# „Jazz“ – Kreativität jedes einzelnen, Perfektion im Zusammenspiel

Hubert Biskup

Rational Software
IBM Deutschland GmbH
Hollerithstr. 1
81829 München
Hubert.Biskup@de.ibm.com

**Abstract:** Die Software-Entwicklung zu industrialisieren ist seit Jahren ein Bestreben der Softwarebranche. Dabei stellt sich natürlich die Frage, was darunter zu verstehen ist. Oft wird ein Vergleich mit der Fertigungsindustrie gezogen. Mit einem kleinen Rückblick in die Geschichte der Softwareentwicklung möchte dieser Beitrag das Augenmerk weg von einer wie auch immer gearteten Automatisierung, hin zu einer flexiblen Unterstützung kreativer Arbeit richten. Die aktuelle Entwicklungsplattform der IBM „Jazz“ ist ein Beispiel, wie kreative Arbeit durch Softwarewerkzeuge unterstützt werden kann.

## 1 Programmieren in der Software-Fabrik?

„The Art of Computer programming“ (Donald Knuth, 1962)[1], ein Klassiker der Informatikliteratur, trug der Situation Rechnung, daß Softwareentwicklung der frühen Tage eine Kunst war, provozierte aber auch gleichzeitig damit. Es wurde deutlich, daß bei zunehmend größer werdenden Software-Systemen die „Kunst des Programmierens“ in eine wiederholbare Tätigkeit, möglichst personenunabhängig und mit gleichbleibender Qualität überführt werden mußte. Eine beliebte Metapher war folglich, Software-Entwicklung mit einer Fabrik zu vergleichen (Ende der 1980er Jahre wurde z.B. zur Förderung der europäischen Software-Industrie ein Projekt namens Eureka Software Factory mit EU-Mitteln gefördert (vgl.[2])). Dieser Vergleich stimmt jedoch nicht wirklich, wenn man sich die Fließbandproduktion industrieller Fertigung ansieht (vgl. Abbildung 1).

[1] Donald Knuth, *The Art of Computer programming,* 1962; Quelle Wikipedia: http://en.wikipedia.org/wiki/The_Art_of_Computer_Programming
[2] Christer Fernström, *The Eureka Software Factory*: Concepts and accomplishments, 1991; www.springerlink.com/index/f4272xvhq213146u.pdf

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Abbildung 1: Fließband in Charles Chaplins Film *Modern Times*[3]

Softwareentwicklung sollte nicht mit der Fließbandarbeit in der Fabrik verglichen werden, sondern mit der ingenieurmäßigen Konstruktion eines Produktes und dessen Serienreifmachung Waren frühe „Software-Schmieden" vom Geist der jeweiligen Softwareentwickler geprägt und eher mit einer Manufaktur vergleichbar, so wurden bereits frühzeitig arbeitsteilige Verfahren entwickelt, um überhaupt in der Lage zu sein, große Softwaresysteme wie z.B. leistungsfähige Betriebssysteme für Großrechner oder Compiler zu entwickeln. Systeme dieser Größe und Komplexität ließen sich nicht mehr von einzelnen Programmierern entwickelt oder überblicken. Viele Methoden des Softwareengineering beschäftigen sich damit, wie man Komplexität meistern kann und große Entwicklungsaufgaben auf viele Schultern verteilen kann.

## 2 Vom Programmieren zum Konstruieren

In den letzten Jahren läßt sich durchaus ein Wandel bei der Software-Entwicklung beobachten. Normen und Standards und darauf basierende Komponenten sind entstanden wie Normteile im Maschinenbau oder Fertigteile im Ingenieurbau. Software wird zu einen Großteil nicht mehr „geschrieben" sondern wird aus fertigen, wiederverwendbaren Komponenten zusammengesetzt. Wer schreibt z.B. noch einen Druckertreiber für seine Tabellenkalkulation? Der wird wie selbstverständlich vom Druckerhersteller mitgeliefert und integriert sich nahtlos in das verwendete Betriebssystem.

