Seminar "Open Source Software Engineering"

Wintersemester 2008

SourceForge.net Data Mining by Greg Madey et al. University of Notre Dame

Sebastian Wojtowicz

wojtowic@mi.fu-berlin.de

Betreuer: Christopher Oezbek

13.03.2009

Zusammenfassung

SourceForge.net eignet sich gut als Beispiel, wenn man die Eigenschaften der OSS Community analysieren möchte. SF hat bereits über 180,000 registrierte Projekte und mehr als 1.9 Millionen registrierte Benutzer (Stand August 2008 [wikipedia.org). Mit dem früh erkennbaren Erfolg von SourceForge.net haben sich Madey et al. von der University of Notre Dame circa 2001 das Ziel gesetzt, ein genaueres Verständnis der Strukturen und der Erfolge dieser OSS Community zu erlangen.

Ich gebe in diesem Paper einen Überblick über die einzelnen Forschungen und Ergebnisse, die Madey im Laufe der Jahre 2002 - 2007 erarbeitet und veröffentlicht hat und gehe ein auf Themen, wie Data Mining mit Hilfe von Web-Crawlern, Netzwerk Analysen auf den daraus erhobenen Daten und die Effizienzsteigerung der Analysen durch späteren Zugriff auf Daten aus den SourceForge.net Datenbanktabellen. Ich zeige wie Madey mit Hilfe von Simulationen - von vier unterschiedlichen Modellen sozialer Sturkturen - Eigenschaften der SourceForge.net Struktur verstehen und bereits vorhandene Theorien unterstützen konnte.
Weiter zeige ich, wie mit Hilfe der Datenbankdaten topologische Analysen auf der OSS Entwickler Community von SourceForge.net ausgeführt wurden und vergleiche zum Schluss globale und temporale Analysen sozialer Positionen und gebe Mögliche Erklärungen für die beobachteten Unterschiede.

Madeys et al. Paper und weitere Informationen zum SourceForge.net Research Data Archive von Madey et. al findet man auf www.nd.edu/ oss/.

Inhaltsverzeichnis

1 Einleitung

Um den Aufbau des Papers zu verstehen, werde ich erstmal die Beweggründe für selbigen erklären.

Meine Ziel war es, ein Paper zum Thema „Data Mining 4 (Source Force): Greg Madey (Notre Dame)“ anhand zwei oder drei anderer Paper von Greg Madey et al. zu erstellen.
Mehr oder minder schnell schien mit Hilfe des Betreuers ein gutes Paper gefunden zu sein. Verständlicherweise kamen dabei auch Fragen auf, die ich folglich mit Hilfe oben erwähnter zwei oder drei anderer Paper zu klären versuchte. Doch dabei entstand ein Problem. Je mehr ich mich am Verstehen und Lesen neuer Paper versuchte, umso mehr neue Fragen kamen auf, was auch verständlich ist, jedoch waren es stets mehr neue, als dass alte beantwortet wurden.
Mal waren es nicht ausreichend definierte Begriffe, wie die „Handyperson“[CM07], mal entstand die Frage nach dem Unterschied solcher Definitionen, ob z.B. die Begriffe „Software User, Project Administrator, Software Developer, Task Management, Bug Reporter, Feature Requester, Handyperson“[CM07] eine Verfeinerung oder etwas ganz anderes als die Developer Rollen „Project Leader, Core Developer, Co-Developer, Active User“[XCM06] sind, womit gleich die nächste Frage enstand, woher diese neuen / anderen Rollen stammen.

Da die Antworten mit den Begriffen und Details in den Papern von Madey et al. stets mit unterschiedlichem Grad auf die Paper verteilt waren - Begriffe mal im chronologisch ersten Paper, mal im chronologisch letzten Paper bzgl. des gleichen Themas erklärt wurden und teilweise in thematisch völlig ungleichen Papern - kam ich zu dem logischen Entschluss die rekursive Herangehensweise aufzugeben und statt dessen alle Paper in chronologischer Reihenfolge zu lesen, womit ich auch zum chronologischen Aufbau meines Papers mit drei grobgeteilten Zeitabschnitten (Phasen) kam.

- In Phase 1 hatte Madey nur Daten zur Verfügung, die ein Webcrawler sammeln konnte, hauptsächlich Zuordnungen von Developern zu Projekten. Mit diesen Daten untersuchte er das soziale Netzwerk.
- In Phase 2 hatte Madey bereits Zugriff auf direkte Kopien der in der SF.net Datenbank enthaltenen Tabellen (ca. 100), sowie CVS Daten und führt erste Untersuchungen auf den User Aktivitäten aus. Damit war er in der Lage, die Benutzer in 5 unterschiedliche Rollen zu teilen.
- In Phase 3 gab es bereits eine große Spanne obiger Daten, mit deren Hilfe Madey et al. dann temporale und globale Untersuchungen hinsichtlich der sozialen Strukturen des SourceForge.net Netzes durchführen und auswerten konnte.

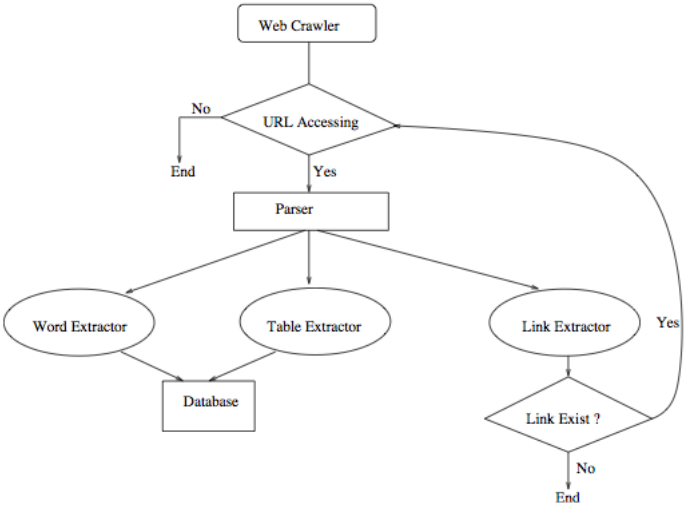


Abbildung 1: Funktionsweise eines Web Crawlers. Bild von [XHM03]

2 Phase 1

2.1 Erste Datensammlung

Madey et al. brauchten am Anfang Geduld. Am Januar 2001 fingen Madey et. al (im Folgenden zur Vereinfachung meist nur noch im Singular als Madey bezeichnet) mit einem Perl Skript an, die SourceForge.net Seiten zu durchsuchen, was man sich entsprechend Abb. 1 vorstellen kann.

Die ersten Daten, die Madey derart sammelte und analysierte, waren hierbei Zugehörigkeiten von Entwicklern zu Projekten. So erhält man beispielsweise unter http://sourceforge.net/project/memberlist.php?group_id=192583 eine Liste aller Member des Projektes 192583 (Skim PDF Reader fuer OSX). Dort wiederum hat man die Möglichkeit per Link auf die entsprechenden Memberseiten zu gelangen und so an die MemberID ranzukommen.

Dieses Prinzip wurde mit Hilfe eines Webcrawlers automatisiert. Dieser hat die SourceForge.net Seiten „wühlt“ und aus den erhobenen Daten eine Liste wie in Abb. 2 erstellt. Anschließend wurden diese Daten in eine Datenbank übertragen. Die ersten Daten, bzw. deren Ergebnisse veröffentlichte Madey dann 2002 - mit Crawlingdaten von Januar 2001 bis März 2002. Ich werde im Folgenden darauf eingehen.

8001|dev378
8001|dev8975
8001|dev9972
8002|dev27650
8005|dev31351
8006|dev12509
8007|dev19395
8007|dev4622
8007|dev35611
8008|dev7698

Abbildung 2: Liste mit Projekten links, Developern rechts. Bild von [MFT02b]

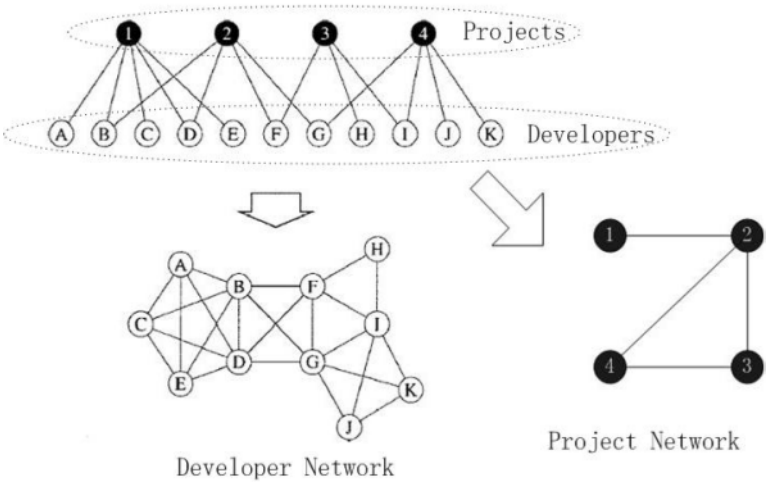


Abbildung 3: Liste mit Projekten links, Developern rechts. Bild von [Mad03]

2.2 Bipartite Graphen und Soziale Netze

Mit den (oben erwähnten) gesammelten Daten (Abb. 2) hat man die Möglichkeit einen Bipartiten Graphen - bestehend aus Entwickler-, sowie Projektknoten - zu erstellen. Die Kanten zeigen, ob ein Developer am entsprechenden Projekt beteiligt ist. Siehe Abb. 3. Weiterhin lässt sich so ein Netzwerk in zwei andere Netzwerke umwandeln: in ein Developer Netzwerk und in ein Projekt Netzwerk. Ein Developer Netzwerk besteht nur aus Developer Knoten. Eine Kante zwischen zwei Developer-Knoten exisitiert, falls beide Developer am selben Projekt beteiligt sind. Beim Projekt Netzwerk ist es genau andersrum.

