NordDesign 2016
August 10 – 12, 2016
Trondheim, Norway

Towards big-data analysis of deviation and error reports in product development projects

Ívar Örn, Arnarsson[1], Johan Malmqvist[1], Emil Gustavsson[2], Mats Jirstrand[2]

[1]Product and Production Development, Chalmers University of Technology
varo@chalmers.se, johan.malmqvist@chalmers.se
[2]Fraunhofer Chalmers Centre (FCC) for Industrial Mathematics, Chalmers Science Park
emil.gustavsson@fcc.chalmers.se, mats.jirstrand@fcc.chalmers.se

Abstract
Large complex system development projects, such as complete truck development projects, take several years to carry out. They involve hundreds of engineers who develop tens of thousands of parts and millions of lines of codes. During a project, many design decisions often need to be changed due to emergence of new information. The bulk of these changes are requested late in the development process. It is known that changes late in the development process are very costly and run a risk of delaying the project. These changes are often well documented in databases, but, due to the complexity of the data, few companies analyze engineering change in a comprehensive and structured fashion. This paper argues that "big data" (specifically data mining) analysis tools can be applied for such analyses and proposes a process for carrying out the analysis and using the results for product and development process improvement. The paper further accounts for experiences gained from testing the approach on a dataset consisting of 4,000 deviation and error reports that were created during a truck development project.

Keywords: Product development, late product changes, big-data analysis.

1 Introduction

Truck development projects involve hundreds of engineers developing tens of thousands of parts and millions of lines of codes. The projects have durations of several years. During the course of a project, many design decisions are made. Many decisions need to be changed during a project due to, e.g., the emergence of new information, simulation or test results. Unfortunately, the bulk of change needs tend to be discovered late in the process, as shown in Figure 1 (Giffin et al., 2009).

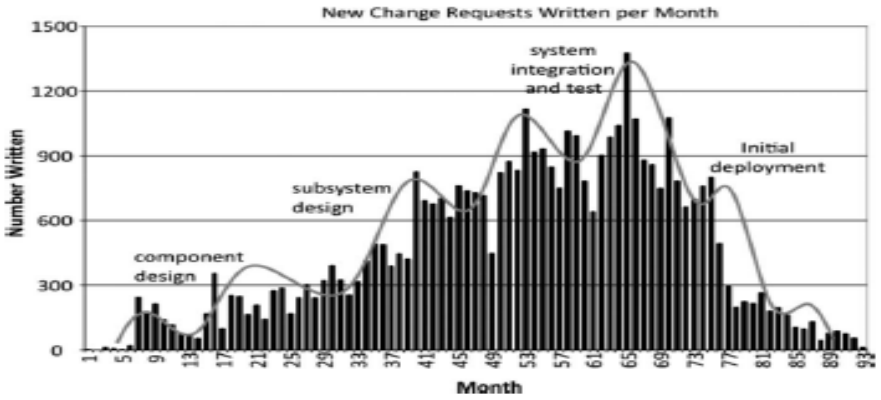

Figure 1. Frequency of change requests during a complex system development project (Giffin et al., 2009), number written per month.

It is well known that changes made late in a development project are very costly and may cause delays (Clark and Fujimoto 1991). Causes of late changes in product development projects can result in failure to meet objectives for budget, schedule, or technical performance. Weak leadership and lack of planning and rigid processes play important roles in this (Thomke & Reinertsen, 2012). The problem with late product changes has been known for several decades but recent studies confirm that it remains a challenge associated with high costs, quality problems, and development lead time delays (Giffin et al., 2009; The Standish Group, 2014; Fernandes et al., 2015). The root causes of these changes are still poorly understood. It seems that mitigations that were proposed during the 1980's such as concurrent engineering and quality function deployment were not sufficient to solve the problem.