Konsequenterweise richtet sich auch die Anwendungsentwicklung auf arbeitsteilige, komponentenorientierte Entwicklung aus und die Arbeitsumgebungen der Softwareentwickler müssen diesen geänderten Vorgehensweisen gerecht werden. Nicht mehr der Editor und der Compiler sind die wichtigsten Werkzeuge des Softwareentwicklers. Komplette Bibliotheken und in unterschiedlichen Zusammenhängen wiederverwendete Softwarekomponenten sind wichtige Bestandteile

3 Quelle Wikipedia: http://www.charliechaplin.com/biography/articles/6-Modern-Times

in der „Werkzeugkiste“ jedes Softwareentwicklers. Die Art und Weise, wie Software entwickelt wird, hat sich in den letzten Jahren grundlegend gewandelt: aktuelle Software-Systeme lassen sich nicht allein „programmieren“. Erst die Verwendung einer Vielzahl von spezialisierten Werkzeugen ermöglicht die Entwicklung komplexer Softwaresysteme.

## 3 Werkzeuge und Automaten

Abbildung 2: Werkzeugsatz eines Handwerkers [4]

Der Entwurf grafischer Oberflächen oder die Modellierung z.B. von Arbeitsabläufen mithilfe einer grafischen Notation sind kreative Tätigkeiten zu deren Ausführung man spezialisierte Werkzeuge benutzt, wie ein Handwerker für bestimmte Tätigkeiten spezielle Werkzeuge aus seiner Werkzeugkiste (vgl. Abbildung 2). Die Metapher von Werkzeug und zu bearbeitendem Material (vgl. [5)] findet man in vielen Software-Werkzeugen wieder. Grafische Werkzeuge besitzen z.B. eine *Werkzeug*palette (vgl. Abbildung 3). Und Arbeitsergebnisse im Entwicklungsprozeß werden als Artefakte in der Projektbibliothek abgelegt.

[4] Quelle Wikipedia:
http://de.wikipedia.org/w/index.php?title=Datei:Furniture_installation_tools.jpg&filetimestamp=20060312001001

[5] Heinz Züllighoven, Werkzeug & Material-Ansatz; Quelle Wikipedia:
http://de.wikipedia.org/wiki/Werkzeug-_und_Materialansatz

