EAF batch process is called *heat*. One useful method of visualizing heats is to use a time-based chart of its more important variables in what is known as the *heat fingerprint.*

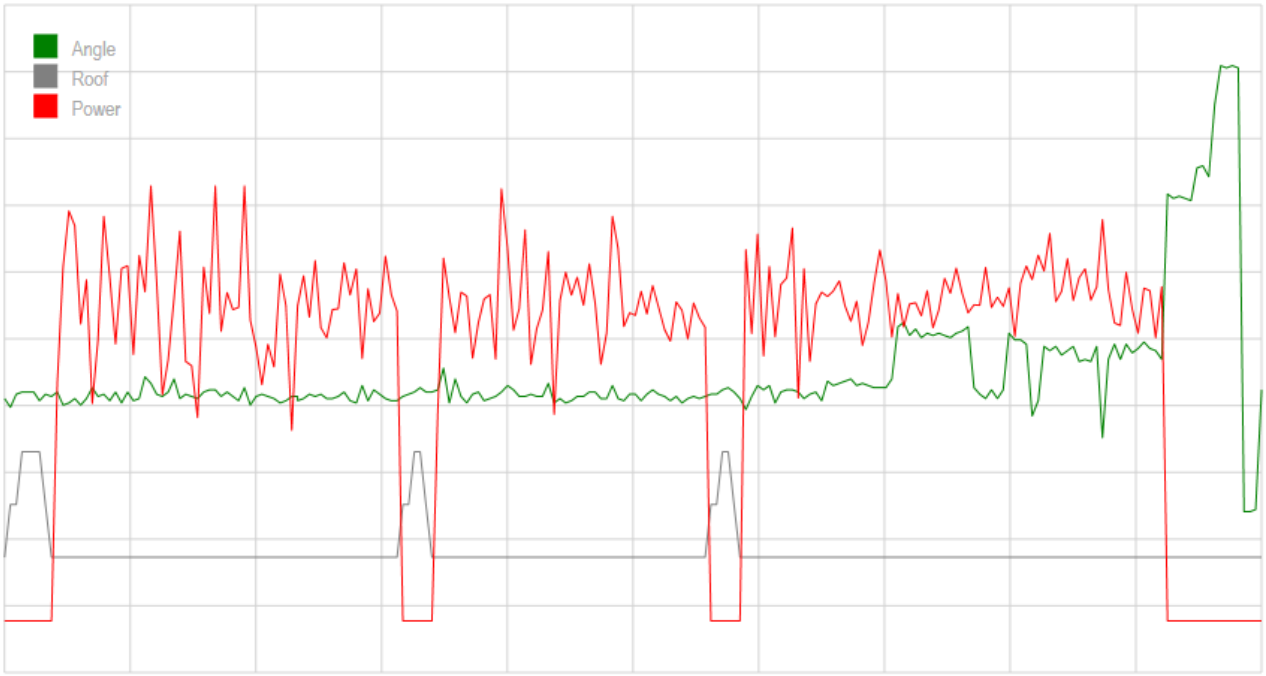

Figure 1 - Actual EAF Heat fingerprint

EAF Finite State Diagram

Unlike the fixed time dimensions found in continuous process such as a blast furnace (e.g. hour, shift, day, etc.), batch processes have dynamic time structures called *phases*. The duration, consumptions and properties of the phases are valuable information. A typical EAF *heat* can be divided into the following phases:

Preparation: Minor maintenance repairs, waiting for sequence.

Charge : Opening EAF roof and dumping scrap into it.

Melting : Applying Electrical Power to melt the solid scrap

Off: Electrical power not being applied to the bath for technological or other reasons

Tapping: Pouring steel into a ladle

The algorithm that detects the process phases and generates summarized information from them is called generically *tracking*. Most tracking algorithms are specified using a Finite State Machine [2] that comprises States and Events. Below is a typical State Diagram for an EAF.

