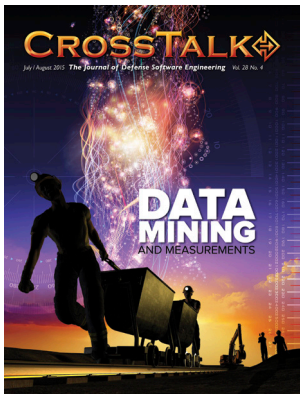
CROSSTALK

July / August 2015 The Journal of Defense Software Engineering Vol. 28 No. 4

DATA MINING AND MEASUREMENTS

TABLE OF CONTENTS

Departments



Cover Design by Kent Bingham

Data Mining and Measurements

CrossTalk

NAVAIR Jeff Schwalb
DHS Joe Jarzombek
309 SMXG Karl Rogers

Publisher Justin T. Hill
Article Coordinator Heather Giacalone
Managing Director David Erickson
Technical Program Lead Thayne M. Hill
Managing Editor Brandon Ellis
Associate Editor Colin Kelly
Art Director Kevin Kiernan

Phone 801-777-9828
E-mail Crosstalk.Articles@hill.af.mil
CrossTalk Online www.crosstalkonline.org

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsor: Ogden-ALC 309 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of **CrossTalk** providing both editorial oversight and technical review of the journal. **CrossTalk's** mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit <www.crosstalkonline.org/subscribe> to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the **CrossTalk** editorial board prior to publication. Please follow the Author Guidelines, available at <www.crosstalkonline.org/submission-guidelines>. **CrossTalk** does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with **CrossTalk.**

Trademarks and Endorsements: **CrossTalk** is an authorized publication for members of the DoD. Contents of **CrossTalk** are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services:
For questions or concerns about crosstalkonline.org web content or functionality contact the **CrossTalk** webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CrossTalk is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing <luminpublishing.com>.
ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CrossTalk would like to thank **309 SMXG** for sponsoring this issue.

Data Mining and Measurements

It was not too long ago that I asked one of our data analysts to see what "gold nuggets" he could find in a metrics data set that had more than 20 years' worth of accumulated software development data that had never been fully analyzed.

I told him that I was uncertain what he would find, but that I bet there was some very valuable information hiding in that data. I even went so far as to tell him that the data was like a big drum of dirt that contained one or more gold nuggets. After some false starts and some frustration, it became apparent that the first problem was to identify what exactly we are looking for in the data. Most people would think the answer to that question would be obvious but quite the contrary was true.

There were in fact some obvious things to look for, but it was not until we started brain storming and coming up with some more obscure ideas and ways to look at the data that we started to learn what we needed to learn from the data set. The better we can define what we are looking for in our data the easier the task of data mining becomes. In addition, the more we understand about what we are looking for the better we can define the metrics that need to be collected for the future.

We have spent a lot of time collecting and analyzing data over the years and it has become clear that we continue to learn about data analysis, data collection and metrics identification but it seems like in many cases it is a slow and difficult process.

The information we mine from our metrics drives improvements in our software development processes but it seems like we can never get enough data to answer all of the questions. This issue of CrossTalk has compiled articles concerning Data Mining and Measurements. I believe these articles can help each of our organizations to make better decisions about data collection, metrics and data analysis.

The articles included hopefully can help each of our organizations learn how to optimize data mining and metrics identification. I hope you enjoy this issue of CrossTalk and I hope it helps your organization to move make significant strides in Data Mining and Measurements.

Karl G. Rogers
Director
309th Software Maintenance Group

Big Data and Deep Learning for Understanding DoD Data

Ying Zhao, Ph.D., Naval Postgraduate School
Douglas J. MacKinnon, Ph.D., Naval Postgraduate School
Shelley P. Gallup, Ph.D., Naval Postgraduate School

Abstract. Today, Big Data infrastructure and analytics intervene with traditional data sciences. We are compelled to ask - What is new? In this article, the authors provide a pragmatic context for how Big Data infrastructure and analytics are related to traditional data sciences including statistical analysis, numerical analysis, machine learning, data mining, pattern recognition and data fusion. The authors also discuss use cases in various categories that demonstrate empirical practicality for understanding and applying Big DoD Data.

1. Why Big Data Now?

"Project pursuit," a method that seeks useful lower dimensional projections[1] from higher dimensional data, was researched extensively in the 80s with a research funding level of $10,000. When "machine learning" emerged in the earlier 90s, the funding for "projection pursuit learning network" [1], for instance, grew to $100,000. It grew to $1,000,000 for "data mining" in the late 90s.[2] The core method remained the same, yet the data size was bigger. Today, Big Data science intervenes with traditional data sciences. We are compelled to ask - What is new? Let us examine the current breakthroughs:

- Big rise in data: data creation is remarkable for its volume, velocity, and variety. "Volume" considers the rise of new data creation platforms of multimedia, social media, mobile devices, the Internet of Things (IOT) and new sensors. "Velocity" considers these new platforms capturing millions of events per second and in real-time. "Variety" considers captured data not only just numbers but also unstructured text, images, audios, videos, geospatial data, and 3D data. Big Data are omnipresent and ubiquitous. In 2012, the Obama administration announced 84 Big Data initiatives across six departments [2].
- Big rise in needs: It is critical for business to transform data into smart data, or actionable knowledge. For example, researchers need to use Big Data to discover new drugs. Marketers need to use social networks, mobile, geo-location, and sensor data to reach more customers. The United States National Security Agency (NSA) needs to process the exabytes (1018) of data collected over the internet in the Utah Data Center [3].
- Big rise in technologies: Traditional data sciences including statistics, numerical analysis, machine learning, data mining, business intelligence, and artificial intelligence are evolved into Big Data analytics. The US Federal Government owns six of the ten most powerful supercomputers in the world [4].

These technologies can be overwhelmingly complex, requiring diversified and extensive expertise.

2. Practicality

2.1 Tools

Big Data is near impossible to process with conventional technologies, requiring instead massively parallel software on thousands of servers. The current technologies are dominated by systems that provide 1) safe storage, 2) parallel/operational processing, and 3) deep analytics.

As part of open-sourced Apache Hadoop ecosystem, Hadoop Distributed File System (HDFS) provides distributed and fault-tolerant data storage. Beehive and Pig are "SQL-like" tools for conventional database queries on a HDFS. NoSQL systems[3] include document and graph databases in a "cloud" such as Amazon and Cloudera. Operational systems for messaging, banking, advertising and mobile devices can utilize Apache Storm to handle day-to-day transactions in real-time, or with no- or low-latency of response.

Map/Reduce is an analytic programming paradigm for Big Data. It consists of two tasks: 1) the "Map" task, where an input dataset is converted into key/value pairs; and 2) the "Reduce" task, where outputs of the "Map" task are combined to a reduced key-value pairs. Apache Spark[5] could replace Map/Reduce for its speed and in-memory computation.

2.2 Challenges

As the data size gets bigger, the statistical significance for an analysis is often guaranteed due purely to the size. This positive impact of the data size can be a great advantage. However, other challenges rise. For example, traditional data sciences used in small- or moderate-sized, analysis typically require tight coupling of the computations of the "Map" and "Reduce" steps. Such an algorithm often executes in a single machine or job and reads all the data at once. How can these algorithms be modified so they can be executed in parallel in thousands of clusters? If the data is processed in parallel and parsed into subsets, how to leverage the art and science of fusing the results as phrased in the "Reduce" step?

An oddity to be further explored in Section 3 is that "data fusion" has been successfully performed in many DoD applications whereas commercially-tempted innovations (i.e., Thinking Machines [6] and Cray Computers) were not successful [7].

2.3 Commercial Trends

Predictive analytics is to turn Big Data into smart data, for example, accurately forecasting high-value targets such as high-value customers, events, and social media sentiment. The topic has been thoroughly studied in supervised learning. Some algorithms are implemented using the Map/Reduce paradigm [8]. 95% of Big Data is unstructured. Text analysis methods (e.g., categorization, summarization and topic discovery) are being adapted to Big Text.

Social network analysis, product cross-selling, recommendation engines, event diffusion, and graph search require graph analyses leveraging massively parallel processors. For instance, viral path predictions are used for predicting how useful events, e.g. new ideas, videos or diseases, become proliferated to a large population or "go viral." Graph algorithms can process petabytes of data and are considered as the core drivers of Big Data analytics. Spark, Titan and Neo4j are used for Big Graph.

One important trend is Deep Learning including unsupervised machine learning techniques (e.g., neural networks) for recognizing objects of interest from Big Data [9], for instance, sparse coding [10] and self-taught learning [11]. The self-taught learning [12] approximates the input for unlabeled objects as a succinct, higher-level feature representation of sparse linear combination of the bases. It uses the Expectation and Maximization (EM) method to iteratively learn coefficients and bases [13]. Deep Learning links machine vision and text analysis smartly. For example, text analysis Latent Dirichlet Analysis (LDA) is a sparse coding where a bag of words used as the sparsely coded features for text[10]. Our methods Lexical Link Analysis (LLA), System-Self-Awareness (SSA), and Collaborative Learning Agents (CLA) can be viewed as unsupervised learning or Deep Learning for pattern recognition, anomaly detection, and data fusion.

3. DoD Big Data Applications

Data sources for DoD applications including disparate, multi-sourced real-time sensors, and archival sources are of extremely high rates and large volumes. In DoD collaboration environments, the needs for information sharing and agility as well as strict security across all domains makes the matter more complex. While commercial applications such as massive marketing may require identifying information with popular and repeatable patterns, emerging and anomalous information are more useful for DoD applications (e.g., intelligence analysis and resource management). Deep learning regarding pattern recognition, anomaly detection, and data fusion can be even more useful. The US Navy has now begun to take initiatives to move Big Bata into the battlefield [14].

In the past, at the Distributed Information Systems and Experimentation (DISE) research group at the Naval Postgraduate School (NPS), we have applied Big Data sciences to understand DoD data. In particular, Lexical Link Analysis (LLA) has been used to analyze unstructured and structured data for pattern recognition, anomaly detection, and data fusion. It uses the theory of System Self-Awareness (SSA) to identify high-value information in the data that can be used to guide future decision processes in a data-driven or unsupervised learning fashion. It is implemented via a smart infrastructure named "system and method for knowledge pattern search from networked agents (US patent 8,903,756)" also known as Collaborative Learning Agents (CLA), licensed from Quantum Intelligence, Inc. [15].

In the following sections, we first describe our approaches of LLA, SSA and CLA briefly and then categorize some DoD applications. We discuss four use cases in these categories. Some use cases were described in more detail in related publications [15-16, 25-30].

3.1 LLA, SSA and CLA

In LLA, a complex system is expressed in specific vocabularies or lexicons to characterize its features, attributes or its surrounding environment. LLA uses bi-gram word pairs as the features to form word networks. Figure 1 depicts LLA with word pairs as groups or themes. Figure 2 shows a detail of a theme in Figure 1. A node represents a word. A link or edge represents a word pair.

LLA is related to bags-of-words (BAG) methods such as LDA [17] and text-as-network (TAN) methods such as the Stanford Lexical Parser (SLP) [18]. LLA selects and groups features into three basic types:

- Popular (P): They are the main themes in the data. Figure 2 is an example of a popular theme centered around word nodes "analysis, model, approach." These themes could be less interesting because they are already in the public consensus and awareness. They represent the patterns in the data.
- Emerging (E): Themes may grow to be popular over time. Figure 3 is an example of an emerging theme centered around word nodes "national, defense, acquisition."
- Anomalous (A): These themes may be off-topics themes that are interesting for further investigation. Figure 4 is an example of anomalous theme centered around word nodes "stock, market(s)."

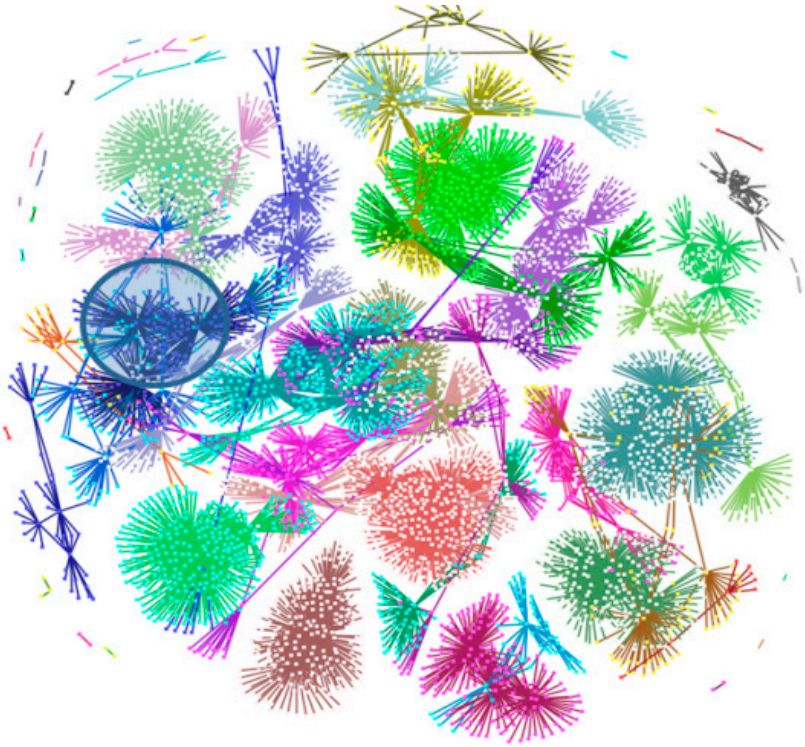


Figure 1. Themes Discovered in Colored Groups

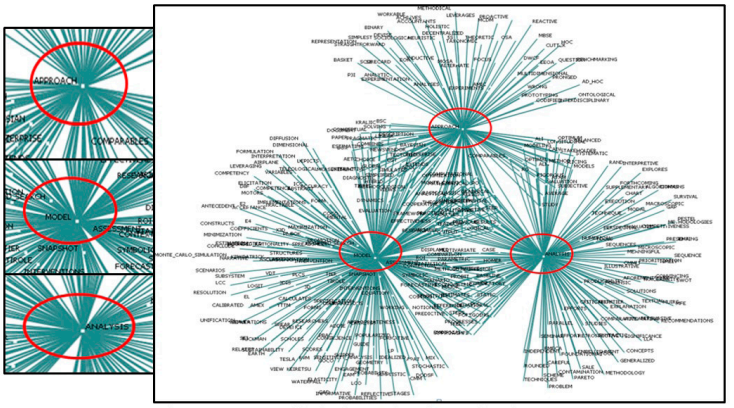


Figure 2. A Detailed View of a Theme in Figure 1

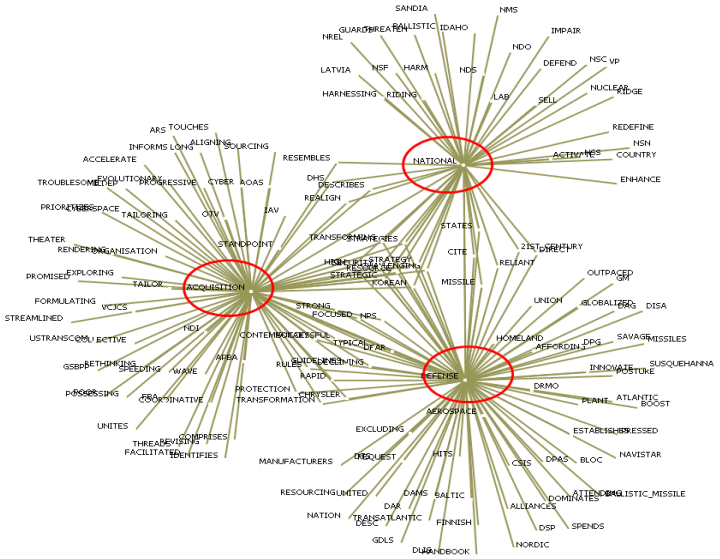


Figure 3. An Example of Emerging Theme

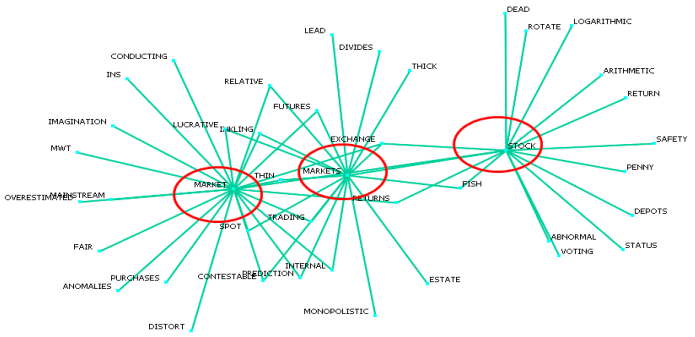


Figure 4. An Example of Anomalous Theme

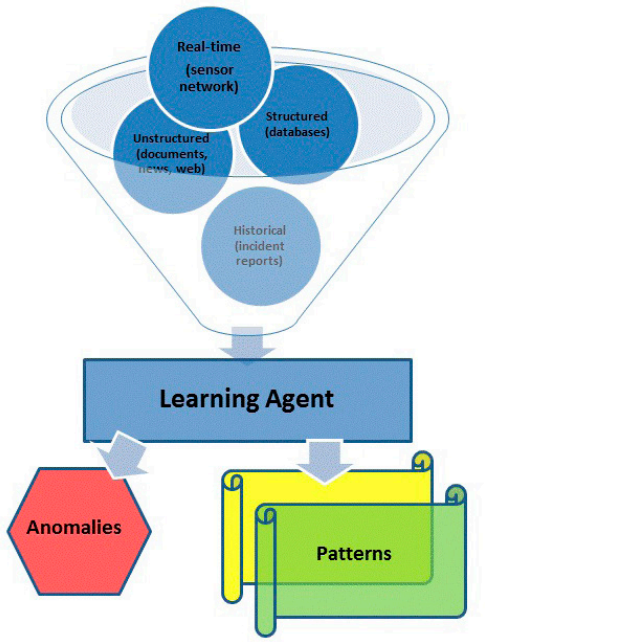


Figure 5(a). A Single Collaborative Learning Agent: Patterns are graded from medium to relevant correlations.

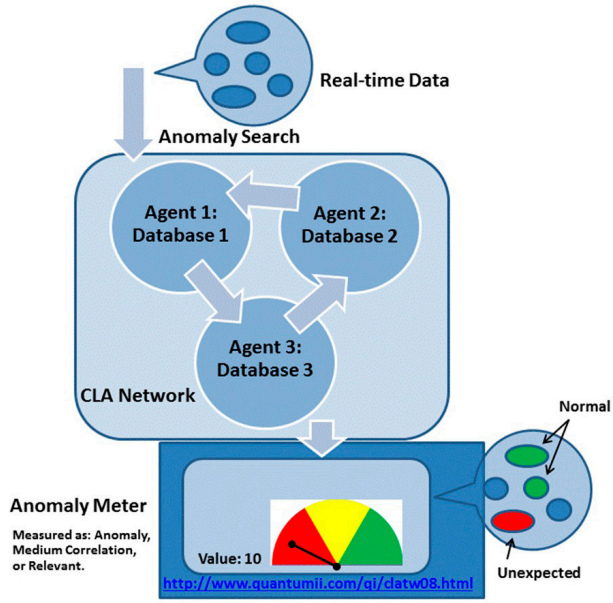


Figure 5(b). Agent Collaboration

The separation of the three types is based on SSA and implemented using CLA. Figure 5(a) shows a CLA as a computer program used to separate and extract patterns and anomalies from multiple data sources. A single agent installed in a single computer node is capable of ingesting and analyzing data sources locally. Multiple agents can work collaboratively in a network and fuse multiple data sources as shown in Figure 5(b).

We define System Self-Awareness (SSA) as the ability for an agent to estimate its global importance by optimizing its total value considering its relations to other agents (authorities and patterns) and its own expertise (anomalies) learned from the local data. SSA is implemented as a fusion mechanism to optimize the overall value R(t,j) using a recursion as shown in Figure 6.

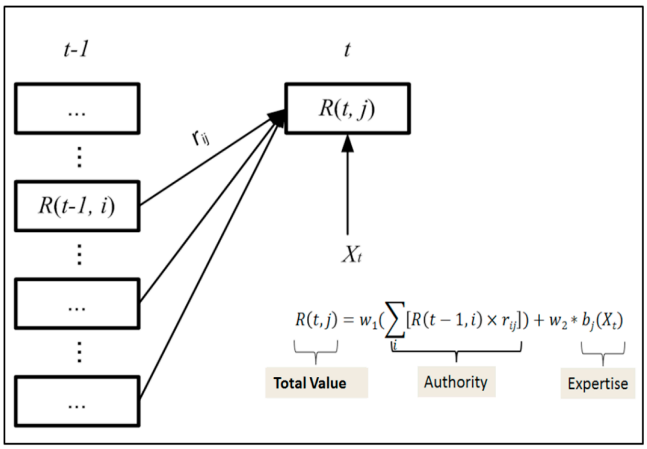


Figure 6. Recursion to Compute the Overall Value of a System R(t, j)

3.2 How Can the Methodology Be Used for Future Decision Processes

We show in the following four real examples (i.e., use cases in Section 3.2.1 to 3.2.3) that the Big Data and Deep Learning methodology in Section 3.1 can be used for future decision processes by

- Processing more data in parallel
- Automating data fusion
- Learning associations and correlations from diversified data sources which may not be standard data
- Performing pattern recognition and anomaly detection

3.2.1 Data Fusion, Optimization of Distributed Resources

DoD Big Data (e.g. sensors) are collected in local compartments and specific to domains. DoD resources are distributed with the strict security requirements [19], fault-tolerance, and agility. These data need to be combined for applications. Data fusion is the process of combing information from a number of different sources to provide a robust and complete description of an environment or process of interest. Distributed and parallel processing is required but is not sufficient for data fusion where analytic algorithms that can combine the results from distributed systems are critically required. Data fusion finds application in many military systems especially when sensor data were collected and must be combined, fused, and distilled to obtain information of appropriate quality and integrity on which future decisions can be made. For many military data fusion scenarios [20], data fusion is often divided into a hierarchy of four processes. Level 1 and 2 fusion is generally concerned with processing raw data using numerical fusion methods such as probability theory or Kalman filtering. Level 3 and 4 fusion is thus concerned with the extraction of high-level knowledge from low level fusions, the incorporation of human judgment and the formulation of decisions and actions.