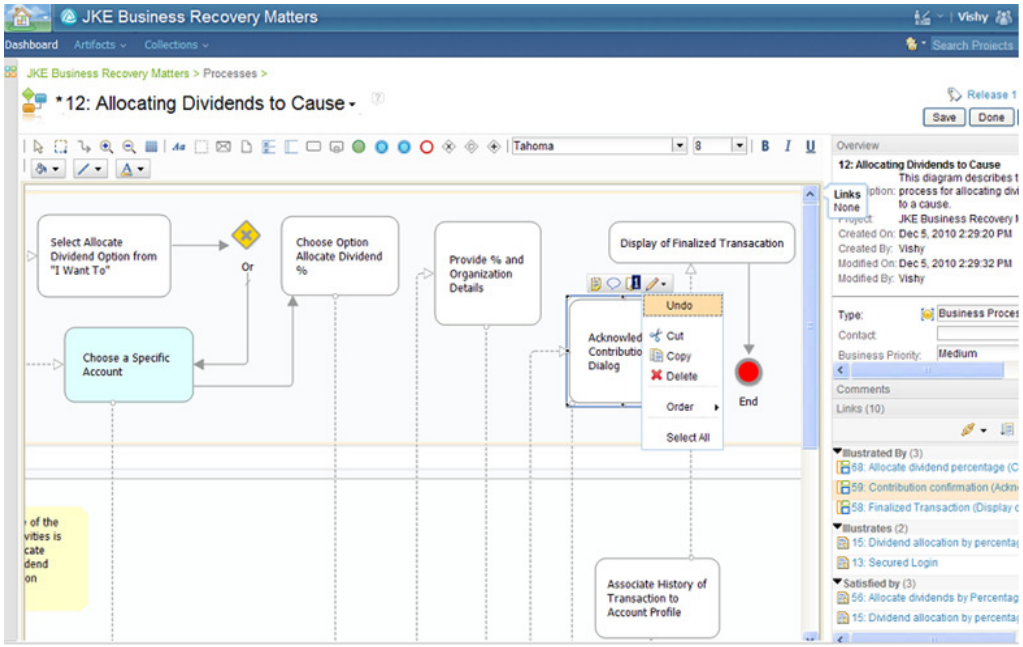


Abbildung 3: "Canvas" mit Toolpalette eines grafischen Softwarewerkzeugs

Andere Aufgaben dagegen, die weniger kreative Tätigkeiten umfassen, lassen sich ganz oder teilweise automatisieren. In der Softwareentwicklung gibt es eine Reihe von Routinetätigkeiten, die gemäß vordefinierter Regeln quasi vollautomatisch erfolgen können, wie z.B. Quellcode übersetzen, ein lauffähiges System in einem vollständigen Build erstellen, oder Regressionstests fahren.

## 4 Die Entwicklungsplattform „*Jazz*“

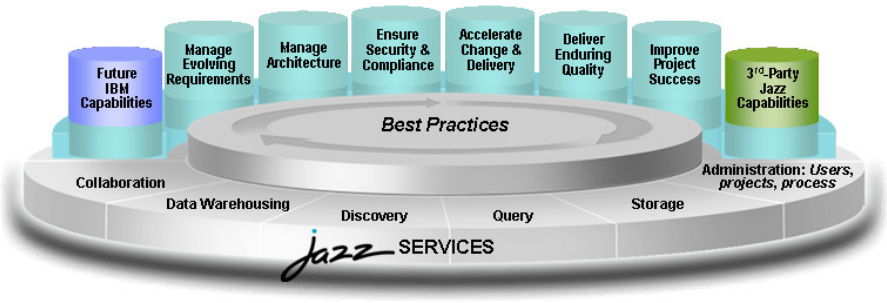


Abbildung 4: Die Jazz-Entwicklungsplattform


ISBN 978-3-88579-286-4

Die IBM-Entwicklungplattform „Jazz“ (vgl. Abbildung 4) unterstützt gezielt Entwicklungsteams bei ihren vielfältigen Aufgaben.

Softwareentwicklung wird strukturiert in Arbeitspakete (work items). Diese werden im Team einzelnen Bearbeiteitern zugeordnet (vgl. Abbildung 5). Anforderungen werden systematisch erfaßt und abgestimmt. Software wird „modelliert“, konstruiert und integriert. Mit abgestuften Testkonzepten wird verifiziert, daß die Anforderungen erfüllt werden. Dies geschieht nach bewährten Verfahren oder in agilen Projekten abgestimmt auf die jeweilige Projektsituation.

