Der Transparente Mensch: Die unvermeidliche Preisgabe von Metadaten

Arne Beer, MN 6489196, University of Hamburg

20.08.2019

1 Abstract

Metadaten spielen eine bedeutende Rolle in der Welt der Datenanalyse. Simple Informationen über einen Menschen, wie zum Beispiel der Standort verknüpft mit Zeitpunkten, verschiedene Aktivitäten oder Vorlieben, liefern enormes Potential, um diesen zu analysieren und kategorisieren. In vielen Fällen wird aktives Sammeln von Daten, Data Mining und Big Data Analysis betrieben und Plattformen wie Facebook, haben sich voll diesem Ziel verschrieben.

Metadaten sind jedoch tückisch und befinden sich an viel mehr Orten als man vielleicht auf den ersten Blick vermutet.

Dieses Paper wird sich speziell mit der Intransparenz in Bezug auf die Freigabe von Metadaten bei bestimmten Tools auseinandersetzen, welche eigentlich nicht für diesen Zweck vorgesehen sind. Speziell betrachten wir in diesem Falle das Tool *Git*, welches hauptsächlich zur Versionskontrolle von Quellcode in Informationstechnischen Projekten verwendet wird. Zudem wird die populäre Website *GitHub* betrachtet, welche als Open-Source Plattform dient, auf der jeder Entwickler seine Projekte öffentlich zur Verfügung stellen kann.

2 Einleitung

Das Ziel dieser Arbeit ist, die Notwendigkeit von Datenerfassung zu betrachten. In einigen Umfeldern, speziell im Informationstechnischem Bereich, ist die Erfassung von Metadaten unvermeidlich und teilweise unabdingbar. Daten werden zur Analyse von technischen Prozessen benötigt oder um z.B. die Verantwortlichkeit eines Entwicklers für eine bestimmte Änderung an einem System festzuhalten.

Leider lassen sich aus simplen Metadaten jedoch häufig mithilfe von Data Mining Techniken mehr Informationen als auf den ersten Blick ersichtlich. Hierbei kann es dazu kommen, dass durch simple Hilfsmittel, welche lediglich die Produktivität und Benutzbarkeit eines Werkzeugs verbessern sollen, private Informationen über den Nutzbar nach außen hin ersichtlich werden. Falls diese Daten nun zusätzlich öffentlich einsehbar sind, können diese von einer nicht kontrollierten Instanz benutzt werden.

Aus diesen Umständen entsteht ein Dilemma, welches aus dem Konflikt zwischen der Notwendigkeit Daten zu erheben und zu veröffentlichen und der unmöglichen Kontrolle über die Verbreitung dieser Daten besteht. Im Folgenden wird dieses Problem in Bezug auf Privatheit und Transparenz am Beispiel der professionellen Open-Source Community und dem Tool Git näher betrachtet.

3 Git und Github

Diese Arbeit erläutert die vorliegende Problematik am Beispiel des Tools *Git* und der Plattform *GitHub*. Hierzu werden im Folgenden die beiden Technologien vorgestellt, wichtige Aspekte erörtert und zusammengefasst.

3.1 Git

Git ist ein heutzutage als Standard angesehenes Werkzeug zum Entwickeln von Projekten im Informationstechnischen Sektor. Beinahe jedes Projekt mit Quellcode besitzt eine Art von Version Control System (VCS), und in den meisten Fällen wird diese Rolle von *Git* eingenommen. Ein VCS ist ein Hilfsmittel, welches

es erlaubt den Quellcode zu versionieren, also zu bestimmten Zeitpunkten eine Version des momentanen Standes des Projekts festzuhalten. Ein Projekt, welches von Git verwaltet wird, wird im Fachjargon ein *Repository* genannt.