So verfuhr auch Madey und hatte folglich die Möglichkeit das Soziale Netzwerk der Developer (und Projekte) zu betrachten und zu analysieren.
Dabei spielten Werte, wie diameter, clustering coefficient, degree distribution eine wichtige Rolle für das Verständnis des Netzes und damit der SourceForge.net Developer Struktur und damit nach Madey auch der OSS Struktur im Allgemeinen, wie man bereits an Hand des Namens des ersten veröffentlichten Papers „Understanding OSS as a Self-Organizing Process“ [MFT02c] ablesen kann.

Bei diesen Analysen machte Madey nun mehrere Beobachtungen:

- Beim Verhältnis Entwickler $\leftrightarrow$ Projekte war eine exponentielle Verteilung zu beobachten.
  Das heißt z.B., dass es sehr viele Projekte gibt mit nur einem Entwickler, und nur sehr wenig (gegen 1) Projekte mit sehr vielen Entwicklern (siehe Abb. 4). Diese Analysen, bzw. Grafiken findet man immer wieder in unterschiedlichsten Papern von Madey.

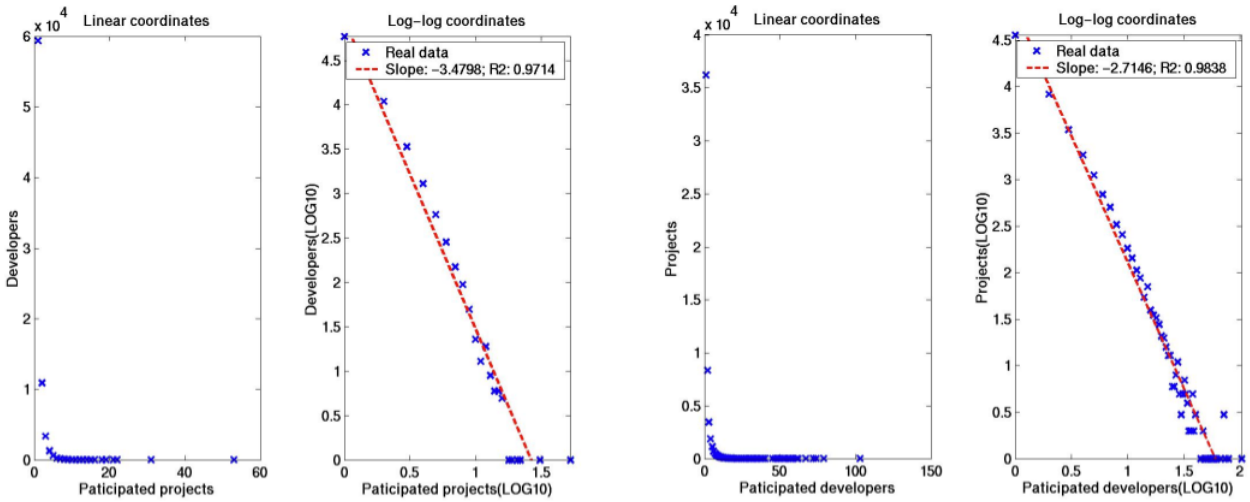


Abbildung 4: Verteilung Projekte auf Developer. Bild von [Mad03]

- Weiterhin war erkennbar, dass die 10 von SourceForge.net höchstgerankten Projekte im Schnitt 20 mal mehr Entwickler pro Projekt hatten (Daten von 2002), als die restlichen Projekte [MFT02b].

Damit war bereits erkennbar, dass es sich beim nun visualisierten Netzwerk von SourceForge.net nicht um ein Netzwerk mit randomisierten Verknüpfungen handelt (in dem Fall wu"rden neue Developer zufällig irgendwo mitmachen), sondern um einen Graphen mit „bevorzugter Anfügung“[MFT02b]. Das heißt, die großen Projekte werden immer größer (der „rich-get-richer“ effect).
Also analysierte man auch die Grapheneigenschaften, wie z.B. den Diameter, den Clusterkoeffizient, den durschnittlichen Grad oder den durschnittlichen Diameter mit folgenden Ergebnissen [GFM03a].

- Für den Diameter, welcher den maximalen aller kürzesten Pfade zwischen allen Knotepaaren darstellt, wurde im Entwickler Netzwerk mit über 70.000 Knoten ein Wert zwischen 6 und 8 festgestellt. Im Vergleich: jeder Mensch ist mit jedem anderen Menschen in den USA im Schnitt über eine Kette von 6 Personen verbunden [wik09].

- Für den Clusterkoeffizienten, welcher den Anteil der vorhandenen Kanten über alle potentiell möglichen Kanten zwischen Nachbarknoten/Cliquen darstellt, wurde ein Wert von 0.7 festgestellt. Im Vergleich: bei einem Netzwerk mit zufälligen Verknüpfungen neuer Developer wären es circa 0.2.

- Für die Grad Verteilung (z.B. wieviele Developer sind an einem Projekt beteilig, wieviele Developer an zwei, wieviele an drei, etc.) wurde festgestellt, dass der Graph fast einer Linie folgt, wenn man den Graphen in log-log Koordinaten abbildet, wie es in Abb. 4 zu erkennen ist. Auch diese Eigenschaft wäre bei einer zufälligen Anfügung nicht zu erwarten.

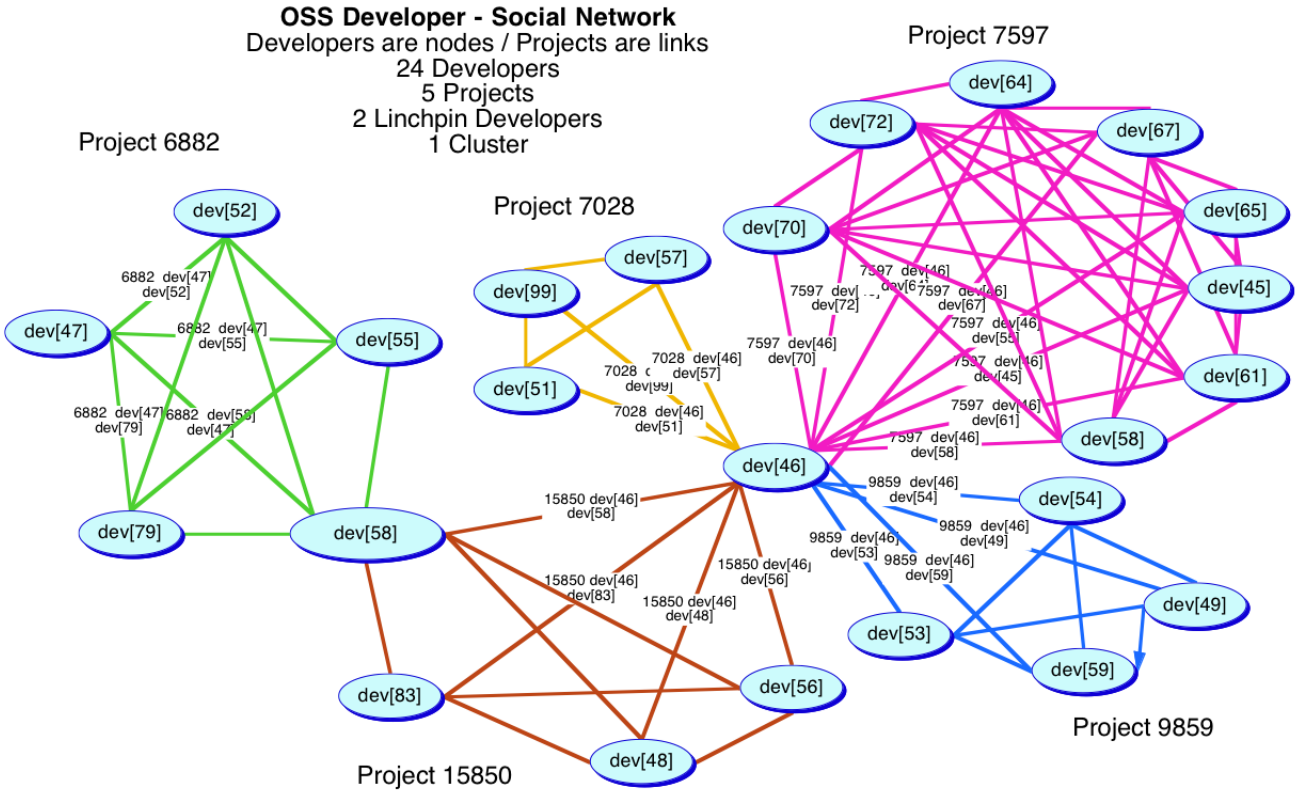


Abbildung 5: Cluster mit 5 Projekten. Bild von [MFTH03]

- Bei evolutioneller Betrachtung war zu erkennen, dass der durschnittlicher Grad aller Knoten wächst, was man zum Beispiel so deuten könnte, dass Leute sich eher bereits vorhandenen Projekten anschließen, als eigene Projekte zu erstellen oder/und Entwickler nicht so schnell von Projekten abspringen, wie sich andere Entwickler bei irgendwelchen Projekten mit anmelden, oder aber, dass die Entwickler einfach zu faul sind sich abzumelden, auch wenn Sie bereits mit dem einen Projekt nichts mehr zu tun haben, dafür allerdings bei anderen Projekten anfangen. Erkennbarer Fakt ist jedoch, dass das Netz dichter wird.