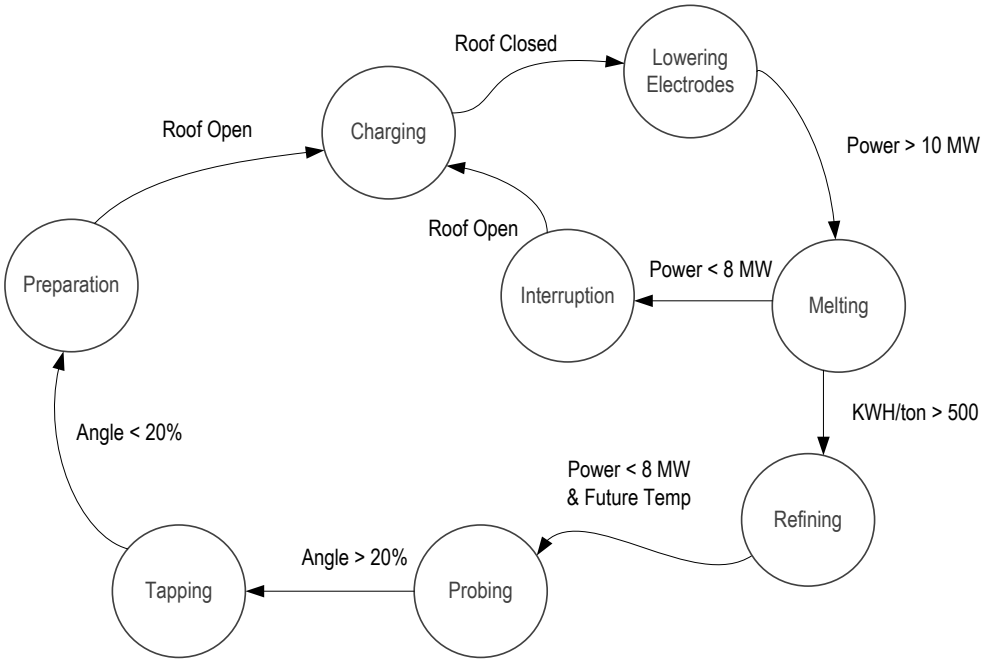

Figure 2 - Finite State Diagram for EAF process

For the sake of simplicity in this paper we are going to simplify it further and work with only 3 signals: Power, Angle and Roof Position. Below is the simplified version of the Finite State Diagram