Durch diese Versionierung, ist es den Entwicklern des Projekts möglich schnell zwischen verschiedenen Versionen hin und her zu wechseln. Sollte also zum Beispiel ein neues Feature einer Software fehlerhaft sein und nicht funktionieren, kann mithilfe eines Befehls ohne weiteren Aufwand zu der vorherigen stabilen Version des Projektes gewechselt werden. Eine solche Version wird in Git als *Commit* bezeichnet. Ein *Commit* kann nun wiederum auf einen oder mehrere andere *Commits* zeigen, welche die *Parents*, also deren Vorfahren, bezeichnet werden.

Wenn ein neuer *Commit*, also eine neue Version, erstellt wird, zeigt dieser also immer auf die vorherige Version, von der dieser abgeleitet wurde. Durch diese Verbindung zwischen den *Commits* lässt sich folglich die komplette Historie des Projektes ableiten und zu jedem Zeitpunkt des Projektes zurückspringen. Diese Struktur wird in Git die *History* genannt, welche man als gerichteten azyklischen Graphen darstellen kann.

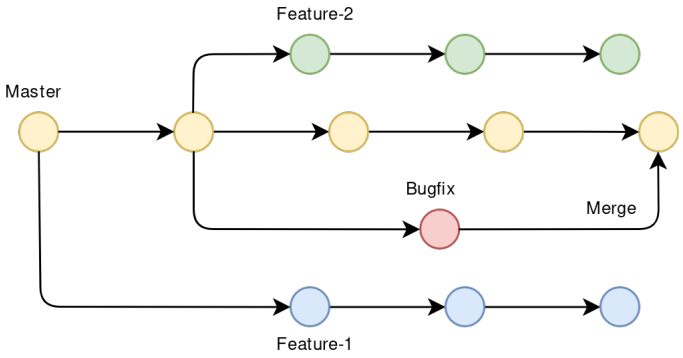

Figure 1: Eine beispielhafte Darstellung einer möglichen *History* in Git.

Wenn man sich nun jedoch einen solchen *Commit* genauer anschaut, fallen einige sehr interessante Details auf. In Listing 1 kann die Struktur einer solchen Datei gesehen werden und es sind mehrere direkte Identifikatoren und Metadaten zu sehen. Zum einen ist der volle Name und die Email Adresse ersichtlich, welche

vermutlich zu einer vollständigen Identifikation ausreichen würden. Zudem ist Timestamp mit dem momentanen UTC Offset des Erstellers des *Commits* eingebunden.

```
1     tree      cd7d001b696db430b898b75c633686067e6f0b76
2     parent    c19b969705e5eae0ccca2cde1d8a98be1a1eab4d
3     author    Arne Beer <test@eintest.de> 1513434723 +0100
4     committer Arne Beer <test@eintest.de> 1513434723 +0100
5
6     Chapter 2, acronyms
```

Listing 1: Eine Git *Commit* Datei. Auf unterster Ebene wird ein *Commit* nur durch eine Textdatei in diesem Format dargestellt.

Es ist folglich für jede neue Version zu sehen, wer diese erstellt hat, wann er sie erstellt hat und in welcher Zeitzone sich diese Person zu diesem Zeitpunkt aufgehalten hat. Ebenfalls verbunden mit einem Commit sind alle Änderungen, die der Autor zwischen dieser und der letzten Version vorgenommen hat.

3.2 GitHub

GitHub ist die zur Zeit größte Open-Source Plattform mit über 96 Millionen öffentlichen Repositories. Es ist eine Seite, die zur Verbreitung von Wissen, öffentlicher Software, zum Lernen sowie zum entwickeln kommerzieller Projekte verwendet wird. Neben dem bloßen Bereitstellen einer Plattform, werden zudem Tools angeboten, welche die Kollaboration zwischen Entwicklern vereinfacht. Dadurch wird die Hürde zum Beitragen an fremden Projekten deutlich gesenkt und dementsprechend sogar Kollaboration gefördert.