To understand Big Data architecture and analytics in DoD applications, we need first understand the existing decentralized and distributed data fusion architectures [20].

- A decentralized data fusion system consists of a network of sensor nodes, each with its own processing facility, which together do not require any central fusion facility. In such a system, fusion occurs locally at each node ensures that the system is scalable as there are no limits imposed by centralized computational bottlenecks. Such a system is also made survivable or fault-tolerant. The decentralized data fusion algorithms are implicitly limited in requiring full communication, e.g. a fully connected sensing network or as a broadcast system.
- In a distributed data fusion system as shown in Figure 7 requires a central processor; however, each sensor also has its own local processor which can extract useful information from the raw sensor data prior to communication. The degree to which local processing occurs at a sensor site varies substantially from simple validation and data compression up to the full construction of tracks or interpretation of information locally.

In a use case entitled "Big Data Architecture and Analytics (BDAA) for Common Tactical Air Picture (CTAP)", the NPS team showed that the data generated by intelligence, surveillance, and reconnaissance (ISR) sensors has become overwhelming and the Navy now needs to apply new architectures and analytics to improve its CTAP. More specifically, accurate, relevant, and timely

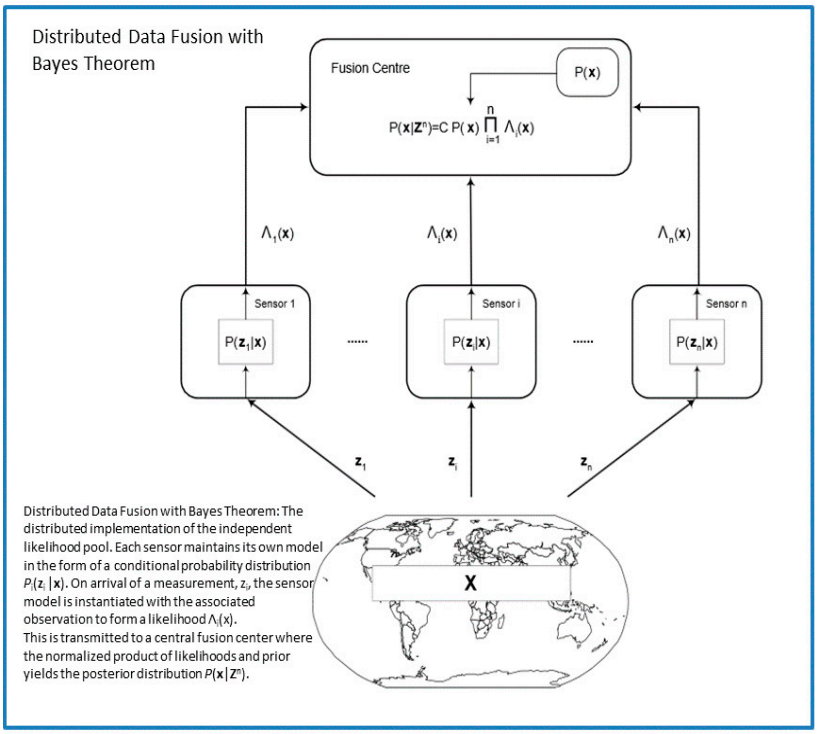


Figure 7: From [20]

Combat Identification (CID) enables the warfighter to locate and identify critical targets.

The NPS team applied LLA, SSA, and CLA jointly with Hadoop, Map/Reduce, and Deep Learning to 1) improve track correlation, continuity, fidelity and latency reductions, 2) discover and learn the patterns in historical data and correlate them with real-time data to detect anomaly; 3) improve real-time targeting recommendations and guide future decision making; 4) optimize the warfare resource management.

In this context, the NPS team cannot use either distributed or decentralized data fusion, instead a recursive data fusion methodology leveraging LLA, SSA, and CLA can be employed as follows:

- An agent j represents a sensor, operates on its own like a decentralized data fusion, however it does not communicate with all other sensors but only with the ones that are its peers. A peer list can be specified by the agent.
- An agent j includes a learning engine CLA that collects, analyzes from its domain specific data knowledge base $b(t_j)$, for examples, $b(t_j)$ may represent the statistics for bi-gram feature pairs (word pairs) computed from LLA.
- An agent j also includes a fusion engine SSA with two algorithms SSA1 and SSA2 that can be customized externally. SSA1 integrates the local knowledge base $b(t_j)$ to the total knowledge base $B(t_j)$ that can be passed along to its peers and used globally in the recursion in Figure 6. SSA2 assesses the total value of the agent j by separating the total knowledge base into the categories of patterns, emerging and anomalous themes based on the total knowledge base $B(t_j)$ and generates a total value $V(t_j)$ as follows:

  Step 1: $B(t_j) = SSA1(B(t\text{-}1, p(j)), b(t_j))$;

  Step 2: $V(t_j) = SSA2(B(t_j))$

  Where p(j) represents the peer list of agent j.

• The total value $V(t,j)$ is used in the global sorting and ranking of relevant information.

In this recursive data fusion, the knowledge bases and total values are completely data-driven and automatically discovered from the data. Each agent has the exact same code of LLA, SSA, and CLA, yet has its own data apart from other agents. This agent work has the advantages of both decentralized and distributed data fusion. It performs learning and fusion simultaneously and in parallel. Meanwhile, it categorizes the patterns and anomalous information. In many use cases investigated, the NPS team found the discovered patterns are often correlated with authoritative information, while anomalies are correlated with new and interesting information requiring further investigation. For example, sorted and ranked information according to authority and anomalousness can be used to improve and automate future decision processes of CID with higher precision and lower latency that optimize the use of long-range weapons, aid in fratricide reduction, enhance battlefield situational awareness, and reduce exposure of U.S. Forces to enemy fire.

3.2.2 Situation Awareness (SA), Decision Making and Command and Control

Situational Awareness (SA) in military parlance is the ability to maintain a constant, clear mental picture of relevant information and the tactical situations (e.g. friends and threats). The traditional SA exists in three levels: the perception of elements in the environment within time and space, the comprehension of their meaning, and the projection of their status in the near future.

SA models focus heavily on human factors for perceiving, comprehending and projecting including mental and team models, sensemaking [21], and communication models in computational linguistics and machine learning [22].

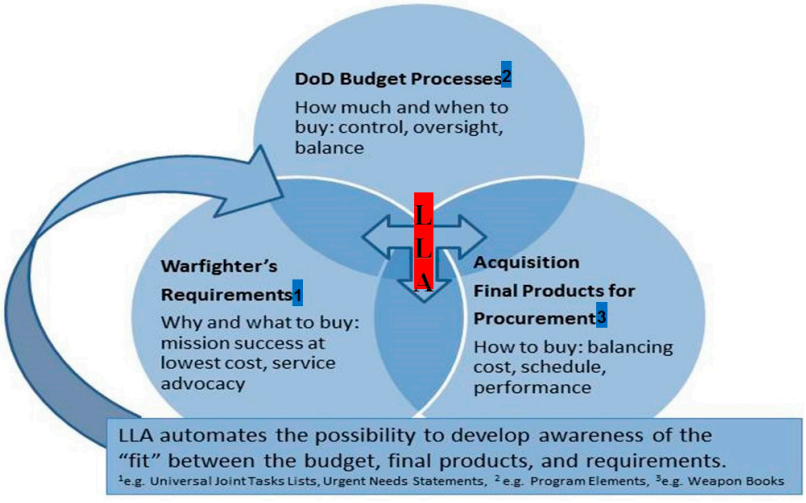


Figure 8. DoD Acquisition Decision Making

Related to SA is a Decision Support System (DSS) which is a computer system that supports decision-making and command and control (C2) activities. A DSS architecture typically includes a knowledge database, a model and a user interface. DSS models often are based on machine learning and artificial intelligence, e.g. decision trees [23] and intelligent agents [24].

There are many differences between SA and DSS. DSS, for instance, may rely on traditional analytics and apply to less dynamic data. SA emphasizes the real-time information gathering, communication methods and collective knowledge that might result in better operational capabilities. Therefore, it may require not merely DSS technologies such as machine learning systems and decision making algorithms but also smart infrastructures to achieve real-time (e.g., in a crisis response situation) and collective intelligence (e.g., in a social web).

In a use case, the NPS team has been studying the DoD acquisition decision making [16, 25-30] since 2009. The US DoD acquisition process is extremely complex. There are three key processes that must work in concert to deliver the capabilities: the warfighters' requirements/needs; the DoD budget planning and the final products for procurement as in Figure 8. Each process produces Big Data. There has been a critical need for automation, validation, and discovery to help acquisition professionals, decision makers and researchers understand the data and optimize the DoD resources.

Since 2009, the NPS team has been working on the research questions, for example, can the Big Data be used to produce the awareness of the fit between DoD programs and warfighters' needs? Can gaps be revealed? The NPS team performed studies in the following areas:

- Compare Urgent Need Statements with Trident Warrior technologies
- Compare congressional budget documents with the warfighers' needs
- Compare categories of data in the Acquisition Visibility Portal

The NPS team took a detailed look at the Research, Development, Test and Evaluation (RDT&E) budget modification practice from one year to the next over the course of ten years and about 450 DoD Program Elements. The NPS team found a pattern that the programs with fewer links (measured by LLA) to warfighters' requirements, received more budget reduction in total but less on average, indicating the budget reduction may have focused only on large and expensive programs rather than perhaps cutting all the programs that do not match warfighters' requirements. Furthermore, the programs with more links to each other received more budget reduction in total, as well as on average, indicating a pattern of good practice of allocating DoD acquisition resources to avoid overlapping efforts and to fund new and unique projects. These findings were useful as validation and guidance for future decision processes for automatically identifying programs to match warfighter's requirements, limit overall spending, minimize efficiencies, eliminate unnecessary cost and maximize the return of investment.

3.2.3 Prediction, Deep Learning, Pattern Recognition and Anomaly Detection

Predictive models are paramount to machine learning their predictive power comes from empirically reviewed data. Machine learning algorithms are divided into supervised learning and unsupervised learning. Supervised learning is accurate but more expensive due to intervention costs. Unsupervised learning focuses on data-driven discovery and often is linear scale-up on the number of machines and the size of the data. Thus unsupervised learning is the proposed core analytic strategy for both commercial and DoD Big Data. The NPS team show LLA, SSA and CLA are important unsupervised learning methods for pattern recognition, anomaly detection and data fusion.

In many cases of the DoD applications, Big Data is buried in the complex business processes and the data fusion has to performed on a vast amount of data sources from the complex business processes. In a use case entitled "Comprehensive Approach to Identifying and Sourcing NATO's Future Capability Requirements," the NPS team applied LLA, SSA and CLA to identify and predict NATO capabilities and force requirements to improve the US EUROPE COMMAND (EUCOM)'s visibility and recommend new collaborations toward "Smart Defense" projects. The NPS team first interviewed USEUCOM's desk officers and planning specialists and gained an understanding of their business processes and the Big Data involved in these processes shown in Figure 9.

The NPS team then conducted the following studies using LLA, SSA and CLA:

1) Compare Chicago Summit Open Sources and Smart Defence (SD) Database, the SD database contains structured and unstructured data about all the SD projects

2) Compare Minimum Capability Requirements (MCR) and SD Database

3) Compare 28 Bluebooks of NATO countries

Figure 10 shows an example of visualization from (1). Themes were discovered automatically and can be drilled down to the original data or the features (word pairs) that describe the consensus and gaps between the Chicago Summit Open Sources and the SD database. Themes were further categorized into popular, emerging, and anomalous concepts. The NPS team showed that popular concepts are highly correlated with the discovered consensus among compared data sources. In contrast, the emerging and anomalous themes are highly correlated with the gaps in the business processes which may need further investigation and could provide guide for future decision processes, for example, discovery of interesting resource relocation opportunities that can be used by the USEUCOM and Smart Defense programs to advance US interests.

3.2.4 Knowledge Management, Collaboration and Network Analysis

Graph and network analysis are important for knowledge management and collaboration. The current research focuses on direct social links among social entities of people or organizations regardless of the contents [30]. The study of centrality has been a focal point for the social network structure studies to discover mavens, leaders, bridges, isolated nodes and peripheries.

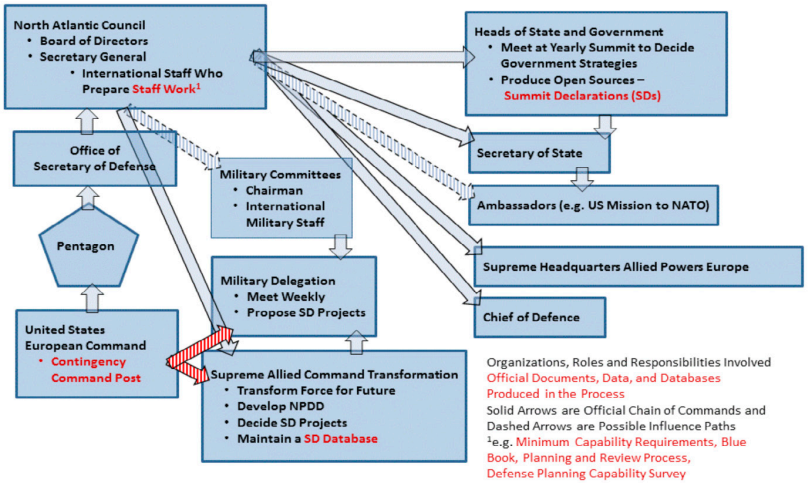


Figure 9. EUCOM Planning Processes

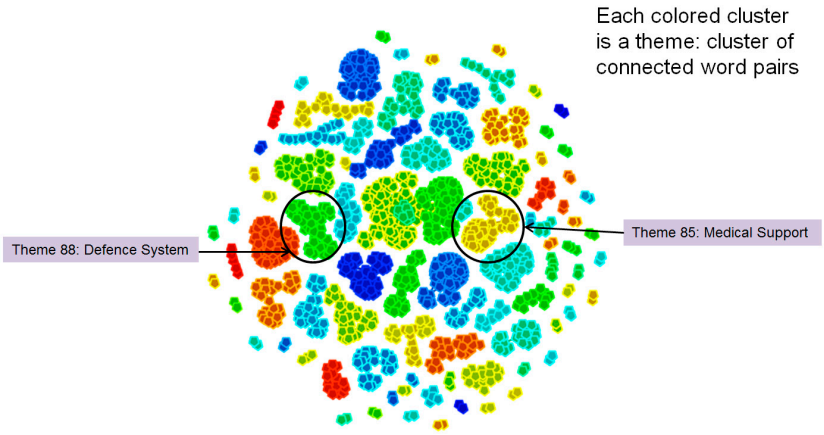


Figure 10. Clustered Depiction

So called "metadata analysis," applied to social entities whose profiles collected from structured data (e.g., Palantir [31]) has also drawn attention. It can infer two people are linked because they share the same metadata attributes [32], for example, two people may share a same metadata attribute such as belonging to a same social club.

LLA-generated semantic networks can infer that two people are linked because they share the same content, for example, two people are both interested in "information assurance." LLA can be used to discover such keywords for interesting connections.

After the Haiti earthquake in 2010, US military and civil organizations provided rapid and extensive relief operations. In a use case entitled "Open Source, APAN Network and Haiti Operation Data Analysis [29][30]," the NPS team applied LLA to show an overall picture of how military and civil organizations actually collaborated. The NPS team first examined ~2600 open source data from the social media platforms Twitter, Facebook and news-feed web sites [29]. The NPS team discovered the synergy patterns and the organizations involved in disseminating information orderly and efficiently in the operation. The NPS team also analyzed the All Partners Access

Network (APAN) data [30] with official briefings of 317 PDF files of situation reports, 1400 forum posts, and 3900 blog messages. By using social, metadata and LLA-generated semantic networks, the NPS team found that these were the organizations which had no social connections with others, however, they shared similar metadata attributes, discussion content, and therefore, may be predicted as potential high-value targets in the future decision and collaboration processes.

4. Acknowledgements

The authors thank Major Henry R. Salmans III, USMC (Retired) of CSC, Technology Services Organization, Programs & Resources, HQMC at the Marine Corps Information Technology Center, who provided many relevant insights and in-depth discussions.

NOTES

1. Colloquially, lower dimensional projections are empirically mined products that allow for wisdom expressed or patterns emerged in a simpler form. For example, projection pursuit was used to discover that a random number generator is not truly random but shows interesting patterns in a lower dimensional space.
2. Funding grew exponentially as the commercial practicality of using "data mining" to gain competitive edge was identified and articulated to industrial executive leadership. This article explores the landscape.
3. NoSQL databases are increasingly used in Big Data and real-time applications because of simplicity of design, horizontal scaling, and finer control over availability. The data structures used by NoSQL databases make some operations faster than those used in relational databases.

ABOUT THE AUTHORS

Dr. Ying Zhao is a research associate professor at the Naval Postgraduate School and frequent contributor to DoD forums on knowledge management and data sciences. Her research and numerous professional papers are focused on knowledge management approaches such as data/text mining, Lexical Link Analysis, system self-awareness, Collaborative Learning Agents, search and visualization for decision-making, and collaboration. Dr. Zhao was principal investigator (PI) for six contracts awarded by the DoD Small Business Innovation Research (SBIR) Program. Dr. Zhao is a co-author of four U.S. patents in knowledge pattern search from networked agents and data fusion and visualization for multiple anomaly detection systems. She received her Ph.D. in mathematics from MIT and is the Co-Founder of Quantum Intelligence, Inc.
E-mail: yzhao@nps.edu

Dr. Doug MacKinnon is a research associate professor at the Naval Postgraduate School (NPS). Dr. MacKinnon is the deputy director of the Distributed Information and Systems Experimentation (DISE) research group where he leads multi-disciplinary studies ranging from leading the Analyst Capability Working Group (ACWG) for the U.S. Air Force, studying Maritime Domain Awareness (MDA), as well as Knowledge Management (KM) and Lexical Link Analysis (LLA) projects. He also led the assessment for the Tasking, Planning, Exploitation, and Dissemination (TPED) process during the Empire Challenge 2008 and 2009 (EC08/09) field experiments and for numerous other field experiments of new technologies during Trident Warrior 2012 (TW12). He teaches courses in operations research (OR) and holds a PhD from Stanford University, conducting successful theoretic and field research in Knowledge Management (KM). He has served as the program manager for two major government projects of over $50 million each, implementing new technologies while reducing manpower requirements. He has served over 20 years as a naval surface warfare officer, amassing over eight years at sea and serving in four U.S. Navy warships with five major, underway deployments.

Dr. Shelley Gallup is a research associate professor at the Naval Postgraduate School's Department of Information Sciences, and the director of Distributed Information and Systems Experimentation (DISE). Dr. Gallup has a multidisciplinary science, engineering, and analysis background, including microbiology, biochemistry, space systems, international relations, strategy and policy, and systems analysis. He returned to academia after retiring from naval service in 1994 and received his PhD in engineering management from Old Dominion University in 1998. Dr. Gallup joined NPS in 1999, bringing his background in systems analysis, naval operations, military systems, and experimental methods first to the Fleet Battle Experiment series (1999–2002) and then to Fleet experimentation in the Trident Warrior series (2003–2013). Dr. Gallup's interests are in knowledge Management and complex systems field experimentation.

