

*Beyond Outsourcing*

# Commercial Open Source Development

Robert Schmitt, Carsten Behrens und Klaus Hense, RWTH Aachen



Prof. Dr.-Ing. Robert Schmitt ist als Inhaber des Lehrstuhls für Fertigungsmesstechnik und Qualitätsmanagement und Direktor des Werkzeugmaschinenlabors WZL der RWTH Aachen. Er ist Direktoriumsmitglied des Fraunhofer-Instituts für Produktionstechnologie (IPT).



Dipl.-Ing. Klaus Hense ist Mitarbeiter am Lehrstuhl für Fertigungsmesstechnik und Qualitätsmanagement. Sein Arbeitsschwerpunkt umfasst Beschaffungs- und Produktstrategien.



Dipl.-Ing. Carsten Behrens ist Mitarbeiter am Lehrstuhl für Fertigungsmesstechnik und Qualitätsmanagement. Sein Arbeitsschwerpunkt umfasst Unternehmens- und Bereichsstrategien.

künftig zu stärken, muss Outsourcing jedoch weiterentwickelt werden. Der Grundgedanke des Geschäftsmodells Commercial Open Source Development (COSD), welches am Werkzeugmaschinenlabor WZL der RWTH Aachen konzipiert wurde, ist es, den Open Source Gedanken aufzugreifen und Outsourcing weiterzuentwickeln. COSD hat das Potenzial, das Spannungsfeld zwischen Kundenorientierung, Individualisierung, Entwicklungskosten und Fehlerwahrscheinlichkeit aufzubrechen und bindet dabei den Anwender großflächig, systematisch und gewinnbringend in die Entwicklung ein. Zwei aktuelle Entwicklungen sprechen für COSD: Technische Produkte können zunehmend durch Software beeinflusst werden und die Open Source Prinzipien haben eine hohe Reife erreicht.

Die Konzentration auf Kernkompetenzen, die Einbindung externen Know-hows, Kostensenkung sowie kapazitive und finanzielle Freiräume durch den Fremdbezug von Leistungen und Produkten sind die vorherrschenden Motive für Outsourcing [1]. Einsatz findet Outsourcing vornehmlich in vertraglich geregelten professionellen Geschäftsbeziehungen (B2B). Die Leistungen werden anhand vorher erstellter Spezifikationen gesteuert und sind in ein striktes Zeit- und Kostencontrolling eingebunden (z.B. Stage-Gates). Im Fokus des Outsourcing-Managements liegen im Wesentlichen die Ermittlung auslagerbarer Prozesse, die Lieferanteneinbindung sowie Controlling und Qualitätssicherung der Leistungen (Bild 1).

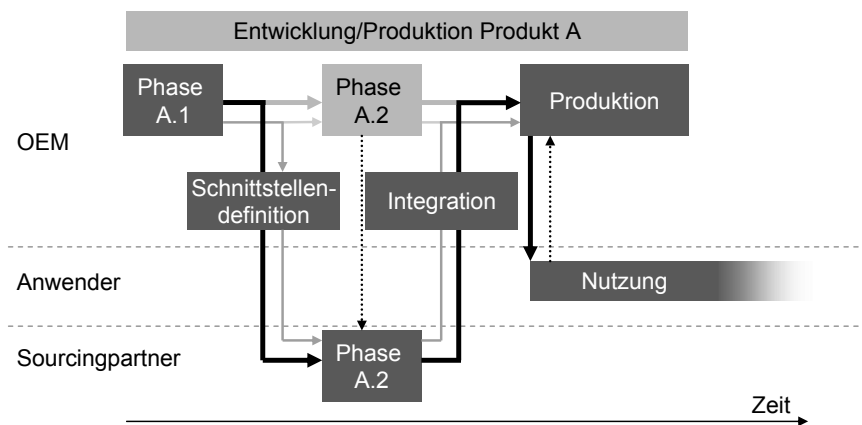
Gravierende Fehlerquellen liegen im klassischen Outsourcing vor allem bei softwareintensiven Produkten wie beispielsweise Mensch-Maschine Schnitt-