Die Folge dieser Tatsachen ist, dass mit dieser Dichte und den Cluster verbindenden „linchpin“ (Dreh- und Angelpunkt) Knoten (siehe dev[46] und dev[58] in Abb. 5), ein kleiner Diameter für das Netzwerk entsteht und sich Informationen und Technologie im Netz somit sehr schnell verteilen und verbreiten können [MFT02b].

2.3 Simulationen von 4 Modellen

Anhand oben gewonnener empirischer Daten wollten Madey et al. als nächstes ein Model finden. Ein Model, dass - sobald simuliert -, möglichst identische Werte hervorruft,

wie zuvor analysiert. Ziel eines solchen Vorhaben ist es dabei erstmal nicht, ein Modell zu entwerfen, um damit die Zukunft vorherzusagen [MFT02a], sondern erstmal ein Verständnis zu entwickeln, wie und warum die einzelnen Elemente des Systems so ein emergentes Verhalten hervorrufen. Sollte allerdings ein passendes Modell gefunden werden, so wäre auch die Vorhersage der weiteren Entwicklung der SourceForge.net Community im Anschluss möglich [GM07].

Madey et al. nutzten für die Simulation von SF.net mehrere unterschiedliche Modelle. Angefangen haben sie hierfür mit der Implementierung aller drei populären komplexen Netzwerkmodelle[GFM03b]. Diese folgen den gleich erläuterten Theorien. Der Ablauf dieser Untersuchungen ist grob in Abb. 7 vogestellt.

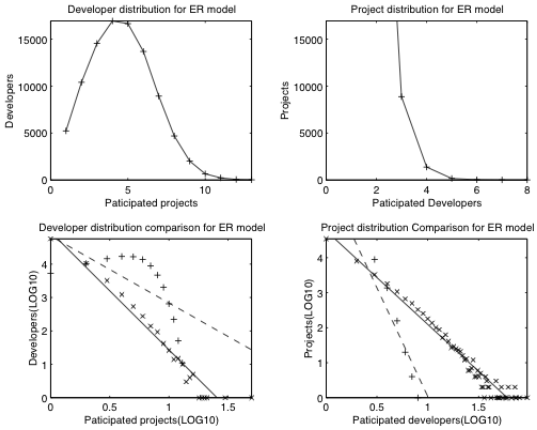


Abbildung 6: Grad Verteilung im ER Netzwerk. Bild von [GFM03b]

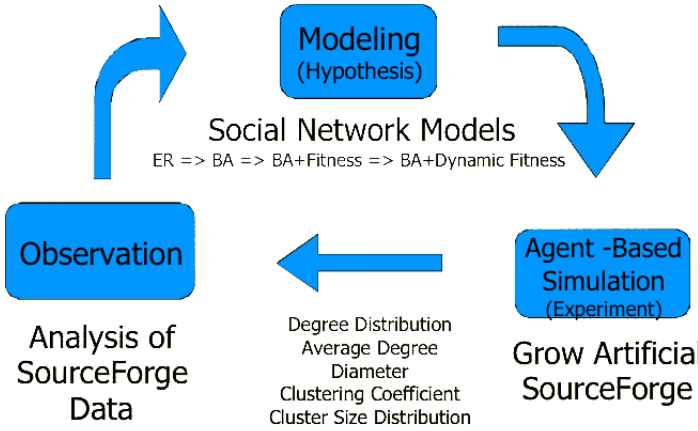


Abbildung 7: Modellierungs Zyklus. Bild von [Mad03]

- „Random Network“ Theorie, implementiert als ER(Erdös und Rényi) Modell.
- „Scale Free Network“ Theorie, implementiert als BA (Barabási und Albert) model
- „Scale Free Network with Fitness“ Theorie, implementiert als BA (Barabási und Albert) model

In der ersten Simulation und damit der Realisierung des ER Modells, wurde ein zufälliges Beitreten neuer Entwickler simuliert. Die Ergebnisse wurden mit den oben erwähnten empirisch gesammelten und analysierten Daten verglichen.
Das Ergebnis war, dass der (simulierte) Diameter kleiner ist, aber wächst. Die Grad Verteilung der Knoten hingegen war komplett anders und erinnert optisch teilweise eher einer Normalverteilung wie in Abb. 6, im Gegensatz zu den zuvor beobachteten Verteilungen gemäß Abb. 4.

In der nächsten Simulation wurde die bevorzugte Bindung („preferential attachment“) simuliert, was Erstankömmlingen eine Vorteil verschuf. Diese Simulation implementierte damit das BA Modell und zeigte nun eine Potenz-Gesetz entsprechende Verteilung ähnlich Abb. 4 mit einem Cluster Koeffizienten nahe 0.7 - wie in den empirischen Daten - und einem Diameter bei 7 (auch ähnlich) und schrumpfend. Allerdings, und das war zu erwaten, gab es selten jüngere Ankömmlinge mit mehr inzidenten Kanten als ältere Ankömmlinge.

Also wurde ein zufälliger Fitness Parameter hinzugefügt - damit das BA Modell mit Fitness realisiert - der jüngeren Ankömmlingen die Möglichkeit gab, ältere Projekte zu überholen, wodurch die empirischen Daten nun genauer Abgebildet wurden. Allerdings immer noch nicht komplett. Was noch fehlte, war eine Art Lebenszyklus einzelner Entwickler und Projekte.

Folglich implementierten Madey et al. (ca. 2003) noch eine vierte Variante, diesmal ein BA Modell mit dynamischer Fitness. Mit der dynamischen Fitness, hinsichtlich des Lebenszyklus eines Entwicklers (und auch hinsichtlich des Projekt-Lebenszyklus) und einer über die Zeit abnehmenden Fitness (anfangs schnelles wachstum, ruhige Eigenschaften in der Mitte des Lebens, abnehmende Aktivität gegen Ende)[MFT+03] wurden nun die statistischen Eigenschaften des vorigen BA Modells beibehalten, jedoch die Entwickler- und Projekt Lebenszyklus-Muster entsprechend der empirischen Daten mit abgebildet. Damit hatte man ein gut passendes Modell gefunden, was nun schlussfolgern ließ, welche Eigenschaften für die Entwicklung des heutigen SF.net Entwickler Netzwerkes verantwortlich sein mochten:

- Entwickler registrieren sich bevorzugt bei populären Projekten.
- Entwickler und Projekte haben eine Art Fitness, wodurch neue Projekte auch mal schneller loslegen und alte überholen können, was sich auch in „the young upstart phenomenon“ [MFT+03] wiederspiegelt.
- Später hingegen können werden Entwickler auch träger werden, was durch eine dynamische Fitness zu erklären ist.

2007 veröffentlichten Madey et al. nochmal Forschungsergebnisse, welche noch übrig gebliebene kleine Unterschiede in den Ergebnissen gut machen sollten. Während die User Fitness zuvor nur in Abhängigkeit der bereits teilnehmenden Projekte berechnet wurde, sollte nun mit einer zuerst zufälligen und später auch noch dynamischen User Energie - also ein (dynamischer) Wahrscheinlichkeitswert, der entscheidet, ob der User eine Aktivität ausführt oder nicht - eine Korrektur in den Unterschieden der Ergebnisse hergestellt werden.[GM07] (für eine Beschreibung der Aktivitäten bitte weiter unten im Dokument schauen). Auch das gelang Madey et al. mit Erfolg, wobei genauere Erklärungen hier den Ramen sprängen würden; erwähnt sei jedoch, dass damit auch Ausreißer, also z.B. Projekte mit besonders vielen Teilnehmern (viel viel mehr als der Durchschnitt restlicher großer Projekte), ähnlich abgebildet werden konnten.

2.4 Ergebnisse der ersten Phase

Madey hat in ersten Jahren nur wenig Daten zur Verfügung gehabt. Im Prinzip wurden alle obigen Forschungen, mit Daten aus einer Tabelle mit nur 2 Attributen gewonnen. Dennoch sind bereits beeindruckende Forschungen damit möglich und wir haben gesehen, was ein soziales Netzwerk ist, die Rolle von Linchpin Knoten und die dadurch verbundenen Cluster kennengelernt. Wir haben auch gesehen, dass man die Graphentheoretischen Statistiken nutzen kann, um das Wachstum und die Eigenschaften eines Netzwerkes zu verbalisieren und mit genaueren Analysen und einem gewissen Verständnis Modellsimulationen bauen kann, die einem ermöglichen, zuvor entstandene Theorien überprüfen zu können, indem man empirische Daten mit simulierten Daten vergleicht. Dabei haben wir das ER Modell, das BA Modell, das BA Modell mit Fitness und das BA Modell mit dynamischer Fitness kennengelernt, sowie die Tatsache, dass letzteres Modell ein ziemlich genauen „fit“ ergab.