Github ist jedoch nicht einfach nur eine Plattform zum Verteilen von freier Software, sie wird ebenfalls benutzt um sich selbst als Entwickler zu profilieren. Viele Software Developer partizipieren in der Erstellung oder Weiterentwicklung von Projekten aus Leidenschaft heraus, allerdings jedoch auch mit dem Wissen, dass diese Beiträge ihr Resume darstellen und dadurch direkten Einfluss auf den Marktwert ihrer Fähigkeiten hat.

Zudem bietet Github einige Features, die an Funktionalitäten aus größeren Social-Media Plattformen erinnern. So erlaubt Github es seinen Nutzern Sterne zu vergeben, wenn ihnen ein Projekt gut gefällt. Diese Sterne werden zudem gerne als Maßstab für den Erfolg und die verbreitete Nutzung eines Projektes verwendet.

4 Transparenz

Dieser Abschnitt befasst sich mit dem Begriff der Transparenz in Bezug auf das Tool Git und die Plattform GitHub. Transparenz wird hier in zwei verschiedenen Konnotationen benutzt werden. Zum einen wird überprüft in wie weit die Veröffentlichung von Metadaten transparent dargelegt wird. Zum anderen wird untersucht, in wie weit der Zugriff und die Verbreitung der veröffentlichten Daten verboten oder zugelassen wird.

4.1 Benutzung

Der Begriff der Transparenz in Bezug auf die Nutzung von Git, bezieht sich in erster Linie darauf, ob klar kommuniziert wird, dass potentiell persönliche Daten veröffentlicht werden und falls ja, welche Art von Daten veröffentlicht werden.

Hier spielt die Komplexität eines Programmes eine große Rolle. So wird es zunehmend schwerer die Auswirkung einer Handlung abzuschätzen, speziell mit der zunehmenden Komplexität des unterliegenden Programms (Floridi and Taddeo, 2016, p. 2).

Git wird in erster Linie als Commandline Tool oder noch höher abstrahiert in einem grafischen Interface verwendet. Auf der Commandline werden bestimmte Befehle auf der Konsole eingegeben, welche interpretiert werden und anschließend die gewünschte operation ausführen. In einem grafischen Interface wird stattdessen mithilfe von Knöpfen und Visualisierungen eine interaktive Oberfläche erstellt, welche jedoch nicht die Feingranularität und bei weitem nicht alle Optionen der Commandline Applikation zur Verfügung ermöglichen. Ein komplexes Programm immer weiter zu abstrahieren birgt stets die Gefahr weniger von der tatsächlichen Funktionalität zu verstehen und sich der Technologie auszusetzen. Doch selbst

beim benutzen des Commandline interfaces, sind die tatsächlichen Metadaten, die bei diesen Operationen erstellt werden, beim alltäglichen Gebrauch nicht ersichtlich und meist nicht notwendig für die Benutzung des Programms. Daher wird auch häufig nicht über diese aufgeklärt. Für interessierte Entwickler, gibt es ein öffentliches Buch, welches genauer auf die unterliegenden Funktionalitäten und Methoden von Git eingeht, jedoch ist dieses wie bereits gesagt keine notwendige Lektüre (Chacon and Straub, 2014).

Auch bei dem weiteren Durchsuchen der offiziellen Hilfe Seite von Git, findet sich keine Hinweis oder Warnung darüber, dass Daten nach außen hin veröffentlicht werden könnten ("Git man page", 2019).

Es wird offensichtlich nicht klar kommuniziert, welche Daten von Entwicklern aufgezeichnet werden und wie diese potentiell missbraucht werden könnten, wenn sie an die Öffentlichkeit gelangen.

4.2 Kontrolle

Die Analyse von Personenbezogenen Daten, kann zu einer Menge an Informationen über die verschiedenen Interessen, Herkunft, Erfahrungen einer Person ergeben (Cristle, 2017, p. 4-5). Im Kontext von Git und GitHub, wird schnell ersichtlich, dass durch die Veröffentlichung von Git Repositories, eine Menge an persönlichen Daten freigegeben wird.

