

## 2. Open-Source-Software (OSS)

### 2.1 Grundlagen

#### 2.1.1 Historische Entwicklung

Die Weitergabe von Software in Quelltextform geht auf die Hackerkultur der 1960er-Jahre<sup>65</sup> zurück. Die Entwicklung von Software wurde insbesondere an Universitäten und im militärischen Bereich durchgeführt. Erst später wurde es aufgrund des Wandels hin zu einem kommerziellen Softwaremarkt<sup>66</sup> die Regel, den Quelltext als Geschäftsgeheimnis anzusehen und nur Binärcode herauszugeben, der vom Nutzer nicht modifiziert werden kann.<sup>67</sup>

Als Gegenpol dieser Kommerzialisierung von Software ist der Begriff Freie Software entstanden, der auf Richard Stallman zurückgeht. Dieser gründete in 1985 die FSF<sup>68</sup> und betonte bei seiner Definition von Software mit frei verfügbarem Quelltext die Freiheit der Software, was durch die Free Software Foundation (FSF) mittels der Free Software Definition<sup>69</sup> kommuniziert wird. Die Freiheit der Software umfasst gemäß dieser nicht nur die Verfügbarkeit des Quelltextes, sondern bspw. auch das Recht diesen zu verbessern und diese Verbesserungen zu veröffentlichen. Die FSF ist zudem der Ursprung der populären OSS-Lizenz GNU GPL.<sup>70</sup> Eine alternative Definition von OSS liefert die 1998 gegründete Open Source Initiative (OSI)<sup>71</sup> mit der Open Source Definition<sup>72</sup>. Diese nutzt den Begriff Open-Source-Software und hebt so die Verfügbarkeit des Quelltextes hervor; die Inhalte der Free Software Definition und der Open Source Definition sind jedoch sehr ähnlich.

Die erste OSS, die im Sinne Freier Software nach Definition der FSF stand, wurde von Richard Stallmann mit der Softwareentwicklungsumgebung GNU zur Verfü-

---

<sup>65</sup> Vgl. Boehm (2006), S. 19, Raymond (2001), S. 4-7.

<sup>66</sup> Dreiling et al. (2005), S. 3 sowie Mundhenke (2007), S. 50f. führen als Auslöser die Beendigung der Bündelung von Software und Hardware durch IBM an, die durch ein Kartellrechtsverfahren des US-Justizministeriums erzwungen wurde.

<sup>67</sup> Der Quelltext wird in diesem Fall vor der Herausgabe in Objektcode umgewandelt. Im Fall von Software, deren Code interpretiert wird, werden Hilfsprogramme genutzt, die den Code verschlüsseln. Siehe für die Programmiersprache PHP bspw. [<http://www.sourceforgeguardian.com>].

<sup>68</sup> Vgl. [<http://www.fsf.org>], Free Software Foundation (2010).

<sup>69</sup> Vgl. Ibid.

<sup>70</sup> Vgl. Brügge et al. (2004), S.10. GNU ist ein rekursives Akronym und steht für GNU's Not UNIX, vgl. [<http://www.gnu.org/home.en.html>]. GPL steht für General Public License, vgl. [<http://www.gnu.org/licenses/licenses.en.html>].

<sup>71</sup> Vgl. [<http://www.opensource.org/about>].

<sup>72</sup> Vgl. Open Source Initiative (2010). Die Nutzung des Begriffs *Open Source* sollte dabei auch einem besseren Marketing der Ideen freier Software dienen, vgl. Goldman und Gabriel (2005), S. 31.

gung gestellt.<sup>73</sup> Das GNU-Projekt zielte darauf ab, ein alternatives und freies Betriebssystem zu erstellen. Wenngleich eine Vielzahl auch heute noch gebräuchlicher Softwarewerkzeuge durch das Projekt erstellt worden ist, so ist bis heute das geplante Betriebssystem GNU/Hurd nicht für einen produktiven Einsatz geeignet.<sup>74</sup>