The product changes in product development projects are often logged and stored in databases where structured data (i.e., numerical data and time stamps) are mixed with unstructured data (i.e., text inserted by engineers). With the development within machine learning and data mining, new techniques for retrieving insights from complex data sets have emerged. Machine learning is the field where algorithms are utilized in order to explore the data and to make predictions from the data (for a survey, see, e.g., Kotsiantis et al., 2007). Data mining is another closely related research field but where the aim is more at detecting patterns and knowledge in the data sets (for a survey, see, e.g., Berkhin, 2006). In the context of this paper, the application of such tools provide an opportunity to connect the documentation of errors and tests made in a product development project with the upstream decisions and working practices that were their root causes. For the sake of compactness, we refer to these data analysis tools as "big data" analysis tools in the remainder of the paper.

The aim of the project outlined in this paper is thus to develop big-data based tools to analyze product deviations and to connect them with their originating root causes, enabling the root cause to be eliminated rather than the symptoms. In the paper, we specifically apply data mining tools for this purpose. We address two research questions:

1. *Is it possible to use product deviations reports from databases and apply data mining analysis to find patterns that will aid in finding root causes for deviations and repeatable failures?*
2. *Is it possible from this analysis to build fact-based hypotheses for how to improve product quality in the late phases of a project?*

2 Earlier work

As noted in Section 1, solutions that were proposed during the 1980's have not been sufficient to solve the problem with late changes. This does not necessarily suggest that the solutions were "wrong" as product development complexity and time pressure have increased during the period. Nevertheless, continued attention to this phenomenon is needed.

Recent studies have identified poor requirements management (Fernandes et al., 2015) and difficulties in predicting impact of design changes resulting in late discovery of problems (Eger et al., 2007, Giffin et al., 2009) as causes for late changes. Closely related to that reason, Thomke & Reinertsen (2012) maintain that companies tend to need more time to adjust to constantly evolving market needs, which can lead to later detection of product weakness. Thomke & Reinertsen further claim that many companies try to over-utilize their product development resources. When product development employees have nearly full plates; speed, efficiency, and output quality decreases (Thomke & Reinertsen, 2012). When resources are highly utilized, queues in projects tend to appear. Queuing can yield e.g. unavailability of resources, longer duration of projects, delayed feedback, and cause developers to be more unproductive. However, there are many more possible causes, including, e.g., lack of or poor use of simulation tools (Söderberg et al., 2013), the use of too few physical prototypes, too few milestones/lack of continuous follow up, and too cumbersome error reporting systems leading engineers to avoid reporting errors. Moreover, previous work has only considered "single" products/systems. Variant-rich, platform-based products such as trucks introduce an additional level of complexity to the task of understanding causes and impact of changes and errors. There is thus a need to look at this situation in a more detailed and comprehensive way, consider a larger set of causes and mitigations, as well as the more complex process of platform-based development.

Focusing on the computational support for such an analysis, some researchers are developing big data mining methods to identify structures or patterns in engineering information. Researchers have developed methods to analyze e-mail databases and social media tools for making inferences about project status and for connecting relevant specialists to queries and issues (Hicks et al., 2013). Earlier studies have also shown that change requests can be analyzed with network graphs (Giffin et al., 2009). Network graphs can assist in visualizing how change requests are related to one another, emerging from a single parent or if they are disconnected. Change analysis during ongoing product development (Eger et al., 2007) use a node-link diagram to allow the designer to monitor the progress of the project. The tool makes a change propagation tree to provide an exploded view of the design links, aiding in identifying changes paths. Tree diagrams and scatter graphs have also been used to analyze data (Giffin et al., 2009). On a more general level, emerging technologies, in particular for searching and browsing, focus on visualization and other structured presentations of information as key to efficient data analysis. Frameworks such as d3js (D3JS.org, 2013) provide building blocks that support tailor made solutions for visual representations of text data, word clouds, patent information as well as data driven dynamic manipulation of documents.

The application of big data mining technology to support engineering development projects is however still in an exploratory phase. In this paper, we focus on application of data mining to analyze a specific kind of engineering documentation, i.e. deviation and error reports (hereinafter referred to as "DE reports").

