david consulting group
TRUSTED ADVISOR

"What are "Software Analytics" and who should care about them?"

October 2013

Scope of this Report

Everyone has a different definition of Software Analytics and so it is fair to say that this is a broad topic. Hence, this report defines Software Analytics as much by the examples it gives as by one single formal definition. In answering the "who should care?" part of the question, the report seeks to identify roles that should be using Software Analytics already and/or could be using it more. We also look into the future to see how Software Analytics might change software development in organizations in the way that business intelligence has impacted some organizations.

What are "Software Analytics"?

This definition of "Software Analytics" comes from a 2011 paper by Zhang et al:

> *"Software Analytics" – to utilize the data-driven approach to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for completing various tasks around software systems, software users, and software development process.*

So, can we think of Software Analytics as "big data" for software development in that the focus is on mining the richness of data about software projects for a surprising new insight? In short, yes. But most practitioners set the bar a little higher than just "new insights." The goal of software analytics is to produce real-time and actionable insights where "real-time" is defined to be within the time frame of a project such that corrective action is feasible more quickly than it would be by just letting the project fail.

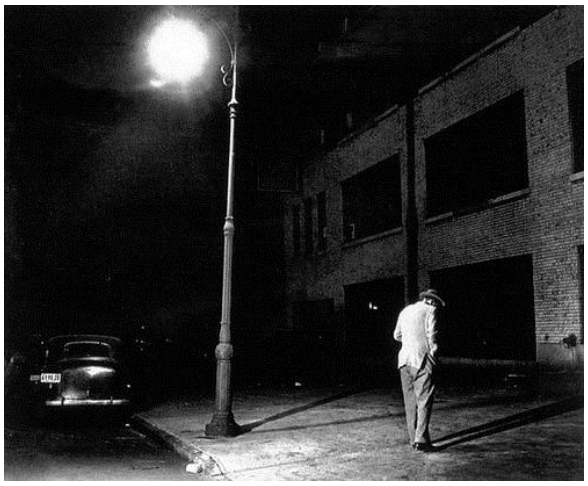
Some examples of Software Analytics include (more examples are provided later in the report):

- Using data from previous projects to size and estimate future projects
- Using defect data to allocate effort in a running project
- Benchmarking projects and project against industry data
- Statistically and/or heuristically analyzing code to look for patterns and anti-patterns.

Menzies and Zimmerman identify one important goal and constraint for Software Analytics – it is about using lessons from one set of projects to improve results on another set of projects. Consequently, the value of Software Analytics is undermined if the right data is not available for analysis (because it doesn't exist, is not sufficiently structured or the owners are not prepared to share) or if the results cannot be readily shared (because they are not sufficiently structured or the owners are not prepared to share).

Current use of the term "Software Analytics" suggests a tipping point. In the past, "Software Analytics" seems to have been used quite narrowly, more often than not in the context of analyzing defects. More

recently and perhaps due to the growth in business intelligence ("Big Data" if you must), an awareness has grown that the software development process and the subsequent operation of the software throws off a lot of data. Historically, some of this data has been analyzed but much has been ignored by organizations. Going forward, under the more broadly defined "Software Analytics" banner, there is an expectation that important insights can be uncovered to improve decision-making around software development and operation.



In his paper, "Searching under the Streetlight for Useful Software Analytics," Johnson makes an excellent point about the importance of unobtrusive data collection. One of the reasons that much data goes to waste is because, in Johnson's words: "For developers, one of the most frustrating aspects of manual data collection is the loop of doing some work and then interrupting it to record what they worked on." We would go further to add that most developers are trained to question the purpose of any measurement and analysis of what they do and smart enough to "game" any data collection system that they don't value.

Different and Distinct Kinds of Software Analytics

Everyone has a different definition of software analytics and this is justified to some extent by the variety. For example:

- Internal versus External Analytics
    - Live (or directly accessible) Data versus Stale (or exported and sent to a 3rd party) Data – Internal analysts are much more likely to be able to access live data than external analysts.
    - Privacy – external analysts may necessarily only have access to anonymous data
- Quantitative versus Qualitative Methods
    - Typically, quantitative analysis of data is thought of as somehow more "accurate" than qualitative data but often quantitative data can be misleading or downright wrong without the benefit of qualitative context.
    - DCG's own ADM Performance Baseline combines quantitative analytics with qualitative interviews to provide our clients with an assessment of how their software development group is performing AND why!
- Data mining Tools versus Interactive Tools

    - As the names imply, data mining tools are usually automatic, configured to run regularly and perform the same analysis every time on a fresh (new, expanded or updated) set of data each time.
    - Today, most Software Analytics falls into the data mining category.
- Practitioner Audience versus Management Audience
    - Different audiences have different data needs for different decisions.
- Exploratory versus Deployment Analytics
    - In Exploratory Analytics, the goal is to explore the data to try to find potentially valuable insights
    - In Deployment Analytics, the goal is to automatically and regularly generate the information from the valuable insight and deploy it out into the organization to inform decisions.
    - An example might be that an exploratory analysis might look for a size of project (in function points) at which the defect rate deteriorates significantly. If found then deployment analysis can be used to look for and report on projects that exceed or are close to exceeding this number of function points.