Bekanntheit hat OSS insbesondere durch GNU/Linux erlangt, ein Betriebssystem-projekt, das von Linus Torvalds ins Leben gerufen worden ist und dem viele, über die ganze Welt verteilte, Entwickler beigetreten sind.<sup>75</sup> Der von Torvalds entwickelte Linux-Kernel wurde dabei mit den durch das von Richard Stallmann initiierte GNU-Projekt zur Verfügung gestellten Programmen zu einem vollständigen Betriebssystem, GNU/Linux genannt, zusammengeführt.<sup>76</sup> GNU/Linux war zu Anfang insbesondere unter technikaffinen Personen bekannt, ist aber inzwischen auch in Distributionsform erhältlich. Distributionen führen das Betriebssystem mit weiterer OSS und zum Teil auch proprietärer Software zusammen und beinhalten üblicherweise eine Paketverwaltung, um die Installation zu erleichtern; sie erfordern daher keine umfassenden Computerkenntnisse und fördern folglich die Verbreitung von OSS auch in weniger technikaffinen Nutzergruppen.

### 2.1.2 Nutzung

Die Nutzung von GNU/Linux hat mit der Durchdringung des Internets zugenommen, was u. a. auf die Verbreitung der Software mittels des Internets, die schon früh integrierten umfangreichen Netzwerkfunktionen sowie für GNU/Linux verfügbare Software für Internetdienste zurückzuführen ist. Neben dem Nameserver Bind<sup>77</sup> und dem Mailserver Sendmail<sup>78</sup> ist hier insbesondere der Apache Webserver<sup>79</sup> zu nennen. Dieser wird oft in Zusammenspiel mit GNU/Linux, der Programmiersprache PHP und dem unter dualer Lizenz<sup>80</sup> stehenden Datenbankmanagementsystem (DBMS) MySQL eingesetzt; eine Umgebung, die auch unter der Abkürzung LAMP (Linux, Apache, MySQL, PHP) bekannt ist.

Während GNU/Linux wie auch Apache allgemeine Kernfunktionen anbieten, erlangt OSS zunehmend auch im Bereich spezieller Anwendungen Bekanntheit, beispielsweise im Bereich der Business Intelligence mit der Pentaho BI Suite<sup>81</sup> und

---

<sup>73</sup> Vgl. Alpar et al. (2011), S. 377.

<sup>74</sup> Vgl. [<http://www.gnu.org/s/hurd/hurd.html>].

<sup>75</sup> Vgl. Raymond (2001). Eine Abhandlung aus Sicht Torvalds' liegt mit Torvalds und Diamond (2001) vor.

<sup>76</sup> Vgl. Brügge et al. (2004), S. 33 f.

<sup>77</sup> Siehe [<http://www.isc.org/software/bind>].

<sup>78</sup> Siehe [<http://www.sendmail.org>].

<sup>79</sup> Siehe [<http://httpd.apache.org>].

<sup>80</sup> Siehe Abschnitt 2.3.1.

<sup>81</sup> Siehe [<http://www.pentaho.com>].

dem Eclipse BIRT (Business Intelligence and Reporting Tool)-Projekt<sup>82</sup>. Selbst Enterprise-Resource-Planning-Systeme sind inzwischen als OSS verfügbar (bspw. SQL Ledger<sup>83</sup> und ADempiere<sup>84</sup>). Die Freiheit der OSS kann bei diesen Anwendungen einen Vorteil gegenüber der nur eingeschränkt offenen Architektur von proprietären Enterprise-Systemen darstellen.<sup>85</sup> Es ist anzumerken, dass OSS oftmals nicht ausschließlich für GNU/Linux verfügbar ist, sondern zunehmend auch für proprietäre Betriebssysteme wie Microsoft Windows.

Neben OSS, die eine Neuentwicklung darstellt, gibt es Beispiele, bei denen vormals proprietäre Software durch ein Unternehmen als OSS freigegeben worden ist. Bekannte Beispiele hierfür sind Mozilla (basierend auf dem von Netscape zur Verfügung gestellten Quelltext), Eclipse (Bereitstellung durch IBM)<sup>86</sup> sowie Open Office (basierend auf Star Office von Sun)<sup>87</sup>.

Wenngleich OSS nicht unbedingt kostenfrei verfügbar sein muss, ist dies oft der Fall.<sup>88</sup> Aus diesem Grund ist OSS sehr relevant für Unternehmen, die ihre Lizenzkosten senken wollen.<sup>89</sup> Tatsächlich scheint Kostenreduzierung einer der Hauptgründe für die Einführung von OSS zu sein, wenngleich eine Vielzahl weiterer Stärken von OSS für deren Einsatz sprechen kann.<sup>90</sup> Beispiele sind ein geringerer Lock-In-Effekt und eine reduzierte Abhängigkeit von einem Entwickler, was auch die Zukunftssicherheit der Software erhöht.<sup>91</sup> Die Sicherheit von OSS profitiert

---

<sup>82</sup> Siehe [<http://www.eclipse.org/birt/phoenix>].

