Opening

Measurable Data

-Variety of methods, both manual and automatic have been measured

-The basics are lines of code and code complexity (don’t really work)

-Use of outdated methods is very common

-More recently methods have evolved to take in more data and produce more relevant studies

-Regression methods, Defect rates, reliability

-Agile has made other methods more common (velocity etc) (Javdeni)

-Customers are also a key metric as output-based results are very important for knowledge workers

-Wearable technology produces even more data

-Source code investigation

-Empirical Software engineering

-Employee interaction and learning (Dittrich et al)

-Work Patterns (Snipes et al)

-Software Intelligence (Hassan)

-Worker Satisfaction

-Effectiveness concerns mentioned here?

-Drucker

-Searching under the streetlight

-Successes and Failures (Fenton and Neil)

-Game like environment methods (Silverman, Truning real-world sotware development into a game)

-The myth of developer productivity (Barnes)

Computational Platforms Available to Perform Work

-Searching under the streetlight lists some key devices

- Kan: IBM and Hewlett-Packard tools, Assessments

- Game like environment methods (Silverman)

- Agile applications like JIRA (Javdani)

- COSMIC (Javdani)

-HackStat, PROM (Snipes) (Johnson) (Sillitti)

-Cloud computing makes processing big data feasible (Pocatilu, Shakeabubakor)

-CMMI, ISO/IEC etc (Grambow)

-Hitachi tool (Yano)

-Zorro (Kou)

-DevRPG (Silverman)

-Surveys

Algorithmic Approaches Available

-Historic methods much simpler (ie counting)

-Agile methods also use some simpler approaches

-Bayesian Belief Networks (Fenton and Neil)

-Network Analysis (Dittrich et al)

-Function Point Analysis (Heemstra)

-Influence Models (Wei Pan)

-Test Driven Development Analysis (Kou)

-Software Process Telemetry (Johnson)

-Pattern Extraction

-Locates patterns, usually prejudice

-Expert Systems

-Reasoning developed behind patterns

-More relatable judgement

-Not able to consider as much data

Ethical Concerns

-Taylor and Amazon: Results of measuring too much can be negative (Kantor and Streitfeld)

-Price of Privacy

-GDPR (Legal ICT)

-Effect of Measuring and Importance of measuring correctly

-Data Manipulation

-Data Sovereignity

-Would greater transparency help this?

-Searching under the Spotlight (Johnson, 2013)

-Knowing all parameters would lead to best knowledge, but also all knowledge about everything employees do

-Cloud servers spend a lot on security

-Fremont Plant: How to do it right?

-Power given to higher ups?

-Conclusion

**1) In-text:**(Johnson, 2013)

**Your Bibliography:**Johnson, P. (2013). Searching under the Streetlight for Useful Software Analytics. *IEEE Software*, 30(4), pp.57-63.

-Collaborative Software Development Laboratory

-Searching under the streetlight is easier and less controversial but is limited in usefulness

- “trade-off is an essential design characteristic”

-Unlikely to find a silver bullet

-PSP

-1996 “A Discipline for Software Engineering” (Watts Humphrey)

-How to adapt organizational software analytics for individual devs

-How analytics can drive improvement

-This books PSP requires manually filling out a lot of forms

-Leads to situation specific responses

-Also flexible, allowing users to search for analytics they need

-Sometimes led to incorrect solutions despite low error rate

-LEAP

-Automates and normalizes data

-Still manual, but some parts of PSP automated and some extra conclusions reached

-Higher level tool support, but automation makes some data easy to collect and other harder

-Hackystat

-Collect software process and product data with little/no overhead

-Sensors attached to Dev tools

-Includes client and server-side data

-Unobtrusive data collecting

-collects a data instance each time active buffer is changed

-Personal and group based development

Problems

-Some people didn’t like data being collected without their knowledge

-Created discord in developing group by providing too much info about individuals working styles

-Unhappy with management having that level of data

-Robert Austin Measuring and Managing Performance in Organizations, Dorset House, 1996. (4)

-Info on collected data for (3)

DevCreek, Ohloh, Atlassian, CAST, Parasoft, McCabe, Coverity, Sonaor and others

-Typically configuration management systems, build systems and defect-tracking system

-Entirely automated and data is already available, simply analysing (coverage, complexity, security etc) to data