REFERENCES

1. Zhao, Y. & Atkeson, C. (1994). Projection pursuit learning: Some theoretical issues. In Computational Learning Theory and Natural Learning Systems. S.J. Hanso, et al.(Eds.). Cambridge: MIT Press.
2. Executive Office of the President (2012). Big Data across the federal government. White House. http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_fact_sheet_final_1.pdf
3. Utah Data Center (2013). http://www.forbes.com/sites/kashmirhill/2013/07/24/blueprints-of-nsa-data-center-in-utah-suggest-its-storage-capacity-is-less-impressive-than-thought/
4. Hoover, J.N. (2010). Government's 10 most powerful supercomputers. Information Week. http://www.informationweek.com/applications/image-gallery-governments-10-most-powerful-supercomputers/d/d-id/1088702?page_number=6
5. https://gigaom.com/2014/02/27/as-mapreduce-fades-apache-spark-is-now-a-top-level-project/
6. Taubes, G.A. (1995). The rise and fall of Thinking Machines. http://www.inc.com/magazine/19950915/2622.html
7. Markoff, J. (1995). Supercomputer decline topples Cray Computer. http://www.nytimes.com/1995/03/25/business/supercomputer-decline-topples-cray-computer.html
8. http://mahout.apache.org/
9. http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html
10. Olshausen, B. & Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature.
11. Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A.Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In ICML.
12. Building high-level features using large scale unsupervised learning. http://arxiv.org/pdf/1112.6209v5.pdf
13. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
14. Navy Big Data (2014). http://defensesystems.com/articles/2014/06/24/navy-onr-big-data-ecosystem.aspx
15. Zhou, C., Zhao, Y., & Kotak, C. (2009). The Collaborative Learning agent (CLA) in Trident Warrior 08 exercise. In KDIR, Madeira, Portugal, INSTICC Press.
16. Zhao, Y., Gallup, S.P., & MacKinnon, D.J. (2011). System self-awareness and related methods for improving the use and understanding of data within DoD. Software Quality Professional, 13(4), 19-31. http://www.nps.edu/Academics/Schools/GSOIS/Departments/IS/DISE/docs/improving-use-and-understanding-of-data-dod.pdf
17. Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1022. http://jmlr.csail.mit.edu/papers/volume3/blei03a/blei03a.pdf
18. The Stanford Natural Language Processing Group. http://nlp.stanford.edu/software/lex-parser.shtml
19. http://iase.disa.mil/cloud_security/Documents/u-cloud_computing_srg_v1r1_final.pdf
20. http://www.acfr.usyd.edu.au/pdfs/training/multiSensorDataFusion/dataFusionNotes.pdf
21. Klein, G., Moon, B. & Hoffman, R.R. (2006). Making sense of sensemaking 1: Alternative perspectives. IEEE Intelligent Systems, 21 (4):70-73.
22. Endsley, M.R. (1997). The role of situation awareness in naturalistic decision making. In Zsambok, C.E. and Klein, G .(Eds.), Naturalistic decision making, Mahwah, NJ.
23. Quinlan, J.R. (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers.
24. Franklin, S. & Graesser, A. (1996) Is it an agent, or just a program?: A taxonomy for autonomous agents. In the 3rd International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag.
25. Gallup, S.P., MacKinnon, D.J., Zhao, Y., Robey, J., & Odell, C. (2009). Facilitating decision making, re-use and collaboration: A knowledge management approach for system self-awareness. In IC3K, Madeira, Portugal, INSTICC Press.
26. Zhao, Y., Gallup, S.P., & MacKinnon, D.J. (2010). Towards real-time program awareness via lexical link analysis. http://www.acquisitionresearch.net/files/FY2010/NPS-AM-174.pdf
27. Zhao, Y., Gallup, S.P., Mackinnon, D.J. (2011, 2012). Applications of lexical link analysis web service for large-scale automation, validation, discovery, visualization and real-time program-awareness. http://www.acquisitionresearch.net/files/FY2011/NPS-AM-11-186.pdf http://www.acquisitionresearch.net/publications/detail/1020/
28. Zhao, Y., Gallup, S., Mackinnon, D.J. (2013). Lexical Link Analysis application: Improving web service to acquisition visibility portal. http://www.acquisitionresearch.net/publications/detail/1220/
29. Zhao, Y., MacKinnon, D.J., Gallup, S.P. (2011). Lexical Link Analysis for the Haiti earthquake relief operation using open data sources. In the 16th ICCRTS. http://www.dodccrp.org/events/16th_iccrts_2011/papers/164.pdf
30. Zhao, Y., MacKinnon, D.J., & Gallup, S.J. (2012). Semantic and social networks comparison for the Haiti earthquake relief operations from APAN data sources using lexical link analysis. In the 17th ICCRTS. http://www.dodccrp.org/events/17th_iccrts_2012/post_conference/papers/082.pdf
31. Palantir. http://www.palantir.com/technologies/
32. Metadata Analysis (2013). http://www.slate.com/articles/health_and_science/science/2013/06/prism_metadata_analysis_paul_revere_identified_by_his_connections_to_other.html

Rapid Deployment of Data Mining for Engineering Applications

Nikhil Dakwala, Broadcom Corporation

Abstract. This paper enables rapid deployment of data mining to improve engineering efficiency and productivity. The paper presents the fundamentals of data mining: structured vs. unstructured data, supervised vs. unsupervised learning, cluster analysis, association learning, and decision trees. The paper explains algorithms, data structures, and fuzzy clustering to extract actionable intelligence. The paper provides pseudocode to mine Integrated Circuit (IC) test data and guides readers to practice mining skills on 2012 Medicare payments data online.

1. Introduction

Data Mining (DM) is the process of discovering knowledge hidden in the underlying data. This process is also called knowledge discovery in data (KDD). This hidden knowledge is the Actionable Intelligence, the critical knowledge that will guide the further course of action to quickly attain the desired goals. Data mining is as old as our civilization. Ancient Egyptians were the first known data scientists. They collected enough empirical evidence to link the arrival (clear visibility) of the star Sirius with the flooding of the river Nile. Based on these observations they derived actionable intelligence of when to plant crops and when to stay away from the river. Human history is full of engineers without degrees performing data mining to solve immediate problems at hand. This compelling history can be read online in reference [14].

Data mining was transformed into machine (artificial) intelligence as computers began to proliferate. Within the last decade, an explosion in the computing platforms has created a data deluge in a variety of industries. Most of this data is never analyzed and is simply archived and forgotten. While not all data is worth mining, embedded monitors, on-chip and onboard instruments, network of industrial sensors, etc. provide continuous real-time data. This type of data needs to be mined to improve performance, production costs, runtime reduction, and security for a wide range of end applications. An engineer is very likely to encounter data deluge which can be immediately mined to improve personal productivity and efficiency. An engineer is also unlikely to have easy and immediate access to Relational Database Management systems (RDBM) that offer built-in DM solutions. Being a quintessential knowledge worker, an engineer is likely to be intimately familiar with the underlying data and to possess the necessary technical skills for KDD. This paper shows how to develop an engineering DM solution without requiring expert DM knowledge or commercial DM software. For readers curious about commercial data mining, review the links provided by Microsoft in reference [12] and Oracle in [15].

The author's first introduction to data mining was in the form of machine intelligence while beginning Master's thesis in 1989 to encapsulate a diagnostic expert's knowledge in the form of heuristics that drove diagnostic software engine and on-chip test logic to detect faults in IC chips. Advances in IC fabrication around the year 2000 shrank chip size and simultaneously boosted their performance. Along came an explosion in the variety and quantity of defects. Consequently, when chips failed, they failed in large numbers. Out of necessity, data mining began to be deployed at several stages in the IC chip lifecycle. Reference [1] uses an engineer's knowledge, called pre-filtering, to detect physical defects in electric circuits. Reference [4] presents Bayesian statistics DM, which starts with a set of previously learned probabilities and dynamically updates them with new data. Reference [6] targets outliers to speed up Boolean state justification. The DM to target timing-critical circuits is presented in [2]. Reference [3] uses a support vector machine (SVM) DM utility from MilDe, a publicly available machine learning package. An excellent guide to DM is in reference [7], an open source book from MIT press, "Principles of Data Mining", by Hand, Mannila, and Smyth.

The author leveraged data mining to counter problems faced while testing IC chips before and after fabrication. Every DM term and solution in this paper is presented in context of testing IC chips. The IC test terminology has been greatly simplified, and the DM solutions are generic enough to be applicable to all fields of engineering. It is still important for the reader to thoroughly read section 2, which familiarizes the reader with the IC test knowledge required to understand DM techniques presented in the paper. The remaining sections are organized as follows. Section 3 explains fundamental DM terms, theory, algorithms, and processes. Section 4 presents details about software tools, pseudo-coded data structures, and algorithms for rapid deployment. Readers can experience DM by participating in the online mining of 2012 Medicare payments data as shown in section 5. While an engineer will be intimately familiar with the data he/she is dealing with, not all engineers are programmers. In such cases, hopefully there will be some programmers on the team. Otherwise, an aspiring engineer can obtain the required programming skills by following the methods and references provided in this paper. The software mentioned in this paper is available from the public domain. For the remainder of this paper, DM, DM software, and DM solutions are used interchangeably.

2. IC Testing 101

Figure 1 shows the IC test flow. Test patterns generated to test IC functionality are validated against the software model of the chip running under a software simulator. Failures at this stage could be due to incorrect test patterns or an incorrect software model. Once the test patterns pass in the software simulation, they are executed on the chip, utilizing a hardware chip tester. Failures at this stage can mainly be due to

manufacturing defects, incorrect test patterns, or a discrepancy between the software model and the actual chip.

IC chips undergo rigorous testing on the hardware tester at multiple voltage and temperature values. Depending on fabrication process variations, certain chips can exhibit sensitivity to different voltage and temperature values, i.e., they will fail at a voltage lower than 10% of the normal operating voltage, but otherwise function correctly at all other operating conditions. This is called low-voltage sensitivity. Consider the fact that a 12-inch diameter wafer can hold 100 or 1000 chips, depending on the size of each chip. When these chips fail, the volume of failure and debug data can become overwhelming.. DM is very useful in debugging the data deluge created due to process variation failures.

Another possibility of data deluge occurs when two different blocks in the chip are utilizing two independent clocks, and they each have multiple data registers. Each register can have one or more bits. It is possible that there is an unintentional path starting from a register bit in one clock domain, crossing into another clock domain, and ending at another register. This is called Clock Domain Crossing (CDC). Such clock domain interactions must be detected and handled through special synchronization circuits, or prevented altogether because they can cause data to be lost while travelling across clock domains running at different frequencies.

From a software perspective, a 32-bit register is a single entity. From a chip design perspective, each bit in this register is a unique entity. This allows us to identify and account for each piece of logic-circuit on the chip. This is similar to having a residential community where US mail can be delivered to each family living in each house by identifying them through a unique home address. An example below shows bit 39 of GPIO_1 register. The delimiter "/" represents a logical hierarchy in the chip. Starting from left to right, the logical hierarchy progresses from the input ports of the chip towards the output ports.

soc_top/core_1/i_biu/gpio_1/reg_39_/q
soc_top/core_2/i_biu/gpio_1/reg_39_/q

Thus, the example above shows two separate register bits with the same name, separately residing in core_1 and core_2.

2.1 Determining DM necessity

DM can be deployed for a variety of engineering requirements. Past data can be mined to extract useful features, discard least used features, and develop new products. Data from current processes can be mined to improve efficiency and productivity. The user needs to have some insight into data,-patterns to look for, process sigma variations to detect outliers, and a set of theories to prove or disprove. Contrary to popular definition of Big-Data, DM does not depend on the size of the underlying data. In fact, as will be explained in section 3, a smaller dataset is initially required to develop and test a DM solution. Once validated on small dataset, it is then applied to the larger set for knowledge discovery. Therefore, DM deployment depends on the user's need and aptitude.

3. Fundamentals of Data Mining

Figure 2 shows the DM flow.

First we will briefly describe each step shown in the DM flow above, followed by details in subsequent sections. Step

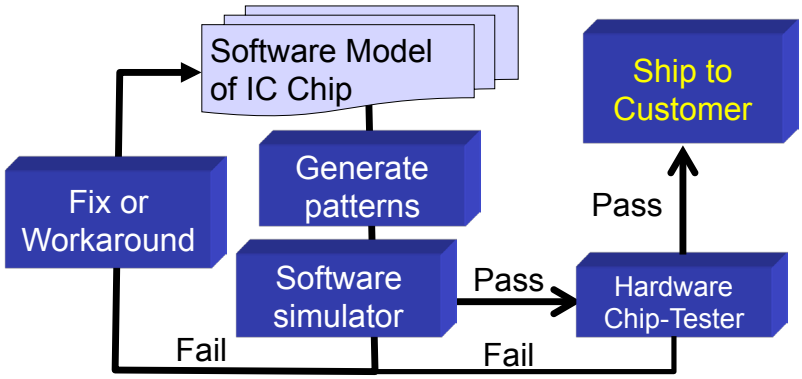


Figure 1. IC Test Flow

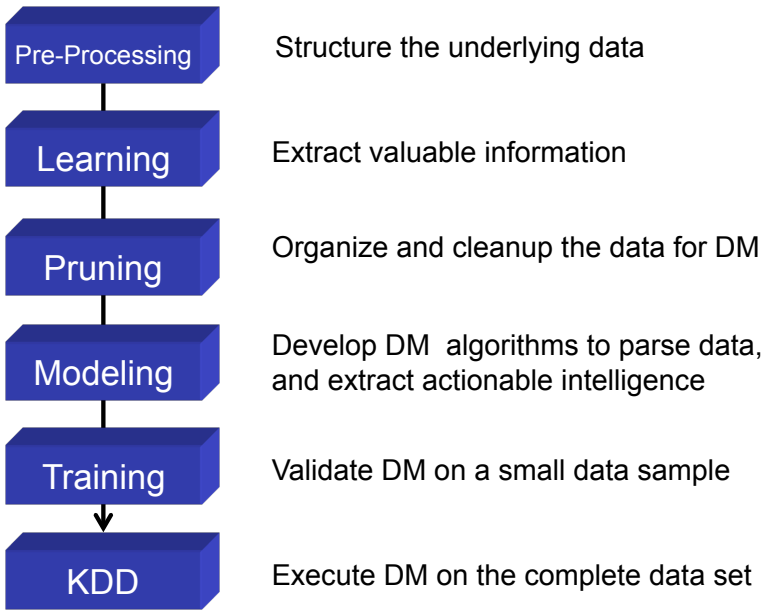


Figure 2. DM Flow

0 is pre-processing which is sometimes needed to properly structure the underlying data. This is explained in section 3.1. Once the data is properly structured, step 1 in developing a DM solution is to learn the underlying data, i.e., to understand the information it represents, its syntax, and semantics. This is explained in section 3.2. Another item that requires attention is data storage and retrieval from the database and hard disks. For engineering applications, we will assume that the data is stored on hard disks and is readily available. Step 2 is to prune the data by parsing/translating it into a format that the DM solution can work with (discarding useless data) and thereby saving compute resources. Pruning relies on the likelihood of relevant data features producing valid and usable results. Pruning is done in the software used to develop DM solution. Step 3 prepares DM models, i.e., algorithms to exploit relevant data features and relationships found through learning and pruning. Modeling is also done in DM software. Both pruning and modeling depend on user's programming skills, aptitude and are not addressed in

this paper. In step 4, the algorithms are proven, or trained, on a small sample size using model(s) that have been just developed. Step 4 is part of due diligence in any engineering process and hence not discussed in this paper. KDD, the final step targets the entire dataset to extract actionable intelligence. KDD is covered in sections 3.3, 3.4 and 3.5.

3.1 Structured vs. Unstructured Data

If your DM can easily extract the information you value from the underlying data, then the data is correctly structured. If, on the other hand, the underlying data is circumspect about the information you value, then it is unstructured data. Let us say that we have a DM to extract the population of each town in Texas by performing a Wikipedia query. Go to www.wikipedia.org and search for Old Dime Box, Texas. You will see the following:

"Old Dime Box is an unincorporated community in Lee County, Texas, United States. According to the Handbook of Texas, the community had an estimated population of 200 in 2000."

A Wikipedia search for another town, Muleshoe, Texas, reveals the following information:

"Muleshoe is a city in Bailey County, Texas, United States. The town of Muleshoe was founded in 1913 when the Pecos and Northern Texas Railway built an 88-mile (142 km) line from Farwell, Texas to Lubbock through northern Bailey County. In 1926, Muleshoe was incorporated. The population was 5,158 at the 2010 census. The county seat of Bailey County, it is home to the National Mule Memorial."

The two examples above show that the population-extraction-DM is dealing with unstructured data. Each town has its population listed with a variety of other details, in no particular order. In computer terms, there is no syntax or semantics that will enable pattern matching. Wikipedia solves this problem by also displaying a table on the far right of the search result screen. This table has the relevant information structured as shown below:

Old Dime Box
Unincorporated community
Location within the state of Texas
Coordinates: 30°22' 38" N 96°51' 47" W
Country United States
State Texas
County Lee
Population (2000)
• Total 200
Time zone Central (CST) (UTC-6)
• Summer (DST) CDT (UTC-5)
GNIS feature ID 1375270

The structured data above can be easily parsed by DM to extract the relevant information.

3.2 Supervised vs. Unsupervised Learning

A typical DM solution employs supervised or unsupervised learning processes to discover detailed information about the underlying data. Supervised learning extracts specific knowledge (patterns, relationships) from specific input data. In other words, an engineer looks at the data and either creates a customized DM or modifies an existing one. Unsupervised learning discovers hidden knowledge from structured and unstructured data. Unsupervised DM is mostly based on advanced neural network and artificial intelligence algorithms. In the author's view, it is impractical for rapid deployment in most engineering applications.

Supervised learning can only work on structured data. It relies on the engineer's intimate knowledge of the underlying problem and data, thereby jump-starting the learning process. By visual inspection of the problem data, an engineer can discern possible cause(s) or failure behavior based on which DM solution can be modeled. This paper only discusses supervised learning. Ambitious readers looking for a challenge can learn about unsupervised learning and a lot more complicated DM in the excellent online book listed in reference [13].

Now, let us travel back in time to Dime Box, Texas and supervise construction of a DM to extract its population from the structured data. Let us assume that we already have a list of all towns in Texas and an automatic method to communicate with the Wikipedia website. We can model our population-extraction-DM as follows:

- First line is the name of the town
- Ignore all lines, unless the line begins with the word "Population", followed by #decimal_year
- Obtain #decimal_population from the next line, after the word: "Total"

If at first you encounter an engineering application producing unstructured data, talk to the owner and reconfigure it to produce structured data. If the owner refuses or otherwise is incapable, you can employ regular expressions, parse out irrelevant information, and impose a structure on the data. Regular expressions will be demonstrated in section 4.

3.3 Knowledge Discovery: Cluster Analysis

Sections 3.3, 3.4, and 3.5 discuss knowledge discovery in data. Theory behind knowledge discovery presented in these sections is just enough to support practical deployment. The reader will find detailed theoretical background in reference [13]. Supervised knowledge extraction depends on the type of knowledge we are seeking and the type of relevant patterns we are able to discern in the data. Knowledge extraction is conducted using cluster analysis, association analysis, and decision trees. In the same order as the three techniques listed, the quality and quantity of KDD obtained, and DM difficulty progressively increases.

Cluster analysis organizes the data in groups or clusters such that each member of the cluster is closer to other members of the same cluster than it is to any other member in any other cluster. This is called supervised K-Nearest Neighbor (K-NN) clustering where users control the number and type of clusters.

For example, we can group Texas towns based on their counties: Lee, Travis, etc. We can also group them based on their population count being greater or smaller than a certain limit. Cluster analysis provides cluster statistics such as X% Texas towns belong to Lee county, Y% belong to Travis county, etc. The statistics can be organized in ascending or descending order for further analysis.

3.4 Knowledge Discovery: Association Analysis

Association analysis derives knowledge by associating data points from similar and different sources. Association requires an engineer to be familiar with the entire system in which the data is generated and consumed. To illustrate association in terms of IC failures, let us say that we have clustered IC chips failing at nominal and ±10% of nominal voltage. We can then associate the failing vectors with software simulation logs to determine the exact functionality being executed at the time of failure. Such associated data can bring out one or more functionalities during which most failures occur.

Going back to Texas, let us say that we perform similar population-extraction-DM on all Texas towns every year, going back 20 years. We can now associate the current year's data with the data from each previous year and learn when the population increase (or decrease) started.

It should be evident by now that supervised DM depends not only on the engineer's knowledge about the data, but also on what kind of information the engineer is looking for. If we are trying to solve IC chip failures, we need to have a theory of possible causes. Then we need to examine the data and detect patterns we can leverage to construct a DM to prove or disprove those theories. Clustering and association show data points that fit with our theories as well as the outliers that disagree with our thinking. The final piece of evidence, the actionable intelligence, comes out through decision trees, which are explained in the following section.

3.5 Knowledge Discovery: Decision Trees

Once the knowledge is clustered and associated, a tree consisting of a series of if-then-else decisions is constructed to draw actionable conclusions, which will prove/disprove the engineer's original theories. This process is called Decision Tree (DT) based DM. The elements of a supervised DT are decisions, conditions, errors, stubs, number of members or size of the dataset, and data attributes. A decision tree is a flowchart of conditions that the data has to satisfy for the engineer to reach a decision. Each condition causes a split in the DT based on it being satisfied or not satisfied. The engineer needs to be cognizant of the types of errors the data can have due to incorrect experiment settings or equipment malfunctions. Stubs are illegal conditions invalidating the complete dataset causing a dead end in the DT. Finally, the data itself has certain attributes, like the test results under high temperature, low voltage, etc.

Quantity and quality of compute power and storage is usually not a concern for engineering rapid deployment. In case resource planning is required, it is possible to estimate the data storage, runtime memory requirements, and the computational complexity. For a given dataset with m members, a attributes, d decisions, c conditions, e errors, and s stubs, the worst-case computation is represented by equation: d.m.a(e+s+2c). The equation to estimate data storage requirements is m(size_of_a). The equation for runtime storage estimates is m(size_of_d). As described in an earlier section, a DM solution can also have pruning, clustering, association, and sorting capabilities, which all affect runtime, speed, and storage. The author does not recommend drawing an actual decision tree because it is a distraction during rapid deployment. A decision table can be constructed in an Excel spreadsheet if needed. It is best to deploy it directly in software using in-built functions like if-then-else, foreach, while loops, regular expressions, and multidimensional arrays.

4. Data Mining Software

The reader can implement DM using any software the reader is familiar with. The author uses PERL software for DM. There are certain applications where the author also uses TCL software. Both PERL and TCL are included in Linux and Unix operating systems. A Windows version is available from www.activestate.com. PERL for Windows is also available at www.strawberryperl.com. DM software needs multidimensional arrays, regular expressions, and subroutines to enable fast pruning, data processing, and DT implementation. Regular expressions are essential for parsing and pruning to impose a structure on the underlying data which will enable knowledge discovery.

4.1 Fuzzy Clustering and Regular Expressions

DM is all about diving into the data and discovering pieces of knowledge. Fuzzy clustering is a contrarian technique which provides an eagle's eye point of view. It enhances cluster analysis by removing uniqifiers to expose global characteristics; e.g., removing register bit indices, circuit flattening uniquifiers, multicore identifiers and hierarchies, etc. to bring out commonality from the sea-of-uniqueness. Fuzzification is done through regular expressions. Most software languages support pattern matching and regular expressions. While the author is most comfortable with PERL, the reader can employ any other software language for regular expressions. For illustration, consider the two registers in Section 2.

```
soc_top/core_1/i_biu/gpio_1/reg_39_/q
soc_top/core_2/i_biu/gpio_1/reg_39_/q
```

Using PERL, regular expression prunes out every piece of information and extracts the register name:

```
/(\/)(\w+)(\/\w+\/q)/;
$register_name = $2;
```

These register names are then counted for their occurrence, stored in an array, sorted, and utilized for further knowledge discovery. The best way to learn PERL is to read the book by PERL's inventor, Larry Wall [16]. To learn TCL, visit the tutorial in [17]. Both these references contain good explanations on regular expressions and pattern matching.