3 Methodology

The data that needs to be analyzed in order to understand the root causes of the changes made during a product development project are of varying types. *Numerica*l data (i.e., time stamps and number of fault points) needs to be coupled with both *categorical* data (i.e., which component caused the issue) and *unstructured* data (i.e., free text inserted by engineers).

In this work, these data science capabilities are placed in a methodology that supports the wider process of extracting data from the set of databases that are used to document a product development project, analyzing the data in order to find root causes for failures, design solutions or actions that eliminate the root causes of the failures, and then implementing these solutions/actions in the next product (Figure 2). More specifically, the process consists of the following steps:

1. Relevant data sources are identified.
2. Data is exported from the data sources and compiled in a suitable form, e.g. as an Excel file.
3. The data quality is evaluated and if needed cleaned up.
4. Hypotheses on how data is connected and can be explored are formulated.
5. An initial set of graphs are created in order to get an overview of the data, e.g. Pareto chart, magnitude over time categorized by departments diagram, change requests per month diagram, and issues per month categorized into severity of fault points diagram are created. The Python library Matplotlib (Hunter 2007) is used to create the visualizations.
6. Text data analysis is performed in Python. Text mining tools such as entity extraction methods (e.g., Maynard et al., 2001) and information retrieval methods (e.g., Nadeau and Sekine, 2008) are employed to analyze and find patterns in product deviations in late phases of a product development project. The computational analysis is performed using the programming language Python (Van Rossum and Drake 1995). For example, the most frequent parts and words analyzed on a group and department level.
7. Pattern and root cause analyses are carried out. These patterns will bring facts on the table that are not visible looking at the raw data and thereby make it possible to correct repeated failures. Variables of interest are but not limited to design location, design organization, technical function, technologies, etc.
8. Solutions are sought that eliminate the root causes.
9. Solutions are implemented in new PD (product development) projects.

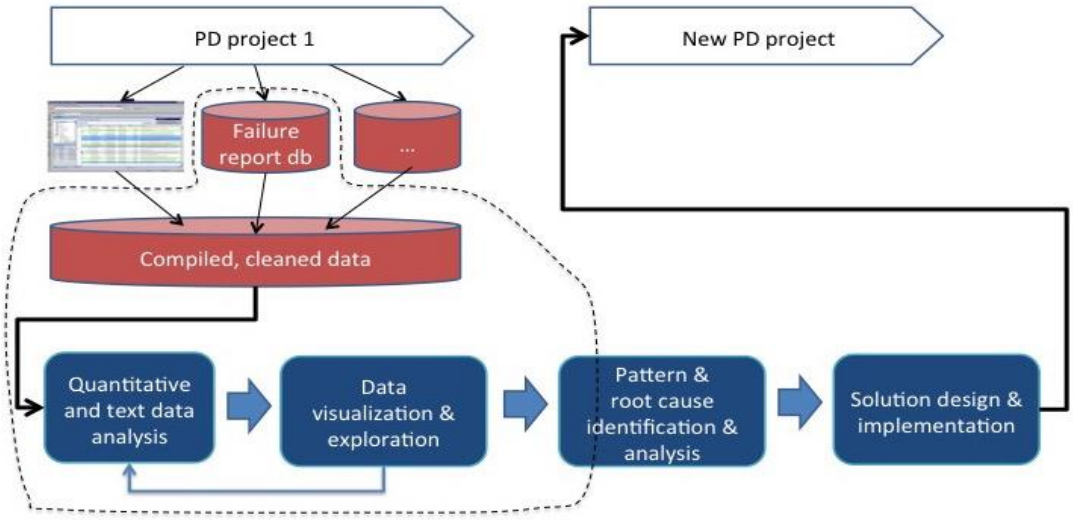

Figure 2. Flowchart of the methodology. Parts within dashed lines tested in case study.