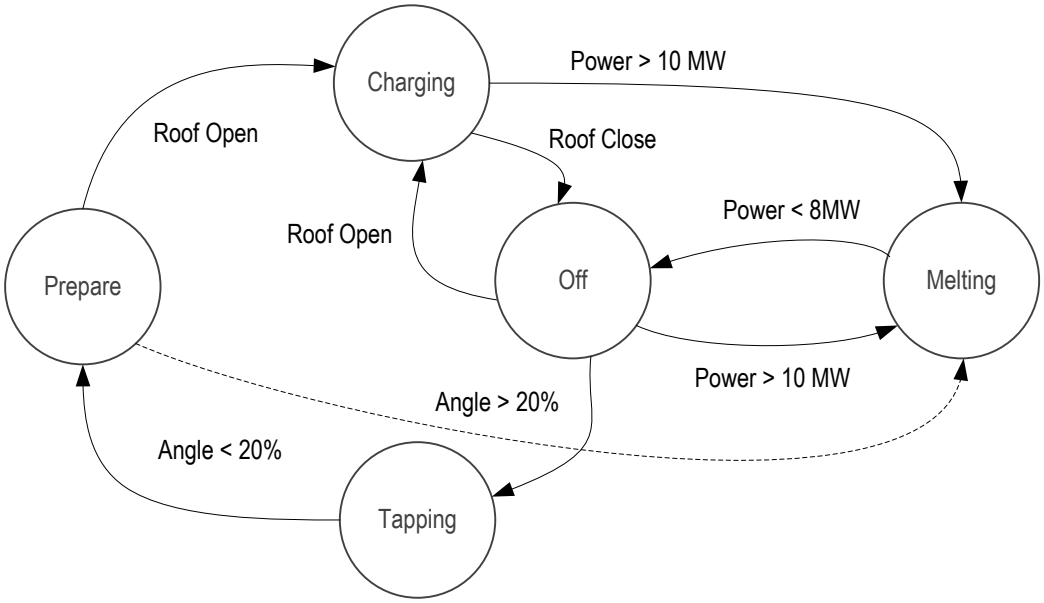

Figure 3 - Simplified Finite State Diagram for EAF

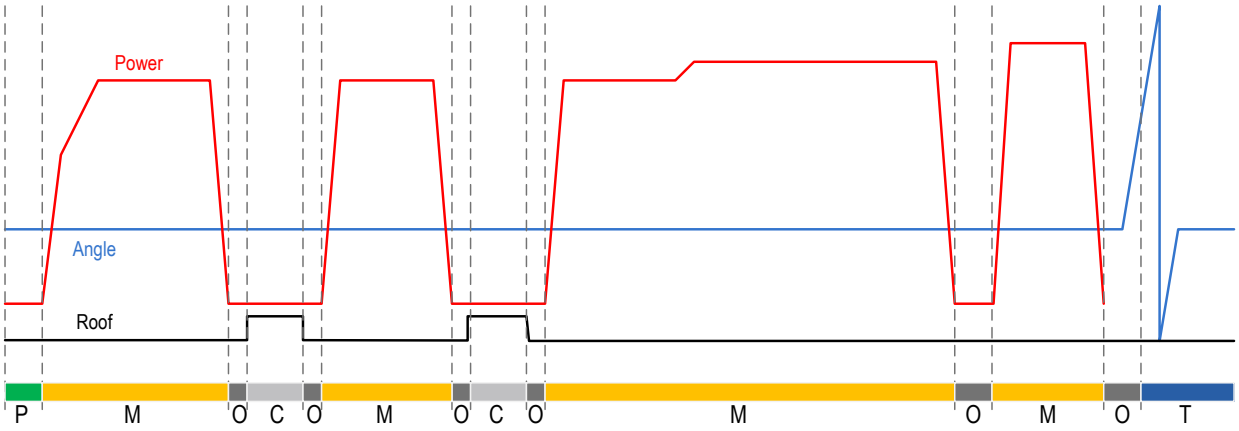

Figure 4 - Simplified EAF Heat fingerprint

Event Detection techniques

In real-time tracking, the system knows the current state and evaluates only those events that would trigger a state transition; not all events and variables are evaluated at all times. However, when applied to historical analysis, this method requires executing the FSM algorithm through the entire dataset, which can be very time consuming and impractical. In order to improve performance using the advantages mentioned in the previous paragraphs, the dataset is pre-processed and reduced to a list of triggering events. Even when not all events generated with this method will trigger a state change, the filtering process is effective in reducing the amount of data required.

In order to generate the events, we evaluate the analog variables against a threshold. Then in order to improve the robustness the following techniques are applied:

- Hysteresis - Threshold Value
- Minimum Time

The hysteresis threshold filter allows the elimination of intermittent state changes due to a variable oscillating close to the threshold limit. The minimum time eliminates high frequency peaks on the variable that can be ruled out as not real when compared with the inertia of the studied system

For example, we want to mark the start of the melting phase when the Active Power is above 5 MW but only if on that cycle it will reach at least 15 MW. Also we want the power to stay in this state for at least 10 seconds. By using this definition we avoid invalid detections due to noise or temporary instabilities.