4.2 DM pseudocode for rapid deployment

We will go back to Clock Domain Crossing (CDC) issues highlighted in Section 2 on IC testing.

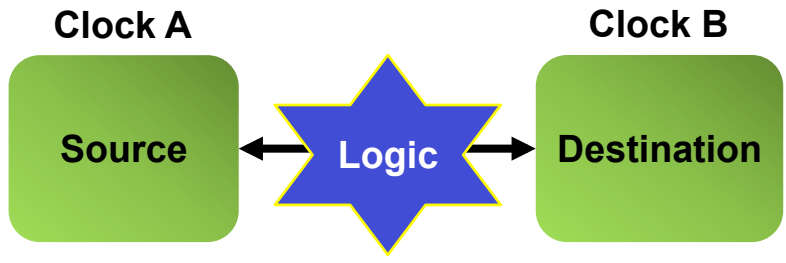


Figure 3. clock domain crossing

Figure 3 shows two clocks, A and B, that can be running at difference frequencies. For illustration purposes, clock A is the source of start points, i.e., individual bits of registers like GPIO, transmit data, receive data, etc. that can go through different logic circuits and end up at individual bits of other registers located in the clock B domain. In reality, clock B can also originate start points that end up in the clock A domain. Circuit tracing applications generate a full report of all such paths in the circuit with the following syntax:

<end_clock> <end_point_name>

<start_clock> <start_point_name>

Figure 4 shows the knowledge we seek.

The goal of mining CDC data is to identify abnormalities where a single start point reaches multiple endpoints across other clock domains. Similarly, identify a single endpoint acting as a terminator for multiple start points. The DM pseudocode is shown below. Note the following:

- It is based on the PERL syntax.
- Data structures are implemented using hash-tables (associative arrays).
- Algorithms are coded into subroutines controlled by variables, which will also control their respective placement in hash tables.
- Extracting relevant start, end–points, and clock names is done through regular expressions.

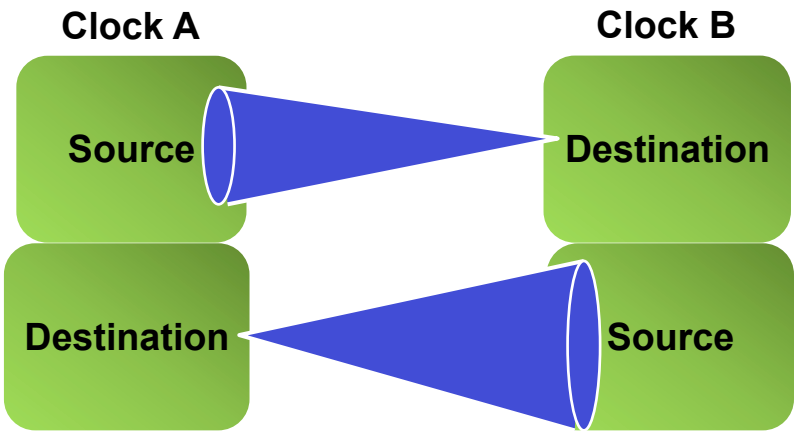


Figure 4. DM Goal: Identify cones

- None of the regular expressions and other software codes for parsing data, sorting hash tables, and compiling statistics are shown because they differ for each application.

```
##################################
#Main Code
#defaults and defines
#parse mandatory, optional arguments, individual files or file list
#open gzip'd file and non-zipped files
#parse data
#extract end-clock and end-point
&process_store_node ($clock, $end_point, "end_point");
#3rd argument is branch/flow control in hash storage

#extract start-clock and start-point
&process_store_node ($clock, $start_point, "start_point");
#
#sort and print the mined jewels
&process_sort_collection("end_point", "full");
&process_sort_collection("start_point", "full");
&process_sort_collection("end_point", "fuz");
&process_sort_collection("start_point", "fuz");
#
# Finish Main Code
##################################
sub process_store_node {
#store node according to flow control, update counts
#each node will have a counter and associated clock variables, #which are not shown here

#Data structure to store full nodes
  $dat_hash{full}{$flow_ctl}{$node};
  $dat_hash{full}{$flow_ctl}{$node}

#fuzzify, store node, update counts
$fuzed_node = &process_fuzzify_name($node);

#Data structure to store fuzzy nodes
  $dat_hash{fuz}{$flow_ctl}{$fuzed_node}
}#sub process_store_node
##################################
sub process_sort_collection {
#counts each hash entry, sort and print in descending order
} #sub process_sort_collection
##################################
sub process_fuzzify_name {
#remove uniquifiers: reg, inst, bit-slices, other unique traits using regular expressions shown earlier
} #sub process_fuzzify_name
##################################
```

Figure 5. shows an example of the actionable intelligence extracted from the CDC DM on a design the author worked on. The worst offenders are shown first followed by others in descending order. The first table lists the registers acting as the start points of paths crossing over to another clock domain. The second table lists registers acting as endpoints (terminations) for paths originating from another clock domain.

#starts	**Register name**
160656	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_rw_cssnoop_vld
42537	vaux_mdio_intf_mdio_slave_pci_addr
18240	vaux_cfg_multi_vf_cfg_vf_cfg_private_vf_flr_in_progress
15840	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_tl_ack

terminations	**Register name**
200836	core_tl_tl_isolate_cfg_tl_demux_cfg_bararr
19899	core_tl_trx_receive_trx_common_interfac_rx_data
18400	core_tl_trx_receive_trx_cmplctl_vf_bdf_table
12842	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_cs_rd_data

Figure 5. Actionable Intelligence

5. Experience DM on 2012 Medicare payments

The Wall Street Journal (WSJ) and several other entities have been mining Medicare payments data made available by the USA Government to identify fraud and misuse. The raw government data can be obtained from www.cms.gov, at the Research & Statistics link. The WSJ has structured 2012 data and put it online. Reference [18] is one of several articles they have published with knowledge discovered from this data. The reader is cautioned not to jump to conclusions based on the billing data. To experience DM algorithms, go to this link: http://projects.wsj.com/medicarebilling/?mod=medicarein#

The data is grouped based on following clusters:

- Last name / Company name
- Specialty / Facility type
- City
- Location

Each of the above has been further clusterized in to several smaller clusters. For example, Specialty/Type has clusters like Psychiatry, Ambulatory services etc. Leave 1st three columns (name/type/city) empty and select "Foreign Country" as location and press the Search button. You will discover foreign individuals collecting from American Taxpayers. Several individual and company names will have a "+" sign to their left, which will expand when clicked, and will show further breakdown of charges.

To experience association analysis, leave name and city columns empty, select "slide preparation" facility type and "all" locations. Click on "state/country" columns to sort based on the State name. Now you have associated the most expensive providers of slide preparation services with the States in which they're located. A visual count will show 48% (12/25) of these providers reside in California.

Decision trees are constructed in software to sort through the clusters and their associations. Limited DT based DM can be performed by copying the "payment" column in to an Excel spreadsheet, and by identifying the average, median and outlier payments. Knowledge extracted through DT is presented in reference [18]

6. Conclusion

Author's primary objective in writing this paper was to enable DM deployment in different engineering applications. The success and speed of such deployment will depend on each application and the user's skills. . The DM pseudocode presented in section 4.2 is a good template to construct reusable DM software. For DM novices, start by learning Perl [16]. Focus on regular expressions, associative arrays and sorting. Author recommends starting with supervised data mining. Cluster analysis is the easiest form of data mining. Engineers with good software skills and basic DM knowledge can deep dive in associative analysis and decision trees. The reader will learn much more DM by practice rather than reading. Readers are encouraged to contact the author with questions, for more information and explanations as needed. Good Luck.

ABOUT THE AUTHOR

Nikhil Dakwala I firmly believe it is possible to improve personal efficiency and productivity through data mining. As mentioned in the paper, my first introduction to data mining was in the form of machine intelligence while working on my Master's thesis at SUNY Buffalo. After obtaining my MS EE in 1991, I joined Motorola's microcontroller division where the goal was to manually detect faults and achieve the highest quality at the lowest cost. Without any automated EDA solution, this experience resembled trench warfare, but it was also a data mining paradise. Out of necessity, I began using cluster fault analysis to detect the maximum number of faults for the maximum coverage increase in the least amount of time.

Since then, I have worked at various companies such as IBM, ARM, and startups. I have been a consultant and currently I am employed at Broadcom. When faced with roadblocks, I resort to data mining to gain deep insight into the data and chart the best course forward. I have conducted research on topographical analysis of silicon failures based on chaos theory. I have presented at IEEE ITC, and STC. Feel free to contact me with questions and DM ideas.

E-mail: ndakwala@broadcom.com

REFERENCES

1. Y. Hirano, et al, "Scrubber induced substrate cracks found by data mining", IEEE ISSM, 2005, pp. 257-259
2. J. Chen, et al, "Mining AC delay measurements for understanding speed-limiting paths", ITC 2010, p 18.3
3. S. Wang, et al, "Machine learning based volume diagnosis", DATE, 2009, pp. 902-905.
4. Daasch et al, "Die level adaptive test: real time test reordering and elimination", ITC 2011, p15.1
5. Caruana and Mizil, "Empirical comparison of supervised learning algorithms", ICML 2006.
6. W. Wu and M. Hsiao, "SAT-based state justification with adaptive mining of invariants", ITC 2008, p7.2
7. "Principles of Data Mining", by David Hand, Heikki Mannila and Padhraic Smyth, MIT Press open source
8. K. Baker and J. Beers, "Shmoo plotting: The black art of IC testing", ITC 1998, L2.3
9. P. Patten, "Divide and conquer based fast shmoo algorithms", ITC 2004, P8.3
10. L. Huisman et al, "Data mining IC fails with fail commonalities", ITC 2004, P232
11. N. Dakwala, "Data Mining Fail Data Through Cluster Analysis and Association Learning", ITC DATA 2011
12. Microsoft SQL Server: "Basic Data Mining Tutorial", http://technet.microsoft.com/en-us/library/ms167167.aspx
13. Pang, Steinback and Kumar: "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/index.php
14. Stephen Wolfram: "Advance of the Data Civilization", http://blog.wolframalpha.com/2011/08/16/advance-of-the-data-civilization-a-timeline/
15. Oracle "Data Mining Document Library", http://www.oracle.com/pls/db102/portal.portal_db?selected=6
16. Larry Wall: "Programming Perl"
17. TCL Tutorial: http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
18. The Wall Street Journal, "Taxpayers Face Big Medicare Tab for Unusual Doctor Billings," June 9 2014, http://online.wsj.com/articles/taxpayers-face-big-medicare-tab-for-unusual-doctor-billings-1402364264

Not All Time Matters

Be Sure To Count What Does

Timothy A. Chick, SEI
Lana Cagle, Naval Oceanographic Office
Gene Miluk, SEI

Abstract. "Time" seems like a simple measure to use when planning and tracking projects. Most accounting systems track employees' work hours. The resulting time measure is in essence a proxy for cost: Labor hours translate into dollars. However, if you have used time that has been tracked in that way to plan and estimate projects that depend on developers, testers, engineers, or other knowledge workers, you have probably found that your estimates aren't as close to the actuals as you would like. If you have difficulty bringing projects in on schedule with the promised functionality, you might want to rethink the "time" you are using.

For this article, we are going to call the hours employees spend working "ordinary time." Ordinary time is very useful for accounting and payroll purposes, but several major issues make it nearly impossible to use for creating accurate estimates or precisely tracking projects, especially in system development.

- Unrelated activities are included. The time being measured for payroll is often a mixture of activities that contribute to costs, but that are not directly related to project deliverables. Ordinary time includes tasks such as attending meetings or training classes, reading email, creating reports, and working on a myriad of administrative requirements.
- Detailed estimating is difficult to accomplish using ordinary time. Accounting systems do not generally track time at the level necessary for bottom-up, detailed estimating. Their data is instead well suited to top-down estimations for payroll and budget purposes.

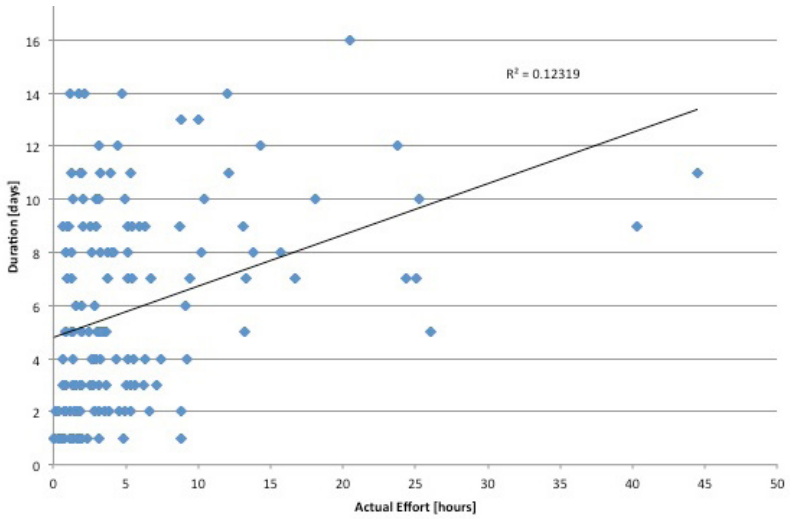


Figure 1: Effort And Duration, Work Packages < 20 Days Of Development

- Precise project tracking is not supported by the data. The time data captured supports gross project tracking, but does not support the precise project tracking that system development projects need to be repeatedly successful. Fred Brooks best states the importance of adequate tracking in his book The Mythical Man-Month when he says, "How does a project get to be a year late? One day at a time."

In short, ordinary time is not what a team of knowledge workers needs to successfully deliver a product or service on schedule and within budget. Using research done at the Software Engineering Institute (SEI) and experience gained working with the Naval Oceanographic Office, we hope to help others avoid the time trap by measuring the time that matters.

The Time Trap

The assumption of a relationship, or correlation, between effort and duration is fundamental to traditional project planning and tracking techniques, but is simply not a valid assumption for knowledge-based efforts. The problem is evident when we examine project planning approaches and tools for system development, which fit into two general categories: traditional and agile-based.

The traditional approach follows the PMBOK [1], which focuses on creating a work breakdown structure (WBS) in order to capture the project scope. The WBS is then used to define the activities and their sequence and to estimate activity resources and durations to facilitate the schedule. Given this, cost can be estimated and a budget generated. In order to generate cost estimates for labor, most financial systems require ordinary time, thus employee's time is planned and tracked for every hour worked. This in turn is used in an earned value management system. The PMBOK uses costs to determine earned value and schedule variance. While these techniques have proven effective in industry, they have been shown to be less effective in software development [2,3,4] and on other knowledge-based projects.

One reason for the ineffectiveness can be seen in Figure 1, which shows that there is no correlation between effort and duration. To illustrate the difficulty in using ordinary time in this traditional approach we took data from 89 software development and knowledge-based projects in which the duration of the work package was less than 20 days, from open to close. The data shows that there is no correlation between the duration of a work package (i.e., ordinary time—in days— spent during development) and the actual effort required to complete it (i.e., only the time spent working directly on project deliverables).

Agile-based approaches and tools usually plan and track progress using story points. First, an ordered list of requirements called a product backlog [5] is generated, then the backlog is prioritized and each item in the backlog is estimated using story points. A story point is an arbitrary measure that represents the effort required to implement a story. The point system is based on the Fibonacci sequence [6]. All effort is performed using an iterative, time-boxed approach, usually called a sprint. Past performance is used to predict how many story points can be implemented per sprint. Most agile teams do not collect effort in terms of hours because they consider it to be a waste of effort [7]. However, there are several problems with relying solely on story points as a project's only source of planning and tracking data.

- Story points and velocity are subjective measures that are calibrated by each team based on previous team performance [8].
- Estimates are biased [9].
- The use of story points is not objective and thus cannot be used to define a standard practice for the estimation of software size [10].

Time That Does Matter

A much more useful measure of effort for estimating, planning, and tracking systems development projects is "task time." Task time is defined as the actual time spent working on a specific task in the plan. To determine task time, each individual working on the team is responsible for tracking the time spent working on each specific task they are assigned in the plan.

Figure 2 shows that there is a very strong correlation between the planned and actual task time required to implement an individual work package. This high correlation enables very effective bottom-up estimating. Figure 3 shows the use of task time in creating accurate estimates. It plots actual task time against planned task time, in hours, for completed projects. This demonstrates that there is a strong correlation between the bottom-up estimates for a project and the actual task time spent on the project.

Task time can be used to overcome the deficiencies of both traditional and agile based approaches. Traditional planning and tracking methods can use task time to overcome the shortcomings of using ordinary time or financial information alone to determine when a project will be completed. Once you can determine when a project will be completed using a given set of resources, you can also determine how much the project will cost. Task time can also be used in place of agile's subjective story points approach, by providing an objective measurement which can be used as a standard estimating practice across teams and organizations, while conforming to the Agile Manifesto and Principles.

What Makes Task Time So special?

The reason task time is more effective for project planning and tracking is because it represents the project's value chain and ignores the other activities, which skew a predictability. The value chain is the set of tasks directly associated with a work package that is carried out to create value for the customer or end user. While effort reports record both ordinary time and time spent on primary tasks such as coding and testing, task time applies only to the primary tasks of a project's value chain. While the cost of project members can be predictable, the amount of task time individuals are able to commit to a project's value chain is highly variable, as seen in Figure 4. Understanding the variability at the individual and team level is key in producing accurate estimates and for precise status tracking.

Task time is measured in hours or minutes. Interrupt time, or off-task time, is not included in the time measure for a task; if there is an interruption during the work, that time is subtracted from the time measurement. Task time only measures the time spent working on a specific work package. In general, off-task time is not measured or tracked since it does not contribute to meeting the stated project goals.

