Anomaly Detection
Feras A. Batarseh
College of Science, George Mason University, Fairfax, VA, USA

Synonyms
Defect detection; Error tracing; Testing and evaluation; Verification

Definition
Anomaly Detection is the process of uncovering anomalies, errors, bugs, and defects in software to eradicate them and increase the overall quality of a system. Finding anomalies in big data analytics is especially important. Big data is “unstructured” by definition, hence, the process of structuring it is continually presented with anomaly detection activities.

Introduction
Data engineering is a challenging process. Different stages of the process affect the outcome in a variety of ways. Manpower, system design, data formatting, variety of data sources, size of the software, and project budget are among the variables that could alter the outcome of an engineering project. Nevertheless, software and data anomalies pose one of the most challenging obstacles in the success of any project. Anomalies have postponed space shuttle launches, caused problems for airplanes, and disrupted credit card and financial systems. Anomaly detection is commonly referred to as a science as well as an art. It is clearly an inexact process, as no two testing teams will produce the same exact testing design or plan (Batarseh 2012).

Anomaly Examples
The cost of failed software can be high indeed. For example, in 1996, a test flight of a European launch system, Ariane 5 # 501, failed as a result of an anomaly. Upon launch, the rocket veered off its path and was destroyed by its self-destruction system to avoid further damage. This loss was later analyzed and linked to a simple floating number anomaly. Another famous example is regarding a wholesale pharmaceutical distribution company in Texas (called: Fox Meyer Drugs). The company developed a resources planning system that failed right after implementation, because the system was not tested thoroughly. When Fox Meyer deployed the new system, most anomalies floated to the surface, and caused lots of users’ frustration. That put the organization into bankruptcy in 1996. Moreover, three people died in 1986 when a radiation therapy system called Therac erroneously subjected patients to
lethal overdoses of radiation. More recently however, in 2005, Toyota recalled 160,000 Prius automobiles from the market because of a software anomaly in the car’s software. The mentioned examples are just some of the many projects gone wrong (Batarseh and Gonzalez 2015); therefore, anomaly detection is a critical and difficult issue to address.

Anomaly Detection Types

Although anomalies can be prevented, it is not an easy task to build fault-free software. Anomalies are difficult to trace, locate, and fix; they can occur due to multiple reasons, examples include: due to a programming mistake, miscommunication among the coders, a misunderstanding between the customer and the developer, a mistake in the data, error in the requirements document, a politically biased managerial decision, a change in the domain market standards, and multiple other reasons. In most cases, however, anomalies fall under one of the following categories (Batarseh 2012):

1. **Redundancy** – Having the same data in two or more places.
2. **Ambivalence** – Mixed data or unclear representation of knowledge.
3. **Circularity** – Closed loops in software; a function or a system leading to itself as a solution.
4. **Deficiency** – Inefficient representation of requirements.
5. **Incompleteness** – Lack of representation of the data or the user requirements.
6. **Inconsistency** – Any untrue representation of the expert’s knowledge.

Different anomaly detection approaches that have been widely used in many disciplines are presented and described in the Table 1.

<table>
<thead>
<tr>
<th>Anomaly detection approach</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection through analysis of heuristics</td>
<td>Logical validation with uncertainty, a field of artificial intelligence</td>
</tr>
<tr>
<td>Detection through simulation</td>
<td>Result-oriented validation through building simulations of the system</td>
</tr>
<tr>
<td>Face/field validation and verification</td>
<td>Preliminary approach (used with other types of detection). This is a usage-oriented approach</td>
</tr>
<tr>
<td>Predictive detection</td>
<td>A software engineering method, part of testing</td>
</tr>
<tr>
<td>Subsystem testing</td>
<td>A software engineering method, part of testing</td>
</tr>
<tr>
<td>Verification through case testing</td>
<td>Result-oriented validation, achieved by running tests and observing the results</td>
</tr>
<tr>
<td>Verification through graphical representations</td>
<td>Visual validation and error detection</td>
</tr>
<tr>
<td>Decision trees and directed graphs</td>
<td>Visual validation – observing the trees, and the structure of the system</td>
</tr>
<tr>
<td>Simultaneous confidence intervals</td>
<td>Statistical/quantitative verification</td>
</tr>
<tr>
<td>Paired T-tests</td>
<td>Statistical/quantitative verification</td>
</tr>
<tr>
<td>Consistency measures</td>
<td>Statistical/quantitative verification</td>
</tr>
<tr>
<td>Turing testing</td>
<td>Result-oriented validation, one of the commonplace artificial intelligence methods</td>
</tr>
<tr>
<td>Sensitivity analysis</td>
<td>Result-oriented data analysis</td>
</tr>
<tr>
<td>Data collection and outlier detection</td>
<td>Usage-oriented validation through statistical methods and data mining</td>
</tr>
<tr>
<td>Visual interaction verification</td>
<td>Visual validation thought user interfaces</td>
</tr>
</tbody>
</table>

However, based on the recent study by National Institute of Standards and Technology (NIST), the data anomaly itself is not the quandary, it is actually the ability to identify the location of the anomaly. That is listed as the most time-consuming activity of testing. In their study, NIST researchers compiled a vast number of software and data projects and reached the following conclusion: “If the location of bugs can be made more precise, both the calendar time and resource requirements of testing can be reduced. Modern data and software products typically contain millions of lines of code. Precisely locating the source...
of bugs in that code can be very resource consum-
ing.” Based on that, it can be concluded that
anomaly detection is an important area of research
that is worth exploring (NIST 2002; Batarseh and
Gonzalez 2015).

Conclusion

Similar to most engineering domains, software
and data require extensive testing and evaluation.
The main goal of testing is to eliminate anomalies,
in a process referred to as anomaly detection.

It is not possible to perform data analysis if the
data has anomalies. Data scientists usually per-
form steps such as data cleaning, aggregation,
filtering, and many others. All these activities
require anomaly detection to be able to verify
the data and provide valid outcomes. Addition-
ally, detection leads to a better overall quality of a
data system, therefore, it is a necessary and an
unavoidable process. Anomalies occur for many
reasons, and in many parts of the system, many
practices lead to anomalies (listed in this entity),
locating them, however, is an interesting engi-
neering problem.

Further Readings

Batarseh, F. (2012). Incremental lifecycle validation of
knowledge-based systems through CommonKADS.
Ph.D. Dissertation Registered at the University of
Central Florida and the Library of Congress.

in contextual software development through data
analytics. Proceedings of Springer’s Software Quality
Journal.

Planning Report for NIST. (2002). The economic impacts
of inadequate infrastructure for software testing. A
report published by the US Department of Commerce.