

ENTWICKLUNGSUMGEBUNG – Lernziele (LZ)



- (1) SOFTWAREENTWICKLUNG BEI C&M
Die Bedeutung der Softwareentwicklung in der Forschungsgruppe C&M wird verstanden
- (2) C&M-TFS UND SCRUM
Die wichtigsten durch den C&M-Team-Foundation-Server (TFS) bereitgestellten Funktionen und deren Nutzung innerhalb der Scrum-basierten Entwicklung bei C&M sind bekannt
- (3) ECLIPSE UND JAVA
Eclipse kann installiert und zur Entwicklung von Java-Programmen benutzt werden
- (4) JAVASCRIPT
Die wesentlichen der Skriptsprache JavaScript zugrunde liegenden Konzepte werden verstanden

1

30.10.2013

WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

(1) In der Forschungsgruppe C&M wird Software im Rahmen von Forschungs- und Projektarbeiten entwickelt mit den Zielen, (i) Studierende in diesem Bereich auszubilden und (ii) prototypische Entwicklungen zur Evaluation von Forschungsergebnissen durchzuführen.

(2) Der Team Foundation Server ist ein Produkt von Microsoft, das die teamorientierte Entwicklung von Software unterstützt und bei C&M zur Unterstützung der agilen Entwicklung auf der Basis von Scrum genutzt wird.

(3) Eclipse ist aufgrund des Plug-in-Konzepts eine flexibel erweiterbare integrierte Entwicklungsumgebung (Integrated Development Environment IDE). Die Java-Entwicklung ist ein zentraler Gegenstand anderer Veranstaltungen, wie z.B. "Programmieren" und "Praxis der Softwareentwicklung (PSE)". In der vorliegenden Kurseinheit werden daher elementare Java-Grundlagen nur noch einmal kurz aufgefrischt werden. Insbesondere wird gezeigt, wie die Java-Programmierung in der Eclipse-Umgebung unterstützt wird.

(4) Ein zentrales Konzept von JavaScript ist das prototypische Programmierparadigma.

C&M	Cooperation & Management
IDE	Integrated Development Environment
PSE	Praxis der Softwareentwicklung

Hauptquellen:

[DF+04] Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy: The Java Developer's Guide to Eclipse. Addison-Wesley, 2004.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

Entwicklung von Software in der Forschungsgruppe C&M



- (1) Softwareentwicklung ist sowohl in der C&M-Lehre als auch in der C&M-Forschung ein zentrales Thema
 - (1) Schwerpunkt ist die strukturierte, modellbasierte Entwicklung von komponenten- und serviceorientierten Web-Anwendungen
- (2) C&M entwickelt Software unter anderen Randbedingungen und mit anderen Zielen als ein Softwareunternehmen
 - (1) Es werden "Lehr- und Forschungsprototypen" und keine "Produkte" entwickelt
- (3) Vorteile einer abgestimmten Softwareentwicklung
 - (1) Komplexitätsreduktion und Kompetenzbündelung
 - (2) Fokussierung des C&M-Lehrangebots im Bereich der Softwareentwicklung
 - (3) Austausch von Softwareartefakten über Forschungsbereiche und Projektgrenzen hinweg

2

30.10.2013

WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[CM-TD-CSE:3]

(1) Die C&M-Lehre (insbesondere diese WASA-Vorlesung und das begleitende Praktikum) weist Berührungspunkte zum Software Engineering auf. In der C&M-Forschung wird Software entwickelt, um ein wissenschaftlich interessantes und neues Konzept umzusetzen.

(1.1) Während die "traditionellen" komponentenorientierten Web-Anwendungen den Schwerpunkt im Bachelor-Studiengang bilden, werden die im Wesentlichen auf Web-Services basierenden serviceorientierten Web-Anwendungen im Master-Studiengang behandelt.

(2.1) An ein Produkt ist die Anforderung zu stellen, dass dieses für den Einsatz im Wirkbetrieb bei einem Kunden geeignet ist.

Ein Lehr- und Forschungsprototyp wird weniger von den Anforderungen des Kunden, sondern von einem Lernziel bzw. einer Forschungsidee geleitet und ist i.d.R. stärker auf einen begrenzten Ausschnitt fokussiert.