<sup>83</sup> Siehe [<http://www.sql-ledger.org>].

<sup>84</sup> Siehe [<http://adempiere.com>]. ADempiere ist als Abspaltung (Fork) aus dem OSS-Enterprise-Resource-Planning-Projekt Compiere hervorgegangen. Zum Begriff Fork siehe Abschnitt 2.2.5.

<sup>85</sup> Vgl. Dreiling et al. (2005).

<sup>86</sup> Die Unterstützung des OpenSolaris-Projektes ([<http://hub.opensolaris.org/bin/view/Main/>]) nach der Übernahme von Sun Microsystems durch Oracle soll jedoch in Zukunft reduziert werden, vgl. Benz (2010).

<sup>87</sup> Ein Framework zur Beurteilung der Chancen einer proprietären Software im Fall der Freigabe als OSS insbesondere in Hinblick auf die Gewinnung von neuen Entwicklern schlagen Kilamo et al. (2010) vor.

<sup>88</sup> Bei OSS, die den Vorgaben der Open Source Definition bzw. der Free Software Definition entspricht, kann bspw. ein Entgelt für die Bereitstellung der Software berechnet werden. Ist dieses gezahlt, kann die Distribution beliebig oft kostenfrei kopiert werden.

<sup>89</sup> Eine Vielzahl von Veröffentlichungen betont dabei die Wichtigkeit der Betrachtung der Total Cost of Ownership (TCO), da eine ausschließliche Betrachtung der Lizenzkosten beispielsweise höhere Kosten für Nutzerschulungen nicht berücksichtigt. Eine Vielzahl von TCO-Betrachtungen vergleichen die Kosten proprietärer Software und OSS; eine eindeutige Tendenz ist dabei nicht festzustellen. Ein Vergleich verschiedener TCO-Berechnungen stellt Brügge et al. (2004) auf den S. 115-124 dar.

<sup>90</sup> Vgl. bspw. Dedrick und West (2004). Gemäß einer aktuelleren Umfrage der Unternehmensberatung Accenture sehen viele Unternehmen die hohe Qualität von OSS inzwischen als ein wichtigeres Kriterium an, vgl. Accenture (2010).

<sup>91</sup> Vgl. Alpar et al. (2011), S. 378f.

zudem durch die Offenheit des Quelltextes, da es allen Beteiligten offen steht, Sicherheitslücken zu finden und zu beheben. Die Vorteile von OSS haben nicht nur zu einem vermehrten Einsatz von OSS in Unternehmen geführt; auch öffentliche Verwaltungen nutzen auf einer großen Zahl von Rechnern OSS. Der Umfang ist dabei unterschiedlich und muss nicht zwangsläufig auch einen Umstieg auf das Betriebssystem GNU/Linux beinhalten. Exemplarisch sei hier die Migration von ca. 15000 Arbeitsplatzrechnern der Stadt München zu einer OSS-basierten Lösung im Rahmen des LIMUX-Projekts<sup>92</sup> erwähnt.<sup>93</sup> Neben der Nutzung von OSS im öffentlichen Bereich erfährt diese auch Unterstützung von politischer Ebene. So verfolgt die Bundesregierung bereits seit einiger Zeit das Ziel, die Abhängigkeit von einzelnen Softwareanbietern zu reduzieren und auf offene Standards zu setzen.<sup>94</sup>

### 2.1.3 Geschäftsmodelle

Aufgrund der meist kostenfreien Verfügbarkeit von OSS kann die Erhebung von Lizenzkosten, wie bei proprietärer Software die Regel, kein sinnvolles Geschäftsmodell für OSS sein. Aufgrund dessen ist eine ganze Reihe von weiteren Geschäftsmodellen entstanden, die im OSS-Bereich angewandt werden. Neben einer aufgrund der Verfügbarkeit als Download nicht mehr als besonders relevant anzusehenden physischen Distribution von Software umfasst dies u. a. die folgenden Bereiche und Tätigkeiten:

- Softwareintegration:

Hierbei sind insbesondere die Hersteller von Distributionen gemeint. Neben Distributionen, die ein breites Publikum ansprechen, sind Distributionen wie Red Hat Enterprise Linux zu nennen, die gezielt Unternehmen ansprechen und zusammen mit Supportverträgen angeboten werden.