2.5 Weitere Forschungen

Erwähnenswert ist noch, dass auf den ebenfalls per Web-Crawler erhobenen Projektstatistiken noch weitere Forschungen ausgeführt wurden.
Diese gehen allerdings auf Clustering Algorithmen ein, mit Begriffen wie k-means, o-cluster, Bayesche Netzwerke, Adaptive Bayessche Netzwerke oder A-Priori, was in Themenbereiche anderer Veranstaltung gehören würde und auch zu weit in fremde Theorien abdriftete. Spätestens als Wikipedia und der Rest des Internets mir keine zusammengefassten Antworten mehr geben konnte, musste ich aufgeben. Interessant ist es dennoch, da man mit diesen Daten gewisse Folgerungen erstellen und finden kann. Hierfür ein in den Forschungen gefundenes Beispiel: wenn die Anzahl der Entwickler und die Anzahl der Forum Aktivität steigt, so kann man mit einer gewissen (hohen) Wahrscheinlichkeit voraussagen, dass auch die Zahl der Downloads steigen wird und damit auch das Ranking/die Popularität des Projekts.
Mit solchen Untersuchungen kann man den Erfolg und die Ursachen für den Projekt Erfolg vorhersagen, bzw. rausfinden, um ihn folglich auch auf andere Projekte übertragen. [XHM03] und [MCM05]

3 Phase 2

3.1 Einführung

Den Schnitt zwischen diesen Phasen habe ich an dem Moment angesetzt, bei dem Madey et al. das erste mal Rohdaten aus der SourceForge.net Datenbank erhalten und somit neue Analysen möglich, ausgeführt und deren Ergebnisse veröffentlicht wurden.

Während der Prozess der Datenextraktion aus den Webseiten mit Web-Bots in Phase 1, oft mehrere Tage brauchte und geplagt war von unvollständigen und fehlenden Daten[XGCM05], kam nun die Datenbank ins Spiel. Madey erhielt in dieser Phase Datenbankkopien von SourceForge.net bestehend aus stets circa 100 Tabellen. Anfangs scheinbar nur eine Kopie (Januar 2003), später (Nov. 2004 - Okt. 2008) monatlich eine Datenbankkopie. Gleichzeitig (hier konnte ich jedoch nicht den genauen Zeitpunkt und die Regelmäßigkeit bestimmen) kamen auch Daten aus den CVS Repositories hinzu [CM05b].

Diese Daten nutzte Madey primär, um Analysen auf den User Aktivitäten durchzuführen, und um Usern damit jeweils eine von 5 Rollen zuzuweisen.

3.2 Von den Aktivitäten zur Rollenverteilung

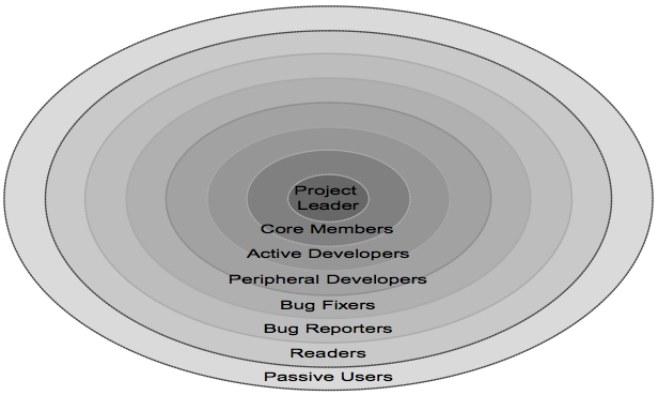


Abbildung 8: Acht User Rollen nach Kishida et al. [NYNK02]

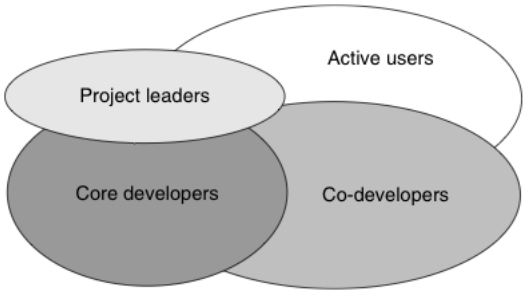


Abbildung 9: Vier Developer Rollen nach Madey et al. [XCM06]

Nakakoji et al., (auch Kishida, bekannt aus unserer Veranstaltung) entwickelten 2002 ein gewisses Rollen-Modell, dessen Bild die Teilnehmer dieses Seminars bereits aus einem anderen Paper kennen, und für uns vom Betreuer als Onion Modell identifiziert wurde (Abb. 8).
N. Xu reduzierte dieses Model laut Madey [XM04] dann auf 6 Rollen. (Hierzu sei erwähnt, dass ich Xu's Paper - genauer Master's thesis - nicht finden konnte, nur Verweise einiger Autoren auf seine Arbeit → „Xu, Neng. 2003. An Exploratory Study of Open Source Software Based on Public Project Archives. Master's thesis, John Molson School of Business, Concordia University, Montreal.“).

- User Gruppe

    1. Passive User: Keine direkte Mitwirkung am Projekt. Nur Download und Nutzen des Codes.

2. Active User: Finden und berichten Bugs, machen Feature Vorschläge, tauschen Informationen aus durch Foren- oder Mailinglisten Beiträge.

- Entwickler Gruppe

1. Periphere Entwickler (Peripheral Developer): Unregelmäßige Tätigkeiten → Bug fixes, Features hinzufügen, bieten Support, schreiben Dokumentation und tauschen (andere) Information aus.
2. Zentrale Entwickler (Central Developer): Regelmäßige Tätigkeiten → Bug fixes, Features hinzufügen, führen Patches aus (submit), bieten Support, schreiben Dokumentation und tauschen (andere) Information aus.
3. Core Developer: Umfangreiche Mitwirkung am Projekt, verwalten CVS Releases und koordinieren die peripheren und zentralen Entwickler.
4. Project Leader: Führt die Vision und Richtung des Projekts.

Madey griff wiederum dieses Modell auf und verkürzte es ebenfalls. Diesmal auf 5 Rollen, in dem er den Peripheren und den Zentralen Entwickler zu einem Co-Entwickler (Co-Developer) zusammenfasste, wodurch er die Trennung zwischen regulären und irregulären Aktivitäten beider Typen aufhob. Der passive User wird häufig ebenfalls weggelassen, da dieser für Madey kein Entwickler im klassischen Sinne ist, und Madey vor allem die Entwickler-Community untersuchen wollte. Relevant und übrig blieb damit rein eine Entwicklergruppe mit den vier Rollen gemäß Abb. 9, welche simbolisiert, dass gewisse Aktivitäten mehreren Rollen gemeinsam zugeschrieben werden können, während ein großer anderer Teil disjunkt bleibt.

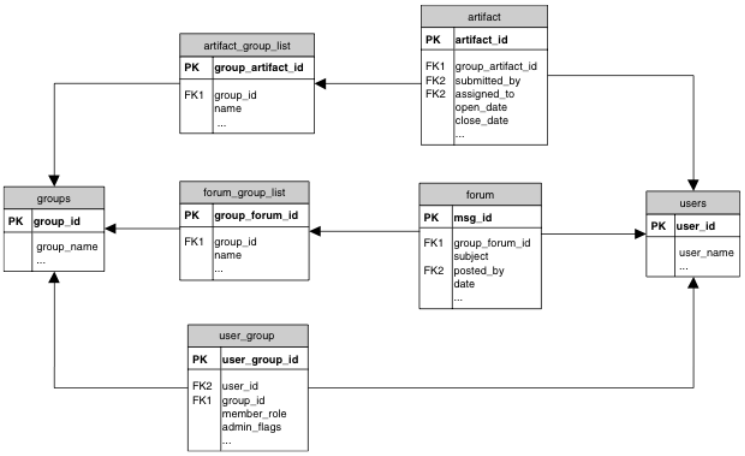


Abbildung 10: Tabellen mit Informationen der Passiven und Aktiven User, sowie der Co-Developer. Bild aus [XCM06]

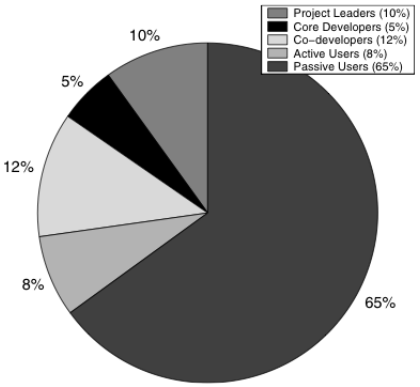


Abbildung 11: Rollenverteilung prozentual. Bild aus [XCM06]

Mit diesen Rollen im Hinterkopf machte sich Madey nun auf die Suche. Während die Project Leader und die Core Developer bereits bei SF gelistet waren, waren die Informationen und damit die prozentuale Verteilung der Co-Developer, der Aktiven und der Passiven User über 7 Tabellen (Abb. 10) verteilt.
Um die User mit ihren Aktivitäten auf 5 Rollen aufzuteilen wurde ein 3-schrittiger Daten Integrations und -Reduktionsprozess auf den Daten ausgeführt. (Ablauf in Abb. 12 visualisiert):

- Im ersten Schritt wurden jeweils die Artifakt(Bug Report, Feature Request, Support Request, Patch, Todo...) und die Forum Daten integriert, da sich einige Attribute zwischen *artifacts* und *forums* unterschieden
- Im nächsten Schritt wurden dann die beiden neu entstandenen Tabellen zu einer einzelnen vereinigt.
- Im dritten Schritt wurde diese Tabelle mit den Daten aus der *user_group* Tabelle, welche die Verhältnisse von Projekt Leadern und Core Developern zu den Projekten enthielt vereingt.