A case-based method was applied in this research, as it is good for investigation of complex configurations of events and structures (Brady and Collier, 2010). An in-depth study was conducted to identify specific behaviors in the project. Data from a large, recently concluded, truck development project was examined. The data sources included a prototype build and test report database, a product documentation database, steering committee meeting protocols and other individual documents such as time-plans and department descriptions. The data set consist of around 4,000 reported change requests. The current paper reports on the encircled elements of the methodology.

4 Findings

Computational text analysis tools have been employed to analyze and find patterns in product deviations in late phases of a product development project. All charts presented in this research are based on real numbers but due to confidential agreements no numbers can be shown on graphs. Moreover, names of parts, teams, and failure modes have had to be coded.

4.1 Compiled and cleaned data

We identified the relevant data sources, exported the data from its sources and compiled it into a suitable form. We used multiple Excel files that included data such as project data, department descriptions and time-plans. Data was cleaned up and the quality of it was evaluated by comparing empty cells to fill out cells in the change request reports for the prototype and test report database. It showed the percentage of columns that were filled in and that were left empty. The average quality of the data used in this research was 97%. It was concluded that important columns like e.g. part name, part no, design team, and function unit all showed sufficient and consistent amount of data for analysis.

4.2 Quantitative and text data analysis

The analysis part began with brainstorming of hypotheses of how the deviation data can be connected to possible systematic causes. Deviations were classified with respect to design teams, functional unit, time, frequency, severity, change requests etc. The first graphs were created to get an overview and visualize the data. Python library Matplotlib enabled us to create graphs such as: Pareto graph for design teams, function units and words, magnitude diagram over time, change requests per month, and issues and the severity composition of issues per month. The text data analyses were performed in Python, which enabled us to count the words from the compiled data. We then analyzed common parts, text and looked for some known failure modes.

4.3 Data visualization and exploration

We can see that information can be extracted from the graphs that can help us building fact-based hypotheses. Some hypotheses that have been explored so far how data can be connected and analyzed: Errors per design team, errors over time per design team, errors related to parts and text, and magnitude of failure modes related to functional units and part names.

The first step of the data visualization and exploration was to plot the magnitude of the issues over time in a bar chart (sum of issues) and a line chart (composition of the severity of the issue). This was followed by the creation of a Pareto graph, a volume graph (over time and in relation to parts and words), and magnitude graph.

Figure 3. The bar chart (top) displays recorded project issue per months throughout the project lifetime. The line chart (below) displays the issues in a group of severity (fault points).

The first step is to get an overview of all the issues in a single graph (top bar chart in Figure 3) that represents the period of the product development project. The blue lines represent issues in a month and they are to scale. It is important to note that the issues in a change report have different values and are ranked on four levels; 100, 25, 5, and 1 (100 points is most severe).
The lower line chart in Figure 3 displays the same scenario as the bar char, but here the four fault groups are represented by colored lines. The data shows the amount of change request generated per month over the project period. Features of interest are not only the curve of the change requests but also the composition of fault points for that curve.

An interesting observation is to make a frequency graph (Pareto chart), where individual design teams were plotted in descending order according to amount of fault points each design team was responsible for (see Figure 4). Each blue bar represents a design team and the numbers, or the height of the bars is to scale. The graph helps us to summarize the total amount of fault points and display the relative importance between the design teams. The red line with the cumulative percentage of fault points tell us approximately where most of the fault points reside within what group. This help us to easily identify the largest contributes when it comes to fault points and we can see that roughly 20 percent of the design teams create 80 percent of the deviation and error reports.