(3) Durch ein abgestimmtes Vorgehen bei der Softwareentwicklung wird erreicht, dass sich C&M bei der Entwicklung der Prototypen stärker auf die eigentlichen Inhalte und Konzepte (und weniger auf die eingesetzten Technologien) konzentrieren kann.

(3.2) Nur im Falle einer Abstimmung der für die Softwareentwicklung genutzten Technologien innerhalb der Forschungsgruppe ist eine gezielte Ausbildung der Studierenden unter Nutzung dieser Technologien möglich. Das betrifft insbesondere das Praktikum, in dem die Studierenden gezielt an die bei C&M praktizierte Softwareentwicklung herangeführt werden sollen.

(3.3) Eine Abstimmung auf technischer Ebene ist eine Voraussetzung für den Austausch von Softwareartefakten, woraus eine stärkere inhaltliche Zusammenarbeit über Forschungsbereiche und Projektgrenzen hinweg ermöglicht wird.

WASA

Web-Anwendungen und Serviceorientierte Anwendungen

[CM-TD-CSE] Cooperation & Management: C&M-Softwareentwicklung, Technische Dokumentation, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck).

Klassifizierung der auftretenden Software



- (1) **Wirkbetriebssoftware**
 - (1) Liegt im Verantwortungsbereich der C&M-Organisationsbereiche
 - (2) Beispiele: Dokumentenablage und -verarbeitung (Office-Anwendungen), Webauftritt (z.B. Web Solution Manager von Opentext), Verwaltungssoftware (insbes. KIT Campus Management von CAS)
- (2) **Software-Entwicklungswerkzeuge**
 - (1) Eclipse-Plattform als die C&M-Standardentwicklungsumgebung
 - (2) C&M-Standard-ergänzende Werkzeuge
 - (3) Projektspezifische Werkzeuge
- (3) **Von C&M entwickelte Artefakte**
 - (1) Beispiele: BPMN-Prozessbeschreibungen, BPEL-Kompositionen, Web-Services, Komponenten

3

30.10.2013

WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[CM-TD-CSE:4]

(1) Zur Wirkbetriebssoftware gehören alle Softwaresysteme, die C&M zur Abwicklung seiner Geschäftsprozesse benötigt. Zu diesen Geschäftsprozessen zählen nicht die Prozesse, die die Softwareentwicklung in Lehre und Forschung betreffen.

(1.1) Die C&M-Technische Infrastruktur sorgt dafür, dass die vernetzte Systeminfrastruktur (Netz- und Systemebene) zur Nutzung der Anwendungssoftware bereitsteht. Für betriebliche Belange auf der Anwendungsebene (d.h., oberhalb der Systemebene) sind die C&M-Organisationseinheiten zuständig, die diese Anwendungen zur Durchführung ihrer Geschäftsprozesse benötigen.

Die Regelungen zu der Wirkbetriebssoftware findet sich in den entsprechenden C&M-weiten bzw. C&M-internen Teamarbeitsdokumenten.

(1.2) Ein momentan von C&M verfolgtes Ziel besteht in einem möglichst weitgehenden Verzicht der selbst betriebenen Server-Infrastruktur, indem zukünftig verstärkt Cloud-Services genutzt werden.

(2) Die zur Softwareentwicklung eingesetzten Werkzeuge setzen auf Teilen der Wirkbetriebssoftware auf.

(2.1) Diese Entwicklungswerkzeuge sind grundsätzlich Open-Source-Software. Konkret wird die "Eclipse IDE for Java Developers" verwendet, die ein Basis-Paket für die Entwicklung von Java-Programmen bereitstellt.

(2.2) Sind Eclipse-Plug-ins, die für spezielle Zwecke in einem C&M-Forschungsbereich benötigt werden und ggf. kommerzielle (keine Open-Source) Software darstellen.