Outsourcing von Entwicklungsleistungen bleibt ein entscheidender Hebel zur Steigerung der Wettbewerbsfähigkeit – dies zeigen aktuelle Studien. Um Innovationspotenzial und Entwicklungseffizienz auch

**Kontakt:**

Dipl.-Ing. Carsten Behrens  
Lehrstuhl für Fertigungsmesstechnik  
und Qualitätsmanagement  
WZL der RWTH Aachen  
Steinbachstraße 19  
52074 Aachen  
Tel.: 0241 / 80-27413  
E-Mail: c.behrens@wzl.rwth-aachen.de

Bild 1: Geschäftsmodell Outsourcing.



Legende: **→** : Materialfluss **→** : Informationsfluss **----->** : Finanzfluss

stellen (MMI) von Automobilen oder Maschinen. Somit wird herkömmliches Outsourcing zukünftig nicht mehr ausreichen, um im Wettbewerb entscheidende Vorteile zu erzielen. Aktuell wird den oben genannten Fehlerquellen begegnet, indem der Einsatz von Entwicklungsressourcen erhöht wird; hierbei entstehen jedoch hohe produkt- und qualitätsbezogene Kosten.

## Open Source in der Software-Entwicklung

Eine kurze Skizze des Open Source Prinzips, wie aus der Softwaretechnik bekannt, ist notwendig, um das Geschäftsmodell des Commercial Open Source Development bewerten zu können.

Bei Open Source wird zwischen Distributoren und Anwendern unterschieden. Die Distributoren stellen regelmäßig aktualisierte Entwicklungsplattformen und Kombinationen bestehender Software-Module zusammen. Diese werden dann gemeinsam mit Supportleistungen vom Distributor vertrieben. Da der Quellcode der Entwicklungsplattform und der Module den Anwendern zugänglich ist, wird eine Monopolisierung des Wissens vermieden. Der Anwender nutzt die Software und hat die Möglichkeit, sie zu kommentieren, weiterzuentwickeln und Fehler zu beseitigen. Diese Ergebnisse der Anwender werden wiederum allen anderen Anwendern und auch den Distributoren zur Verfügung gestellt; es ergibt sich eine Entwicklungsgemeinschaft, die „Community“ (Bild 2). Untersuchungen

Bild 2: Open Source Entwicklung.

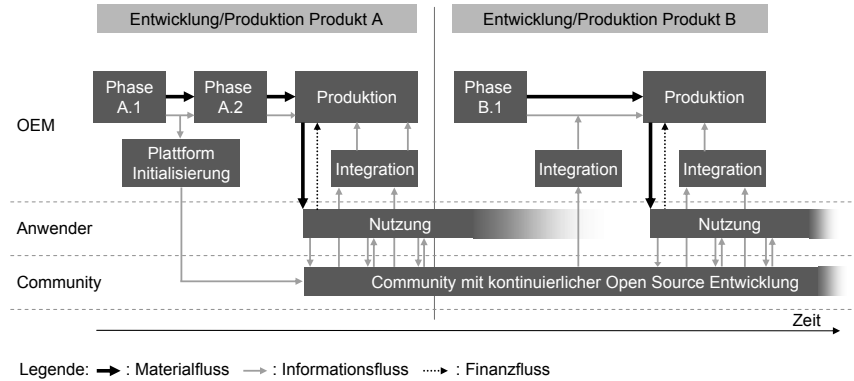
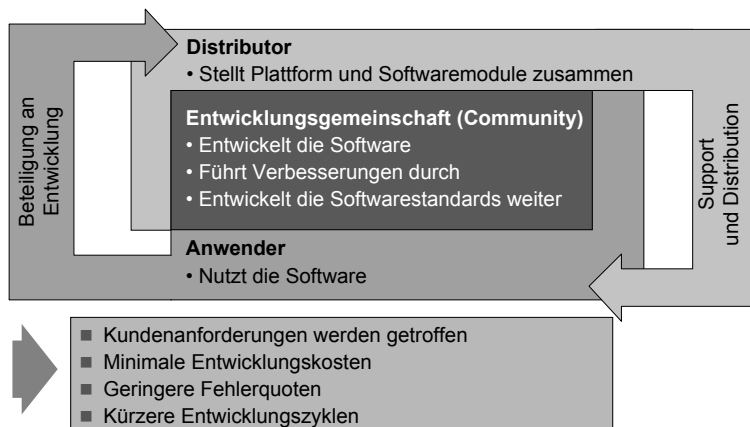