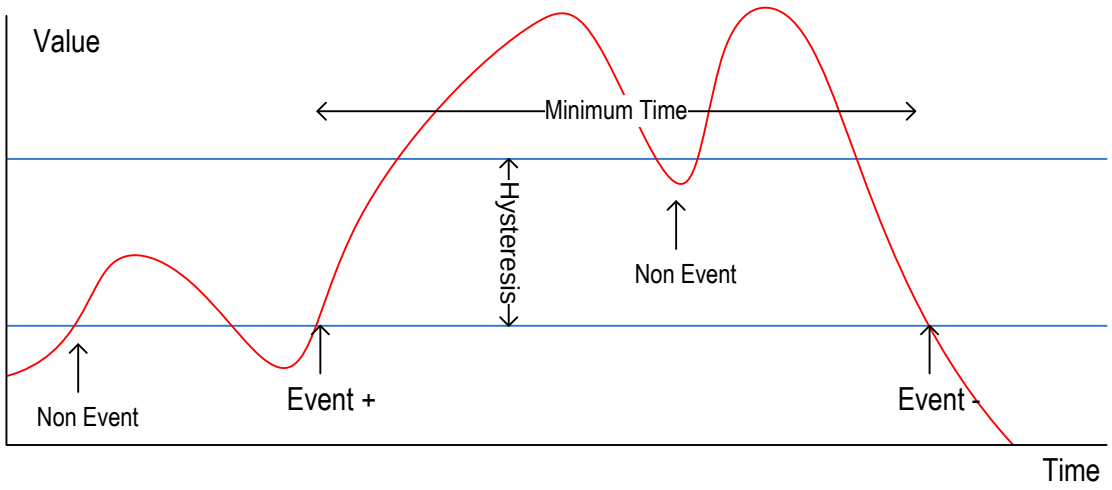

Figure 2 - Hysteresis Threshold Filter

The same analysis can be applied if we want to mark the end of a cycle when the power comes from 15 MW and goes below 5 MW. If we use the same set of limits for detecting the positive events and negative events on a variable, the entire plane is divided into positive sections and negative sections with no superposition or empty spaces. We call this detection *binary* implying that, at any point in time, the evaluation of the variable will be on the positive or negative side.

In order to implement hysteresis threshold in queries using SQL language, we define 3 bands per variable and label them 0,1, and 2 as in the figure below and run a query that evaluates in what band the variable is at any moment.

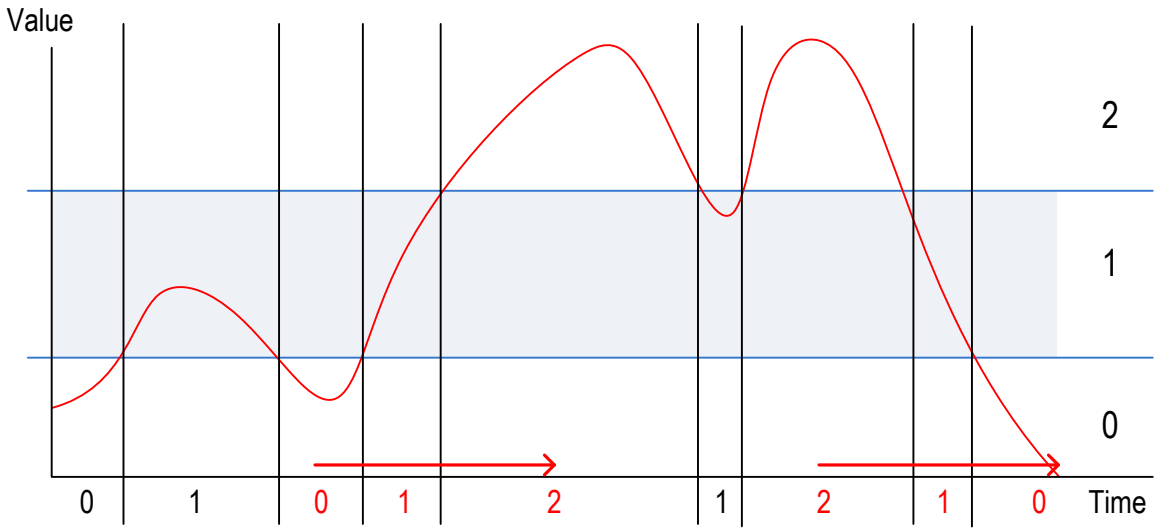

Figure 3 - Calculation of events on an analog variable

This query allows us to simplify the detection of the triggering events. Based on this characterization the events will be generated only when a sequence of **0,1,2 (positive)** or **2,1,0 (negative)** is detected

A suitable set of parameters for the filters described in our EAF example is the following:

Variable	Limit 1	Limit 2	Minimum Time
Power	5 MW	15 MW	10 sec
Angle	-20 Deg	-10 Deg	10 sec
Roof	N/A	N/A	5 sec

Table I - Sample Parameter set

Running the FSM algorithm

After we run the event detection in all the dataset, we obtain a reduced table consisting of only the filtered events; we call this table the *event list*. The following step is to use the FSM diagram and translate it into a *transition matrix*. The transition matrix is another way to express a Finite State Diagram which is suitable for SQL database evaluation. The transition matrix for the simplified FSM above is:

State	Power +	Power -	Angle +	Angle -	Roof +	Roof -
Melting		Off			Error	
Charging	Melting		Error			Off
Off	Melting		Tapping		Charging	
Tapping				Prepare	Error	
Prepare	Melting				Charging	
Error	Melting					

Table II - EAF Transition Matrix

There are a couple of things that become evident in the above table. One is that the Power+ column determines the next state as Melting in most of the cases. We can use this to our advantage and mark the Power+ event as an eigenvalue event, that is, an event that uniquely determines the next state regardless of what state it is detected in. This is useful to bootstrap our detection algorithm.

We also see an "Error' state in the last row of the matrix. This state corresponds to errors that might appear during the adjustment phase of our FSM and are a possible sign that the tracking parameters (thresholds and minimum time) are not correct or that our dataset has errors. We can have one generic error state or label different transition errors with different names. In our EAF example if we detect that the roof is opening while the furnace is tapping, it might be an indication of an error and we should check the parameters or the dataset. In our transition matrix, the only way to get out of the error state is to detect Power+ and we use this to start up and synchronize the algorithm.

When we apply the event list to the transition matrix, we get a list of all states (or phases) in chronological order.

Identifying the Batches

As mentioned before, each batch has definite set of phases and one of them can be considered the first phase. If that phase is unique in the process, we can then group together all the states belonging to the same batch and generate the "list of batches" or heat list. In the case of the EAF tracking, we can identify the **next state after tapping** as our first state, called preparation. Using that state as a starting point, we can get the list of heats.

Obtaining more data and knowledge

Once the batches are identified and properly time-tagged, then it is a straightforward process to query the historian database to extract the rest of the information. An EAF Heat tracking algorithm would be able to detect a number of key process indicators (KPIs) such as:

- Preparation Time
- Number and duration of charges
- Number and duration of melting interruptions

- Time from power off to roof open command
- Time from roof open command to roof open
- Time from roof close command to roof closed (roof speed)
- Time from roof closed to power (electrode speed)
- O2 / Carbon / NG steps duration
- Time from refining to temperature probe
- Time from temperature probe to tapping
- Refining duration (obtained from KWH/ton or arc stability events)
- Tapping time
- Transformer Tap change events and duration
- De-slagging events and duration
- Off Gas Damper / Furnace pressure events and duration

A number of energy related KPIs can be also obtained using off gas/water flow rates and temperatures to estimate bath atmosphere conditions:

- Heat losses on off gas and its peak timestamp
- Heat losses on cooling water and its peak timestamp

And finally the standard KPIs most steel shops have:

- Consumptions per phase (Energy, Oxygen, Natural Gas, Carbon)
- Total values for Power On Time, Power Off, Total Energy, Total Oxygen etc.

Parameter Adjusting Techniques

Most of the tracking algorithms in the industry run by checking conditions and events in real-time as they occur in order to provide feedback to the process and/or the operators. In our case, we only want to generate information for analysis, and we have the complete time-series for all the involved signals reading to be used with the power of SQL. When analyzing a point in time, our algorithm can take advantage by accessing all the past and future values. Additionally, one of the most difficult parts of designing a batch tracking algorithm is to estimate and adjust the detection parameters. By using a historian database, the parameter finding process can be iterative, which makes the procedure faster and more robust.

Once a parameter set is proposed, it is run against the whole dataset. The results are evaluated and, if they require further adjustment, they can be corrected and tested again. The following diagram explains the basic procedure.