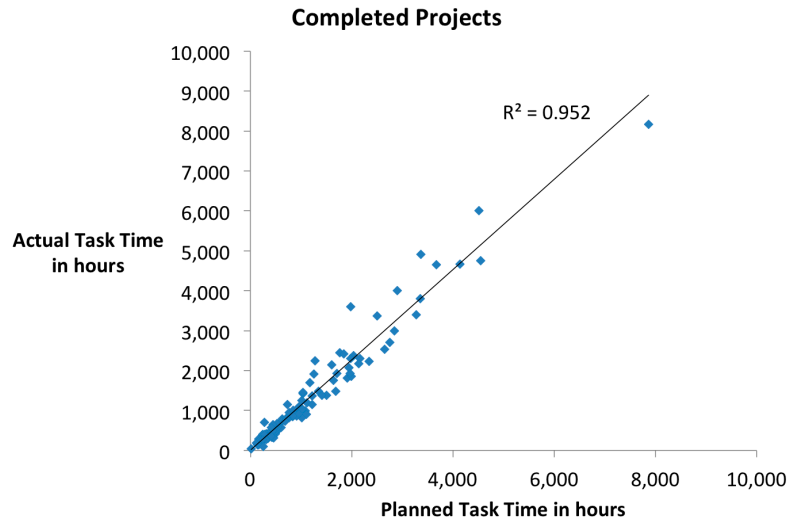


Figure 2: Actual Versus Planned Task Time For Completed Projects

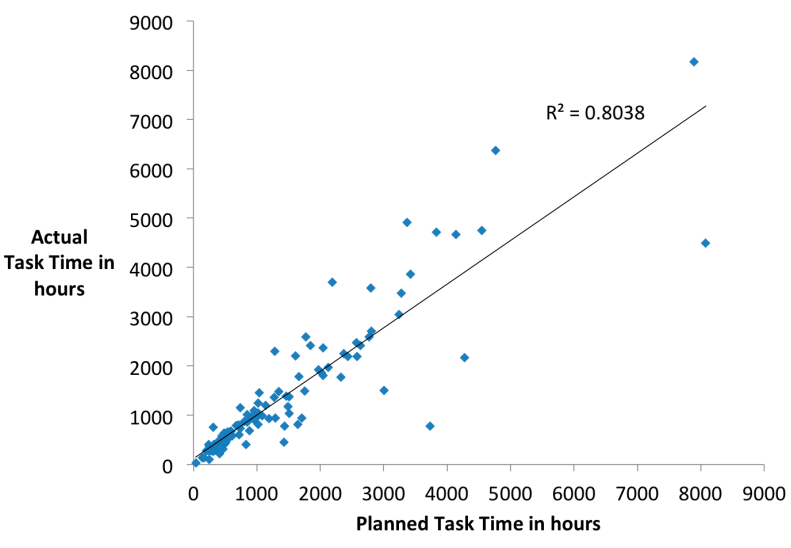


Figure 3: Actual Task Time Versus Plan Task Time

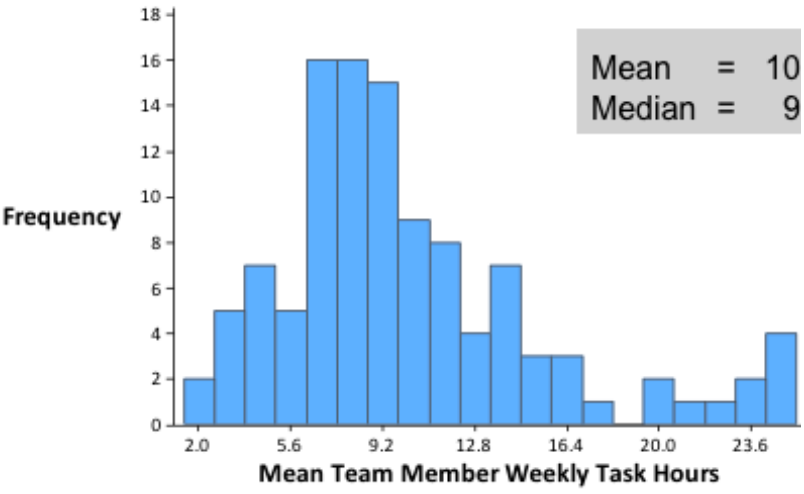


Figure 4: Average Weekly Task Time, In Hours, Per Individual

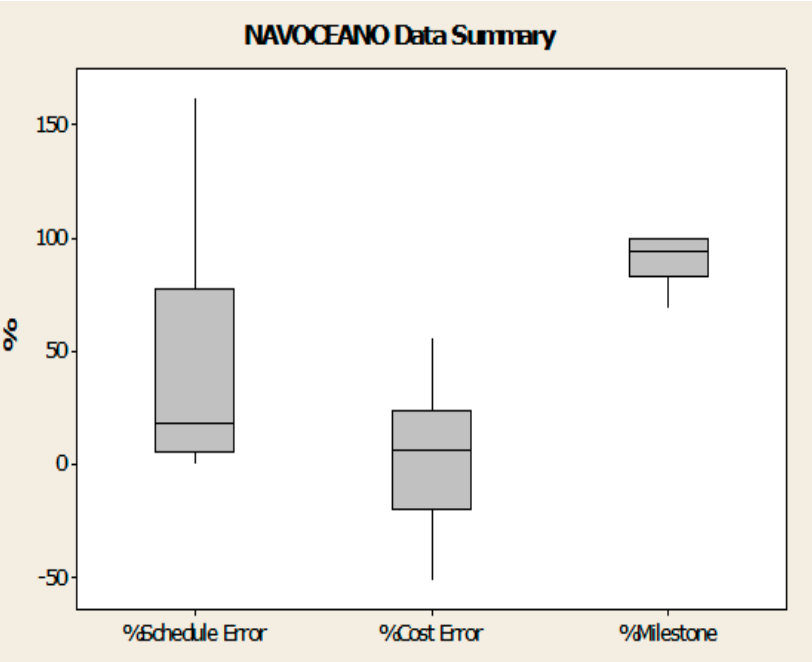


Figure 5: NAVOCEANO Summery Results

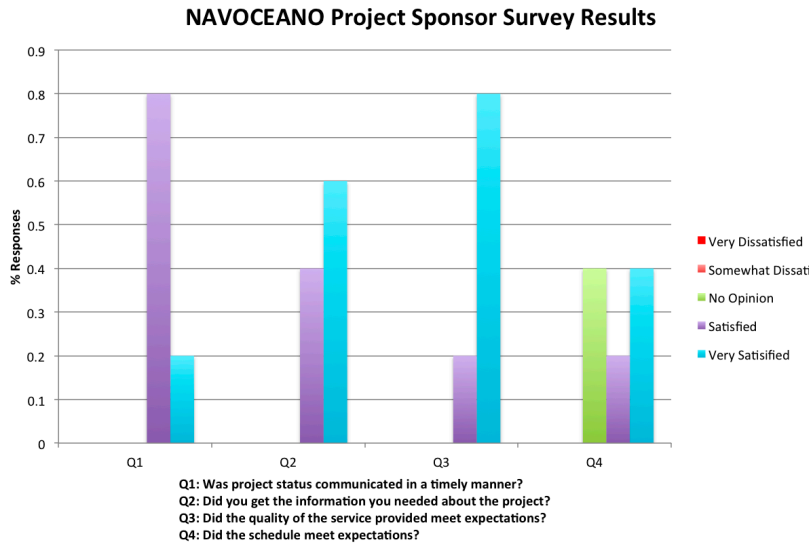


Figure 6: NAVOCEANO Project Sponsor Survey Results

The Naval Oceanographic Office Experience Using Task Time

In 2010 the Naval Oceanographic Office (NAVOCEANO) asked us to provide some support for their measurement program. NAVOCEANO is responsible for providing oceanographic products and services to all elements of the Department of Defense. As such they are operationally focused, and they focus less on development and engineering than many of the other commands. Their primary mission is to collect data through production and analysis to provide warfighters with the best available knowledge of the maritime battle space [11]. Thus, most of their staff consists of specialized knowledge workers [12].

Two of the key objectives they hoped to achieve by improving their measurement program were to provide more accurate estimates and to implement more precise status tracking. Given these objectives, we examined the mechanisms available for planning and tracking product development work and began to define the measures necessary to meet their requirements. We found that understanding and accurately tracking their "time" measure was vital to meeting their goals.

NAVOCEANO established a broader process improvement initiative in 2010, and one of their goals was to improve the organization's ability to plan, monitor, and control projects within the organization, specifically ones that crossed multiple organizational boundaries. As one of their managers put it, "We do a pretty good job of meeting our commitments within our individual silos, but when we try to work across silos we spend a lot of time and don't always deliver." Realizing that traditional command and control techniques were not optimal for certain types of work and based on past experience using the SEI's Team Software Process (TSP) for software engineering development [13], they decided to work with the SEI to more broadly apply the task-time-based, team-focused planning and tracking techniques to non-software teams within NAVOCEANO. Their approach included developing training specific to general knowledge workers, which was taught as just-in-time training for projects using the task-time approach. They also established a process working group to identify projects and support the transition.

In addition to learning what task time was, the knowledge workers were taught a systematic planning and tracking framework. The framework involved defining a project's scope or requirements, and then breaking down the requirements into manageable parts to create a conceptual design. The conceptual design was then organized in a WBS.

Once the work packages were defined, tasks could be determined. The tasks were based on the processes selected or created to produce the work packages. The work was then estimated. Historical data from past projects, contained in an effort database, and resource availability for the current project were used in the planning framework to develop the effort estimates and schedule.

During a three-year period NAVOCEANO established 12 different projects, which in total consisted of 23 "cycles." A cycle is the project's detailed planning horizons, which ran between 3 and 23 weeks and averaged 16 weeks in duration. All of the projects consisted of full-time employees working on the assigned project on a part-time basis. Every cycle included deliverables and milestones. Most projects consisted of members from across multiple organizational silos and ranged in size from 6 to 18 members, with an average of 9 members.

Each project was planned and executed incrementally. During initial planning, near-term tasks were planned in detail and distant tasks were planned at a higher level. These detailed plans were used to guide the work of individuals and allow them to precisely track their progress. As the measurement framework was implemented, managers for cross-silo projects found they had a new ability: to obtain accurate data about the status of their projects.

Figure 5 provides a summary of their results. It shows that they had a tendency to deliver late about 19% of the time, which was mostly due to the part-time nature of the work and the changing demands from the operational environment. A big difference from previous efforts was that managers now had

the data to be able to communicate the state of the project in a quantitative, accurate, and defensible way. This allowed management to be more proactive in negotiating resources across silos and setting expectations with customers.

Had they been able to spend the time on the project they had planned, they would have probably delivered early. This is due to the fact that the teams had a tendency to underspend, which is shown in the 7% Cost Error. That said, 95% of the projects objectives were met, which was a huge improvement from past experiences.

At the end of each project the project's sponsors were surveyed. Figure 6 shows the sponsors' evaluations of the projects. In general the survey shows that the projects were able to communicate their status in a timely manner, meet the sponsor's information needs and schedule expectations, and still deliver quality services.

Conclusion

Understanding the dynamics of your system development process in a quantifiable way increases the accuracy of estimates, the reliability of status reporting, and perceived customer quality and satisfaction. The key to any fundamental knowledge of system development dynamics and economics is an accurate, reliable, and repeatable measure of time. Using experiential and quantitative data, this paper demonstrates that the most common means of capturing time for system development projects is often inaccurate for the intended purposes. The use of task time with a well-defined process value chain improves accuracy and reliability while providing what customers desire most: quality products on schedule and within budget.

REFERENCES

1. A Guide to the Project Management Body of Knowledge – Fourth Edition, 2008. Project Management Institute, An American National Standard, ANSI/PMI 99-001-2008.
2. Cabri, Anthony, and Mike Griffiths. "Earned Value and Agile Reporting." Web. 30 May 2014.
3. Corovic, M. By Radenko. Why EVM Is Not Good for Schedule Performance Analyses (and How It Could Be...). Web. 30 May 30.
4. Solomon, Paul. "Perfomance Based Earned Value." CrossTalk, 2005: http://www.crosstalkonline.org/storage/issue-archives/2005/200508/200508-Solomon.pdf
5. "Scrum (software Development)." Wikipedia. Wikimedia Foundation, 30 May 2014. Web. 30 May 2014.
6. "What Is a Story Point ?" Web log post. AgileFaq. Web. 30 May 2014.
7. Berteig, Mishkin. "Agile Advice." Agile Advice. N.p., 29 July 2013. Web. 30 May 2014.
8. Moser, R., Pedrycz, W., Succi , G. 2007. Incremental effort prediction models in Agile Development using Radial Basis Functions. Proc. 19th International Conf. on Software Engineering & Knowledge Engineering (Boston, MA, USA, July 9-11, 2007), SEKE'07, pp. 519-522.
9. N.C. Haugen, "An empirical study of using planning poker for user story estimation," Minneapolis, MN, Agile 2006 Conference Proceedings
10. Javdani, T., Zulzalil, H., Ghani, A., Sultan A., Parizi, R. 2012. "On the Current Measurement Practices in Agile Software Development," IJCSI, Vol. 9, Issue 4,No 3, July 2012
11. Naval Oceanographic Office. US Navy, n.d. Web. 30 May 2014. http://www.public.navy.mil/fltfor/cnmoc/Pages/navo_home1.aspx
12. Rouse, Margaret. "Knowledge Worker." Web blog post. What Is ? Web. 30 May 2014.
13. Battle, Ed. Leading & Learning – Using TSP at the MSG Level, Sept. 2009. Web.
14. "Partners - Find or Become an SEI Partner or CMMI Institute Partner." Partners Clearmodel. N.p., n.d. Web. 30 May 2014.
15. Kasunic, Mark. SEI Interactive Session: Empirical Study of Software Engineering Results, September 18, 2013. Web.

Acknowledgements

Many people have participated in the work that led to this article, but we would like to give special thanks to Mark Kasunic, William Nichols, and James Over. Without their hard work analyzing all the data, these finding would not have been as compelling or insightful. We would also like to thank Erin Harper for all her technical editorial improvements, which enhanced the overall readability.

The data reported in figures 1, 2, 3, and 4 comes from 113 different software development projects, which reported their results to the SEI through its Partner Network [14]. The projects started between July 2000 and July 2012 and had an average project duration of 119 calendar days, with a team size of less than 20 people. All the projects in this data set used the same definition of task time and a planning framework [15].

ABOUT THE AUTHORS

Timothy A. Chick is a Senior Member of the Technical Staff at Carnegie Mellon University's Software Engineering Institute (SEI). He is a certified PMP, and Scrum Master. He also holds a Lean Six Sigma Black Belt certification and was a certified CMMI-DEV/SVC instructor. Prior to the SEI, he worked for Naval Air Systems Command, as the Software Acquisition Lead for the Fire Scout and former software project manager for the E-2C Hawkeye Program.

Lana Cagle is the Quality Advisor for the Systems Integration Division of the Naval Oceanographic Office, Stennis Space Center, Mississippi. She has worked as a process improvement change agent for 20 years, serving in a lead capacity since 1997. For the last four years she has been assisting in transitioning process improvement methods to the larger organization. She is a certified Team Software Process Mentor Coach and a Lean Six Sigma Green Belt.

Dr. Gene Miluk is currently a Senior Member of the Technical Staff at the Software Engineering Institute (SEI), Carnegie Mellon University. For the past 20 years Gene had been working with SEI client organizations undertaking software process improvement, software acquisition improvement and technology transition. He is a TSP instructor and SEI Certified Team Software Process Mentor Coach . Gene is also a Six Sigma Black Belt, a certified SCRUM Master and certified Project Management Professional.

Hybrid-Agile Software Development

Anti-Patterns, Risks, and Recommendations

Paul E. McMahon, PEM Systems

Abstract. Many organizations are driving toward increased agility in their software development practices. However, due to various constraints (e.g. project size, team physical distribution, compliance requirements, technical complexity) a pure Agile approach is not always feasible. This leads to what today is commonly referred to as a "hybrid-agile" [1, 2] approach. Using a hybrid-agile development approach requires that organizations think carefully about process tailoring and metrics decisions to ensure they stay aligned with their performance goals.

The purpose of this paper is to provide motivation for hybrid-agile approaches, identify common challenges hybrid-agile projects face today, and to provide recommendations that can help teams using a hybrid-agile approach reason through their challenges leading to more effective process tailoring and metrics decisions. Anti-patterns and related risks commonly observed today on large complex hybrid-agile efforts are identified and employed as an aid in demonstrating the reasoning process.

Introduction

The Goal - Question – Metric (GQM) approach to determine optimum metrics [3] has long been accepted as the gold standard for metrics identification. With GQM you start by asking:

"What is our goal?"

Then you formulate a set of questions that can help you assess how well you are doing toward achieve the goal. This leads to a set of metrics to collect that will help answer those questions.

What is the goal of measurement on most software development projects?

First, most organizations measure to understand where they are with respect to where they planned to be so that corrective action can be taken when necessary. An example could be to add resources when your measurements indicate you are behind schedule.

A second reason many organizations take measurements is to help improve performance. These improvements could be on the next project, or the next iteration of the current project.

Understanding your goal is important because the best measurements to collect in a given situation depend on what you are trying to achieve. Let's now look closer at what organizations are trying to achieve when using a hybrid-agile approach.

Why Hybrid-Agile?

In today's rapidly changing, competitive, fast- paced world organizations need to be able to get product to market faster, and they need to be able to respond rapidly with product changes to address changing customer needs. However, many of these same organizations also live in highly regulated environments where compliance to standards is equally critical to business success.

Part of being effective in responding to change and complying with regulations is being able to predict how long all the work will take to get new or modified features ready for stakeholder use while ensuring no critical steps are bypassed.

To predict we must be able to estimate the work effort, but unfortunately accurate software cost/schedule work estimation has alluded the software community since the early days of software development. This observation is not new. Over 15 years ago Tom Demarco and Tim Lister explained the problem as follows [4]:

"Most software managers do a reasonable job of predicting the tasks that have to be done and a poor job of predicting the tasks that might have to be done."

While the problem of predicting the tasks that might have to be done has been difficult for the software community for a long time-- and especially difficult on large complex software efforts-- the recent agile movement has given us some new ways to think about, predict, and measure progress.

What is Different in How Agile Projects Measure Progress?

A major reason why the agile movement continues to gain steam even after ten years is the recognition that many stakeholders do not fully understand the requirements for their desired software system at the start of their endeavor. Furthermore, most professionals involved in software development today know we need a better way to deal with rapidly changing requirements even late in the project.

Agile practices emphasize the need for development teams to work closely with stakeholder representatives to uncover the stakeholder's real needs and manage the resultant work from the start of the endeavor through its completion.

Nevertheless, as organizations have tried to implement these promising new agile practices-- particularly in large, constrained environments-- the resultant agile-tailoring's have led to difficulties and a number of commonly observed anti-patterns.

Anti-Patterns, Observations, Risks and Recommendations

In this section seven anti-patterns commonly observed on hybrid-agile efforts are identified and discussed. For each anti-pattern, observations, risks and recommendations are provided demonstrating a reasoning process[1] [6] that could help organizations using a hybrid-agile approach make better process tailoring and metrics decisions.

Anti-Pattern One: *Creating an "aggregated" team velocity metric and using it to predict and drive progress*

Observations, Risks and Recommendations

Team velocity is a metric that is used to measure the amount of work an agile team estimates it can complete in the next Sprint[2] [7] based on recent team performance.

Most organizations looking to increase agility today understand the importance of creating small Scrum[4] teams even on large complex efforts [8], but many don't yet understand the importance of empowering these small teams with the responsibility to measure themselves.

Often what we see on large hybrid-agile projects is the velocity being set at a high level in the organization and given to the small teams, rather than allowing the small teams to measure their own velocity and then set the target work for the next iteration based on their own recent past performance.

The most accurate estimates of effort to complete work are based on recent performance of each specific small team doing the work. Each small team should measure its own velocity. Organizations should anticipate varying team velocities for each small team.

Defining team velocity from the top defeats the purpose of the velocity metric. When used appropriately Scrum teams should get better at predicting the work that can be accomplished in each sprint as they progress from sprint to sprint learning from their own velocity.

It is recommended that you let your project manager know where he or she can see each small team's velocity measurements in a place where it is visible to the whole team, such as on the wall in a room where each small team holds their daily standup meetings[5] [7]. This will keep the true velocity visible to the team and will reduce the temptation for intermediate and senior managers to use this metric inappropriately.

Anti-Pattern Two: *Telling the team to work harder to improve performance.*

Observations, Risks and Recommendations

When progress is not being achieved per the plan too often the response has been to tell the team members to work harder, such as by putting in overtime hours, to improve performance. While overtime can help to improve performance in isolated instances, excessive regular overtime risks team burn out, and it does not get to the underlying root cause. Furthermore, in these situations, team members often respond by cutting corners and not following their agreed to way of working (e.g. reducing their planned testing or peer reviews). Ultimately this leads to more latent defects and longer schedules.

One approach that can help is to encourage your teams to use the story point efficiency metric. Story point efficiency [6] is defined to be the ratio of the estimated time to complete a user story[6] [9] divided by the actual time it took. This metric can help teams quickly identify problem areas in their requirements/user stories and investigate those problems in a timely way to learn what is hindering the team from achieving their estimate. It is recommended that you keep each small team's story point efficiency measures visible to the whole team, and encourage them to use this measurement data to continually improve their velocity.

When teams don't hit their estimates it usually isn't hard for them to figure out why, if you give them the time to investigate the situation right when the problem is happening. This is also the best time to resolve the problem because it doesn't delay the resolution to the next project, or even the next iteration of the current project. It can help the team's performance right when the problem is happening by raising the visibility of the problem to the right level and getting it resolved in a timely way.

Agile teams must own their practices and their continual improvement

A key strength teams gain from using the story point efficiency metric is that it gives teams the data they need to put improvements in place. This includes identifying weaknesses that have slowed them down in the past and putting improvements, such as better checklists, in place to catch similar potential problems in the future. If they still don't see velocity improvements after implementing changes, it is likely they are not putting the right improvements in place to resolve the problem. Agile teams are self-directed which means it is the responsibility of the team members to identify and resolve performance issues.

Agile practices can work only if the agreed way of working within the organization empowers the team to make timely changes necessary to improve without going through bureaucratic approvals outside the team.

Anti-Pattern Three: *Failure to actively manage risk exposure at the small team level.*

Observations, Risks and Recommendations

Today on many large complex efforts we see risk management carried out at the senior management level, and not effectively implemented deep into the organization. A key agile principle is to take on risky work early driving overall project risk down as the project proceeds [10]. Risks should be actively identified and managed at the small team level, and then rolled up consistently to the higher level—not the other way around. When small agile teams plan their work for each sprint they actively discuss risks to ensure they are driving the risks down early.

It is recommended that you keep the risk trend visible to the team to help the team discuss the right issues when making key work related decisions for the next sprint. The risk assessments at the small team level should be rolled up in a consistent way to provide an accurate overall project risk assessment.

Anti-Pattern Four: *Failure to actively manage stakeholder involvement and stakeholder representation competency.*

Observations, Risks and Recommendations

Too often we see organizations trying to increase their agility at the development team level, but failing to recognize that agility requires changes in the behavior of the stakeholder community as well. To gain the benefit of agility stakeholder representatives with the right competencies must be assigned, agree to their responsibilities, and be given the time to carry out their responsibilities in a timely fashion.

Too often we see the stakeholder representatives that are assigned are people with inadequate competency to carry out this critical role. The stakeholder representation role requires people with the ability to gather, communicate and balance the needs of other stakeholders, and accurately represent their views—not just their own views. [6]

Anti-Pattern Five: *Using "proxy releases" rather than formally selling-off products incrementally.*

Observations, Risks and Recommendations

On large projects, due to various constraints, it is often the case that products cannot be made operational every sprint. Therefore large projects typically conduct multiple sprints leading to each release. On very large projects a release could be every 6 Sprints (or every 6 months). However, these releases should be real "sell-offs" of product to the real stakeholders/customer.

An anti-pattern commonly observed is for these releases to be conducted as "proxy releases." By "proxy release" I mean a release which is not a formal/official sell-off to the real stakeholders. Rather it comprises some level of testing and demonstration in the presence of someone representing the stakeholders, but not authorize to formally accept the product.

The rationale provided for "proxy releases" often relates to constraints such as inability to operationally deploy partial functionality, and/or the lack of availability of key stakeholder representatives. However, the fact that the product cannot be deployed on short cycles should not get in the way of stakeholders being involved, responsible, and given the time to collaborate and accept functionality on short cycles.

The risk with "proxy releases" is that too often, when a real product acceptance is not conducted, we see the development teams and stakeholder representatives just "going-through-the-motions" and not rigorously testing against the agreed to requirements/features allocated to that release.

Often these "proxy" events do not have stakeholder representatives that are authorized, and knowledgeable to provide real answers and to conduct a thorough review of the product. Ultimately this leads us back to the traditional integration and test problems late in the project and the goal of getting product and product changes to the customer rapidly is not achieved.

I have probably heard most, if not all, of the reasons why "we can't" sell-off incrementally.

Yes, it takes authorized and knowledgeable stakeholder representatives from the customer side.

Yes, the team needs to complete all their work that they have committed to following their agreed way of working including thorough testing.

Yes, all the key stakeholders need to agree that the software system is worth making operational.

But these are the points why agile works. A key agile practice is to get the work done in short increments and get product to customer sooner which also reduces the risk of late surprises, extended schedules, and cost over-runs.