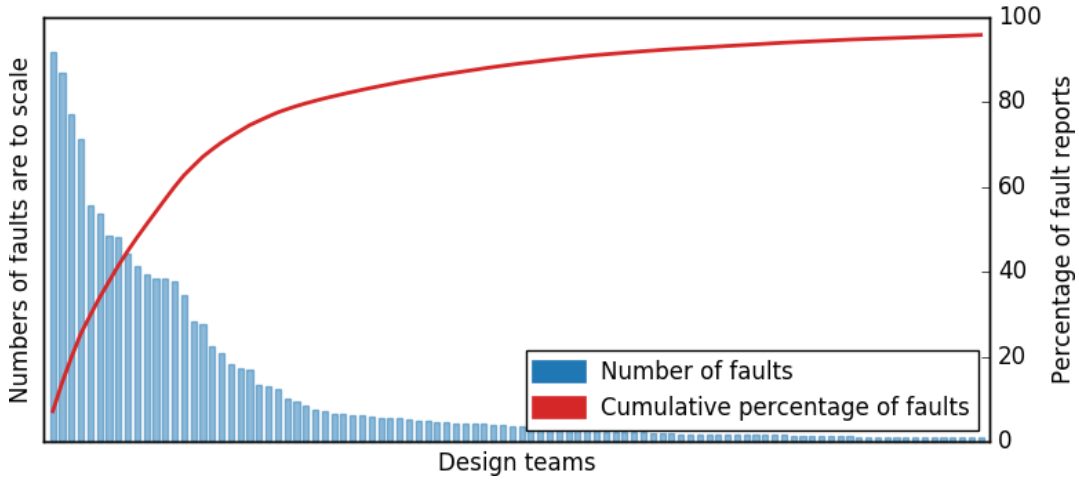

Figure 4. Blue bars represent the fault points for each design team in the project, in descending order. Their cumulative total is then represented by the red line.

Next we focused on the top 5 design teams with the most fault points and plotted them into a volume diagram, see Figure 5. Each graph represents the fault points a design team had over the period of a product development project. A design team indicates a general area that the team works on, e.g. electronics or chassis. This is a good way to analyze the frequency and the magnitude of causes by looking at specific variables over time. The graphs tell us at what point in time a major issue started and how long it took to control it. It is good method to visualize and isolate different design teams so we can further analyze group related issues. We can see when in time major issues happened and easily compare then to other period. The issue that each team has can then further be analyzed into e.g. functions units, failure modes, material groups, etc.

We can see in Figure 5 if design teams are creating DE reports through the whole lifetime of the project, e.g. Team B seems to do some, but Team A does it mainly from 2010 to 2013. Team C seems to report the most issues after Team B and E have gone through long periods of issues.

Figure 5. Volume diagram showing the top 5 design teams with the most fault points, plotted over time.

With the help of computational text analyst, we could broaden the analyses from only numerical but also to text analysis. The prototype build and test report database contains also report title, description of fault/problem/cause/background and addition comments. Figure 6 shows what we could enable with text analysis. It displays the three design teams with the highest number of issues over time. The issues are plotted for each team in different colors, and displaying the most frequent parts and text in a descending list to the right of the figure. This enables us to see what parts and text were the most problematic for each team. Through this, we discovered families of similar parts that we recurring most often in the DE reports, and by knowing that we can narrow the analysis and the specific parts and see what they have in common.

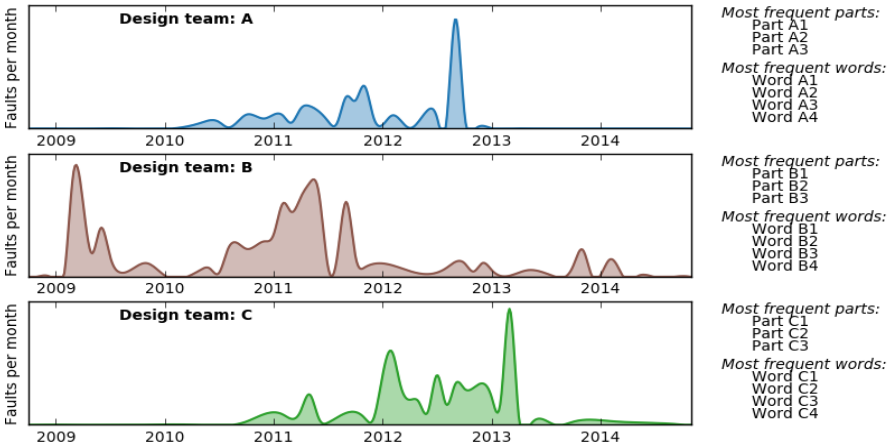

Figure 6. The three design teams displaying number of issues over time. To the right are the most frequent parts and word listed in a descending order.