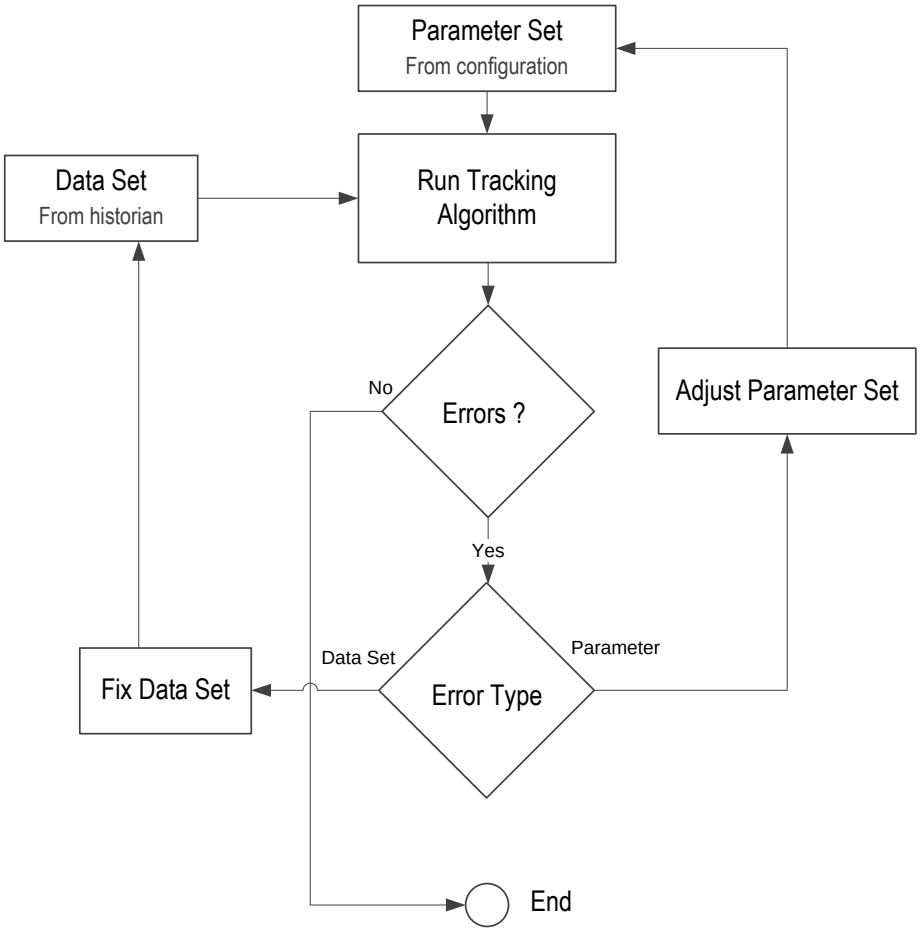

Figure 4 - Parameter Adjustment Flowchart

EXPERIMENTATION

The method described was tested against datasets collected in two different steel plants, each with different historian software. The results were compared with the MES or Level 2 records of heats. The following table summarizes the tests:

Plant	Historian Software	MES / Level 2 Heats	Detected Heats
Plant #1	Wonderware InSQL	3580	3562
Plant #2	Industrial Historian	2694	2645

Table III – Summary of algorithm performance

The discrepancies found on the detected and recorded heats were analyzed and they were found to be related to abnormal operations, such as pour backs and tapping problems. It is possible to improve the success ratio by adding more signals such as “EBT Open” and improving the Finite State Machine to take non-standard operations into account.

CONCLUSION

A method to obtain valuable information from time-based historians using relational database tools was developed and applied to simplified particular case such as the EAF process. The method was tested in two different environments and yielded acceptable results.

REFERENCES

1. Brian Courtney, "Industrial big data analytics: The present and future" *InTech Magazine 2014 Jul-Aug* https://www.isa.org/intech/20140801/

2. Brookshear, J. Glenn (1989). "Theory of Computation: Formal Languages, Automata, and Complexity". Redwood City, California: Benjamin/Cummings Publish Company, Inc.ISBN 0-8053-0143-7.