Why is this so important?

If you aren't measuring features accepted by the customer incrementally, you haven't really understood why agile development can help you achieve your ultimate goal, and you probably won't.

Anti-Pattern Six: *Essentially traditional development with a few "agile practices" sprinkled in to make the project appear "agile."*

Observations, Risks and Recommendations

I have observed many large projects that claim to be using agile methods to be essentially traditional development with a few agile practices being conducted by the development teams. When I have looked close at these projects that are essentially traditional I often have found a "business as usual" attitude even though they may use the "agile" buzzword. The risk in this approach is that it is highly unlikely they will ever experience the real potential value of agility.

One way to counter this risk is to use the "how agile are we?" metric. The "how agile are we?" metric gives you an indication of the degree of agile practice adoption by your team. There are numerous ways to measure how agile you are, many of them by survey. Examples include the Shodan Adherence Survey and the ComparativeAgility Assessment [11].

For many years I did not like the "how agile are we?" metric because my view was it didn't matter how agile an organization

How Have We Traditionally Measured Progress?

Traditional approaches to measure progress are well known along with their shortcomings. The most widely used approach on large complex software efforts is Earned Value Management (EVMS) [5]. The fundamentals of EVMS include breaking the work down into small pieces, estimating the cost of each piece, and monitoring actual expenditures against an agreed to baseline plan.

One weakness with EVMS rests in the assumption that we know all the work that must be done when we plan it, and we know how long each piece will take to complete. Another weakness is the fact that even if we get all the identifiable work done it doesn't necessarily mean the stakeholders will be satisfied that the software system is ready for use.

Traditionally many organizations that use EVMS have addressed these weaknesses by applying risk management practices. However, the way these practices have been carried out in many organizations have failed to adequately address these weaknesses in a timely way.

For example, in one of my client organizations I heard that when a potential risk is raised so much collateral evidence had to be gathered that effectively they had to prove the risk was already a problem before the risk board would accept it.

In another client organization I heard that no one ever raises a schedule risk because the culture in the organization had become one that just accepted the fact that all good schedules had to be aggressive and risky given today's business climate. So risk in a schedule was no longer perceived as a risk that needed to be managed. Rather it was now perceived as an acceptable part of the normal way of working due to the aggressive business climate.

was, it was more important that they had the "right level of agility" given their situation. But I have since discovered that the "how agile are we?" metric can give an organization an early indication of the likelihood that their "brand-of-agile" will help them achieve their performance goals. This metric can give you a good idea early in your project of the level of commitment your organization really has to agile practices.

A Simple Way to Measure "How Agile Are We?"

While hybrid-agile projects can vary across a continuum they could be characterized at the extremes and in the middle through a simple three level scale as follows:

Case 1: Fully Committed

Fully committed means the project has adopted an agile approach across the entire program including systems engineering, test, and stakeholder participation. Characteristics of the fully committed case include:

- Requirements expressed in user story form in a backlog
- The backlog provides the one and only list of requirements
- The backlog is refined, and reprioritized at the start of each sprint
- Systems engineering and test is integrated into the small scrum teams
- Authorized stakeholders provide product owner role attending sprint demos accepting product deliverables at sprint level

Case 2: Hybrid Agile/Traditional

There could exist varying levels within this case, but the typical characteristics include:

- High level requirements completed early in project by systems engineering and allocated to multiple sprint releases
- Lower level requirements generated as user stories and managed by small scrum teams through backlog
- Some level of stakeholder representative involvement at sprint demos
- Multiple sprints lead to release sell-off at sprint release level with authorized stakeholder representatives present

Case 3: Essentially traditional program with a few agile practices conducted by the software teams.

Characteristics of this case include:

- Requirements developed up front by systems engineering
- Requirements developed and managed traditionally in tool such as DOORS[7]
- Software scrum teams break high level requirements down into user stories and manage through backlog within sprints
- Multiple sprints lead to release sprint
- Sprint demos may be conducted to get early feedback at release sprint, but no or minimal acceptance/sell-off of product at sprint release level
- All or majority of requirements tested at end of project through traditional integration and test/acceptance

Risks Associated with the Three "How agile are we?" Hybrid-Agile Cases

Case 1: Requires significant investment by customer to train and commit customer personnel to participate regularly in project activities throughout lifecycle.

Case 2: When choosing a hybrid agile/traditional approach, if personnel involved lack agile experience, there is risk of poor tailoring decisions (e.g. practices, metrics) leading to failure to achieve the intended agile benefits.

Case 3: A major value in using agile approaches is to improve contractor-stakeholder communication and reduce the risk of unexpected latent defects and cost/schedule overruns. This value is unlikely to be achieved in Case 3 since the authorized and knowledgeable stakeholder representatives are not engaged throughout the endeavor.

Anti-Pattern Seven: *Using the requirements volatility measure inappropriately to control scope creep.*

Observations, Risks and Recommendations

Requirements volatility is a common metric that has been used traditionally to manage requirements scope creep. Often this metric continues to be used in an inappropriate way on hybrid-agile endeavors. When using agile practices trying to control project cost and schedule by minimizing requirements changes can conflict with recognized best agile practices. With agile practices you collaborate with your customer to provide the best value for the available resources. Therefore as the project proceeds it may be fine for changes to occur in priority and content of the requirements backlog. When you move to an agile approach continual and close collaboration with the customer trumps controlling requirements volatility.

If you have an effective collaborative relationship with your customer it can be beneficial to allow requirements volatility (e.g. requirements changes late). Requirements stability isn't the end goal. Stakeholder satisfaction is.

Summary

The purpose of this paper has been to provide motivation for hybrid-agile approaches, identify common challenges hybrid-agile projects face today, and to provide recommendations that can help teams using a hybrid-agile approach reason through their challenges leading to more effective process tailoring and metrics decisions. Anti-patterns and related risks commonly observed today on large complex hybrid-agile efforts were also identified and employed as an aid in demonstrating the reasoning process.

ABOUT THE AUTHOR

Paul E. McMahon, Principal, PEM Systems (www.pemsystems.com) has been an independent consultant since 1997. He has published more than 45 articles and multiple books including "15 Fundamentals for Higher Performance in Software Development." Paul is a Certified Scrum Master and a Certified Lean Six Sigma Black Belt. His insights reflect 24 years of industry experience, and 17 years of consulting/coaching experience. Paul has been a leader in the SEMAT initiative since 2010.

E-mail: pemcmahon@acm.org

NOTES

1. The Essence framework was used as a guide in developing the referenced "reasoning process"
2. The heart of Scrum is a Sprint. A time-box of one month or less during which a "done", usable, and potentially releasable increment of software is created.
3. Experience has proven that effective Scrum teams should be no larger than 8-10 people
4. Scrum is a framework for developing and sustaining complex products.
5. A daily standup meeting, also referred to as a daily Scrum, is a time-boxed 15 minute meeting each day where the team synchronizes activities for the next 24 hour day.
6. User stories is one way to describe requirements when using agile software development
7. DOORS is a commercially available requirements management tool

REFERENCES

1. Ambler, Scott, Lines, Mark, Disciplined Agile Delivery, IBM Press, 2012
2. McMahon, Paul E., "Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement, Addison-Wesley, 2011
3. Humphrey, Watts, "A Discipline for Software Engineering", Addison-Wesley, 1995
4. Demarco, Tom, Lister, Timothy, "Waltzing with Bears", Dorset House Publishing, 1995
5. Solomon, Paul J., "Practical Performance-Based Earned Value", Crosstalk, The Journal of Defense Software Engineering, May, 2006
6. McMahon, Paul E., "15 Fundamentals for Higher Performance in Software Development", PEM Systems, http://amzn.com/099045083X. July, 2014
7. Sutherland, Jeff, Schwaber, Ken, "Scrum Guide-- The Definitive Guide to Scrum: The Rules of the Game", July 2013
8. McMahon, Paul E., "Lessons Learned Using Agile Methods on Large Defense Contracts," Crosstalk, The Journal of Defense Software Engineering, May, 2006
9. Cohn, Mike, "User Stories Applied: For Agile Software Development, Addison-Wesley, 2004
10. McMahon, Paul E., "Defense Acquisition Performance: Could Some Agility Help?", Crosstalk, Journal of Defense Software Engineering, February, 2009
11. Cohn, Mike, "Succeeding with Agile: Software Development Using Scrum, Addison-Wesley, 2009

CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for the areas of emphasis we are looking for:

Software - A People Product
Jan/Feb 2016 Issue
Submission Deadline: Aug 10, 2015

Cyber Workforce Issues
Mar/Apr 2016 Issue
Submission Deadline: Oct 10, 2015

Integrated Warfighting Capabilities
May/Jun 2016 Issue
Submission Deadline: Dec 10, 2015

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <www.crosstalkonline.org/submission-guidelines>. We accept article submissions on software-related topics at any time, along with Letters to the Editor and BackTalk. To see a list of themes for upcoming issues or to learn more about the types of articles we're looking for visit <www.crosstalkonline.org/theme-calendar>.

Using Hubs and Cyclicality to Relate Software Architecture and Quality

Tyson R. Browning, Texas Christian University
Jürgen Mihm, INSEAD
Manuel Sosa, INSEAD

Abstract. Recent studies of 17 open source applications have shown two salient characteristics of software architecture, hubs and cycles, to have strong relationships with software quality. Components in cycles were significantly more likely to contain bugs than other components, and architectures utilizing hub components tended to have fewer defects. Identifying hub and cycle components should therefore help software developers focus their quality control efforts.

Software architecture matters. This is not a new revelation, since past studies have noted the significance of architectural characteristics such as coupling, cohesion, and modularity [e.g., 1, 2, 3]. Yet, our recent studies [4, 5] demonstrate that two additional characteristics, hubs and cycles, can also have a major impact on quality. These studies empirically link hubs and cycles to quality and identify particular ways these relationships play out. We found that components involved in cycles were significantly more likely to contain bugs and that architectures utilizing hub components were significantly less so.

Software architecture pertains to the structure of components (e.g., a Java class) and their dependencies (e.g., function calls). A cycle occurs when a component indirectly "calls" itself via a chain of other components. (Architectural cycles occur at a much higher level than the cycles measured by conventional metrics such as cyclomatic complexity.) A hub component is a relatively highly-connected component in the architecture—i.e., one with many dependencies linking it to other components.

Cyclicality Study

Our study of architectural cycles included 28,394 observations of 7,103 components across 111 major releases (versions) of 17 open source applications (an average of 6.5 versions of each application and 256 components per version) from the Apache Software Foundation in mid-2008. To collect architectural data, we downloaded the precompiled version (JAR file) of each major release of each application from the Apache archives or the application's website and used LDM, a tool developed by Lattix, Inc. (www.lattix.com), to build (instantaneously) a design structure matrix (DSM) representation from the source code and extract the module membership of components. To collect data on quality, we developed web crawlers to extract (patched) bug information on each component in each version from Apache's *Bugzilla* and *Jira* bug-tracking systems and SVN repositories. We also used each version's source code and release notes to determine a number of control variables. See [5] for further details about the procedures of the study.

Two examples of the DSM representations are shown in Figures 1 and 2. The DSM is a square matrix (equal number of rows and columns) where the diagonal cells represent an element (here, a component such as a Java class) and off-diagonal cells represent directed dependencies of the component in column *j* on the component in row *i*. The DSM in Figure 1 shows the application Ant (version 1.1) with 62 components and 195 directed dependencies. (It is therefore one of the smaller applications in our study.) This DSM is called "flat" because it does not show the arrangement of the components into modules. Rather, the components in this flat DSM have been ordered to place as many of the dependencies as possible below the diagonal. Any dependencies remaining above the diagonal are brought as close to the diagonal as possible, thereby grouping any cyclical components as closely together as possible (as indicated by the larger block shown along the diagonal). Ant 1.1 has only one cycle containing five components. The components in this group are designated as "in-cycle" components.

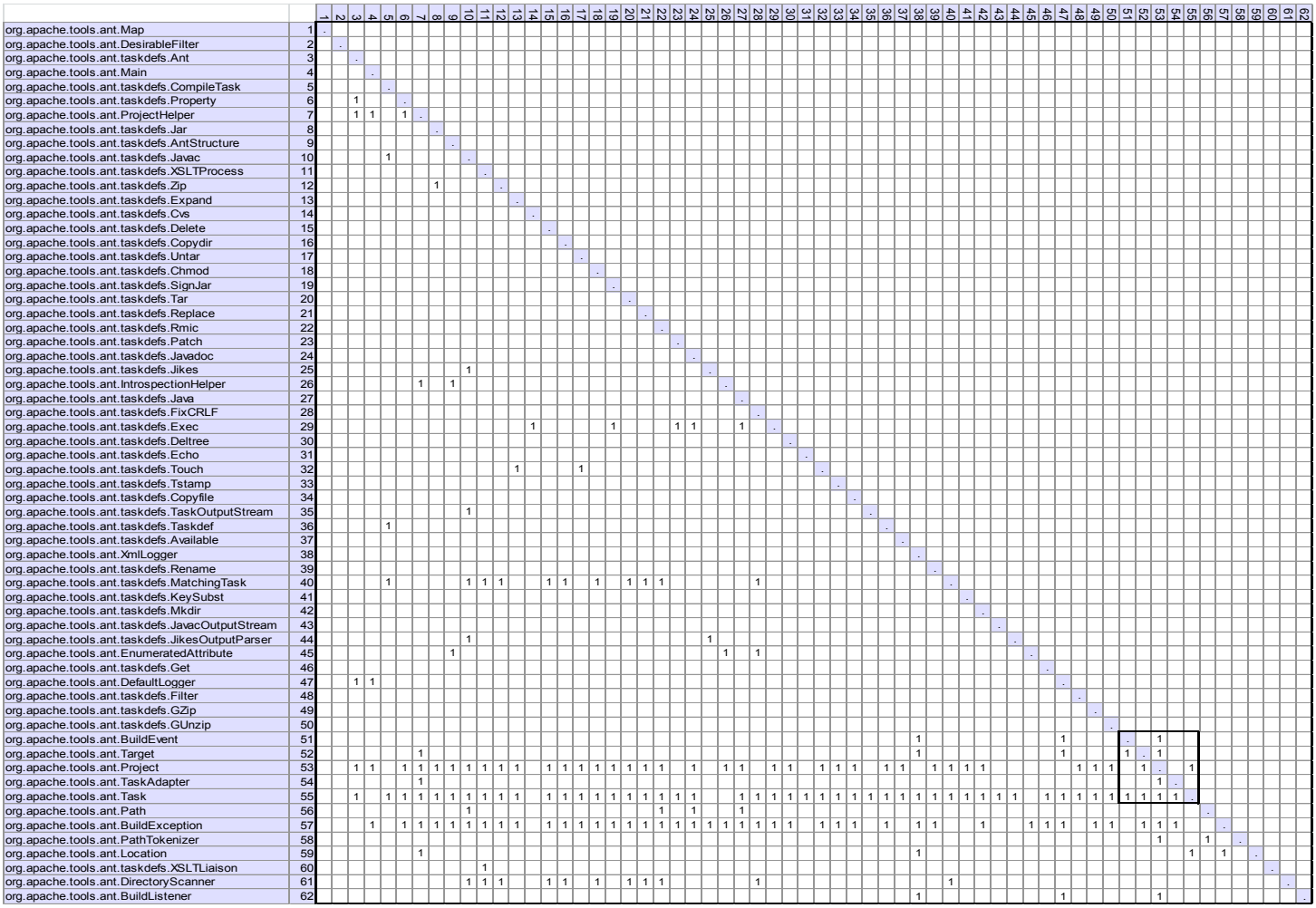


Figure 1: Flat DSM representation of Ant (version 1.1).

The DSM in Figure 2 shows the same application but with the rows and columns sorted according to the nested subdirectory structure of the code—i.e., showing the assignment of the components to the hierarchical modules designated by the architects. The modules are indicated by the differentiated shadings and blocks along the diagonal, as well as by the labeling to the left of the matrix. In this example, the five cyclical components reside inside a single module, but this is not always the case. Overall, DSMs allow us to visualize the dependency structure (or designed structure) of the architecture in various ways that help to highlight important patterns such as modularity and cycles.

The main dependent variable in our study of cyclicality was the number of bugs in a component, and our independent variables were component fan-in and fan-out, whether or not the component was part of a cycle, and the number of modules involved in each cycle. Component fan-out is the fraction of the other components on which the component depends, either directly or indirectly. Component fan-in is the opposite, the fraction of other components that depend on a component. Fan-in and fan-out have been studied extensively, with the latter having been shown to be a significant architectural factor.

We also included a number of system-level control variables for each version of each application, including the following: number of new and modified features since last major release, number of unpatched bugs, application age, time since last major release, application size (KSLOC), average cyclomatic complexity of all components, average fan-in and fan-out visibility of all components, and breadth and depth of the hierarchical module structure. Our component-level controls included the following for each version of each application: component age, number of non-bug changes to the component, cumulative number of changes associated with the component prior to this version, cumulative number of committers associated with the component prior to this version, cumulative number of authors associated with the component prior to this version, average cyclomatic complexity of the component's methods, and component size (KSLOC).

Our cyclicality study yielded three main findings. First, cyclicality really hurts quality. Even when controlling for the aforementioned factors, components involved in cycles exhibited a significantly higher number of defects. Moreover, this effect increased with the size of the cycle: the larger the cycle, the greater the expected number of bugs associated with each component therein. These effects were statistically significant at the 99% confidence level. The best fitting random coefficient regression model indicated that an in-cycle component had, on average, 58.3% more defects than a non-cycle component.

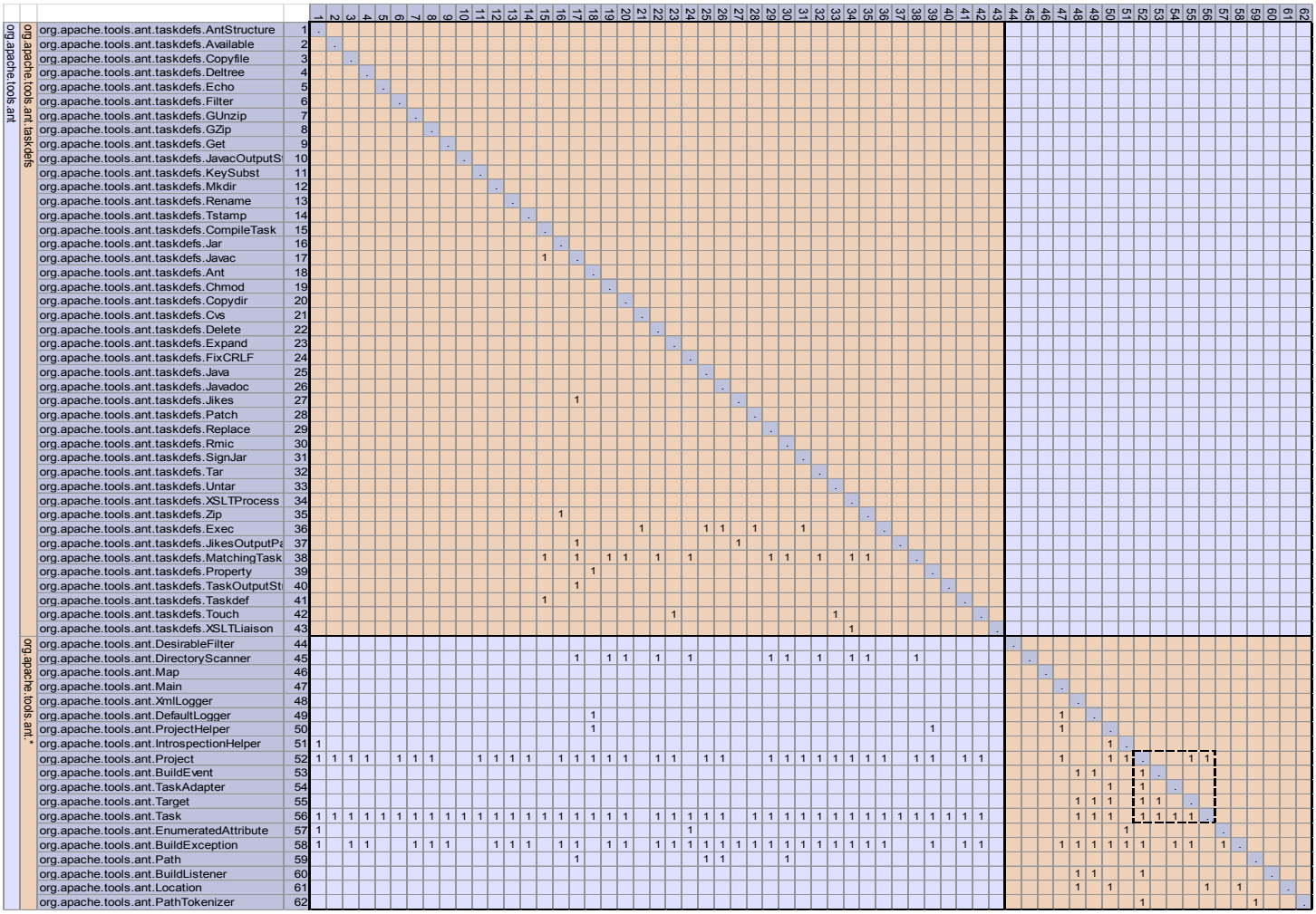


Figure 2: Hierarchical DSM representation of Ant (version 1.1).

In comparison, an increase of a single standard deviation in a component's fan-out corresponded to 35.3% more defects. Furthermore, in an analysis of a subsample of 6,064 components that were matched and balanced with respect to fan-in and fan-out, the in-cycle components averaged 80% more bugs than the non-cycle components. Overall, these results suggest that the average effect of cyclicality is of the same order of magnitude as the effect of component fan-out (modularity).