- Hardwareintegration:

Eine Hardwareintegration meint das Angebot einer Kombination von Hardware mit einer darauf abgestimmten Software. Ein Beispiel hierfür ist das Hardware-Angebot von IBM, das u. a. GNU/Linux als Betriebssystem enthält.

- Support:

Neben den bereits zuvor erwähnten Distributoren bieten u. a. Beratungsunternehmen Support an. In diesen Bereich fällt zudem der Support von OSS-

---

<sup>92</sup> Siehe [<http://www.muenchen.de/lmux>].

<sup>93</sup> Eine Übersicht über weitere Umstellungsprojekte sowie über Studien von Migrationen zu OSS bietet u. a. Ven et al. (2008), S. 194-199.

<sup>94</sup> Vgl. bspw. Marquardt (2004), S. 58ff. Die aktuelle Regierungskoalition von CDU/CSU und FDP erwähnt OSS explizit im Koalitionsvertrag, spricht dabei jedoch lediglich von einer Prüfung der Unterstützung von Open-Source-Lösungen. Vgl. CDU (2009), S. 102.

Projekten bspw. durch Mediatoren wie SourceForge.net<sup>95</sup>. Der Begriff Mediator bezeichnet hier Websites, die als Mittler zwischen den Entwicklern und Nutzern von OSS agieren. Sie stellen den Entwicklern hierzu Hilfsmittel wie bspw. Versionsverwaltungssysteme und Wikis sowie Speicherplatz meist kostenfrei zur Verfügung.

- Publikationen:

Dies umfasst Veröffentlichungen zu Themen und konkreten Produkten des OSS-Bereichs.

- Auftragsentwicklung:

Auftragsentwicklungen werden für viele populäre OSS-Produkte angeboten.<sup>96</sup> Sie ermöglichen bspw., dringende Erweiterungen einer OSS zu realisieren. Eine Weitergabe des Quelltextes an das OSS-Projekt kann u. U. dazu führen, dass die Erweiterung Teil des offiziellen Release der OSS wird.

- Erzeugen kommerziellen Mehrwertes:

Beispiele hierfür sind das Bereitstellen einer Entwicklungsumgebung für eine freie OSS im Embedded-Bereich sowie die Bereitstellung einer Serverkonsole für einen OSS-Server.<sup>97</sup>

- Duallizenzierung:

Eine solche Lizenzierung wird beispielsweise bei dem DBMS MySQL<sup>98</sup> umgesetzt, um die Möglichkeit zu eröffnen, das DBMS unter einer weniger restriktiven Lizenz zu nutzen.<sup>99</sup>

- Kommerzielle Verbesserung von OSS:

Dies ist mit einer Auftragsentwicklung zu vergleichen. Ziel ist hier aber explizit die Verbesserung der ursprünglichen OSS.

---

<sup>95</sup> Vgl. hierzu Saleck (2005), S. 79.

<sup>96</sup> So u. a. auch für das in Abschnitt 4.1 dargestellte Pligg Content-Management-System, bspw. unter [[https://www.odesk.com/groups/pligg/vt\\_cmp=pligg](https://www.odesk.com/groups/pligg/vt_cmp=pligg)].

<sup>97</sup> Vgl. Golden (2005), S. 34.

<sup>98</sup> Vgl. [<http://www.mysql.com/about/legal/licensing/oem/#4>], Brügge et al. (2004), S. 49f. und 112f. Bezüglich MySQL ist anzumerken, dass das Copyright für Entwicklungsbeiträge Dritter, um in das offizielle Release aufgenommen werden zu können, auf die Firma MySQL AB übertragen werden muss.

<sup>99</sup> Siehe hierzu auch Abschnitt 2.3.1.