Applications of Software Analytics

Since the types of Software Analytics vary, it is reasonable to expect the applications of software analytics to have even greater range. For example:

From Menzies & Zimmerman in IEEE Software (references to the underlying case studies are included in the Menzies & Zimmerman paper):

• Combining software product information with apps store data	• Using process data to predict overall project effort
• Using software process models to learn effective process change	• Using operating system logs that predict software power consumption
• Exploring product line models to configure new applications	• Mining natural language requirements to find links between components
• Mining performance data	• Using XML descriptions of design patterns to recommend particular designs
• Using email lists to understand the human networks inside software teams	• Linking emails to source code artifacts and classifying their content
• Using execution traces to learn normal interface usage patterns	• Using bug databases to learn predictors that guide inspection teams to where the code is most likely to fail
• Using security data to identify indicators for software vulnerabilities	• Using visualization to support program comprehension
• Using software ontologies to enable natural language queries	• Mining code clones to assess the implications of cloning and copy/paste in software

The variety can exist across organizations and within organizations. For example, Microsoft have at least two systems: Czerwonka at al have reported on the development of CODEMINE at Microsoft; Zhang et Al have reported on the Stackmine Project. CODEMINE is a software development analytics platform for

collecting and analyzing engineering process data. It includes constraints and pivotal organizational and technical choices. Examples of uses include:

- Trend monitoring and reports on development health
- Risk evaluation and change impact analysis
- Version control branch structure optimization
- Social-technical data analysis
- Custom search for bugs and debug logs, speeding up investigations of new issues.

StackMine was developed to perform Software Analytics on a very particular set of data in a Microsoft system called PerfTrack which logs reports of poor end user performance. For example, PerfTrack measures how long it takes for a user to receive a response from the system when the user clicks on a Windows Explorer button to create a new folder. If the response time exceeds a satisfactory threshold, PerfTrack sends execution traces (containing call stacks collected during the preceding time interval) back to Microsoft for debugging. All the collected call traces in total could contain more than 1 billion call stacks – much more than performance analysts can expect to analyze without automation.

Who should care about Software Analytics?

Everyone who has to make decisions about software development and software operation should care about Software Analytics. Not only because data-driven decisions are usually better than other kinds but also because if Software Analytics are not being used to drive decisions then the collection of that data is almost certainly a waste of time and money.

Johnson and his colleagues experience with designing Software Analytics led him to suggest three dimensions for evaluating approaches (we have simplified):

- Data Collection: Automation versus Manual collection
- Adoption Barriers: Comprehensiveness versus Intrusion (social or political)
- Applicability of Approach: Universal (e.g. all projects) versus Narrow (e.g. only java projects)

Optimizing an approach across these dimensions requires the attention of all those who have a stake in the outcome and/or incur a cost through the implementation.

If Software Analytics is, to some degree, about “Big Data” then is it really relevant to smaller companies? Our experiences with smaller clients at DCG is absolutely “yes” and, again applying some objectivity, those experiences correlate very well with the experience of Amisoft reported by Robbes, Vidal and Bastarrica. While only a few DCG clients introduce Software Analytics as part of a CMMI initiative, this was one of the drivers for Amisoft but they also reported the following additional drivers:

- Determining whether employees were really following the company’s process
- Measuring actual adherence to the company’s process
- Gathering evidence of whether the process was a net positive for the company
- Increasing the visibility of activities
- Locating opportunities for improvement

Amisoft saw benefits from the Software Analytics program in the following areas:

- Strategic decisions
    - Scheduling – analysis of historical data allows for optimization of the contingency buffer applied to project delivery dates

    - Requirement volatility – measurement showed that requirement volatility was so high that it was detrimental to project success. Steps were taken to reduce volatility and continue to monitor it.
    - Verification and Validation – the company realized that they were not putting enough time into this part of the process and quality of the finished product was at risk.
- Tactical Decisions
    - Personnel – Project managers monitored several aspects of employee performance actively during projects to allow for immediate correction of problems instead of retrospective analysis of why the project failed.
    - Client Interaction – Project managers used the reports on Requirements volatility and incidents to manage the client relationship.
    - Rescheduling – planning and task status data was used to prompt rescheduling or reassignment of at-risk tasks.

Menzies and Zimmerman make some predictions about changes we might expect by 2020 due to the coming impact of Software Analytics. You should care about Software Analytics if any of these predicted changes might impact you:

- More and different data;
- More algorithms;
- Faster decision making with the availability of more data and faster release cycles;
- More people involved in analytics as it becomes more routine to mine data;
- More education as more people analyze and work with data;
- More roles for data scientists and developers as this field matures with specialized sub-areas;
- More real-time analytics to address the challenges of quickly finding patterns in big data;
- More analytics for software systems such as mobile apps and games;
- More impact of social tools in analytics.

Conclusion

Software Analytics is the process of extracting meaningful insights from the data produced during software development and operation. Software analytics should be sought after and used by all those people who have to make decisions about software development and operation. Data-driven decisions are usually better than decisions based on “gut feel”, imperfect memory or business as usual.

Sources

- Dongmei Zhang, Yingnong Dang, Jian-Guang Lou, Shi Han, Haidong Zhang, and Tao Xie. "Software Analytics as a Learning Case in Practice: Approaches and Experiences". In Proceedings of International Workshop on Machine Learning Technologies in Software Engineering (MALETS 2011), Lawrence, Kansas, November 2011.
- “Software Analytics: So What?”, Tim Menzies & Thomas Zimmerman, IEEE Software, July/August 2013
- “Searching under the Streetlight for Useful Software Analytics”, Philip M. Johnson, IEEE Software, July/August 2013

- “CODEMINE: Building a Software Development Data Analytics Platform at Microsoft”, Jacek Czerwonka, Nachiappan Nagappan, Wolfram Schulte, Brendan Murphy, IEEE Software, July/August 2013
- “Software Analytics in Practice”, Dongmei Zhang, Shi Han, Jian-Guang Lou, Haidong Zhang, Tao Zie, IEEE Software, September/October 2013
- “Are Software Analytics Efforts Worthwhile for Small Companies? The Case of Amisoft”, Romain Robbes, Rene Vidal, Maria Cecilia Bastarrica, IEEE Software, September/October 2013
- “IEEE Software”, July/August 2013.
- “IEEE Software”, September/October 2013.