Second, the number of defects exhibited by a cyclical component increased with its centrality in the cycle. To understand centrality, consider all of the paths that link back to a focal component in a cycle (touching another component in the cycle at most once). For any in-cycle component, at least one such circuit must exist, although some components will have more than one such path. Centrality is this number of circuits or paths. Central components will have to incorporate more potential changes propagated via their many connections to other components. Our study showed that increased component centrality correlated with more bugs. That is, components in two different cycles of the same size could exhibit different average levels of defects, depending on their centrality.

Third, failing to encapsulate cycles within a module hurts quality even more. The average number of defects exhibited by an in-cycle component increased with the number of modules involved in the cycle. In our sample 87% of the cycles spanned at least one module, which suggests that multi-module cycles are common and that, at least in an open source software development context, developers seemed to neglect the negative consequences of dealing with cyclical dependencies across modules, or were not aware of such cycles.

These results suggest some useful metrics and practices for software developers. System cyclicality indicates the percentage of components involved in cycles. It would be desirable to keep this number as low as possible, especially since the presence of cycles indicates one or more violations of the architecture's design rules. Cycles should be eliminated or at least reduced in size, especially when that reduction also reduces the number modules spanned by the cycle. At the component level, a component's membership in a cycle should be indicated, as well as its centrality in the cycle. Components in cycles, and especially those with the greatest cycle centrality, should receive extra attention and testing. Cycles can be broken by rerouting function calls and other dependencies. Cycles can be avoided in the first place by enforcing design rules that prohibit function calls to different modules, and by providing tools that show whether any potential function call would create or enlarge a cycle.

Hubs Study

Our study of hubs was similar to our study of cyclicality, although it looked at only 105 versions of 16 of the Apache applications used in the former study. See [4] for further details about the procedures of the study. Our dependent variable was the number of bugs in an application version, and our independent variables were the skewness of the application version's degree distribution and its fraction of hub components.

A component's degree is its number of incoming and outgoing dependencies (the sum of its in-degree and out-degree), and a version's degree distribution is essentially a histogram showing the number of its constituent components with various degrees. It is often useful to normalize the degree distribution by expressing its horizontal and vertical axes as percentages of the whole, and a further step can be taken to represent the cumulative version of the distribution. Figure 3 shows the cumulative, normalized, out-degree distributions for the 105 application versions in our sample. These distributions exhibit the Pareto principle in terms of a few components having a very large degree (the long tail to the right of the distribution) and the majority of components having a relatively small degree. Skewness measures this effect, where increasing (positive) skewness indicates increasing distinction among these two types of components (e.g., going from an "80-20 rule" to a "95-5 rule"). We developed a procedure for determining a degree threshold (which can differ for each application version) above which components can be identified as hubs. Note from Figure 3 the large variation in the fraction of hub components in different application versions. The average normalized degree of all components in our sample is 0.02, and the top 20% most-connected components have normalized degree 0.15 or greater. Thus, the normalized degree of a hub component (if 0.15 is used as the threshold) is more than seven times larger than that of the average component.

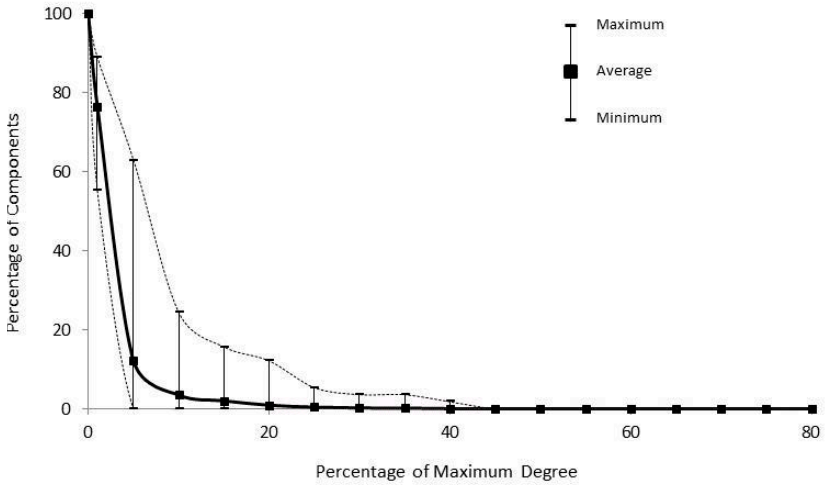


Figure 3: Cumulative, normalized, out-degree distributions for the 105 application versions in our sample (adapted from Sosa et al. 2011).

Two main findings emerged from our study of hubs. First, architectures containing hubs tended to be less defect-prone. The more right-skewed the degree distribution of an application version, the fewer defects it exhibited on average. In particular, increased out-degree skewness is significantly associated with fewer bugs (at the 99.9% confidence level).

Second, we compared different hub-designation thresholds (for both in-degree and out-degree) and found that there are optimal percentages of hub components that minimize the expected number of defects. A one standard deviation increase in the percentage of components with normalized in-degree greater than or equal to 0.15 correlated with 67% fewer defects, and in the case of out-degree it was 69% fewer defects. When the threshold is raised to 0.35, a one standard deviation increase in the percentage of components with normalized out-degree greater than or equal to the threshold correlated with 86% fewer defects.

However, it would be unreasonable to infer that merely increasing the percentage of hub components (e.g., by adding dependencies) would continually increase quality. We would expect points of diminishing and even negative returns. This is indeed what we found when we added a quadratic term to our regression models. When the in-degree threshold is set at 0.15 or greater for identifying hub components, then an application version is likely to have the fewest number of defects if 9.0% of its components are involved in hubs. The application versions in our sample had an average of 2.3% hub components at this in-degree threshold, with a maximum of 13.5%. Thus, to have fewer defects on average, an application version should have more hub components than did the average application version (9% vs. 2.3%), but applications with an even higher percentage of hubs (up to 13.5%) started to have a higher expected number of defects. We found similar results in testing higher in- and out-degree thresholds. Overall, we found substantial empirical evidence that systems with in- and out-degree distributions with "thicker than average" right tails (i.e., an above-average fraction of hub components) were more likely to have fewer defects, but that there are points past which this benefit subsides.

Implications

Previous research has focused on modularity as the most salient architectural characteristic in the architecture-quality relationship, yet these two recent studies have demonstrated that additional architectural characteristics, cycles and hubs, are similarly important. Although both of these studies used the same open-source applications, the cyclicality study focused on the component level while the hubs study focused at the application level. The analyses described herein can be performed on any source code, during development as well as after release. By analyzing their architectures in light of these characteristics, software developers should be able to increase their return on investment in verification and testing.

Since both studies addressed only open-source software, future research could compare with results from "closed-source" contexts. Next steps for this line of study also include the incorporation of cyclicality and hub metrics into software

architecture tools—a process which has already begun in the case of the LDM tool—and the linking of such metrics to further outcome data as part of a formal program of quality assessment.

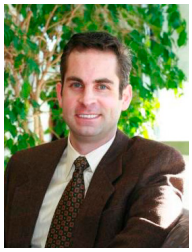
Acknowledgements

The authors are grateful to Neeraj Sangal and Frank Waldman of Lattix Inc. for use of the LDM tool and for insightful feedback on the research. The first author is grateful for support from a grant from the U.S. Navy, Office of Naval Research (grant N00014-11-1-0739). The last two authors are grateful for financial support from the INSEAD R&D committee (grants 2520-360 and 2520-519).

REFERENCES

1. K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Investigating Effect of Design Metrics on Fault Proneness in Object-Oriented Systems," *Journal of Object Technology*, vol. 6, pp. 127-141, 2007.
2. L. C. Briand, J. Daly, and J. Wüst, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 25, pp. 91-121, 1999.
3. R. Burrows, F. C. Ferrari, O. A. L. Lemos, A. Garcia, and F. Taïani, "The Impact of Coupling on the Fault-Proneness of Aspect-Oriented Programs: An Empirical Study," in *Proceedings of the IEEE 21st International Symposium on Software Reliability Engineering (ISSRE2010)*, San Jose, CA, 2010.
4. M. E. Sosa, J. Mihm, and T. R. Browning, "Degree Distribution and Quality in Complex Engineered Systems," *Journal of Mechanical Design*, vol. 133, pp. 101008, 2011.
5. M. E. Sosa, J. Mihm, and T. R. Browning, "Linking Cyclicality and Product Quality," *Manufacturing & Service Operations Management*, vol. 15, pp. 473-491, 2013.

ABOUT THE AUTHORS



Dr. Tyson R. Browning is Professor of Operations Management in the Neeley School of Business at Texas Christian University, where he conducts research on managing complex projects and teaches courses on project, operations, and risk management. He has previous work experience with Lockheed Martin and other companies and has also consulted for several organizations. He earned a B.S. in Engineering Physics from Abilene Christian University and two Master's degrees and a Ph.D. from MIT.

TCU Box 298530, Fort Worth, TX 76129
Web: www.TysonBrowning.com
E-mail: t.browning@tcu.edu
Phone: 817-257-5069

Jürgen Mihm is Associate Professor of Technology and Operations Management at INSEAD. His research interests include all management aspects of large engineering projects, with recent focus on understanding the management of design. Previously, he was a long-standing consultant with McKinsey & Company, Inc. in Frankfurt. He holds a doctorate in technology management from Wissenschaftliche Hochschule Koblenz (WHU) and a joint degree in business and electrical engineering (Dipl. Wirtsch. Ing.) from Technische Universität Darmstadt.

INSEAD, Boulevard de Constance, 77305 Fontainebleau, France
E-mail: jurgen.mihm@insead.edu

Manuel Sosa is Associate Professor of Technology and Operations Management at INSEAD and the Director of INSEAD's Heinrich and Esther Baumann-Steiner Fund for Creativity and Business. His research focuses on coordination and innovation networks in complex product and software development organizations. His work experience includes systems engineering in the petrochemical industry and computer-aided engineering software applications in the automobile and aerospace industries. He received his master's and doctoral degrees in mechanical engineering from MIT.

INSEAD, 1 Ayer Rajah Avenue, Singapore 138676.
E-mail: manuel.sosa@insead.edu

Testing Earned Schedule Forecasting Reliability

Walt Lipke, PMI® Oklahoma City Chapter

Abstract. Project duration forecasting using Earned Schedule (ES) has been affirmed to be better than other Earned Value Management based methods. Even so, the results from a study, employing simulation techniques, indicated there were conditions in which ES performed poorly. These results have created skepticism as to the reliability of ES forecasting. A recent paper examined the simulation study, concluding through deduction that ES forecasting is considerably better than portrayed. Researchers were challenged to examine this conclusion, by applying simulation methods. This paper uses real data for the examination, providing a compelling argument for the reliability of ES duration forecasting.

Background

Those of you who have submitted articles to the *CrossTalk* review process know that the critique and suggestions made consistently lead to a much improved article. Sometimes it doesn't feel like it, but it is nevertheless true. The most significant suggestion to my initial submission of this article was it needed more material on Earned Schedule (ES). The critical thought was readers would have to perform research of other articles and, possibly, books to gain much from the article as it was proposed. After brief reflection, I realized the reviewers were correct.

My anticipation in preparing the article was that only those familiar with Earned Value Management (EVM) and ES would be readers, and, thus, descriptions of these management methods was unnecessary. In taking this approach, I limited the usefulness and value of what I had to say. With the inclusion of foundational material, it is logical that reader interest is widely expanded.

However, with the addition of the descriptions, the article is constructed somewhat unconventionally. There is *background*, including the EVM and ES descriptions, followed then by the *introduction*. Having the fundamentals in-place, the *introduction* prepares the reader for the article's objective.

Earned Value Management

EVM is a management method succinctly depicted in figure 1. The method uses three measures: actual cost (AC), planned value (PV), and earned value (EV). PV is created from the cost estimates made for each task comprising the project. Using the schedule for the tasks, PV is accumulated at periodic time increments, concluding at budget at completion (BAC). This time-phased accrual of PV is commonly termed the performance measurement baseline (PMB); i.e., planned expenditure of the project budget. AC, of course, is the actual project cost accrued at the various status points, while EV is the accomplishment summed over the project tasks. EV for each task is measured in relation to its estimated PV; at task completion, the task EV will equal its PV, and at project completion the totals for EV and PV are equal to BAC.

The vertical dashed line in figure 1 represents a point in time when the project manager (PM) assesses performance of the project. From the three measures described, the performance indicators are derived:

Cost Variance: $CV = EV - AC$
Schedule Variance: $SV = EV - PV$

Cost Performance Index: $CPI = EV / AC$
Schedule Performance Index: $SPI = EV / PV$

When the difference for the variance formulas is positive, the project is doing well, and when it is negative further analysis is warranted. The indexes are indicators of performance efficiency. When their value is greater than 1.0 the project is doing well, while less than 1.0 indicates the need for improvement.

The indicators for cost are reliable and converge to the actual result at completion of the project. For example, if the project completed at more than its BAC by $1000, the computed CV would equal minus $1000. As well, if the project BAC equals $1000 and at completion AC is equal to $2000, CPI would

equal 0.5; i.e., the cost performance efficiency for the project is 50 percent (a very poor value).

The ability to compute CPI facilitates the capability to forecast the final cost for a project. The most used formula is

$$IEAC = BAC / CPI$$

where IEAC is the Independent Estimate at Completion.

Just as the indicators for cost always converge to the actual result, the forecast does, as well.

To this point, the discussion is fairly straightforward. However, there is a problem: the EVM schedule indicators do not exhibit reliable behavior. They do not converge to the actual result and during execution for late performing projects the indicators do not accurately portray performance. This characteristic for late performance has been observed as early as when the project is 50 percent complete. The reason this occurs is the measures needed for the schedule indicators, EV and PV, are constrained to the value BAC. Because of this failure mode, EVM is not considered to be a useful method for evaluating project schedule performance.

Earned Schedule

ES resolves the problem with the EVM schedule indicators, and does so without requiring additional data. The fundamental concept of ES is shown in figure 2. As the description reads, "The idea is to determine the time at which the EV accrued should have occurred." The time duration associated with the point on the PMB where PV is equal to EV is Earned Schedule; that is, the point in time where the EV should been accomplished. For the EV accrued, ES provides a measure of how much has been earned of the planned duration (PD) of the project.

ES is computed from the simple formula:

$$ES = C + I$$

C is determined by comparing EV to the periodic values for PV, i.e., PV_n. C is the largest value of n satisfying the condition, $EV \geq PV_n$. I is an interpolation over one period of the PMB, using the equation:

$$I = (EV - PV_C) / (PV_{C+1} - PV_C)$$

Having ES, the time based schedule indicators are formed, Schedule Variance (time) and Schedule Performance Index (time), abbreviated as SV(t) and SPI(t), respectively. The indicators are computed by applying the following formulas:

$$SV(t) = ES - AT$$
$$SPI(t) = ES / AT$$

where AT is the actual time, i.e. the duration from the start of the project to the time (status point) at which EV is measured.

These time-based schedule indicators perform reliably for both late and early performing projects, thereby supplementing and improving EVM. Furthermore, the time-based indicators always converge to the actual result at project conclusion, as do the EVM cost indicators.

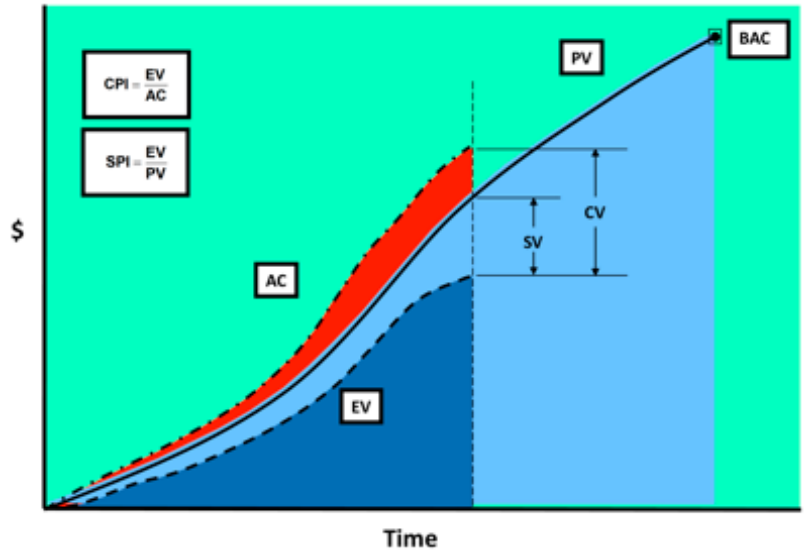


Figure 1. Earned Value Management

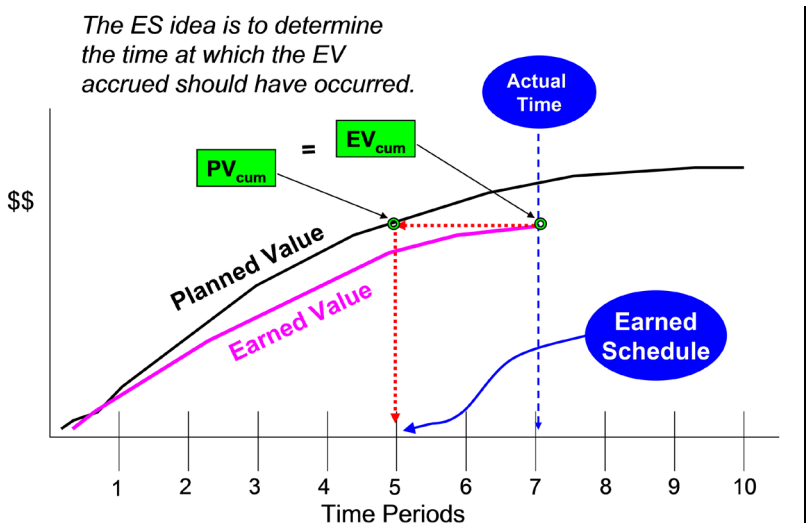


Figure 2. Earned Schedule Concept

In similar fashion, the SPI(t) indicator has made forecasting project duration possible from EVM performance data, using the simple formula [1]:

$$IEAC(t) = PD / SPI(t)$$

where IEAC(t) is Independent Estimate at Completion (time-based).

Similar to EVM forecasting, the ES forecast of project duration always converges to the actual result.

Introduction

A research study of project duration forecasting was made several years ago, employing simulation methods applied to created schedules having several variable characteristics [4]. The overall result from the study was that forecasts using

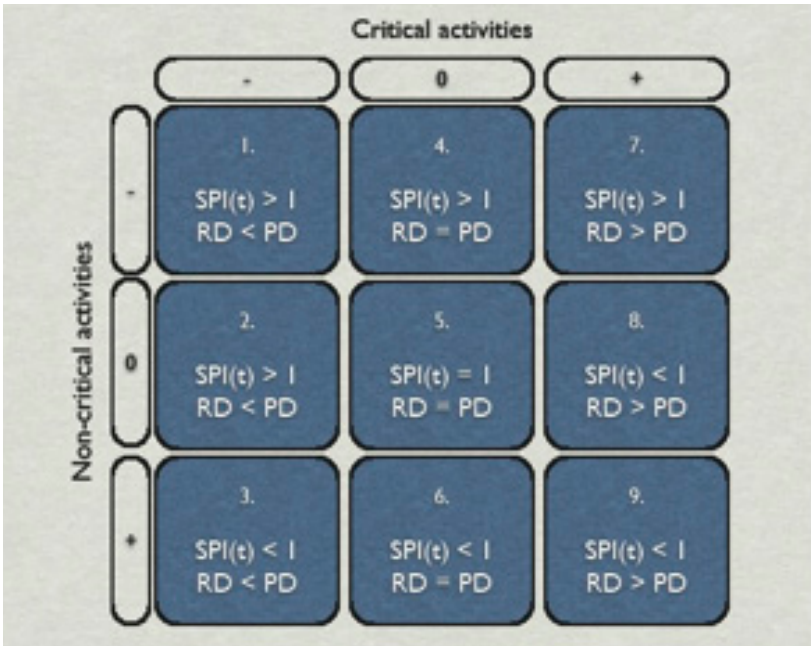


Figure 3. Schedule Performance Scenarios

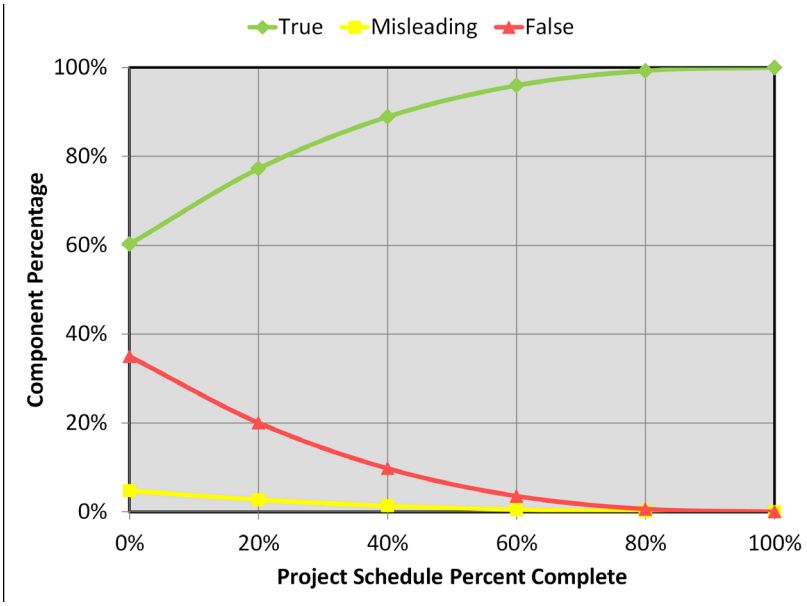


Figure 4. ES Forecasting Reliability Theory

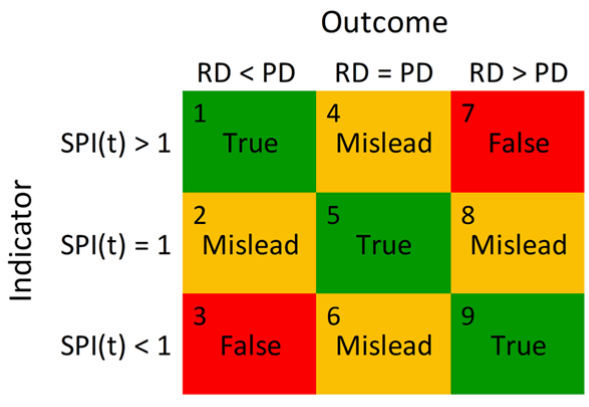


Figure 5. Indicator vs Outcome Scenarios

Earned Schedule (ES), on average, are better than other Earned Value Management (EVM) based methods. However, in certain instances the ES forecast was not.