Die volle Transparenz eines Nutzers kann große Nachteile nach sich ziehen, besonders wenn der Nutzer sich nicht direkt klar über die Ausmaße der Transparenz ist. Wenn ein System ohne eine klaren Vorstellung implementiert wird, in welcher klar geregelt ist, warum welche Daten veröffentlicht werden sollten, kann die Transparenz die Privatsphäre des Nutzers bedrohen (Ananny and Crawford, 2016, p. 6)

GitHub stellt eine API zur Verfügung, die es erlaubt die Website programmatisch zu erkunden und mit dieser via Code zu interagieren. Durch diese API ist es möglich mit mit relativ guter Präzision die meisten Repositories zu der eine einzelne Person oder eine Gruppe von Personen beigesteuert hat, zu finden und von diesen Daten zu extrahieren (Beer, 2017, p. 14-16).

Dieser Mechanismus ist zwar nicht perfekt, aber reicht in vielen Fällen aus, um eine ausreichende Menge an Daten für verschiedene Analysen zu erfassen. Zudem bietet die Web-Oberfläche eine Visualisierung der kompletten Aktivität eines Nutzers. Diese ist zwar nicht über die API einsehbar, jedoch könnte man mithilfe eines Scrapers diese Oberfläche benutzen um sämtliche öffentlich einsehbaren Daten eines Nutzers der Plattform zu sammeln.

Es herrscht kein wirklicher Mechanismus um großflächiges Sammeln von Daten zu verhindern und es existieren sogar bereits Projekte, wie zum Beispiel GHTorrent, welche alle Daten von Github live beobachten und jedem öffentlich zur Verfügung stellen. Dadurch kann jeder, auch Personen ohne größeres technisches Verständnis, diese Daten herunterladen und benutzen.

Durch Daten dieser Art, ließ sich zum Beispiel eine großangelegte Studie durchführen, ob Software Entwickler in den Mozilla Open-Source Projekten eher während der Nacht oder am Wochenende entwickeln (Claes, Mäntylä, Kuutila, and Adams, 2018).

Jedoch lassen sich diese Analysen nicht nur auf eine Gruppe von Personen anwenden, sondern auch auf eine Einzelperson. So war es zum Beispiel möglich mithilfe von Github gewonnenen Daten sehr präzise die Urlaubs und Krankheitsausfälle mehrerer Arbeiter eines Unternehmens zu analysieren (Beer, 2017, p. 37-39). Diese Daten können hoch sensible sein, da durch die Veröffentlichung dieser potentiell Nachteile auf dem Arbeitsmarkt entstehen können. Falls ein Nutzer innerhalb eines Jahres vermehrt an einer Krankheit gelitten hat, sollte sich dies nicht negativ auf seinen Wert auf dem Arbeitsmarkt auswirken.

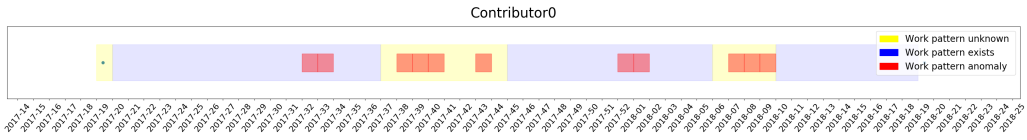

Figure 2: Analyse eines Mitarbeiter auf Krankheitsfälle und Urlaub. Rote Blöcke sind garantierte Fehltage. Blaue Abschnitte stellen normal erkennbare Werktagsmuster dar. Gelbe Abschnitte stellen Anomalien dar, die jedoch nicht als ganzer Ferien- oder Krankheitsausfall erkannt werden(Beer, 2017, p. 37-39).