Damit erhielt man die Tabelle *User_project_role*, deren Daten folgend reduziert werden konnten auf die „Member set"s *Project leaders*, *Core developers*, *Co-developers*, *Active users* und *Passive Users*.

Mit diesen Daten konnte nun

- die prozentuale Verteilung (Abb. 11 und Abb. 13) der einzelnen User Rollen
- sowie die in Phase 1 errechneten Werte, wie z.B. der Cluster Koeffizient, der Diameter und die größen der Cluster neu

berechnet werden, zumal die jetzt erfassten Daten jetzt auch die Daten der Co-Developer und der Aktiven User mit einbezog. Während die Netze in Phase 1 also nur aus Project Leadern und Core Developern bestanden, waren nun auch die an den Projekten teilnehmenden Co-Developer und Aktiven User sichtbar.

3.3 Ergebnisse der zweiten Phase

Kleine Projekte enthalten (Details Abb. 13)

- 47,8 % Project Leader
- 20,6 % Core Developer

Große Projekte enthalten

- 3,6 % Project Leader + Core Dev.

- 55,8 % Co-Developer
- 40,6 % Active User

Das zeigt unter anderem, dass es ratsam ist, soziale Netze wie SourceForge.net, bzw. deren Verteilungen einzelner Benutzerrollen nicht nur im Gesamten zu betrachten, da ein Durchschnitt von vielen kleinen Projekten stark verzerrt werden kann, während die großen Projekte meist eine viel relevantere Rolle spielen. Viel mehr Sinn ergibt es also, nur Projekte gleicher oder ähnlicher Größe miteinander zu vergleichen.

Weiter sah man nun, dass der Diameter in Wirklichkeit noch viel kleiner war, als in den in Phase 1 vollzogenen Analysen, was sich mit den nun einbezogenen Co-Developern und den Aktiven Usern erklären lässt. Der Wert Betrug nun nur noch circa 3 [XGCM05] (vs. 6-8 in Phase 1). Der Clustering Koeffizient zeigte nun ebenfalls eine höhere Dichte mit einem Wert von über 0.8[XGCM05].

Diese Fakten unterstützen die Vermutung, dass Co-Developer und Aktive User eine entscheidende Rolle bei dem Verbinden, der SourceForge Community bilden. Mit ihrer Existenz können Informationen schneller durch die gesamte OSS Community fließen. So vermuteten Madey et al., ließe sich zum Beispiel erklären, warum bei 676 Text-Editor-Projekten 50 % aller Teilnehmer gerade bei den 6 erfolgreichsten mitmachen. Ein Grund könnte der kurze Kommunikationspfad sein, durch welchen die Leute schneller Projekte finden, die sie anziehen[XGCM05].

Die Eigenschaft der schnellen Kommunikation kann auch ein Faktor für den Erfolg der OSS Development Community sein. So gibt es bei „closed source“ Software Entwicklungen, keine Co-Developer oder Aktiven User, welche in engen Kontakt mit den Entwicklern stehen [XGCM05].
Die Co-Developer und Aktiven User scheinen damit die Funktion von „linchpin“ (Abb. 5), also Dreh-und Angelpunkt Knoten zu übernehmen. Ihre Verbindungen zu den einzelnen Projekten stellen „weak ties“ dar, also quasi die schwachen, einzelnen Kanten zwischen den sonst einzeln verbleibenden Clustern. So besteht das größte Cluster mit letzteren User Rollen aus circa 40.000 Usern, während das größte Cluster ohne jene Rollen aus nur circa 15.000 Usern bestehen würde [XCM06].
Durch die Vergrößerung dieser Cluster und der damit einhergehenden Verkürzung der Wege, kann also die Relevanz der Aktiven User und der Co-Developer geschlussfolgert werden.

4 Phase 3

4.1 Einführung

Der Schnitt zwischen Phase 2 und Phase 3 ist diesmal fließender. Der Datenbestand auf dem gearbeitet wird ist quasi der gleiche, die Untersuchungen mit Cluster-Algorithmen jedoch anders und die Suche nach Temporalen und Globalen Sozialen Stellungen neu. Es werden fast ausschließlich die Aktivitätsdaten betrachtet.

4.2 Acitivitäts Typen

Madey hat genau 29 Acitivitäts-Typen klassifizifiert (Auflistung und Details Abb.14), wobei die ersten 21 aus der SF.net Datebank kommen und die restlichen 8 Aktivitäten im CVS Repository beschreiben. Die ersten 21 Typen wurden hierfür aus diversen Datenbanktabellen (-konstrukten) zusammengesammelt.

- „Artifact Activities“.
  Sie beschreiben die ersten zwölf Aktivitätsyypen, zu denen beispielsweise *bug reports*, *support requests* oder *feature requests* gehören, welche bereits standardmäßig von SF.net vorgegeben sind. Zusätzlich können Nutzer auch ihre eigenen Typen definieren. Ganz typisch gibt es etwa noch die *todo*'s welche ebenfalls seperat gelistet wurden, da sie wichtig sind, und man sie leicht mit String Matching Algorithmen herausfiltern konnte. Die restlichen von den Usern anders benannten Artifaktdaten werden als *other artifacts* zusammengefasst [CM05b]. Um sich, das Data Mining besser vorzustellen hier ein Query, welcher auf den Tabellen aus Abb. 15 ausgeführt wurde und die Artifakttypen herausfilterte.

```
select
 a.artifact_id, b.group_id, b.name, a.assigned_to, a.submitted_by,
 a.open_date, a.close_date
from
 artifact a, artifact_group_list b, groups c
where
 c.group_id = b.group_id and b.group_artifact_id = a.group_artifact_id;
```

- Weiterhin gibt es die „Forum Acitivity“.
  Hier enthalten sind neue Beiträge und deren Folgebeiträge.
- „Project History Activity“
  Das sind projektrelevante Modifikationen.
- „File Release Activity“
  Das sind Releases neuer Version.
- „Project Task Acitivity“
- „Documentation Activity“

- und „Job Activity“
  Tritt ein, wenn bei einem Projekt ein Job Angebot, bzw. ein „help-wanted“ geposted wird.

Die restlichen 8 Aktivitäten beschreiben CVS Aktionen, wie den Checkout, den Export, das Hinzufügen oder das Updaten von Code.

4.3 Soziale Stellungen finden

Madey et al. wollten jetzt mit Hilfe der Aktivitätsdaten soziale Stellungen (social positions) finden. Ein soziale Stellung kann man sich nach Madey als eine bestimmtes Verflechtungsmuster im sozialen Netzwerk vorstellen. Zwei Aktoren die ähnlich ins Netz eingebunden sind, haben damit die gleiche soziale Stellung. Auf diesen sozialen Stellungen kann man dann auch Rollenanalysen machen. Als Beispiel gibt es die Rollen des Entwicklkers und des Testers. Der Entwickler produziert Codeänderungen, kommuniziert diese Änderungen mit dem Tester, der Tester probiert dann diese Änderungen aus und gibt dem Entwickler wieder Feedback.

Um solche sozialen Stellungen zu finden wandten Madey et al. einen Novel Algorithmus an. Dieser hat den Vorteil, dass er keine Vermutung über die Verteilung der Cluster (wie zum Beispiel eine Gaussche Verteilung) annimmt oder eine passende Ähnlichkeitsmetrik verlangt, wie bei anderen Algorithmen, wie z.B. K-means, K-mediod, Expectation-Maximization [CM05a]. Statt dessen benutzt der Novel Algorithmus ein ganz simples Prinzip, dass auf User/Projekt Paaren(siehe Abb. 16) arbeitet und grob wie folgt zusammengefasst werden kann [CM05a].

Abbildung 16: Ein User-Project-Paar, mit allen Aktivitäten, die der User auf dem Projekt auswührt, w entspricht der Menge der Aktivitäten des entsprechenden Aktivitätentyps

- User/Project Paar vergleichen
- Statistischer Test auf Verteilung der Variablen
  - Sind zwei Paare verschieden, werden sie ausgesondert
  - ansonsten läuft der Algorithmus weiter
- Dadurch werden alle Paare rausgeschmissen die unterschiedlich sind
- Die restlichem müssen mit bestimmten Konfidenz Faktor gleich sein und bilden also ein Cluster

- Das wird solange wiederholt bis keine separaten User/Projekt Paare mehr existieren

4.4 Ergebnis der globalen Untersuchung der sozialen Stellungen

Ausgeführt auf allen User/Projekt Paaren, die vom Beginn von SourceForge.net bis 2005 entstanden sind, errechnete der Novel Algorithmus 45 Cluster mit je mehr als 100 Leuten. Die manuelle Analyse deutete auf eine Aufteilung mit ungefähr 7 sozialen Stellungen hin. Davon wurden 6 bereits als sperate Cluster vom Algorithmus gefunden, die verbliebenen 39 bildeten die Position des Software Developers, da bei allen offensichtlich Source Code Operationen als primäre Aktion identifizierbar war und Madey diese nicht weiter zu unterscheiden beabsichtigten. Verteilt auf 39 Cluster waren sie vermutlich, weil sie alle mit unterschiedlichem Grad unterschiedliche Source Code Operationen ausführten[CM07].