- Spezialisierte Produkte:

Dies bezieht sich beispielsweise auf die Umsetzung einer Firewall-Appliance<sup>100</sup>, bei der GNU/Linux für die Nutzung als Bestandteil einer Hardwarelösung angepasst wird.<sup>101</sup>

Neben der Umsetzung dieser Geschäftsmodelle im Kontext bestehender OSS-Projekte kann für Unternehmen zudem ein Anreiz bestehen, selbst ein OSS-Projekt zu starten.<sup>102</sup>

## 2.2 Entwicklung von OSS

### 2.2.1 OSS-Projekte und -Communitys

Der Start eines OSS-Projekts erfolgt „klassisch“ aufgrund des Bedarfs eines einzelnen Entwicklers („scratching a developer’s personal itch“<sup>103</sup>), der bei der OSS-Community um Mithilfe bei der Entwicklung wirbt.<sup>104</sup> Wenngleich dies nicht in allen Fällen gegeben ist, steht doch oft keine Gewinnerzielungsabsicht hinter der Gründung. Die Entwicklung von OSS wird in OSS-Projekten durchgeführt, die aus einer oftmals weltweit verteilten Entwicklungsgemeinschaft bestehen. Diese Entwicklungsgemeinschaften werden auch als OS(S)-Communitys bezeichnet.<sup>105</sup> Sie setzen sich üblicherweise aus freiwilligen Mitarbeitern zusammen, die in ihrer Freizeit an den OSS-Projekten mitarbeiten.

Es sollte bereits bei der Gründung des OSS-Projekts hinterfragt werden, ob sich eine ausreichende Anzahl von Benutzern und Entwicklern finden wird.<sup>106</sup> Während die offiziellen Releases von den Hauptentwicklern zur Verfügung gestellt werden, können Anpassungen zwischen Releases von den Hauptentwicklern oder jeder anderen Person zur Verfügung gestellt werden. Die Bereitstellung der Anpassungen ist dabei abhängig von Nutzung und Lizenz nicht immer verpflichtend; folglich können auch Anpassungen in Organisationen durchgeführt werden, die nicht für die Öffentlichkeit verfügbar gemacht werden.

---

<sup>100</sup> Vgl. Karels (2003), S. 50ff.

<sup>101</sup> Vgl. Golden (2005), S. 35.

<sup>102</sup> Vgl. Koch (2003), S. 57, sowie die Darstellung zur Quelle Goldman und Gabriel (2005) in Abschnitt 5.1.

<sup>103</sup> Vgl. Raymond (2001), S. 23.

<sup>104</sup> Vgl. Fitzgerald und Ågerfalk (2005). Dies wird als Hinweis auf eine Einstufung von OSS-Projekten in den Bereich des Kollektivismus verstanden. Dennoch finden sich insbesondere im Bereich der individuellen Motivation der Entwickler Hinweise auf einen individualistischen Charakter von OSS-Projekten.

<sup>105</sup> Vgl. Brügge et al. (2004), S. 79.

<sup>106</sup> Vgl. Koch (2003), S. 57.

Das OSS-Entwicklungsmodell weist hinsichtlich des Stellenwertes des Codes, der kurzen Entwicklungszyklen, der intrinsischen Motivation der Beteiligten und der Bereitschaft, Software als ein sich ständig änderndes Produkt anzusehen, Parallelen zur Entwicklung mit agilen Methoden auf.<sup>107</sup> Aufgrund der Erfolge von Softwareentwicklungen mittels des OSS-Entwicklungsmodells gibt es Versuche, Aspekte des OSS-Entwicklungsmodells auch in die Entwicklung proprietärer Software einzubringen.<sup>108</sup>

### 2.2.2 Teilnehmer an OSS-Projekten

Für OSS-Projekte ist es kritisch, eine ausreichende Anzahl an Benutzern und Entwicklern zu gewinnen.<sup>109</sup> In der Anfangszeit von OSS waren daher insbesondere Projekte erfolgreich, bei denen die Nutzer der OSS zugleich über Programmierkenntnisse verfügten, da folglich ein hoher Anteil möglicher Personen vorhanden war, die zur Weiterentwicklung beitragen konnten.<sup>110</sup>

Die Motivation für Personen, die an einem OSS-Projekt teilnehmen, ist dabei oftmals rein intrinsisch. Sie kann aus der Reputation durch die Teilnahme und dem Spaß am Programmieren entstehen; insbesondere für private Entwickler entsteht oftmals ein Nutzen aus der Beschäftigung mit der Software selbst, dem Lernen und Weiterentwickeln sowie einer durch die Beteiligung entstehenden Sozialwertsteigerung. Besteht ein eigener Bedarf an der Software, so kann durch die eigenen Beiträge und die offene Lizenzierung eine Verbesserung der Marktchancen entstehen.<sup>111</sup>