Bild 3: Geschäftsmodell Commercial Open Source Development (COSD).

zeigen, dass Open Source Entwicklungsergebnisse günstiger, schneller und fehlerärmer sein können [2].

## Von Outsourcing zu Commercial Open Source Development

Im Maschinen- und Fahrzeugbau sind zunehmend Funktionalitäten durch Software beeinflussbar. Dies führt dazu, dass der Anteil der notwendigen Entwicklungshardware im Sinne von Versuchs- und Produktionsmaterial pro entwickelter Funktion sinkt. Für die Entwicklung werden neben den Entwicklern (der Brainware) primär das Grundprodukt (die Hardware) und geeignete Schnittstellen zur Manipulation der Softwareanteile des Produkts benötigt. Hinzu kommt eine Tendenz hin zu Produktinnovationen, die durch die Kunden initiiert sind. Je nach Branche

können zwischen 20 und 80 % aller Neuproduktentwicklungen auf eine Idee (und oft auch ersten Prototyp) der Nutzer zurückverfolgt werden [3].

Diese neuen Rahmenbedingungen eröffnen das Potenzial, Entwicklungen und Anpassungen durch den Anwender des Endprodukts durchführen zu lassen. Die große Entwicklerressource „Anwender“ ist somit grundsätzlich auch in softwarefernen Branchen prinzipiell ähnlich erschließbar wie bei Open Source Entwicklungen in der Softwarebranche.

Das Geschäftsmodell COSD verfolgt den Ansatz, auch Nicht-Softwarebranchen die Potenziale von Open Source zugänglich zu machen. Der OEM gewährt dabei den Anwendern seiner Endprodukte einen Zugriff auf seine Produkte, sodass sie Manipulationen durchführen können. Diese Manipulationen können wieder in bestehende Endkunden- und neue Entwicklungsprodukte überführt werden. Der bisher übliche Einkauf von Entwicklungsdienstleistungen kommerzieller Partner wird so durch die Einbindung intrinsisch motivierter Anwender in eine Entwicklungs-Community ersetzt.

## Funktionsprinzip Commercial Open Source Development

Drei Phasen gliedern das Funktionsprinzip von COSD: Die Initialisierung, die kontinuierliche Open Source Entwicklung der Anwender und die OEM-Ergebnisintegration (Bild 3).

Die erste Phase von COSD (Initialisierungsphase) enthält alle Schritte, die der OEM durchzuführen hat, damit eine sichere Entwicklungsplattform gewährleistet ist. Dies erfolgt technisch über eine Modularisierung der Software in eine System-Software (entspricht Betriebssystem und Entwicklungsumgebung), in gekapselte sowie in Open Source-Funktionen, -Parameter und -Darstellungen. Drei wesentliche Gründe erfordern die Kapselung bestimmter Module: Sicherheitskritische Funktionen dürfen nicht manipuliert oder ausgehebelt werden können; hochkomplexe Module sind nicht für semi-professionelle Entwickler geeignet; die Wettbewerber sollen keinen Einblick in markenbildende Funktionen erhalten. Der für COSD freigegebene Bereich ist skalierbar von grafischer Konfigurierbarkeit der Anwenderebene über funktionale Konfigurierbarkeit hin zu funktionaler Entwicklung. Um möglichst viele Anwender einbinden zu können, sind COSD-Bereiche so einfach wie möglich zu gestalten (z.B. grafisches Auswahlmenü). Die Manipulationsschnittstelle kann onboard oder offboard (Notebook + Schnittstelle) ausgeführt werden; allerdings eignet sich nur die offboard-Schnittstelle für komplexere Manipulationen.