Die gefundenen Sozialen Rollen (siehe auch Abb. 17) waren:
der „Software User“, welcher das größte Cluster bildete. Primär mit Forumaktivitäten und Source-Code-Checkouts. Der „Project Administrator“ mit Projektmodifikationen und Dateiveröffentlichungen, der „Software Developer“ mit der größten Sammlung an unterschiedlichen Aktivitäten (*source code add/remove/checkout/modify/update*, auch *file releases* und *project modifications*, der Verantwortliche für „Task Management“, der „Bug Reporter“, der „Feature Requester“ und die „Handyperson“, welche eine signifikante Anzahl an Aktivitäten für viele unterschiedliche Aktivitätstypen aufweist, inklusive Source-Code-Modifikationen, Bug-Reporting, Projektmodifikationen, Datei-Releases und Projekt-Tasks. Uncategorisiert blieben 14818.

4.5 Ergebnis der temporalen Untersuchung (Abb. 18)

Während obiges Ergebnis alleine stehend erstmal keine besonderen Reaktionen hervorruft, so wird es allerdings spannend, wenn man obiges Ergebnis mit einer temporalen Untersuchung vergleicht.
Wie im vorigen Abschnitt ließen Madey et al. den selben Algorithmus nochmal laufen, diesmal seperat für mehrere Monate bei relativer Zeitmessung.
Erklärung: der Moment der ersten Aktivität eines Users wird als Zeitpunkt 0 betrachtet und alle Aktivitäten innerhalb der nächsten 30 Tage in Monat 0 zusammengefasst. Monat 1 sind dann alle Aktivitäten der nächsten 30 Tage und so weiter.
In der temporalen Analyse wurden dann erstmal alle User/Projekt Paare des Monats 0 zusammengefasst analysiert (mit dem Novel Algorithmus), ganz gleich, ob ein User seinen Monat 0 im Jahr 1999 erlebte oder irgendwann Ende 2004 und das Prinzip wiederholt für die Monate 1, 2, .. , 20.

Das erstaunliche Ergebnis war nun, dass die Rolle des „Software Users“ plötzlich

verschwunden war. Ebenso die des „Bug Reporters“ und die des „Feature Requesters“. Hinzu kamen dafür der „Message Poster“ und der „Release Manager“. Die Vermutung hierfür ist, dass 40 % der Individuen bei den Projekten als „Sotware User“ tätig sind, ihre primären Aktionen jedoch in größeren Abständen ausführen [CM07].

Nächste Festellung ist, dass während die Verteilung der Rollen im ersten Monat (0) noch recht normal aussieht, der Projekt Administrator im zweiten Monat komplett und der Message Poster fast verschwunden ist und sich die Zahl aller User/Projekt Paare auf weniger als ein fünftel verkleinert hat.

In einem Paper mit Daten von Beginn von SF.net bis 2003, jedoch ohne CVS Aktivitäten (vs. bis 2005 inkl. CVS Daten) wurden andere Soziale Position inditifiziert und eine mit dem Namen „Brief Flame“ [CM05a]. Der Name deutet darauf hin, dass diese Rolle kurz aufflammt mit (möglicher Weise) einigen unterschiedlichen Aktionen, dann jedoch wieder ganz plötzlich verschwindet (bei damaligen Untersuchungen im 2ten Monat verschwunden). Die primären Aktivitäten dieser Rolle waren Forum-Aktivitäten und Projekt Modifikationen. Ähnliche Operationen finden wir nun beim „Message Poster“ und beim „Project Administrator“, welche beide zusammen auch bei diesen Analysen wie der „Brief Flame“ fast komplett verschwunden sind. Die Bezeichnungen „Project Administrator“ und „Message Poster“ sind somit also nicht unbedingt treffend hinsichtlich der Beobachtungen, dafür aber erklärender und Verständnis schaffend.
So kann man nun schnell erfassen, um welche Rollen es sich handelt und leicht zustimmen, dass der „Project Administrator“ höchstwahrscheinlich nur in der Anfangsphase gebraucht wird, um die SourceForge Projektseite zu gestalten, Freunde/Bekannte als Member hinzuzufügen oder andere initialisierungsrelevante Aktionen auszuführen. Anschließend sind solche Aktivitäten nicht mehr nötig (zumindest nicht im selben Maße).
Dass der „Message Poster“ verschwindet, ist weniger trivial, lässt sich aber damit begründen, dass die meisten Nachrichten für die Koordination im ersten Monat stattfinden und anschließend höchstwahrscheinlich erstmal eine Arbeitsphase folgt, in der gewisse Ideen erstmal umgesetzt und getestet werden müssen. Ein anderer Teil der Leute wird wahrscheinlich abgesprungen sein, nachdem der erste „Hype“ verflogen war.
In dem 3ten Monat (Monat 2) hat die „Handyperson“ einen enormen Zuwachs und bleibt im vierten Monat zusammen mit dem „Software Developer“ als einzige Stellung(en) übrig.
Die Aktivitätenverteilung der letzten vier erwähnten Rollen kann man sich in Abbilung 19 anschauen, wobei sich dortige Nummern auf die Nummern der Tabelle in Abb. 14 beziehen.
Madey et al. vermuten, dass der Wechsel z.B. vom anfänglichen „Message Poster“ zur „Handyperson“ oder zum „Software Developer“ das Schlüsselereignis ist, um das Aktivitätsniveau über die ersten Monate hinaus aufrecht zu erhalten.[CM07]. Auch scheint die Möglichkeit sich Rollen aussuchen zu können und und diese dann im richtigen Moment wieder wechseln zu können, Open Source die Effizienz und schnelle Anpassungs-

fähigkeit zu verleihen, welche den „closed source“ Projekte zu fehlen scheint. Man muss hierfür auch bedenken, dass diese meist nur mit den Rollen „Projekt Administrator“ und „Software Developer“ besetzt sind.

5 Rückblick

Ich habe einzelne Untersuchungen von Greg Madey und seinen Kollegen vorgestellt. Dabei bin ich eingegegangen auf die einzelnen Datenerhebungsmaßnahmen und den Fortschritt selbiger. Ich habe gezeigt, mit welch wenigen Daten bereits große Ergebnisse in Hinsicht auf die (Soziale) Netzwerkstruktur von SourceForge.net erarbeitet werden können und wie diese mit Hilfe von Simulationen auf unterschiedlichen Modellen verstanden und Theorien damit unterstützt werden können. Dabei habe ich den Diameter, den Clustering Koeffizienten, den durschnittlichen Grad und die Grad Verteilung des Netzwerkes vorgestellt, sowie die Grundideen der Modelle ER, BA, BA mit Fitness, BA mit dynamischer Fitness und wie die ersten Daten mit einem Web-Crawler gesammelt wurden. Weiter habe ich gezeigt, wie mit Zunahme der Datenmenge mehr und mehr individuenspezifische Untersuchungen betrieben werden konnten.
Ich habe gezeigt, wie Madey et al. vom uns bekannten „Onion Model“ mit 8 Rollen zum Developer-Typen Modell mit 5 Rollen gekommen ist. Ich habe die einzelnen Rollen vorgestellt, den Filterungsprozess dieser Rollen aus den einzelnen Datenbanktabellen erklärt und die prozentuale Verteilung dieser bei großen und bei kleinen Projekten miteinader verglichen.
Zum Schluss bin ich auf die einzelnen „Activity Typen“ aus Madeys Forschungen eingegangen und habe gezeigt, wie Madey et al. zu der Einteilung mit 29 Typen kamen und wie man mit Hilfe des Novel Algorithmus auf User-Projekt-Paaren die einzelnen sozialen Stellungen/Positionen, durch Herausfilterung unähnlicher Stellungen, voneinander separieren kann. Anschließend habe ich die Ergebnisse der globalen Untersuchung mit denen der temporalen Untersuchung verglichen.
Mein Paper bietet damit einen Überblick zu den einzelnen Forschungen rund um Madey und dem interessierten Leser einen guten Einstiegspunkt für das Lesen zukünftiger aber auch mehr ins Detail gehender Paper von Madey in Zusammenhang mit SourceForge.net und OSS Development im Allgemeinen.

Literatur

[CM05a] S Christley and G Madey. An algorithm for temporal analysis of social positions. In *NAACSOS2005*, Notre Dame, IN, Jun 2005.

[CM05b] S Christley and Greg Madey. Collection of activity data for sourceforge projects. Technical Report Technical Report TR-2005-15, Dept. of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, Oct 2005.

[CM07] S Christley and G Madey. Social positions at sourceforge. net. Limerick, Ireland, Jun 2007.

[GFM03a] Y Gao, V Freeh, and G Madey. Analysis and modeling of the open source software community. In *NAACSOS Conference 2003*, Pittsburgh, Jun 2003.

[GFM03b] Y Gao, V Freeh, and G Madey. Conceptual framework for agent-based modeling and simulation. In *NAACSOS Conference 2003*, Pittsburgh, Jun 2003.

[GM07] Y Gao and G Madey. Towards understanding: A study of the sourceforge. net community using modeling and simulation. pages 145–150, Norfolk, VA, USA, Mar 2007. The Society for Modeling and Simulation International (SCS).

[Mad03] Greg Madey. An investigation into the free/open source software phenomenon using data mining, social network theory, and agend-based. *UIUC - NFS Workshop on Continuous (Re)Design of Open Source Software*, page 43, Oct 2003.

[MCM05] D Mack, N Chawla, and G Madey. Activity mining in open source software. In *NAACSOS2005*, Notre Dame, IN, Jun 2005.

[MFT02a] G Madey, V Freeh, and R Tynan. Agent-based modeling of open source using swarm. In *8th Americas Conference on Information Systems (AMCIS2002)*, pages 1472–1475, Dallas, TX, 2002.