Wenn nun weitere Daten von GitHub eingebunden werden, könnte ein sehr genaues Profil eines aktiven Entwicklers erstellt werden, welches die Schlafzeiten, Krankheitsausfall, beherrschte Technologien, die Erfahrungen mit einer bestimmten Technologie und Produktivität umfassen könnte. Auch wenn Punkte wie Technologien und Erfahrung eventuell gewünscht dargestellt werden, sind andere Aspekte wie Schlafzeiten und allgemeine Produktivität eventuell ungewünscht einsehbar. Solche Daten können immer missbraucht werden, um auf den Lebensstil und die momentanen Umstände einer Person Rückschlüsse zu ziehen.

Ein solcher Datensatz wäre beispielsweise perfekt geeignet um gezielte Rekrutierungen von Entwicklern mit einem bestimmten Skill Set durchzuführen.

Ebenfalls ist vorstellbar, das durch die Analyse von Repositories Mitarbeiterüberwachung durchgeführt werden kann. Ein Repository ist ein beinahe perfektes Logbuch, in welchem die gesamte geleistete Arbeit eines jeden Programmierers an einem Projekt einsehbar ist. Durch die Anzahl der Commits und die Länge und Qualität des hinzugefügten Codes könnte die Effizienz eines Mitarbeiters bestimmt werden und Mitarbeiter, welche eine bestimmte Quota nicht erfüllen, könnten Opfer einer Benachteiligung werden. So existieren bereits mehrere Methoden, um die Effizienz eines Entwicklers anhand von geschriebenen Zeilen von Code zu berechnen (y Parareda, 2007).

Wie z.B. die Snowden Enthüllung gezeigt hat, werden Daten heutzutage im großen Stil gesammelt, von den verschiedensten Quellen und sie werden auf auf bisher nie dagewesene Arten miteinander verbunden und auf neue Weisen angewendet (Lyon, 2014, p. 4). Eine Anwendung zur Analyse von Git repositories und Github im professionellen Sektor ist nicht unrealistisch und höchstwahrscheinlich schon vorhanden.

5 Privatheit

Um die Auswirkung auf die Privatsphäre einer Person, betrachten wir hierzu zuerst eine Definition der Privatsphäre von W.A. Parent.

> Privacy is the condition of not having undocumented personal knowl-

> edge about one possessed by others. A person's privacy is diminished exactly to the degree that others possess this kind of knowledge about him. Documented information is information that is found in the public record or is publicly available (e.g. information found in newspapers, court proceedings, and other official documents open to public inspection) (Parent, 1985, p. 203).

Mit dieser Definition und dem Wissen aus dem vorherigem Abschnitt, lässt sich erschließen, dass wir es in unserem Fall mit dokumentierter Information haben. Zudem handelt es sich um volle Transparenz in Bezug auf die veröffentlichten Projekte und die Arbeitszeiten und Fähigkeiten des Nutzers.

Der Nutzer hat zwar stets die Möglichkeit seine Projekte zu verbergen oder erst gar nicht auf GitHub oder einer anderen Seite zu veröffentlichen, jedoch würde dies gegen das Grundsätzliche Prinzip von Open-Source sprechen. Der gesamte Sinn hinter GitHub ist, eine Plattform für die freie Verbreitung von Wissen zu bieten, welches ein nobles moralisch wünschenswertes Ziel ist.

6 Diskussion

Die Erfassung von Daten, speziell in der Informationstechnischen Umgebung, ist häufig zwangsweise Notwendig und gewünscht (Ananny and Crawford, 2016, p. 6). Beim Beispiel von Git, wird durch Erfassung von Daten Einsicht und eine verbesserte Kontrolle über das Projekt ermöglicht.

Zusammenarbeit mit mehreren Parteien wird erleichtert, da stets klar ist, welche Partei für welche Änderungen verantwortlich sind. Im Zweifelsfall, kann stets die verantwortliche Person angeschrieben und gefragt werden, wenn sich Fehler ergeben sollten. Auf der gegenüberliegenden Seite, kann jedoch auch stets die verantwortliche Person zur Rechenschaft gezogen werden, da jede Änderung fein säuberlich dokumentiert ist.