Damit den Anwendern ein funktionsstüchtiges Produkt als Plattform für die kontinuierliche Open Source-Entwicklung zur Verfügung steht, führt der OEM nach Entwicklung und Kapselung der Plattform vor der Auslieferung zunächst selbst eine initiale Applikationsentwicklung durch.

Anschließend startet die 2. Phase, die Open Source-Entwicklungsphase. Die Anwender können Funktionalität, Regelverhalten oder Darstellung nach eigenen Wünschen und Vorstellungen auf der Basis der COSD-Plattform anpassen. Im Systemkontext hat der Anwender die Möglichkeit, sein Ergebnis selbst zu testen (das Produkt steht ihm ja zur Verfügung) und die Modifikation anschließend der Community zur Verfügung zu stellen. Die Community übernimmt die Rolle des kritischen Regulativs, regt weitere Verbesserungen

an oder führt diese auch selbst durch. Über die Regulativ-Funktion ist eine intensive und schnelle Qualitätssicherung bei gleichzeitig hoher Innovationsrate sichergestellt.

Parallel zur Open Source-Entwicklungsphase startet die 3. Phase, die OEM-Ergebnisintegration, bei welcher der OEM die Community-Ergebnisse auswertet. Als Einstiegslösung ist die statistische Auswertung der Konfigurations-Vorlieben in Abhängigkeit von Benutzergruppen zu nennen; der OEM kann so künftig ohne aufwändige Benutzerstudien die präferierten Konfigurationen als Grundeinstellung übernehmen. Die Übernahme und Integration neuer Softwarefunktionen und -architekturen ist bedeutend anspruchsvoller, aber auch nutzenbringender. Die Open Source-Ergebnisse können sowohl in der laufenden Serie über eine Release-Steuerung, als auch in der Entwicklungsphase von Folgeprodukten der gleichen Produktgattung integriert werden. Dies ist auch ein möglicher Ausweg, um die aktuell divergierenden Zeiträume für Innovation und Produktnutzung wieder zu schließen. Wenn sich über die evolutionäre Open Source-Entwicklung keine Verbesserungen mehr erzielen lassen, muss die Entwicklungsplattform grundlegend aktualisiert werden. Open Source in der Software-Branche zeigt, dass erforderliche Anpassungen aufgrund von Kernel-Umstellungen durch die enorm breite Entwicklerressourcenbasis zügig abgearbeitet werden.

Bei COSD hat der OEM also drei Stellhebel, um die Entwicklung zu steuern: die Ausgestaltung der Entwicklungsplattform, die Steuerung des Freigabeumfangs und die Integration der Anwender-Entwicklungen.

### Beispielanwendung in der Automobilindustrie

Ein Beispiel für eine onboard COSD Lösung veranschaulicht das Geschäftsprinzip: Eine nicht intuitiv gestaltete Menüführung multimedialer Benutzerschnittstellen zur Steuerung einer Vielzahl von Fahrzeugfunktionen, war in jüngerer Vergangenheit häufiger Kritikpunkt an

Fahrzeugen, welche bis hin zum Kundenverlust an Wettbewerber führte.

COSD ermöglicht eine evolutionäre Entwicklung und Individualisierung der Menüführung in den drei Phasen Initialisierung, kontinuierliche Entwicklung und Ergebnisintegration. Der erste Schritt der Initialisierungsphase ist die Modularisierung der Bordcomputersoftware in Funktionen, Strukturparameter und Menünamen. Diese werden so ausgestaltet, dass die Funktion (zunächst) nicht manipulierbar ist. Zur Manipulation der Menüführungen kann eine onboard Applikation geschaffen werden, welche die freie Konfiguration der Menüstruktur und -namen mittels der vorhandenen Hardware ermöglicht. Dabei sind die Wiederherstellbarkeit des Auslieferungszustands und mehrere Speicherplätze für die Kundenkonfigurationen zu berücksichtigen. Der zweite Schritt ist die Definition einer Schnittstelle zum Auslesen der Manipulationsdaten. Das in modernen Fahrzeugen bereits vorhandene Bordinformationsnetz ist prinzipiell geeignet, muss jedoch zu diesem Zweck überarbeitet werden. Der dritte Schritt umfasst die Bereitstellung eines Kommunikationsforums im Internet, welches Kommentare und Anregungen der Anwender bündelt.