-Low overheads since data is collected from a repository

-Uncontroversial data

-Streetlamp searching, easy but limited

-No behavioural client-side data collection

-No test driven development

Trade-Offs

-Degree of automation and overheads incurred to obtain analytics

-Barrier to adoption (social or political)

-Level of generality

-Ultimately no convergence on a single best approach

-Current tools aren’t necessarily advancements on old ones, they just make different trade- offs

-Combining automatic and manual data could be the way forward

-Modern approaches to privacy could be an issue

**1)In-text:**(Fenton and Neil, 1999)

**Your Bibliography:**Fenton, N. and Neil, M. (1999). Software metrics: successes, failures and new directions. *Journal of Systems and Software*, [online] 47(2-3), pp.149-157. Available at: http://www.pauldee.org/se-must-have/new-software-metrics.pdf [Accessed 11 Oct. 2018].

-Software metrics has had little impact on industrial practice

-History of Software Metrics

-Lines of Code measure used routinely from 1960s

-Increased diversity of languages calls for more

-Software complexity took over in mid 70s

-More recent

-mechanics of implementing metrics programs

-Use of metrics in empirical software engineering

-New methods should require empirical evidence as backup

-theoretical underpinnings of metrics

-improving level of rigour

-Success and Failure

Success

-Huge amount of research has gone into it

-Most major IT companies now have software metrics activity occurring

Failure

-No match between increased level of metrics in academia and industry

-Most increased metrics activity is based on metrics from early 70s

-Most academia is irrelevant in scope (small systems) and content (not focused on process improvement)

-Implementation is poorly motivated (usually response to an external assessment)

-Poorly executed

-Use outdated methods

-Limitations of Metrics

-1998 study showed many methodologies and theories were incorrectly implemented

-Contrary data about bugs pre and post release (NB)

-New Directions

-Complexity and Size measures alone cannot assess software defects

-Pre-release info about defects is uncorrelated to post-release

-Regression based methods are inappropriate for defect prediction

-Correlation != Causation

-Size and fault density are not causal despite popular belief

-Model needs to handle diverse evidence, detect genuine cause and effect relationships, deal with uncertainty and incomplete information

-Bayesian Belief Nets (3)

-Building realistic models has only been made possible due to recent algorithms and tools

-Uses data already being collected

-Implemented in decision making for propulsion systems on Space Shuttle

-Hugin tool does calculations

**2) In-text:**(Kan, 2014)

**Your Bibliography:**Kan, S. (2014). *Metrics and models in software quality engineering*. 2nd ed. Addison-Wesley.

Ch1: Software Quality

-Popular View

- “I’ll know it when I see it”

- “Connotes luxury, class and taste” as opposed to reliability

-Professional View

- “conformity to requirements”

- “fitness for use”

-Customers are a key metric in quality, since they are the users

-Software quality: Lack of bugs (1)

-Defect rate and reliability

-Customer satisfaction

-IBM: CUPRIMDSO (capability, functionality, usability, performance, reliability, installability, maintainability, documentation/information, service and overall) (2)

-Hewlett-Packard: FURPS (functionality, usability, reliability, performance, and serviceability)

-Different weights required for different customers (2)

Ch2:

-SPR assessment method

-Software Productivity Research Inc.

-300 questions with a rating from 1-5 (1 being excellent, 5 being poor)

-major themes:

-Findings about the projects/software products assessed

-Findings about technologies used

` -Findings about software processes used

-ergonomics and work environment of staff

-personnel and training

-as of 2000, performed assessments for nearly 350 companies and 50 government organizations

-Malcolm Baldrige National Quality Award Assessment

-Prestigious quality award in the US

-Evaluates approach, deployment, and results

-Report gives places for company to improve on

**3) In-text:**(Dittrich, Gunes and Dascalu, 2013)

**Your Bibliography:**Dittrich, A., Gunes, M. and Dascalu, S. (2013). Network Analysis of Software Repositories: Identifying Subject Matter Experts. [online] Available at: https://pdfs.semanticscholar.org/d6b0/b5abe47a782ad0866f524e1e5b3e10d49a89.pdf [Accessed 11 Oct. 2018].