Neben nicht-professionellen Programmierern sind an OSS-Projekten oftmals Personen beteiligt, die auch beruflich Software entwickeln. Hierbei ist zu unterscheiden in professionelle Programmierer, die in ihrer Freizeit an einem OSS-Projekt mitarbeiten, der Einbeziehung von professionellen Programmierern als bezahlte Mitarbeiter in OSS-Projekten<sup>112</sup> und der Freistellung von Mitarbeitern durch Unternehmen, die an der Weiterentwicklung einer bestimmten OSS interessiert sind<sup>113</sup>. OSS-Projekte können bezahlte und freigestellte Programmierer einsetzen und ggf. gezielt für Aufgaben einsetzen, die für Freiwillige weniger attraktiv sind<sup>114</sup>, wie bspw. das Durchführen von Tests, die Erstellung der Dokumentation sowie die

---

<sup>107</sup> Vgl. Strahinger Ibid., S. 14.

<sup>108</sup> Vgl. bspw. Schneidewind et al. (2002), Lussier (2004) sowie Martin und Hoffman (2007). Dieses Vorgehen wird z. T. als *inner source* bezeichnet, vgl. Fitzgerald (2011), S. 28.

<sup>109</sup> Vgl. Koch (2003), S. 57.

<sup>110</sup> Vgl. Brügge et al. (2004), S. 58.

<sup>111</sup> Vgl. Ibid., S. 58-60 und S. 100, Teupen (2007), S. 60-64.

<sup>112</sup> Vgl. Berdou (2006).

<sup>113</sup> Verschiedene Firmen unterstützen OSS-Projekte nicht (nur) mit finanziellen Mitteln, sondern mittels der Unterstützung durch Programmierer, wie beispielsweise IBM dies im Fall von Linux praktiziert.

<sup>114</sup> Vgl. Michlmayr et al. (2008), S. 2f.

Fehlerbehebung.<sup>115</sup> Für Unternehmen kann die Einstellung bzw. Freistellung von zentralen Beteiligten eines OSS-Projektes zur Vollzeit-Mitarbeit im OSS-Projekt auch deshalb sinnvoll sein, da diese Personen sehr umfangreich im Projekt mitarbeiten können und zugleich, im Idealfall im Sinne des Unternehmens, die langfristige Entwicklung des OSS-Projekts überwachen können.<sup>116</sup> Professionelle Programmierer wiederum können durch die freiwillige Teilnahme an OSS-Projekten versuchen, ihre Aussichten auf interessante Jobangebote zu verbessern.<sup>117</sup> Die weitgehende Autonomie der Teilnehmer von OSS-Communitys, das in diesen Communitys vorzufindende „kreative Chaos“, die reichhaltige Verfügbarkeit von Informationen bezüglich des Projektes und dessen Aufgaben sowie die vielfältigen Qualifikationen der Teilnehmer werden als wichtige Stimuli für eine offene Innovation (*open innovation*) angesehen.<sup>118</sup>

### 2.2.3 Organisation von OSS-Projekten

Die Organisation von OSS-Entwicklungsprojekten weicht deutlich von der Organisation der Entwicklung proprietärer Software ab. Raymond (2001) hat in diesem Zusammenhang den Vergleich der Kathedrale und des Basars geprägt, der die feste Struktur der proprietären Softwareentwicklung mit dem Bau einer Kathedrale und die für neue Entwickler offene Struktur der OSS als den Ablauf auf einem Basar darstellt.<sup>119</sup>

Die Organisation von OSS-Projekten kann sehr unterschiedlich gestaltet sein. Wenngleich manche Projekte nach wie vor von wichtigen, zentralen Personen kontrolliert werden (die zum Teil als einzige das Recht haben, neuen Quelltext zu dem Projektcode hinzuzufügen<sup>120</sup>), so ist eine dezentrale Kontrolle üblich. Die Projekte können Strukturen wie bspw. Konsortien herausbilden, die für die Koordination zuständig sind; unabhängig davon ist die Steuerung üblicherweise weniger zentral als die der traditionellen Softwareentwicklung.<sup>121</sup>

Beteiligte an OSS-Projekten können dabei verschiedene Rollen einnehmen. So unterteilen bspw. Holck und Jørgensen (2004) anhand der Beispiele Mozilla und FreeBSD in *Top-Level Management*, *Release Management*, *Module Owners*, *Reviewers*, *Committers* und *Contributors*.<sup>122</sup> Alternativ kann eine Einteilung der Be-

---

<sup>115</sup> Vgl. German (2004), S. 205.