Die Phase der kontinuierlichen Anwenderentwicklung beginnt mit der Auslieferung der ersten Fahrzeuge. Einige Anwender werden selbst aktiv, oder beauftragend entsprechend ihren Vorstellungen, an der Entwicklung teilnehmen. Die Rückkopplung zum OEM kann mittels der Konfigurationsauslesung bei Inspektionen und anderen Werkstattbesuchen durchgeführt werden, abhängig vom aktuellen Stand der Vernetzung von Werkstätten mit den OEM.

Die abschließende Phase der Ergebnisinterpretation beginnt mit einer statistischen Auswertung der präferierten Menüführungen; diese können in Fahrzeugen der laufenden Produktion integriert oder in die Entwicklung des Folgemodells aufgenommen werden. Der OEM kann auch mehrere, kundengruppenspezifische Konfigurationen anbieten – sei es bei der Auslieferung, als auch nachträglich über das bestehende Händler- bzw. Werkstattnetz.

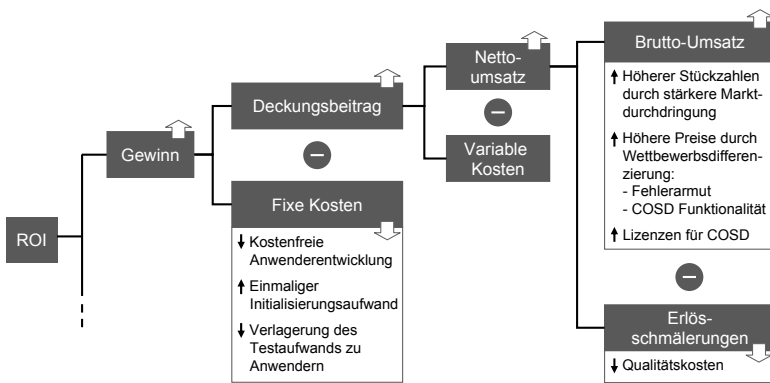


Bild 5: Beispiel-Applikation COSD: Multimediale Benutzerschnittstelle im PKW.

Der Anreiz und damit die intrinsische Motivation zur Beteiligung an der Entwicklung ist im Bereich Automobil speziell im Kundenwahrnehmungsumfeld wie Anzeigen bis hin zu Schaltlogiken von Automatikgetrieben und Fahrwerkregelsystemen besonders ausgeprägt. Die Freigabe zur Anwenderentwicklung dieser Funktionalitäten ist von daher auch gut zu vermarkten und die Entwicklungsergebnisse durch Adaption an Kundenforderungen gewinnbringend zu verkaufen.

Für komplexere Konfigurationen und Entwicklungen beispielweise bezüglich Schaltlogiken und Fahrwerkregelsystemen sind jedoch offboard Lösungen für COSD erforderlich. Bedingung hierfür ist eine geeignete Infrastruktur bestehend aus einer für den Anwender nutzbaren Kommunikationsschnittstelle am Fahrzeug sowie einer – Open Source entwickelten – Entwicklungsoberfläche im PC. Offboard COSD ermöglicht über eine Community Internet-Plattform den selbstständigen Austausch der Entwicklungsergebnisse unter den Anwendern. Dieser Mikroentwicklungszyklus beschleunigt die COSD Entwicklung signifikant, wie aus der Softwarebranche bekannt ist.