The scenarios examined in the 2007 study are depicted in figure 3. The scenario model indicates nine possible outcomes. These outcomes are grouped into three categories: *true*, *misleading*, and *false*. True outcomes are associated with reliable forecasts, whereas the misleading and false categories indicate unreliable ES duration forecasting.

The three groupings are more fully explained as follows:[1]

The true scenarios (1, 2, 5, 8, 9)[2] have the characteristic that the relationship of the real or final project duration (RD) to the planned duration (PD) can be inferred from the schedule performance efficiency indicator, SPI(t).[3] Using scenario 1 for example, SPI(t) is greater than 1 (indicating good performance), while RD is less than PD (as one would expect from the indicator); i.e., the indicator is consistent with the duration result.

The misleading scenarios (4, 6) are characterized by the critical activities being completed as planned, while the non-critical activities are not.[4] The RD equals PD; however, SPI(t) is either greater or less than 1. Thus, the indicator is inconsistent with the duration outcome.

The false scenarios (3, 7) occur for two circumstances: 1) When non-critical activity performance is good and critical performance is poor, or 2) When critical activity performance is good and non-critical is poor. For these scenarios, the indicator, SPI(t), infers an outcome in opposition to the actual duration.

As indicated by the model only five of the nine possible outcomes are true (SPI(t) consistent with the final duration). Thus, a negative perception is created as to the reliability of ES forecasting.

A recent paper [2] examined the reliability question. Because of the convergence characteristic of ES forecasting, it was hypothesized that the misleading and false scenario indications resolve to consistency between SPI(t) and RD as the project progresses to conclusion. The evolution of scenario categories was illustrated in the paper by figure 4. As the project progresses, true scenarios increase, while misleading and false scenarios decrease. Thus, ES forecasting is theorized to become increasingly reliable as the project proceeds to completion.

In the final comments of the 2014 paper, a challenge was made to researchers to test the hypothesis that misleading and false scenarios migrate to true with project progress. For the proposed testing, the performance scenarios are categorized as shown in figure 5. The definitions of the categories are similar to those described for figure 3:

The true scenarios (1, 5, 9)[6] are characterized by SPI(t) being consistent with the relationship of RD to PD.

The misleading scenarios (2, 4, 6, 8) are identified when SPI(t) is inconsistent with RD, but are not regarded as false.

The false scenarios (3, 7) are determined when SPI(t) infers an early finish, while RD is greater than PD, or when it infers a late finish and RD is less than PD.

It is to be noted that the scenarios do not include the distinctions of critical and non-critical activities. They are unnecessary for the testing. The object is to determine the consistency of SPI(t) with the actual duration of the project, thereby providing evidence of ES forecasting reliability.

The research challenge was made intending for the hypothesis to be tested using simulation methods. The advantage of employing simulation is a large data sample can be created for the evaluation. This paper, however, performs the evaluation using data from sixteen real projects.

The motivation for this study is to provide information to managers, thereby enhancing their endeavor to effectively guide projects to successful completion. In this regard, the reliability of project duration forecasting is considered essential. The objective of this paper is to establish, at minimum, an initial understanding of ES forecasting reliability and provide confidence in its application should the testing yield positive results.

Description of Project Data

A total of sixteen projects is included in the study. Twelve (1 through 12) are from one source with four (13 through 16) from another. The output of the twelve projects is high technology products. The remaining four projects are typed as information technology (IT).

The primary data characteristic is the projects have not undergone any re-planning. This enables evaluation of the forecasting results without having undue outside influence. All sixteen projects performed from beginning to completion without baseline changes.

Table 1 illustrates the schedule performance of the projects in the data set. The twelve high technology projects are measured in monthly periods whereas the four IT projects are measured weekly. Two projects completed early, three as scheduled, and the remaining eleven delivered later than planned.

Method of Evaluation

For each project status point, the SPI(t) value and the relationship of RD to PD is used to classify the performance to one of the nine scenarios of figure 5. The scenario identification is then grouped to one of the three categories (true, misleading, or false) and associated with the schedule percent complete.[7] The tabulations of the categories are then assembled into ten percent increments of project completion. The results from all sixteen projects are then summed to form a composite. The composite results are normalized to percentages for each 10 percent increment, as shown in Table 2.

The process described is then re-evaluated taking into account quality of the forecast. Each misleading or false determination is examined for closeness of the forecast to the final duration. When the forecast is within 10 percent of RD, the determination is reassigned to true. It is reasonable to say that a forecast within 10 percent of the actual project duration is neither misleading nor false.

The assessment of whether ES forecasting is more reliable than previously portrayed in the literature is made from graphical analysis. The hypothesis that SPI(t) resolves to consistency with RD is credible, when it is demonstrated that the true percentage increases to 100 while the misleading and false components decrease to zero, as the project progresses to completion. Forecasting is considered reliable when the value from the linear fit of True% is approximately 60 percent at 25 percent schedule completion.

Schedule Performance								
Project	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Planned Duration	21m	32m	36m	43m	24m	50m	46m	29m
Actual Duration	24m	38m	43m	47m	24m	59m	54m	30m
Project	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Planned Duration	45m	44m	17m	50m	81w	25w	25w	19w
Actual Duration	55m	50m	23m	50m	83w	25w	22w	13w
Legend: m = month w = week								

Table 1. Schedule Performance

Pct Gp	@ 00	>5<=15	>15<=25	>25<=35	>35<=45	>45<=55	>55<=65	>65<=75	>75<=85	>85<=95	@100
Graph	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
True%	33.3%	72.7%	80.3%	86.1%	72.3%	72.5%	73.8%	92.3%	95.3%	100.0%	100.0%
Mislead%	44.4%	13.6%	11.8%	6.9%	10.6%	13.7%	18.5%	4.6%	4.7%	0.0%	0.0%
False%	22.2%	13.6%	7.9%	6.9%	17.0%	13.7%	7.7%	3.1%	0.0%	0.0%	0.0%

Table 2. Normalized Composite Results

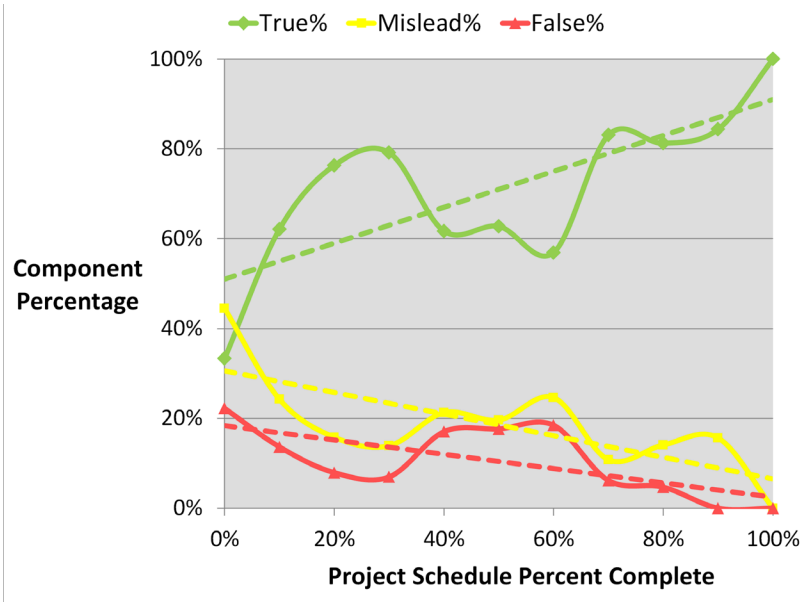


Figure 6. Composite Graph

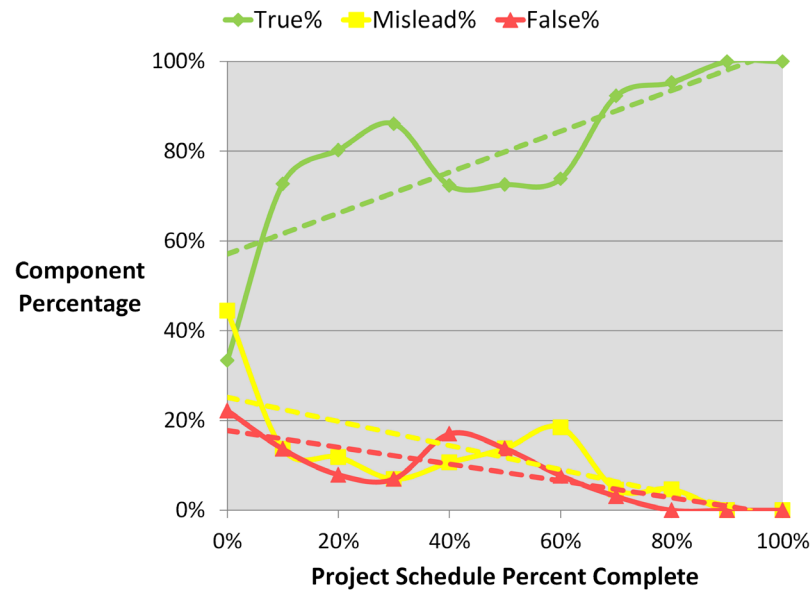


Figure 7. Composite Graph with 10% Margin

Analysis of Results

Two graphs depict the results. Figure 6 indicates results using the scenarios from figure 5. The scenario evaluation, depicted in figure 7, includes the reassigned category determinations from applying the 10 percent margin forecasting variance. Each graph begins at zero percent completion using the percentage of scenarios aligned to each performance component. For example, three scenarios align with true; thus, the initial point for True% is 33.3 percent.

Figure 6 depicts the trends of the forecast components. The compiled results clearly show the percentage of the true component increasing with project progress, while the unreliable components, misleading and false, simultaneously are decreasing. The graphs conclude with the true component at 100 percent and, consequently, the misleading and false components at zero percent.

For figure 7, as stated earlier, the True% includes the reassigned false and misleading results. The graph strongly indicates the convergence characteristic of ES forecasting. With the inclusion of the 10 percent margin, the true component approaches 100 percent much sooner. And overall, the misleading and false components are significantly smaller throughout.

Viewing the plot of True% from figures 6 and 7, the impact of including the 10 percent forecasting margin can be made. From figure 6, ES forecasting is approximately 60 percent reliable at 25 percent schedule completion, and 80 percent reliable at approximately 75 percent complete, reasonably good numbers. However, when the 10 percent margin is considered, figure 7 shows ES forecasting to be 60 percent reliable at approximately 5 percent complete, and 80 percent reliable at about 50 percent complete. These numbers are impressive, indicating ES forecasting for this set of data is good to excellent for 95 percent of the project duration.

Summary and Conclusion

Recently it was theorized that ES forecasting is considerably more reliable than how it has been portrayed previously in the literature. The essence of the theory is that due to the convergence characteristic of ES forecasting, the reliability of the forecasts must increase as the project progresses toward completion.

To test the theory, sixteen projects of real data were used. The performance values for SPI(t) and RD were categorized into the nine scenarios of figure 5 and subsequently grouped for each project into tabulations of true, misleading, and false components at ten percent progress increments. Subsequently, the project tabulations were summed to create a composite for evaluation.

The evaluation was made graphically. For the set of data tested, figures 6 and 7 clearly demonstrate that ES forecasting reliability increases with project progress. The true, or reliable, component increases while the unreliable components, misleading and false, decrease. It was also shown that when the ten percent forecasting margin was considered, the values for the True% component increased significantly. Overall, with the margin included, ES forecasting was assessed as good to excellent for 95 percent of the project duration.

Although more testing would be welcomed, it is reasonable from the results of this study to conclude that project managers employing EVM can have confidence in the forecasts made using ES.

NOTES

1. The true, misleading, and false grouping explanations are taken from [2].
2. The numbers in parenthesis for the groupings refer to the nine numbered cells of figure 3.
3. The relationship inference is obtained from the forecasting equation, IEAC(t) = PD / SPI(t).
4. The terms critical and non-critical refer to activities in relation to the schedule critical path.
5. Figure 3 is from the presentation [3].
6. The numbers in parenthesis for the groupings refer to the nine numbered cells of figure 5.
7. Schedule percent complete is equal to ES divided by PD, multiplied by 100.

REFERENCES

1. Henderson, K. (2004). "Further Developments in Earned Schedule," The Measurable News, Spring : 15-22
2. Lipke, W. (2014). Examining Project Duration Forecasting Reliability. PM World Journal, Vol. III, Issue III.
3. Vanhoucke, M. (2008). Measuring Time: a simulation study of earned value metrics to forecast total project duration. Earned Value Analysis Conference 13. London.
4. Vanhoucke, M., & Vandevoorde, S. (2007). A simulation and evaluation of earned value metrics to forecast the project duration. Journal of the Operational Reserach Society, Issue 10, Vol 58: 1361-1374.

ABOUT THE AUTHOR

Walt Lipke retired in 2005 as deputy chief of the Software Division at Tinker Air Force Base. He has over 35 years of experience in the development, maintenance, and management of software for automated testing of avionics. During his tenure, the division achieved several software process improvement milestones, including the coveted SEI/IEEE award for Software Process Achievement. Mr. Lipke has published several articles and presented at conferences, internationally, on the benefits of software process improvement and the application of earned value management and statistical methods to software projects. He is the creator of the technique *Earned Schedule*, which extracts schedule information from earned value data. Mr. Lipke is a graduate of the USA DoD course for Program Managers. He is a professional engineer with a master's degree in physics, and is a member of the physics honor society, Sigma Pi Sigma (ΣΠΣ). Lipke achieved distinguished academic honors with the selection to Phi Kappa Phi (ΦΚΦ). During 2007 Mr. Lipke received the PMI Metrics Specific Interest Group Scholar Award. Also in 2007, he received the PMI Eric Jenett Award for Project Management Excellence for his leadership role and contribution to project management resulting from his creation of the Earned Schedule method. Mr. Lipke was selected for the 2010 Who's Who in the World. At the 2013 EVM Europe Conference, he received an award in recognition of the creation of Earned Schedule and its influence on project management, EVM, and schedule performance research. Most recently, the College of Performance Management awarded Mr. Lipke the Driessnack Distinguished Service Award, their highest honor.

Upcoming Events

Visit <http://www.crosstalkonline.org/events> for an up-to-date list of events.

INCOSE 25th Annual Symposium IS 2015
13-16 July 2015
Seattle, WA
http://www.incose.org/newsevents/events/details.aspx?id=255

29th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy
July 13-15, 2015
Fairfax, VA
http://dbsec2015.di.unimi.it

2015 Software and Supply Chain Assurance Forum
31 August - 3 September 2015
McLean, Virginia
No Charge to Attend
https://register.mitre.org/ssca
http://www.gsa.gov/portal/content/220411

SDL Forum 2015: 17th International System Design Languages Forum
August 12-14, 2015
Berlin, Germany
http://sdlforum2015.informatik.hu-berlin.de

ISPA 2015: IEEE ISPA-15
August 20-22, 2015
Helsinki, Finland
http://comnet.aalto.fi/ISPA2015

RE 2015: Requirements Engineering
August 24-28, 2015
Ottawa, Canada
http://re15.org

FSE 2015: Foundations of Software Engineering
August 31-September 4, 2015
Bergamo, Italy
http://esec-fse15.dei.polimi.it

SEDE 2015: 24th International Conference on Software Engineering and Data Engineering
Oct 12-14, 2015
San Diego, Ca
http://www.cse.unr.edu/SEDE

STC 2015, the 27th Annual IEEE Software Technology Conference
October 12 - 15, 2015
Long Beach, CA
http://ieee-stc.org

SLE 2015: ACM SIGPLAN Software Language Engineering
Oct 25-27, 2015
Pittsburgh, PA
http://conf.researchr.org/home/sle2015

SEMCMI 2015: The International Conference on Software Engineering, Mobile Computing and Media Informatics- Part of the Fourth World Congress on Computing and Information Technology
Kuala Lumpur
October 27-29, 2015
http://sdiwc.net/conferences/semcmi2015

30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)
November 9-13, 2015
Lincoln, Nebraska
http://ase2015.unl.edu/#tab-main

It's Later Than You Think –
Do You Know Where Your Software Is?

I am a proud Air Force "Brat." In fact, I have never been without a military ID card – first as a dependent, then as a member of the USAF (23 years), and now as USAF retired.

I was born while my Dad was stationed near Edinburgh, Scotland at RAF Kirknewton, which had a U.S. Security Service (USAFSS) squadron attached there. We then moved to Shaw AFB (Sumter SC), Orlando Air Force Base (which eventually became the Orlando Naval Training Center), Chanute AFB (Rantoul, IL), Istanbul Turkey (The US Logistics Group – TUSLOG), Sheppard AFB (Wichita Falls, TX) and finally, McCoy AFB (formerly Pinecastle AFB, Orlando, FL). All of these moves by the time I was 15. (I had another 8 bases under my belt by the time I retired, but that's another column).

Back in 2004, I realized I wasn't getting younger, and I decided to visit all of the houses I had lived in as a child. I wanted to relive some memories. My parents had kept careful records – so it would appear that all I had to do was plug the addresses into Google Earth, and spend a few $s for a couple of airplane tickets.

Of course, nothing is as easy as it seems. In 2004, I took a vacation to Edinburgh, Scotland – I had not been back since I left at the age of 4 in 1959. (NOTE: in case my former boss is reading – I REALLY went to the conference, and only vacationed during weekends. Honestly!) RAF Kirknewton was still there, and I was able to find the house I grew up in, located in the sleepy village of Balerno. Next stop was Sumter, SC. Shaw AFB was still there, of course, but the street name had changed. It took a while, but eventually I was able to find our old house (really, a townhouse) – which seemed SO big as a child, but was unbelievably small. The entire block was under renovation – I was lucky to visit when I did. One more house checked off.

Orlando – that's an easy one. Dad retired there – and even though Orlando AFB became Orlando Naval Training Center (and then closed in 1999) the house we rented was still there. Likewise for the house when he was assigned to McCoy AFB (formerly Pinecastle AFB – also closed) in 1969 – we lived off base, and the house was still there.

Chanute AFB in Rantoul, IL? Not so easy. It had closed in 1993 – and while most of the houses were still there, all streets were renamed. Luckily, there is the Chanute Air Museum where the base headquarters used to be, and with the help of several volunteers who worked there – old maps were brought out, and the two houses we lived in during our 4 years there were located. Both still there, both were visited, and both marked off my list.

Sheppard AFB? Still there, and when I drove through back in 2007, I was lucky enough to arrive the week before they were scheduled to tear down my old house. Pictures taken, memories relived, and another one marked off. (NOTE: We left Sheppard in 1969, and I vaguely remember a cute red-haired girl who lived down the block. I met her again in 2010, 41 years later – and we got married. Some things are just meant to be.)

Only one house to go – the one in Istanbul, Turkey. There was no actual base in Istanbul, and we had rented a house near the detachment. I had the address – how hard could this be? My first trip back to Istanbul was in 2005 (39 years after we had left), and even with addresses and maps and a pretty good memory, it appeared that my house was long gone. The address no longer existed – several taxi drivers were unable to locate it. I walked all over the area I thought it should be in – and I never saw anything familiar. Of course – when we lived there in the 1960s, Istanbul was a big city – about 1 million people. In 2005 – bigger! 12 million! I gave up.

Until 2009. I got the chance to visit Istanbul again to attend a conference (and yes, I actually attended the conference. EVERY DAY!) This time, I had done my homework. In addition to the old address, I had old several maps that I could use, and thanks to several web sites that I had discovered, I had a few landmarks to use as points of reference. It took me less than 2 hours to re-discover our old house (instantly recognizable, but now a small girl's school!)

And my mission was complete. Every house I grew up in as a child I had re-visited as an adult. You are probably asking "What in the world does this have to do with a topic for BackTalk? Not much – it was just fun for me to relive a few memories. Wait – maybe it IS relevant! I was lucky. A lot of military brats (with similar backgrounds to mine) can't revisit where they grew up – based closed, houses torn down, missing addresses, etc. It's all about documentation, directions, addresses and maps. And timing. It's not enough to just know where you are going. Like software, I knew my goal. I "had a vision." However, along the way, I missed turnoffs, got sidetracked, and occasionally got lost. I was lucky in Turkey – almost everybody spoke a bit of English. I just kept asking and asking, and eventually found my way.

I have to ask – "It's halfway through your budget – do you know where your software project is at?" I am teaching a senior-level project management course this semester and there are just SO many things you have to keep track of to be a good project manager. Budgets, risks, configuration management issues, personnel, training. Milestones, earned value, etc.

Why do we have project management tools? Just to make our job simple? Not really – project management is seldom (never?) simple. Project Management tools are there to help you manage the data. It's not enough to just know where you are going. You have to know that you are heading in the right direction, and you need something to make sure you are getting closer (and closer and closer and …) to your goal. And whatever data you have – you need more data, better data, more accurate data. Data that helps you find your way to your goal.

And if that lesson isn't about Data Mining in Metrics, then I'm still lost.

David A. Cook, Ph.D. (Major, USAF, Retired)
Stephen F. Austin State University