<sup>116</sup> Vgl. Berdou (2006), S. 205f.

<sup>117</sup> Vgl. Fitzgerald und Ågerfalk (2005). Eine umfangreiche Untersuchung von Motivation und Anreizen in OSS-Projekten findet sich in Mundhenke (2007), S. 69-89.

<sup>118</sup> Vgl. Fitzgerald (2011), S. 27.

<sup>119</sup> „The cathedral and the bazaar“ ist zugleich der Titel seines Werkes, das aus einer Aufsatzsammlung hervorgegangen ist. Vgl. Raymond (2001).

<sup>120</sup> Vgl. Koch (2003), S. 58. Koch stellt dies als Organisationsform „Owner/Maintainer“ dar, die er als einfachste Organisationsform eines OSS-Projektes bezeichnet.

<sup>121</sup> Vgl. Karels (2003), S. 48, Michlmayr et al. (2008), S. 4.

<sup>122</sup> Vgl. Holck und Jørgensen (2004).



teiligten gemäß des Umfangs der Beteiligung im Projekt erfolgen; bspw. in *passiver Benutzer*, *aktiver Benutzer*, *peripherer Entwickler*, *Mit-Entwickler*, *Kern-Entwickler* und *Projektleiter*.<sup>123</sup>

Bei den Kern-Entwicklern handelt es sich um eine vergleichsweise kleine Gruppe von Personen, die in erheblichem Umfang an der Entwicklung des Programmcodes beteiligt sind.<sup>124</sup> Der Übergang von einem „peripheren Entwickler“ in Richtung der Kerngruppe der Entwickler kann für einen Einzelnen folglich sehr aufwändig sein.<sup>125</sup> Eine Untersuchung des Apache-Webserver-Projekts hat beispielsweise ergeben, dass 91 % aller Änderungsanfragen von lediglich 10 % aller Beitragenden gestellt werden.<sup>126</sup> Die Verweilzeit der Beteiligten in diesen zentralen Positionen im OSS-Projekt ist vergleichsweise lang; für den Fall der GNU/Linux-Distribution Debian wurde dies von Michlmayr et al. (2008) untersucht; etwas weniger als die Hälfte der Personen, die aus OSS installationsbereite Debian-Pakete erstellen, waren nach über sieben Jahren noch aktiv.<sup>127</sup>

Der Projektleiter oder eine bzw. mehrere Personen, die der obersten Ebene des Projektes direkt zugeordnet sind, sind zudem für das Releasemanagement der OSS zuständig. Dies kann die Steuerung der Veröffentlichung von OSS im Sinne eines Projektmanagements darstellen, was auch das Setzen der Deadlines für Releases umfasst.<sup>128</sup> Die Aufteilung in eine Gruppe von Hauptentwicklern und eine größere Zahl weiterer Entwickler, die dieser Gruppe nicht angehören, wird als einer der Gründe für das Fehlen der Ineffizienz großer Teams bei der OSS-Entwicklung angesehen.<sup>129</sup>

#### 2.2.4 Koordination in OSS-Projekten

Wenngleich somit die Stabilität hinsichtlich zentraler Personen oftmals gegeben ist, erscheint die Entwicklung von OSS im Vergleich zu der Entwicklung proprietärer Software insbesondere vor dem Hintergrund der geografisch verteilten Entwickler unter dezentraler Führung vergleichsweise unkoordiniert. Um den verteilten Entwicklungsprozess von OSS handhabbar zu machen, müssen verschiedene Bedingungen gegeben sein, die die Koordination unterstützen sowie für eine anhal-

---

<sup>123</sup> Vgl. Koch (2003), S. 58. Es herrscht keine einheitliche Benennung oder Wahl der Granularität; so unterteilen bspw. Ye et al. (2005) in *Project Leader*, *Core Member*, *Active Developer*, *Peripheral Developer*, *Bug Reporter*, *Reader* und *Passive User*.

<sup>124</sup> Vgl. Koch (2003), S. 60.