Words can also be analyzed according to common failure modes, e.g. in a magnitude diagram over time. The magnitude of issues equals to the sum of fault points related to the functional unit or part. We can then systematically analyze subsystems and related components into each failure modes, e.g. fracture, misalignment, wear, etc. That knowledge can aid in finding the failure of each component.

4.4 Pattern identification and analysis

The visualization and exploration of the data have helped identifying patterns and how the data can be analyzed in more ways. We can now e.g. see what design teams and functional groups are responsible for majority of the issues, and how that graphs out over time. Issues not visible by looking at a list of raw data and therefore making it possible to see patterns. Here are some hypotheses that we see are worth exploring: When building a product/prototype for the first time changes often happen. If resource allocation early in projects is related to late changes, then more resources allocation early on will reduce late changes. If DE report closure is related to design gates, then the same problems should not emerge again. If a DE report is closed with a no-action, then the same problem should not emerge again.

Identified patterns support the compilation of background facts and questions for interviews with engineers that have been involved in the project. We can then connect the engineer's opinions about the development project with the facts from our analysis to see what is worth further investigating.

5 Discussion

Let us now consider the findings in relation to the research questions, the validity and transferability of the approach and some ideas for future work.

RQ 1 was stated as "*Is it possible to use product deviations reports from databases and apply data mining analysis to find patterns that will aid in finding root causes for deviations and repeatable failures?".* We find that the big data analysis tools provide a multitude of ways of filtering, visualizing and exploring the data. We can find out what parts, departments etc tend to log most DE reports, as well as which tend to be later than others. We can find patterns, such as repeated DE reports on the same part. We can find out what failure modes (e.g., "wear") are most frequent. All of these observations will provide a good starting point for unscrambling the root causes of the problems.

RQ 2 was stated as "*Is it possible from this analysis to build fact-based hypotheses for how to improve product quality in the late phases of a project?"* Our conclusion on this point is that while the data mining analysis of the DE report provides a good map of symptoms and patterns, they rarely explain the "why" of a failure (e.g., why a part has corroded). Hence, the next step is qualitative, i.e. to conduct interviews with engineers and managers to better understand the data and to get feedback on the quantitative analysis. For example, if we look at Team E in Figure 5, we can see that problem started to ramp up in 2010 and that it looks like they were controlled and eliminated at the end of 2011. Then there is a possibility that the problem emerged again and escalated to more changes, and at the end of 2012. Interviews would focus on what was going on at each peak, to obtain a deeper understanding, of why the issues kept recurring.

The tests of the approach were conducted on a dataset consisting of 4,000 DE reports. The reports were limited to hardware components (mechanical and electronic). This is a real and significant size of the dataset. As the patterns we see in the data are similar to other reports (e.g., Giffin et al., 2009), we argue that the test confirms the feasibility and potential of the approach, and that it could be applied in analysis of complex systems development projects in other industries, such as defense or telecommunications. Nevertheless, more tests are desirable with multiple data sources (e.g., PDM systems, software change request databases, manufacturing error databases). It would also be interesting to compare DE reports across multiple product development projects to see if patterns seem to repeat themselves.

The next step in this research project is to apply more elaborate machine learning methods like classification, clustering, and regression for analyzing the data. The analysis will be performed using various machine learning modules in Python such as scikit-learn, tensorflow, and GraphLab-Create. An important step towards a fundamental understanding of root causes of failures is to build a dependency graph describing the dependencies between the components in the truck. For this analysis graph database systems such as Nej4j will be utilized together with graph analytics modules such as NetworkX.