-Uses network analysis to identify subject matter experts

-See how well team works together

-Identify any risks of a member leaving the team if there is only one expert on a certain topic

-Identify

-Associating software metrics with specific authors can make them feel threatened

-examine revision control repository of topics and associate authors with these

Does not say to what effect they were involved

-Many others describe different ways of associating authors from control repositories

-This article uses log files only

-Gathering Data

-Bipartite graph

-Connect each author to the files they changed on a graph (vertices: authors and files, edges: each change to a file by an author

-Collected from log data

-Initial additions of files are ignored

-May be worth filtering by date

-Two authors may have worked on the same stuff but never at the same time

-Analyze

-One-mode author graph connects authors by files they both changed

-Core Devs: have the most connections to a project

-Measure the vertex’s degree

-Eigenvector centrality for an author indicates how connected an author is in another author’s network

-Various measures for identifying communities

-If each author is connected to each other author, at least two authors have worked on every project

-Changing a file may not necessarily represent any knowledge of how the file works

-All changes are given equal weight by this measure

**3) In-text:**(Javdani et al., 2012)

**Your Bibliography:**Javdani, T., Zulzalil, H., Ghani, A., Sultan, A. and Parizi, R. (2012). On the Current Measurement Practices in Agile Software Development. *Journal of Computer Science Issues*, 9(4), pp.127-133.

-Agile requires different measurement practices

-Lack of documentation means some methods don’t work

-Adaptability requires more measurement for cost estimation

-Individuals and interaction over software and tools

-(4)

-Measurement: (These tools have become very popular recently)

-Effort Estimation: -Needs well-defined user requirements documents

-These are subject to frequent change with agile

-Usually effort estimation is commonly subjective

-Simple and easy, but highly biased (street lamp)

-Can be collaborative estimate

-User story points (JIRA)

-Less bias but still subjective

-Abrahamsson (P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," Banff, AB, 2011, pp. 400-403.) -Prediction using story points

-Velocity (Productivity): -Number of completed user stories in iteration, used to estimate remaining time until completion

-Velocity should be measured over several successive fixed length periods

-Changes to team will invalidate velocity, depends on same people

-Burndown Chart:

-Estimates amount of remaining work

-Compare actual to estimate

-Cumulative Flow:

-Shows stage of each piece of work

-Shows size of backlog

-Response to Change:

-Used as indicator of ability to adapt

-Measure hours spent reworking project

-Business Value:

-Relies on budgeted costs (which aren’t available in agile)

-For agile, sum of weight of stories completed

-If scope is uncertain, measure is useless

-therefore better for small/medium agile projects

-Total Effort Estimation:

-Managers generally require up-front prediction of work required to dedicate a necessary budget.

-Team can use velocity and backlog to determine

-COSMIC published an adopted version of COSMIC function points

-Software Function Measurement is widespread as an input for effort and cost estimation.

-Most popular methods (FPA, De Marco’s bang model, Mark II, Boeing 3D function point, Full Function Point, COSMIC FFP) are not adaptable to agile despite some being accepted as international standards.

-COSMIC method is more applicable in agile since it doesn’t require detailed specifications

-Derived in terms of user requirements and estimations

-Desharnais proposed COSMIC in agile in 2011 and later COSMIC published an official agile version

-Used for size measurement not effort estimation based on user stories

-More standard than the estimates given by team.

-Future:

-Productivity over velocity ad change requirement measurement are other issues for future study.

**2) In-text:**(Snipes et al., 2013)

**Your Bibliography:**Snipes, W., Augustine, V., Nair, A. and Murphy-Hill, E. (2013). Towards recognizing and rewarding efficient developer work patterns. *35th International Conference on Software Engineering (ICSE)*, pp.1277-1280.

-Proposes system to motivate developers to adopt best practices and tools

-Propose 3 step automated technique recommendation system

1.Understanding Work Patterns:

-A work pattern is a build-up of sequences used by a developer for session types (debugging, extending functions etc.)

-Efficient and inefficient work patterns can be identified by comparison

2.Observing Work Patterns

- “fine-grained data”

-Use Mylyn Monitor, Hackystat or PROM to collect info

-Markov chain of steps taken during development process or recent research in clickstream analysis.