<sup>125</sup> Vgl. Krogh und Spaeth (2007), S. 240.

<sup>126</sup> Vgl. Asundi (2005).

<sup>127</sup> Vgl. Michlmayr et al. (2008), S. 20.

<sup>128</sup> Vgl. Holck und Jørgensen (2004), S. 11-15, die als Beispiele die großen OSS-Projekte Mozilla und FreeBSD nennen. Das Setzen von Deadlines wird von verschiedenen Autoren jedoch als untypisch für ein OSS-Projekt angeführt, vgl. exemplarisch Hang et al. (2005), S. 225. Eine Untersuchung des Release Managements dreier OSS-Projekte liegt mit Erenkrantz (2003) vor.

<sup>129</sup> Vgl. Strahringer (2003), S. 13.

tende Motivation der Entwickler sorgen. Brügge et al. (2004) nennen folgende die Koordination von OSS unterstützende Bedingungen:<sup>130</sup>

- „Eindeutige Schnittstellen zwischen den Programmteilen,
- klare Regeln für die Übermittlung von Programmcode,
- Innovation und Weiterentwicklung durch inkrementelle Schritte,
- hohe Transparenz,
- weltweite Vernetzung und Kommunikation zu günstigen Preisen,
- „natürlich“ anerkannter Projektleiter und Antiforking-Norm,
- niedrige Eintrittsbarrieren und großer Entwicklerpool sowie
- ausreichende Ressourcen“.

Die Koordination wird in OSS-Projekten in hohem Maße technisch unterstützt, um einen möglichst geringen Koordinationsaufwand zu erzeugen, aber dennoch eine redundante Durchführung von Arbeiten weitestgehend zu vermeiden.

Die Dokumentation von Änderungen in OSS-Projekten wird daher üblicherweise mittels verschiedener Softwarewerkzeuge durchgeführt, die Arbeiten automatisieren und als Teil der Koordinationswerkzeuge in OSS-Projekten anzusehen sind.<sup>131</sup> Sie helfen, die Zusammenarbeit einer großen Anzahl von Entwicklern ohne zusätzlichen manuellen Aufwand sicht- und nachvollziehbar zu machen.

Viele OSS-Projekte nutzen Versionskontrollsysteme (bspw. Concurrent Version System, CVS<sup>132</sup> und Apache Subversion, SVN<sup>133</sup>), um Änderungen am Quelltext festzuhalten. Obwohl diese Systeme auch von Entwicklern proprietärer Software genutzt werden, sind sie vor dem Hintergrund der üblicherweise geografisch verteilten Entwicklung von OSS als Werkzeug für diese Projekte besonders geeignet.<sup>134</sup> Versionskontrollsysteme unterstützen Projekte mit einer großen Zahl von Entwicklern, die berechtigt sind, Änderungen in das Versionskontrollsystem einzubringen (sogenanntes „commit“). Während das Mitlesen in Versionskontrollsystemen oftmals für alle Personen freigegeben ist, wird die Vergabe der Rechte zum Einreichen von Quelltext üblicherweise reglementiert. Wenngleich aufgrund der Versionskontrolle Änderungen vergleichsweise leicht rückgängig zu machen sind, wird oft nur einer kleinen Gruppe von Personen ein *Commit*-Recht gewährt. Nicht berechtigte Benutzer müssen in diesem Fall Quelltext über einen berechtigten Be-

---

<sup>130</sup> In einem Fall gekürzte Überschriften aus Brügge et al. (2004), S. 55-58.

<sup>131</sup> Vgl. Joode et al. (2008), S. 104f. Aufgrund der üblicherweise freien Verfügbarkeit der enthaltenen Daten werden Inhalte dieser Koordinationswerkzeuge auch wissenschaftlich ausgewertet. Eine Übersicht über Tools, die helfen, die genannten Koordinationswerkzeuge als Datenquellen wissenschaftlich auszuwerten findet sich bspw. in Robles et al. (2009).

<sup>132</sup> Siehe [<http://www.nongnu.org/cvs>].

<sup>133</sup> Siehe [<http://subversion.apache.org>].

<sup>134</sup> Vgl. Jansen und Brinkkemper (2006).

Anpassung von Open-Source-Software in  
Anwenderunternehmen

Keßler, S.

2013, XIX, 209 S. 29 Abb., Softcover

ISBN: 978-3-658-01954-9