### COSD: Goldesel für OEM und Zulieferer?

Der Einfluss des Geschäftsmodells COSD auf den Gewinn und den ROI eines Unternehmens lässt sich qualitativ anhand des DuPont Kennzahlensystems [4] darstellen: Die Hauptstellhebel von COSD liegen in den Bereichen der Steigerung des Bruttoumsatzes bei Reduktion der Erlösschmälerungen sowie einer

Reduktion der Fixkosten, speziell den Produktentwicklungskosten (Bild 5).

Die Steigerung des Bruttoumsatzes wird gefördert durch den Mehrwert des Produkts für den Kunden. Dies umfasst vorzugsweise Mehrfunktionalität im Sinne von Produktinnovationen und Funktionsalternativen, welche zu einer zusätzlichen Produktindividualisierung führt. Darüber hinaus ist COSD mit seinen Freiheiten und Möglichkeiten selbst ein wettbewerbsdifferenzierendes Merkmal, für das viele Kunden bereit sind, höhere Produktpreise zu akzeptieren. Gebührenmodelle für die COSD-Funktionalität können – gezielt eingesetzt – den Umsatz zusätzlich steigern.

Fixkosten werden bei COSD durch die kostenfreie Anwenderentwicklung bei einem begrenzten und einmaligen Initialisierungsaufwand gesenkt. Dies gilt speziell für Variantenentwicklungen und evolutionäre Fortentwicklungen. Der zum Kunden verlagerte Testaufwand der Entwicklung ermöglicht ebenfalls, die entwicklungsbezogenen Kosten zu reduzieren.

### Vorteile, Risiken und Fazit

Kostenreduktion sowie die bessere Erfüllung von Kundenforderungen sind die Motivation für Commercial Open Source Development. Die Mikrozyklen der Open Source Entwicklung begünstigen eine hohe Innovations- und Entwicklungsgeschwindigkeit bei gleichzeitiger Fehlerarmut durch die Parallelisierung von Testen, Fehlerbeseitigung und Verbesserung. Die Einbindung von Kunden mit hoher Produktaffinität

durch intrinsische Motivation ist dabei der Schlüssel zum Erfolg. Die Anpassung an seinen Bedarf, das Debugging, sowie das Ansehen, das in der Community erlangt werden kann, sorgen beim Anwender für eine hohe Motivation.

Risiken von COSD wie die Integrations-Komplexität, die Produktsicherheit und Piraterie werden am Werkzeugmaschinenlabor WZL in weiterer Forschung abgegrenzt und in branchenspezifische Lösungskonzepte überführt.

Die Einführung unter Steuerung der Risiken des Commercial Open Source Developments kann sukzessive über eine stufenweise Freigabe der Entwicklungsbereiche erfolgen; die Weichen müssen jedoch bereits heute gestellt werden.

### Literatur

- [1] Schmitt, R., Hense, K., Klenter, G., Lesmeister, F.: In or Out. In: QZ 50 (2005) 9, S. 22-24.
- [2] Saleck, T., Safari-Fard, M.: Chefsache Open Source. Kostenvorteile und Unabhängigkeit durch Open Source. Wiesbaden 2005.
- [3] Reichwald, R., Piller, F.: Interaktive Wertschöpfung – Open Innovation, Individualisierung und neue Formen der Arbeitsteilung. Wiesbaden 2006.
- [4] Horváth, P.: Controlling. München 1998, S.551.

### Schlüsselwörter:

Outsourcing, Open Source, Kundeninnovation, Anwenderentwicklung

### Commercial Open Source Development – Customer Development for Free

A new business model, conceived by WZL of the University of Aachen, bases on Open Source development and advances Outsourcing. Commercial Open Source Development (COSD) has the capability to break open the area of conflict between customer orientation, individualisation, product design costs and error probability by embedding the customer systematically and lucratively into product development. Two trends of product design support COSD: Functions of technical products can be increasingly affected through Software solutions; Open Source principals have reached a high level of maturity.

Keywords: outsourcing, Open Source, customer innovation, user development