-identify similarities, user profiles and usage patterns

-Can be used to identify unnecessary steps in processes

3.Motivating Improvement

-Gamefications

-Saatchi study says 75% or 18-45 year-old employee respondents were at least somewhat interested in a working environment with some game elements

-Also called achievement motivation

-A system that offers achievement to the use of better techniques, practices and tools

Related Work

-Zorro uses HackyStat data to determine if developers adhere to test-driven development

-Murphy-Hill, Parnin and Black use Mylyn to explore if developers use the Eclipse refractor command over manually refracting code

-Robillard, Coelho and Murphy explore more effective bug finding and fixing

-More success when a detailed plan of change, structured navigation and keyword/cross-reference searches and only review methods once during search

**4) In-text:**(Barnes, 2015)

**Your Bibliography:**Barnes, D. (2015). *The Myth of Developer Productivity*. [online] Dev9. Available at: https://dev9.com/blog-posts/2015/1/the-myth-of-developer-productivity [Accessed 17 Oct. 2018].

- “Any metric you come up with to measure developers will be gamed”

-Hours: The more hours beyond 40, the less effective you are

-Lines of Code: Easy to game

-Bugs Closed: Easy to open bugs

-Defect Rate: Discourages talking new problems

-Estimation: -Just estimate longer

-Reduces Flexibility

-Story Points: Inconsistent between developers

-These not working are all generic, only the ones related to programming are programming specific

-These methods are all selected with no consideration for correlation at all, simply because they sound good

-“Developers are treated like blue collar workers”

-Doing the same job over and over leads to predictability

-Doctors and Lawyers are measured based on customer satisfaction

-Measuring impediments: Show manager why work takes longer than estimated

-What do you define as an impediment

-Impediments can arise outside of work

-Time before Delivery: Time between requesting work and when it is available, will show consistency if units of work are of similar size

-Flow Control: Working on too much at once can also slow productivity, so should focus on one item and drive to completion (proof here is illogical)

-Map value stream, define start and end points, limit work in progress, adjust empirically

-Seems to me like what is being done

**2) In-text:**(Murphy-Hill and C. Murphy, 2011)

**Your Bibliography:**Murphy-Hill, E. and C. Murphy, G. (2011). Peer Interaction Effectively, yet Infrequently, Enables Programmers to Discover New Tools. *in Proceedings of the ACM 2011 Conference on Computer supported cooperative work*. [online] Available at: https://people.engr.ncsu.edu/ermurph3/papers/discovery.pdf [Accessed 23 Oct. 2018].

-Discusses how tool discovery takes place on a social level

-Much research on increasing tool awareness on a technical level, much less on a social level

-Pair programming is the best way for this to occur

-Peer observation and recommendation were listed as most effective but not the most commonly occurring methods of learning new tools

-Barriers exist including isolated programming and higher ups requesting certain tools be used even if they are not the most effective

-Making it more obvious when tools are used and what they do can improve peer learning

-Trust and relevance is a key reason as to why peer training is more effective than social media and blogs

Cloud

- **In-text:**(Pocatilu, Alecu and Vetrici, 2010)

**Your Bibliography:**Pocatilu, P., Alecu, F. and Vetrici, M. (2010). Measuring the Efficiency of Cloud Computing for E-learning Systems. *WSEAS TRANSACTIONS on COMPUTERS*, [online] 9(1), pp.42-51. Available at: https://pdfs.semanticscholar.org/0d6f/2e0ee9dac8e6d8682c05c12c1e2b7bc01b08.pdf [Accessed 24 Oct. 2018].

- Amazon, Google, Yahoo, Microsoft etc provide cloud computing resources

- Cloud computing becomes very popular because it moves the processing efforts from the local devices to the data center facilities

- devices, like an Internet connected phone, could be able to solve complex equations by simply passing the specific arguments to a service running at the data center level that will be capable to give back the results in a very short time

-Adv:

- cost is low

- the cost of licensing different software packages is moved to the data center level, so there is no need to upgrade the local system when new service packs or patches are released

- crash recovery is nearly unneeded. If the client computer crashes, there are almost no data lost because everything is stored into the cloud.