Die Grundintention anderen Menschen zu helfen und Produkte jahrelanger kostenloser Arbeitszeit frei zur Verfügung zu stellen ist wie bereits erwähnt durchaus nobel und sollte gesellschaftlich angesehen und belohnt werden.

Es entsteht das Dilemma zwischen dem Willen zu helfen und der gleichzeitigen Bestrafung durch Einschränkungen der Privatsphäre durch Veröffentlichung von Daten.

Das Verschleiern von Daten im Open-Source Umfeld würde eventuell dabei helfen die Privatsphäre von Entwickler zu verstecken, als Gegenzug würde darunter jedoch sehr die Kollaboration und Konversationen in der Community leiden, welche das Rückgrad derselben bilden.

7 Zusammenfassung

Das Dilemma zwischen Veröffentlichung und der Notwendigkeit Daten zu erfassen, um ein System lauffähig zu halten lässt sich nicht ohne weiteres lösen. In jedem Fall leidet einer der beiden Standpunkte. Sobald striktere Einschränkungen auf die Erfassung von Daten angewendet, leidet sofort die Nutzbarkeit der Tools und die Kollaborationsfähigkeit der Open-Source community darunter.

Es gibt keine offensichtliche Strategie, mit der sich die Situation im Open-Source Umfeld verbessern lässt. Im professionellen Umfeld und im Gebiet der Mitarbeiterüberwachung hingegen, lässt sich mit klaren Richtlinien, der Missbrauch von Metadaten zu Analysezwecken verhindert.

In dieser Arbeit wurden zudem nur die Aspekte der Transparenz und der Privatheit angesprochen. Jedoch sollte ebenfalls die Fairness und Sauberkeit der Daten berücksichtigt werden. Git Repositories haben selten eine komplett saubere und nachvollziehbare History. Commits können beispielsweise bearbeitet, gelöscht, überschrieben und zusammengefasst werden, um nur ein paar der Möglichen Fehlerquellen zu nennen. Durch diese Fehlerquellen könnten Analysen verzerrt werden. Die Errungenschaften einer Person könnten versehentlich auf eine andere übertragen werden und ungerechtfertigte Benachteiligungen oder Anschuldigungen wären die direkte Folge einer solchen Analyse.

References

Ananny, M., & Crawford, K. (2016). Seeing without knowing: Limitations of the transparency ideal and its application to algorithmic accountability. *New Media & Society*, *0*, 146144481667664. doi:10.1177/1461444816676645

Beer, A. (2017). *Privacy implications of exposing git meta data*. Retrieved from https://github.com/Nukesor/thesis/blob/master/thesis/thesis.pdf

Chacon, S., & Straub, B. (2014). *Pro git, 2nd edition*. Apress.

Claes, M., Mäntylä, M., Kuutila, M., & Adams, B. (2018). Do programmers work at night or during the weekend?

Cristle, W. (2017). Corporate surveillance in everyday life. Retrieved from https://crackedlabs.org/en/corporate-surveillance

Floridi, L., & Taddeo, M. (2016). What is data ethics? *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, *374*, 20160360. doi:10.1098/rsta.2016.0360

Git man page. (2019). Retrieved August 25, 2019, from http://man7.org/linux/man-pages/man1/git.1.html

Lyon, D. (2014). Surveillance, snowden, and big data: Capacities, consequences, critique. *Big Data & Society*. doi:10.1177/2053951714541861

Parent, W. A. (1985). Ethical issues in the use of computers. In D. G. Johnson & J. W. Snapper (Eds.), (Chap. Privacy, Morality, and the Law, pp. 201–215). Belmont, CA, USA: Wadsworth Publ. Co. Retrieved from http://dl.acm.org/citation.cfm?id=2569.2682

y Parareda, B. M. (2007). Measuring productivity using the infamous lines of code metric.