[MFT02b] G Madey, V Freeh, and R Tynan. The open source software development phenomenon: An analysis based on social network theory. In *8th Americas Conference on Information Systems (AMCIS2002)*, pages 1806–1813, Dallas, TX, 2002.

[MFT02c] G Madey, V Freeh, and R Tynan. Understanding oss as a self-organizing process. In *he 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002)*, Orlando, FL, 2002.

[MFT+03] Greg Madey, V Freeh, R Tynan, Y Gao, and C Hoffman. Agent-based modeling and simulation of collaborative social networks. In *AMCIS 2003*, Tampa, FL, Aug 2003.

[MFTH03] G Madey, V Freeh, R Tynan, and C Hoffman. An analysis of open source software development using social network theory and agent-based modeling, slides. In *The 2nd Lake Arrowhead Conference on Human Complex Systems*, Lake Arrowhead, CA, USA, 2003.

[NYNK02] K Nakakoji, Y Yamamoto, Y Nishinaka, and K Kishida. Evolution patterns of open-source software systems and communities. *... International Workshop on Principles of Software Evolution*, Jan 2002.

[wik09] 2009.

[XCM06] J Xu, S Christley, and G Madey. *Application of Social Network Analysis to the Study of Open Source Software*. Elsevier Press, Jan 2006.

[XGCM05] J Xu, Y Gao, S Christley, and G Madey. A topological analysis of the open source software development community. In *The 38th Hawaii International Conference on Systems Science (HICSS-38)*, Hawaii, Jan 2005.

[XHM03] Jin Xu, Yingping Huang, and Greg Madey. A research support system framework for web datamining research: Workshop on applications, products and services of web-based support systems. In *The Joint International Conference on Web Intelligence (2003 IEEE/WIC) and Intelligent Agent Technology*, pages 37–41, Halifax, Canada, Oct 2003.

[XM04] J Xu and G Madey. Exploration of the open source software community. In *NAACSOS 2004*, Pittsburgh, PA, Jun 2004.

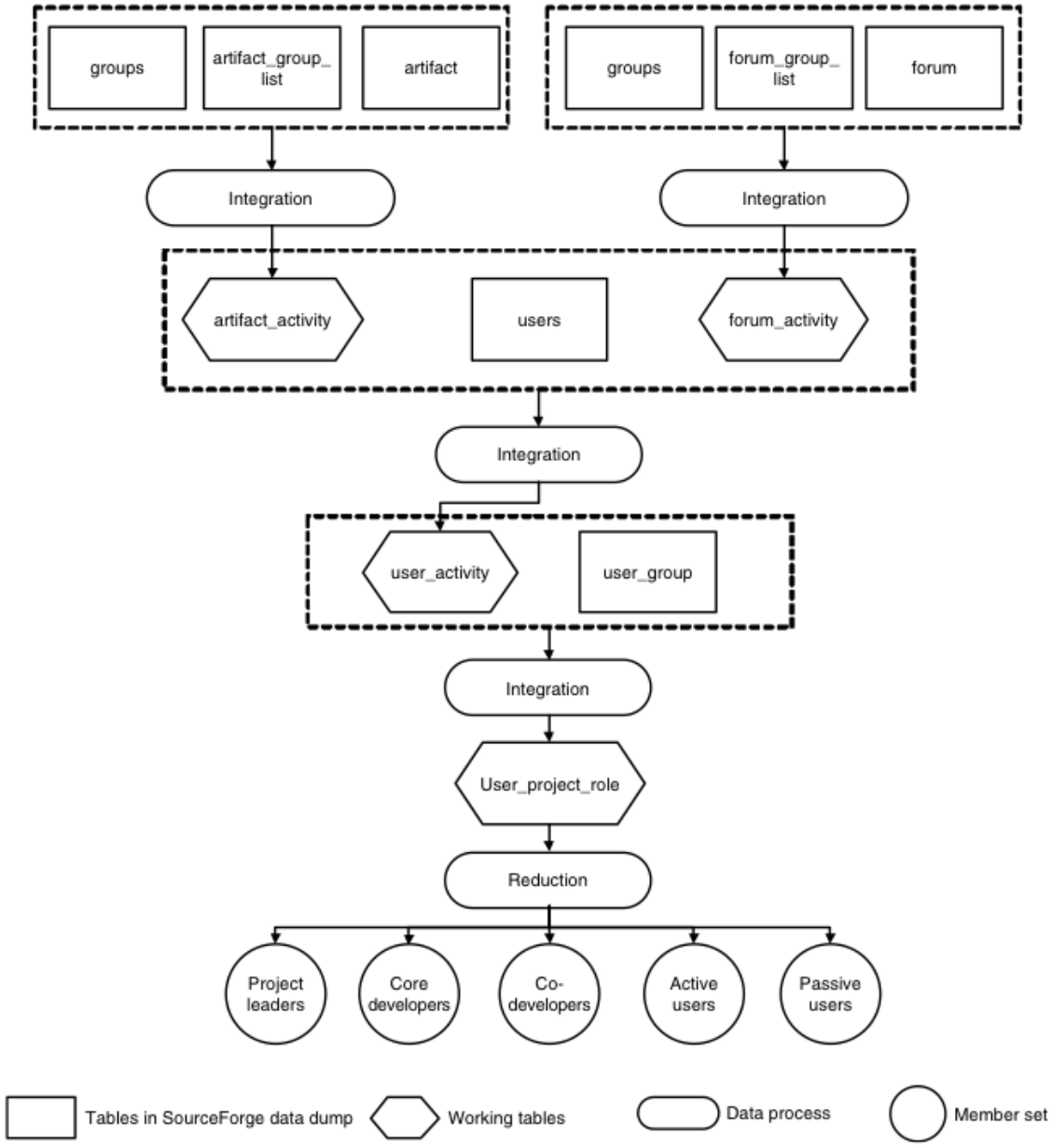


Abbildung 12: Daten Integrations und Reduktionsprozess. Bild aus [XCM06]

Developer Size	Project Number	Project Leaders		Core Developers		Co-developers		Active Users	
≤ 88	64847	80329	(47.8%)	34659	(20.6%)	33275	(19.8%)	19941	(11.8%)
> 88 and ≤ 279	193	590	(2.1%)	1703	(5.7%)	17334	(60.3%)	9124	(31.7%)
> 279	70	798	(0.9%)	2576	(2.7%)	53030	(55.8%)	38593	(40.6%)

Abbildung 13: Verteilung der Developer Rollen in großen, mittelgroßen und kleinen Projekten. Bild aus [XGCM05]

	Activity Type	Activity Description
Artifact Activity	submit bug(1)	Person submits a new bug report.
	assign bug(2)	Bug report is assigned to person.
	submit support request(3)	Person submits a new support request.
	assign support request(4)	Support request is assigned to person.
	submit patch(5)	Person submits a new patch.
	assign patch(6)	Patch is assigned to person.
	submit feature request(7)	Person submits a new feature request.
	assign feature request(8)	Feature request is assigned to person.
	submit todo(9)	Person submits a new to-do item.
	assign todo(10)	To-do item is assigned to person.
	submit other artifact(11)	Person submits an artifact that is not one of the predefined categories of bug report, support request, patch, feature request, or to-do item.
	assign other artifact(12)	Uncategorized artifact is assigned to person.
Forum Activity	new forum message(13)	Person posts a new forum message.
	followup forum message(14)	Person posts a forum message that is a followup to an existing forum message.
Project History Activity	modify project(15)	Person makes an administrative modification to the project; the modification is uncategorized, but they are typically tasks like adding/removing members, changing permissions, updating project settings, etc.
File Release Activity	file release(16)	Person posts a new file release; this is typically associated with releasing a new version of the software to the public.
Project Task Activity	new project task(17)	Person creates a new project task.
	assigned project task(18)	A project task is assigned to person.
	modify project task(19)	Person modifies an existing project task.
Document Actity	create document(20)	Person creates a new document.
Job Activity	create people job(21)	Person posts a new job; these are similar to help-wanted ads where a project is looking for somebody with particular skills.
CVS Activity	checkout source code(22)	Person checks out source code from CVS repository.
	export source code(23)	Person exports source code from CVS repository.
	release source code(24)	Person releases check out of source code from CVS repository.
	tag source code(25)	Person tags source code in the CVS repository with a label.
	add source code file(26)	Person adds a new source code file to the CVS repository.
	remove source code file(27)	Person removes a source code file from the CVS repository.
	modify source code file(28)	Person commits a source code modification to the CVS repository.
	update source code(29)	Person updates local checked out source code with any changes in CVS repository.