Disadv:

- service quality is crucial and the need of the backups is critical when speaking about data security

- Grid Computing: use a grid is to remotely run an application on a different computer than the one on it is usually executed. If a computer is busy, the application can be executed on another idle machine from the grid network. The remote machine must meet hardware, software and resource requirements of the application

-High security provided by the cloud:

1- It is almost impossible for any interested person (thief) to determine where is located the machine that stores some wanted data is

2- monitoring of data access becomes easier in view of the fact that only one place should be supervised, not thousands of computers

-Should discuss the following when pursuing cloud computing:

-Who has specialised access to the data

-Does vendor have security qualifications/compliance

-Data location

-Ensure encryption occurs at multiple levels and is adequately tested

-What happens in case of data breech

-What happens to data when company goes out of business (4)

-What are procedures for moving data elsewhere if required/desired

-Pareto principal: 80% of bugs come from 20% of code

- **In-text:**(Shakeabubakor, Sundararajan and Razak Hamdan, 2015)

**Your Bibliography:**Shakeabubakor, A., Sundararajan, E. and Razak Hamdan, A. (2015). Cloud Computing Services and Applications to Improve Productivity of University Researchers. *International Journal of Information and Electronics Engineering*, [online] 5(2). Available at: http://www.ijiee.org/vol5/521-F005.pdf [Accessed 24 Oct. 2018].

-University researchers are knowledge workers like software engineers

-dynamic environment with constant change

-Productivity: Ratio of output and input (including time)

-Foundation of profitability

- The rapid advancement in technology and increased awareness among people has made educational organizations and universities across the world to be productive, competitive and sensitive to changes in the learning methods

- Motivation and morale of people are very important factors that determine productivity (4)

- **In-text:**(Grambow, Oberhauser and Reichert, 2013)

**Your Bibliography:**Grambow, G., Oberhauser, R. and Reichert, M. (2013). Automated Software Engineering Process Assessment: Supporting Diverse Models using an Ontology. *Int'l Journal on Advances in Software*, [online] pp.213-224. Available at: https://pdfs.semanticscholar.org/9a9d/5251aa571aa31cb30d01e1bf2464364ba08c.pdf [Accessed 24 Oct. 2018].

- Current software engineering process assessment reference models rely primarily on manual acquisition of evidence of practices

- manual data acquisition is inefficient and error-prone, however automation also has issues

-Automation has trouble covering the dynamically changing requirements of software engineering which is why it is currently less widely used

- Context-aware Software Engineering Environment Event-driven frameworK (CoSEEEK) developed to improve this automation

-Automatically collecting data from environment and user ad producing info for rapid process optimisation.

- Hope to implement in-the-loop assessment support

- Three of the most mature and prevalent process assessment approaches used in software projects (CMMI, ISO/IEC 15504 / SPICE, and ISO 9001)

-CMMI: Capability Maturity Model Integration

- Comprises five maturity levels (1- ‘Initial’, 2-‘Managed’, 3-‘Defined’, 4-‘Quantitatively Managed’, 5-‘Optimizing’) indicating ‘Degree of process improvement

- A maturity level (e.g., ‘2’) has process categories (e.g., ‘Support’) that have process areas (e.g., ‘Configuration Management’) that have specific goals (e.g., ‘Establish Baselines’) that finally have specific practices (e.g., ‘Identify Configuration Items’)

- To quantify the assessment, CMMI has a performance scale (1-‘unrated’, 2-‘not applicable’, 3-‘unsatisfied’, 4- ‘satisfied’). Rates each generic and specific goal of a process area using the introduced performance scale. A maturity level is achieved if all process areas within the level and within each lower level are either 2 or 4

-Also features generic goals and practices which are subordinate to capability levels (0-‘Incomplete’, 1-‘Performed’, 2-‘Managed’, 3-‘Defined’, 4-‘Quantitatively Managed’, 5-‘Optimizing’)

- ISO/IEC 15504 (SPICE): Software Process Improvement and Capability Determination

- comprises six capability levels (0-‘Incomplete process’, 1-‘Performed process’, 2-Managed process’, 3- ‘Established process’, 4-‘Predictable process’, 5-‘Optimizing process’). Each of the latter has one or multiple process attributes (e.g., ‘2.1 Performance Management’)

- SPICE does not use assessments of practices to directly determine whether an overall capability level is achieved, but uses them to assign to each process one or more capability levels and to use them to recursively calculate assessments for projects and organizations

- ISO 9001:

- ISO 9000 comprises a family of standards relating to quality management systems. ISO 9001 [5] deals with the requirements organizations must fulfil to meet the standard.