## 5 Projekte mit Rational Team Concert durchführen

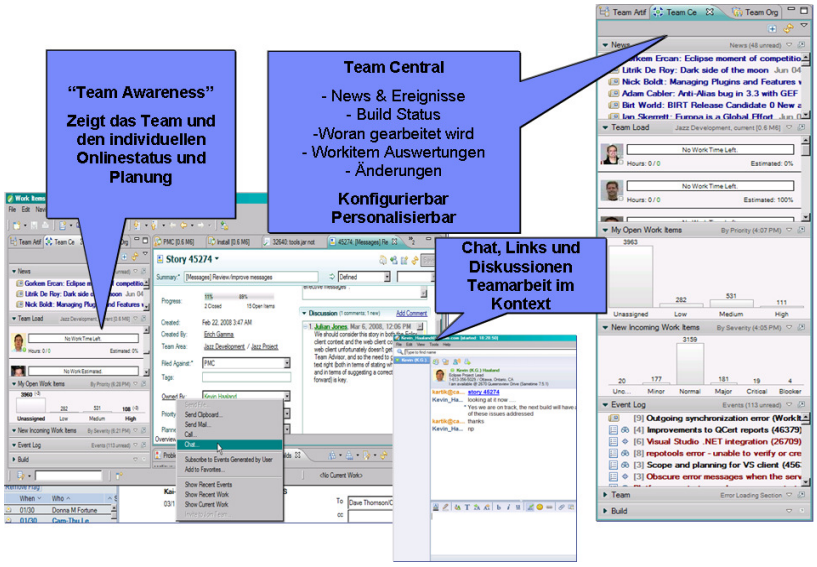


Abbildung 5: Teamorganisation mit Rational Team Concert

Rational Team Concert (RTC) auf Basis von Jazz bietet dem Softwareentwickler auf seine Aufgaben abgestimmte Unterstützung mit einer vollständigen Software-Entwicklungsumgebung. Bewährte Arbeitsweisen werden durch die Definition von Regeln unterstützt, die an kompletten Vorgehensmodellen wie z.B. RUP oder auch an agilen Verfahren wie z.B. Scrum ausgerichtet sein können.

Routineaufgaben, die man sinnvollerweise einer Maschine überläßt, wie z.B. Quellcode übersetzen, ein Build erstellen, oder Regressionstests durchführen, lassen sich mit RTC entsprechend automatisieren (vgl. Abbildung 6).


ISBN 978-3-88579-286-4

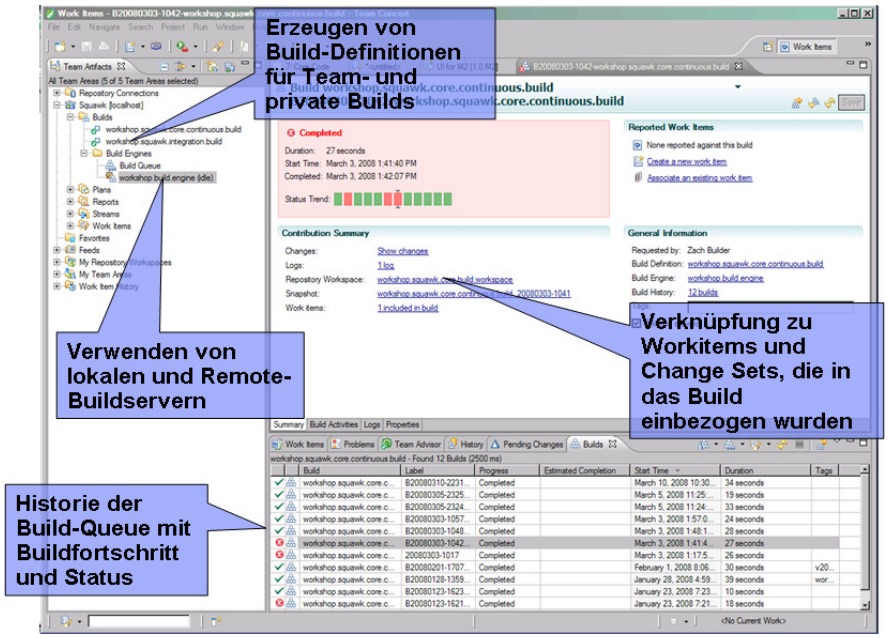


Abbildung 6: Automatisierung der Builds in RTC

## 6 Zusammenfassung

Die IBM-Entwicklungplattform „Jazz“ bildet eine offene Integrationsplattform für eine Vielzahl von Werkzeugen für den Softwareingenieur, die sowohl kreative Tätigkeiten adäquat unterstützen als auch Routineaufgaben „automatisieren“ können.

„Jazz“ ist dabei eine Metapher, die die Kreativität jedes einzelnen Mitspielers hervorhebt, erst im Zusammenspiel aller entsteht das Gesamtwerk in seiner Perfektion.

Softwareentwicklung ist schließlich keine Fließbandarbeit.


ISBN 978-3-88579-286-4