Further, there is an opportunity for the research to identify how complex system development projects can improve the contents of their DE reports. In order to support future analysis on DE reports thoughts should be put into how the data is logged, in order for a more comprehensive analysis. One way is to classify failure modes in a systematic way according to taxonomy of failure modes (see, e.g. Otto and Wood, 2001, p. 567). That can enable better groupings and improve the quality of the logged data.

6 Conclusion and future work

The number of DE reports in a truck development project may exceed 10,000. The DE reports contain both quantitative and qualitative (text) data and are thus challenging to analyze in a systematic manner. As a consequence, product deficiencies detected in tests tend to be resolved in an ad-hoc manner. The root causes are not identified and errors tend to be repeated.

Data mining tools provide means for analyzing large volumes of complex data. Such analyses include combining data sources and data types, data clean up, charting, analyses of large text datasets, visualization and multi-facet and multi-level data exploration. A number of potential benefits of such analyses can be identified including:

- Finding patterns in DE report data, such as repeated DE reports on the same part, occurrence of similar failure modes (e.g. wear, corrosion, misfits, …)

- Comparing deviation data across projects or departments / sites
- Predicting occurrence of late DE reports – through early warning signs detected through analysis of earlier projects
- Connecting deviations to root causes and to systematic identify mitigating actions way.

This work has so far developed a basic data mining approach for analyses of DE reports data and on testing the approach on a limited dataset. The approach has been tested on a dataset comprising 4,000 DE reports. The approach seems to work well. Planned future work includes compiling data from multiple sources (e.g. by adding a software DE report database) and to compare data across multiple projects.

References

Berkhin, P. (2006). A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data*, 25-71. Springer Berlin Heidelberg.

Brady, H. E., & Collier D. (2010). *Rethinking Social Inquiry: Diverse Tools, Shared Standards*. Rowman & Littlefield Publishers.

Clark, K. B., & Fujimoto, T. (1991). *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Cambridge: Harvard Business School Press.

D3JS.org. (2013). *Data-Driven Documents.* Retrieved from http://d3js.org/

Eger, T., Eckert, C. M., & Clarkson, P. J. (2007). Engineering Change Analysis during Product Development*, Proceedings of ICED'07*, 629-630, Paris, France.

Fernandes, J., Henriques, E., Silva, A., & Moss, M. A. (2015). Requirements Change in Complex Technical Systems: An Empirical Study of Root Causes. *Research in Engineering Design*, *26*(1), 37-55.

Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., & Clarkson, P. J. (2009). Change propagation analysis in complex technical systems. *Journal of Mechanical Design*, *131*(8), doi:10.1115/1.3149847.

Hicks, B. (2013). The Language of Collaborative Engineering Projects. *Proceedings of ICED'13*, Seoul, South Korea.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science and engineering*, 9(3), 90-95.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised Machine learning: A Review of Classification Techniques. *Informatica*, 31: 249-268

Maynard, D., Tablan, V., Ursu, C., Cunningham, H., & Wilks, Y. (2001, September). Named Entity Recognition from Diverse Text Types. In *Recent Advances in Natural Language Processing 2001 Conference*, 257-274.

Otto, K. N., & Wood, K. L. (2001). *Product design: Techniques in Reverse Engineering and New Product Development*. Upper Saddle River, NJ: Prentice Hall.

Söderberg, R., Rosenqvist, M., Falck, A.-C., Wahlborg, P.-J., Silow, N. (2013). *Proaktiv monteringsergonomisk och geometrisk kvalitetssäkring för hållbar produktion*, Projektbeskrivning, Wingquist Laboratory, Chalmers tekniska högskola, Göteborg. In Swedish.

The Standish Group. (2014). *Standish Group 2014 CHAOS Report.* https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.

Thomke, S., & Reinertsen, D. (2012). Six Myths of Product Development. *Harvard Business Review*, 90(5), 84-94.

Van Rossum, G., & Drake Jr, F. L. (1995). *Python Reference Manual*. Amsterdam: Centrum voor Wiskunde en Informatica.