- Formal ISO 9001 certifications have gained great importance for organizations worldwide

-Automated Process:

- aim of our approach is not to replace manual ratings of processes conducted by humans or to be used in formal process audits. It shall rather contribute to the quality awareness of a company and provide information on the current state of the process as it is executed. still integrates and relies on manual ratings or confirmations for ratings

-CoSEEK Framework

**In-text:**(Yano et al., 2015)

**Your Bibliography:**Yano, K., Akitomi, T., Ara, K., Watanabe, J., Tsuji, S., Sato, N., Hayakawa, M. and Moriwaki, N. (2015). Measuring Happiness Using Wearable Technology. *Hitachi Review*, [online] 64(8), pp.517-524. Available at: http://www.hitachi.com/rev/pdf/2015/r2015\_08\_116.pdf [Accessed 24 Oct. 2018].

-found a hidden signal representing a person’s happiness within the basic pattern of physical activity known as the “1/T rule.” This result uncovered a close relationship between the “trinity” of physical activity = happiness = productivity. Combined with a technique for using artificial intelligence to generate KPIs automatically,

-a person’s happiness significantly influences performance. Compared to people who are unhappy, it has been found that people who are happy have 37% higher work productivity and 300% higher creativity

-identified characteristic patterns of physical activity that have a strong correlation with happiness

- “1/T rule”(9). It categorizes physical activity during each unit of time as either inactive or active, and looks at the “active” times when the person is moving and how long they last

-some groups follow the rule comparatively closely while others diverge considerably

-1/T fluctuation is strongly correlated with happiness

-There are some people who think this way: As a subjective concept, happiness could be just an indication of self-satisfaction; if people are placed in a comfortable environment, they may lose the motivation to achieve; it should be true that it is when people feel threatened or anxious that they exert their power to the fullest. Nevertheless, the data indicates otherwise.

-Hitachi’s “H” artificial intelligence (an abbreviation of “Hitachi Online Learning Machine for Elastic Society”) can automatically generate individual KPIs (key performance indicators) that include symbiotic variables and the evaluation functions used to calculate them from big data

-H automatically generates a large number of hypotheses about the factors that influence outcomes (with a number of candidate variables ranging between one hundred and one million), narrows these down to a small number of important factors, and then generates the evaluation function

-By having H automatically update these evaluation functions using daily data, it is possible to work continually toward system-wide optimization in a changing environment by having KPIs adapt to the circumstances.

-eg: Instead of conventional air conditioning that maintains a constant temperature, using happiness measurements and artificial intelligence, it is possible to operate the air conditioning in a way that maximizes the overall happiness of the people in the building

**(3) In-text:**(Wei Pan et al., 2012)

**Your Bibliography:**Wei Pan, Wen Dong, Cebrian, M., Taemie Kim, Fowler, J. and Pentland, A. (2012). Modeling Dynamical Influence in Human Interaction: Using data to make better inferences about influence within social systems. *IEEE Signal Processing Magazine*, [online] 29(2), pp.77-86. Available at: <http://fowler.ucsd.edu/modeling_dynamical_influence.pdf> [Accessed 25 Oct. 2018].

- “influence model,” which utilizes independent time series to estimate how much the state of one actor affects the state of another actor in the system

-Another way to measure employee productivity

-Influence model is based on idea that each item in network is influenced to different degrees by its neighbours and moves as such

-Consists of entities interacting with each other. All entities’ states at t-1 will influence each entities’ state at time t with the strength of influence varying from entity to entity

-Uses machine learning to dictate parameters

-The influence model has been applied to various social systems, particularly those that have been monitored by sociometric badges; personal devices that collect individual behavioural data including audio, location, and movement

-methodological advances that allow the model to incorporate dynamic changes in the influence matrix itself

-Emulating the dynamic reality it is meant to model

-The volume of data collected here emphasises usefulness of cloud computing

-The influence model shares the same issues with other machine learning models: inference requires sufficient training data, and tuning is necessary for best results.

-If we find that behavior between two individuals is correlated, it could be due to influence, but it could also be due to selection (I choose to interact with people like me) or to contextual factors (you and I are both influenced by an event or a third party not in the data).

-this does not make observational data worthless. It just means that we should carefully consider alternative mechanisms that may underlie correlated behaviour

**-In-text:**(Kou, Johnson and Erdogmus, 2009)

**Your Bibliography:**Kou, H., Johnson, P. and Erdogmus, H. (2009). Automated inference of test-driven development with Zorro. *Automated Software Engineering*, [online] 17(1). Available at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=956E1624304502A89128F90EAF854128?doi=10.1.1.69.4525&rep=rep1&type=pdf [Accessed 25 Oct. 2018].