Abbildung 14: Insgesamt 29 Aktivitäts-Typen. Tabelle aus [CM07], Beschriftung von S. Wojtowicz

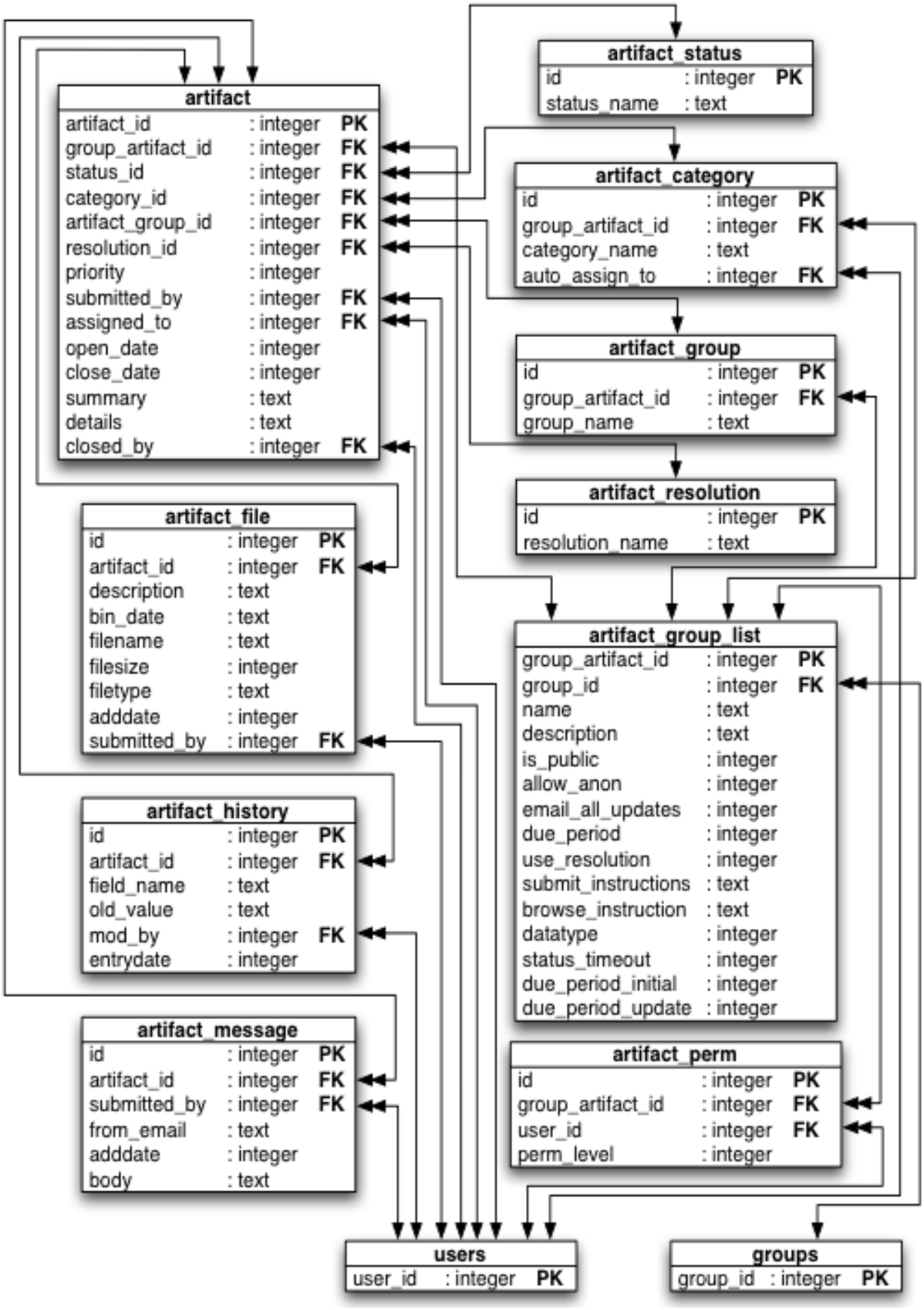


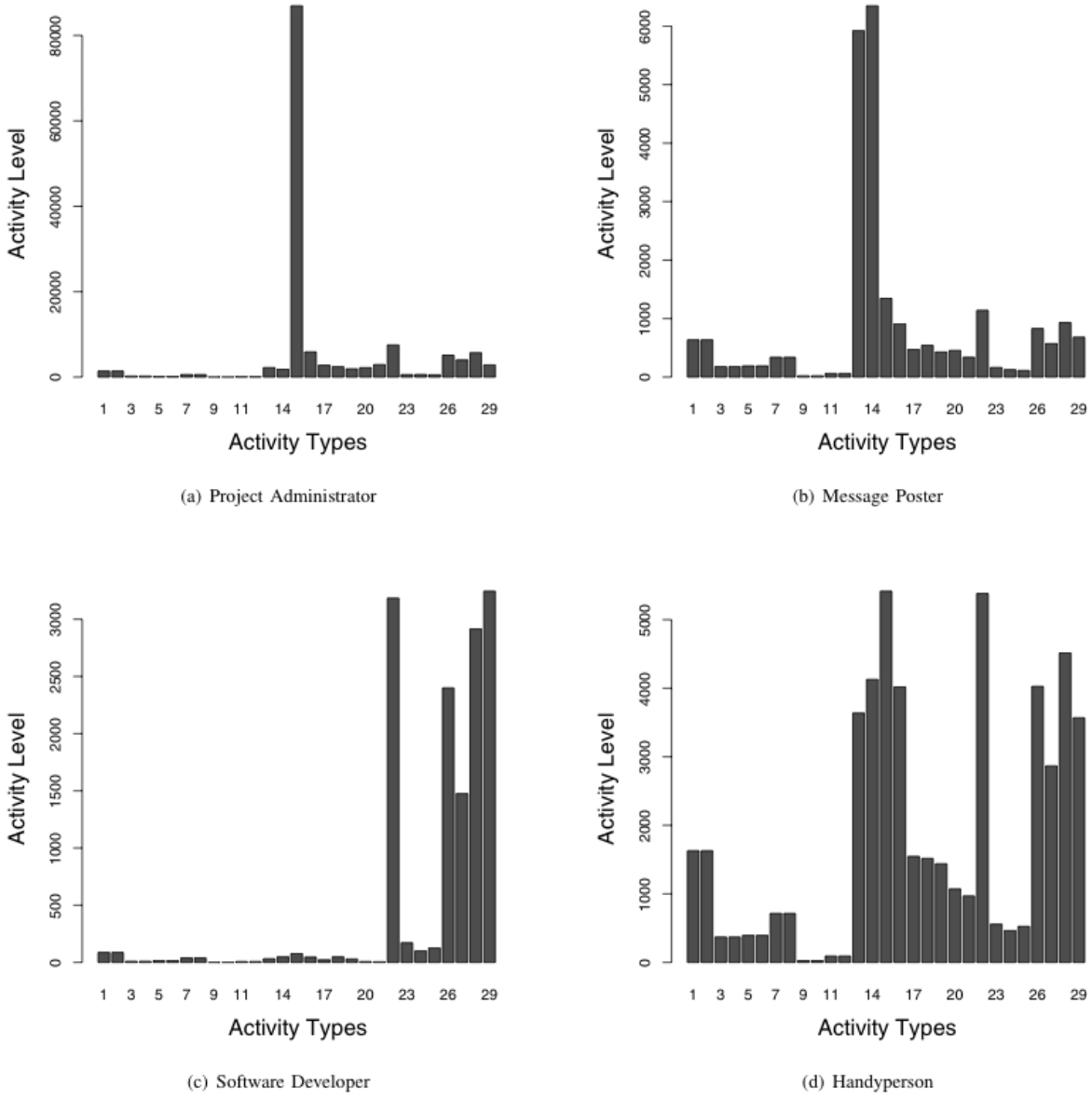
Abbildung 15: Schema für die Artifact Daten in der SourceForge.net Datenbank. Bild aus [CM05b]

Social Position	Description	Size
Software User	The largest cluster with the primary activities of posting new forum messages, followup forum messages, and checkout out source code.	111889
Project Administrator	The second largest cluster with the primary activity of making project modifications; the project administrator also performs file releases, but most other activities are relatively minor or non-existent.	93199
Software Developer	Primary activities are source code operations like checking out source code, add/remove source code files, modify source code, and update source code. The social position contains 39 clusters all with different relative proportions of the source code operations, and some software developers have significant levels of project modification and file release activities.	47495
Task Management	Significant usage of the project task management provided by SourceForge.net.	2181
Bug Reporter	Significant bug reporting activity with a slight amount of features requests, support requests, and patches.	1138
Feature Requester	Primary activity was submission of feature requests but also has a significant amount of bug reporting.	370
Handyperson	The handyperson has significant activity for many different activity types including source code modifications, bug reporting, project modifications, file releases, and project tasks.	217
Not Categorized	The remaining very small clusters that were not analyzed.	14818
	Total user/project pairs	271307

Abbildung 17: Globale soziale Stellungen. Bild aus [CM07]

Social Position	Description	Month 0	Month 1	Month 2	Month 3
Project Administrator	Primary activity of making project modifications; other activities are relatively minor or non-existent.	86951	0	0	0
Message Poster	Primary activity of posting new or followup forum messages; other activities are relatively minor or non-existent.	96052	7315	0	0
Software Developer	Primary activities are source code operations like checking out source code, add/remove source code files, modify source code, and update source code. Multiple clusters with different relative proportions for the source code operations; some software developers have significant levels of project modification and file release activities.	67488	32126	21054	18239
Release Management	Primary activity of file release but also significant project modification activity.	11700	7227	0	0
Task Management	Significant usage of the project task management provided by SourceForge.net.	1775	0	0	0
Handyperson	The handyperson has significant activity for many different activity types including source code modifications, bug reporting, project modifications, file releases, and project tasks.	1768	120	14050	10709
Not Categorized	The remaining very small clusters that were not analyzed.	3066	1638	1712	1611
	Total user/project pairs	268800	48426	36816	30559

Abbildung 18: Temporale Untersuchung der sozialen Stellungen. Bild aus [CM07]



(a) Project Administrator

(b) Message Poster

(c) Software Developer

(d) Handyperson

Abbildung 19: Verteilung der Aktivitäten bei 4 ausgewählten sozialen Positionen. Bild aus [CM07]