-Zorro is a system designed to automatically determine whether a developer is complying with an operational definition of Test-Driven Development (TDD) practices

-Test Driven Development’s response has been mixed, which article attributes to poor introduction (seems to be a recurring theme)

-second problem with TDD research involves compliance, or verification that the participants who are supposed to be doing TDD are doing TDD

-Zorro, a system for automated recognition of TDD practices.

-gathers a stream of low level developer behaviours (such as invoking a unit test, editing production code, invoking a refactoring operation) while programming in an IDE, partitions this event stream into a sequence of development “episodes”, then applies a rule-based system to determine whether each episode constitutes an instance of a TDD practice.

-enabling researchers and practitioners to precisely characterize the extent to which the given definition of TDD was applied

-Uses Hackystat to collect info, SDSA (Software Development Stream Analysis), a Hackystat application that supports generic analysis of development event streams; and Zorro, an SDSA application, which defines the specific rules and analyses necessary for recognition and interpretation of the TDD behaviour of a developer.

-begins by merging the events collected by various sensors into a single sequence, ordered by time-stamp, called the “development stream”. This is followed by a process called tokenizing, which results in a sequence of higher-level “episodes”. These constitute the atomic building blocks for whatever process is being recognized. For any given application of the SDSA framework, tokenization involving defining the specific events to be combined to generate the development stream, as well as the boundary condition that separates the final event in one episode from the initial event in the next. Once the development stream has been abstracted into a sequence of episodes, the next step in SDSA is to classify each episode according to whatever process is under analysi

-Zorro requires unit test invocations (and their results), compilation events (and their results), refactoring events (such as renaming, moving), and editing (or code production) events (all can be collected via Hackystat), requires the use of the Java programming language, the JUnit testing framework, and the Eclipse IDE

-Once each episode instance has been assigned an episode type by the SDSA rule set, the final step in the Zorro classification process is to determine the TDD conformance of that instance

-Zorro classification system replaces the simplistic three episode type (red, green, yellow) approach to TDD developer behaviour with a more sophisticated classification scheme based upon 22 distinct episode types.

-Output shows whether an episode is classified as TDD conformant and why/why not.

-Goes into detail about validation tests run to determine Zorro’s effectiveness

-By building Zorro on top of the more generic SDSA and Hackystat frameworks, its architecture makes it more easily possible to study not only TDD, but other interesting developer “best practices” on this fine-grained level.

**-(3) In-text:**(Johnson et al., 2005)

**Your Bibliography:**Johnson, P., Hongbing Kou, Paulding, M., Qin Zhang, Kagawa, A. and Yamashita, T. (2005). Improving Software Development Management through Software Project Telemetry. *IEEE Software*, [online] 22(4), pp.76-85. Available at: https://pdfs.semanticscholar.org/5871/29969f8856c1dc5d233fcbaaa300e129f758.pdf [Accessed 25 Oct. 2018].

-a recent case study of more than 600 software professionals revealed that only 27 percent viewed metrics as “very” or “extremely” important to their software project decision-making process.4 The study also revealed that most respondents attempted to use metrics only for cost and schedule estimation.

-telemetry is a “highly automated communications process by which measurements are made and other data collected at remote or inaccessible points and transmitted to receiving equipment for monitoring, display, and recording.”

-software project telemetry has the following essentials

-The data is collected automatically by tools that regularly measure various characteristics of the project development environment

-The data consists of a stream of timestamped events where the time-stamp is significant for analysis.

-Both developers and managers can continuously and immediately access the data.

-Improves engagement

-Analysis includes in-process monitoring, control, and short-term prediction.

-Software project telemetry also relates in interesting ways to both the Capability Maturity Model Integration and Agile methods

-Agile: telemetry provides a measurement infrastructure that isn’t focused on achieving a particular long-range target but on process and product measurements that support adaptation and improvement.

-Implemented via Hackystat

-Gathers data about development, building, execution and user interaction

-Telemetry Display Language that we use with the Hackystat Web server to interactively define telemetry streams and specify how practitioners should compose them together into charts and reports

-Telemetry Control Centre: Screen set up constantly displaying various scenes in an office

-makes all the metrics more comparable, accessible, and current.

Drawbacks

-many legitimate and productive activities, including meetings, email, and hallway conversations, are outside the telemetry-based measurement’s scope

-Telemetry data’s decision-making value is only as good as the data that sensors can obtain

**-In-text:**(Sillitti et al., 2003)

**Your Bibliography:**Sillitti, A., Janes, A., Succi, G. and Vernazza, T. (2003). Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data. *Proceedings of the 29th EUROMICRO Conference “New Waves in System Architecture*. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.6806&rep=rep1&type=pdf [Accessed 25 Oct. 2018].

-PROM (PRO Metrics), an automated tool for collecting and analyzing software metrics as well as personal software process (PSP) data

-There are two kinds of measures in engineering: product metrics and process metrics. The former describes product’s qualities such as dimensions, physical data, etc; the latter describes process’ qualities such as effort required, production time, etc. In software, the former includes code length, complexity, reusability, maintainability, the latter includes editing time, number and type of changes in a class or in a file, etc

-Difficulty in collecting metrics

-time expensive task

-manual data collection is an unreliable

-Personal Software Process (PSP) [7] provides a rigorous methodology to monitor software development from the early stages of a project to the shipped product.

-continuous monitoring helps developer to trace their own performances, compare them to the planned schedule, and find out whether specific elements (i.e. environment, behaviors, interactions, etc.) affect them

-Hackystat is a framework that allows individuals to collect and analyze PSP data automatically

-Info given to developers to improve themselves

-PROM helps managers in applying ABC collecting data.

-Activity-Based Costing (ABC) to manage costs. It is very difficult to apply such methodology in software development because human activities are hard to track

-Managers cannot access single developer’s data due to privacy issues but they can access to the same information in an aggregated form that summarize the status of the whole project

-PROM collects and analyzes data at different levels of granularity: personal, workgroup and enterprise. This differentiation takes a picture of the whole software company and preserve developers’ privacy providing to managers only aggregated data.

-Designed to be extensible to new and different IDEs as well as lowing devs to work offline

-Composed of ;

-PROM Database

-PROM Server: Interfaces the database with high-level commands

-Plug-ins Server: Collects data from all plug-ins sending results to PROM server for offline use and devs keeping track of work on other devices

-Plug-in: Collects and sends data to Plug-Ins Server with timestamps & user identification

-PROM Server also allows for simple data analysis and visualisation

-Also supports manual data insertion via web page.

-managers do not need to know projects details do to their job. So are only able to see aggregated data

-The aim of PROM is the collection and analysis of the whole development process including activities not strictly related to coding and target users include all members of the development team (including developers, managers, etc.). Hackystat focuses on the coding activity and target users are developers.

-PROM is useful to monitor and improve the whole development process; Hackystat focus on the improvement of the performances of the single developer.

**-In-text:**(E. Hassan and Xie, 2010)

**Your Bibliography:**E. Hassan, A. and Xie, T. (2010). Software Intelligence: The Future of Mining Software Engineering Data. *FoSER 2010*. [online] Available at: https://people.engr.ncsu.edu/txie/publications/foser10-si.pdf [Accessed 25 Oct. 2018].

-Software Intelligence (SI) as the future of mining software engineering data, offers software practitioners (not just developers) up-to-date and pertinent information to support their daily decision-making processes

-many decisions related to a software system are based on intuition and gut feeling

- leads to wasted resources and increased cost of building and maintaining large complex software systems.

-as systems age and as developers move across companies individuals with high knowledge about a system becomes far rarer and low amounts of documentation means ad hoc decision making is less feasible

-Mining Software Repositories field analyzes and crosslinks the rich data available in software repositories to uncover interesting and actionable information about software systems and projects

-While software repositories are often used in practice as recordkeeping repositories, they are rarely used to support decision-making processes.

-By transforming these repositories from static record-keeping repositories into active ones, we can guide decision-making processes in modern software projects.

Suggestions

-Software Intelligence should be applied beyond code developing to managers, deployers, support teams etc.

-Should use not only historical data repositories but also developer interaction

-SI should adapt more data mining algorithms and outlooks by collaboration

-SI tools need to be easily explained and understood for organisations to feel confident in these systems

-Potentially building SI tools over BI tools could help advance the field further