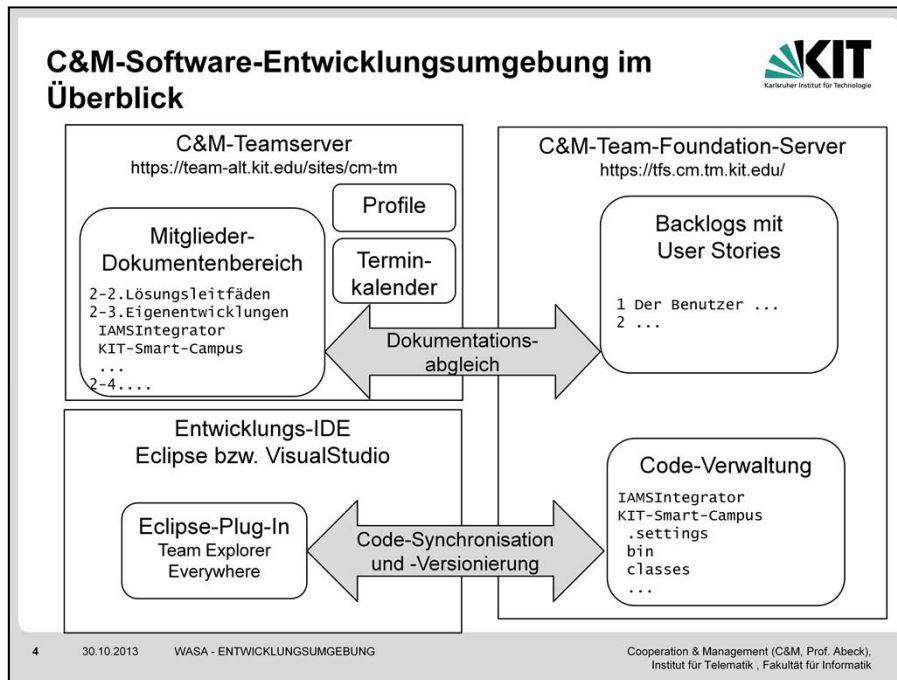
(2.3) Alle nicht auf Eclipse basierende Werkzeuge; häufig keine Open-Source-Software.

(2) (3) Diese beiden Klassen von Software sind Gegenstand dieses Dokuments.

(3) Zur teamorientierten Entwicklung und Versionskontrolle der Artefakte nutzt C&M Subversion (C&M-SVN).

BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notation
HIS	Hochschul-Informationen-Systeme
IDE	Integrated Development Environment

[CM-TD-CSE] Cooperation & Management: C&M-Softwareentwicklung, Technische Dokumentation, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck).



Die von C&M eingesetzte Software-Entwicklungsumgebung besteht aus insgesamt drei Systemen.

(C&M-Teamserver) Der wichtigste Teil des C&M-Teamservers [CM- im Hinblick auf die C&M-Software-Entwicklung ist der Ordner "2-3.Eigenentwicklungen". Hierin befinden sich zu jedem von C&M selbst entwickelten Software-System die meist zu jeder Phase (Analyse, Entwurf, Implementierung) erstellte Dokumentation sowie ein Benutzerhandbuch. Diese Dokumentation ist abzugleichen mit den Informationen im C&M-Team-Foundation-Server.

Andere auch für die Software-Entwicklung relevante Informationen sind die Lösungsleitfäden, und hier insbesondere der Haupt-Lösungsleitfaden C&M-SOFTWARE-ENTWICKLUNG im Ordner "2-2.Lösungsleitfäden".

In der Liste "Profile" befinden sich weitere Informationen zu den beteiligten Entwicklern. Die Liste "Terminkalender" dient u.a. zur Koordination der Treffen zwischen den Entwicklern.

(C&M-Team-Foundation-Server) Hinsichtlich der Dokumentation beinhaltet der C&M-Team-Foundation-Server (C&M-TFS) Informationen, die im Team erarbeitet werden und unmittelbar zur Aufteilung der Arbeiten auf die beteiligten Entwickler benötigt werden. Dieses sind insbesondere die in den (Product- bzw. Sprint-) Backlogs enthaltenen User Stories. Eine zu hohe Redundanz ist bei dem Dokumentenabgleich zu vermeiden. Die in den auf dem C&M-Teamserver abgelegten Dokumenten enthaltene Information sollte dazu dienen, einen Überblick und eine Orientierung hinsichtlich der im C&M-TFS abgelegten Informationen zu erhalten.

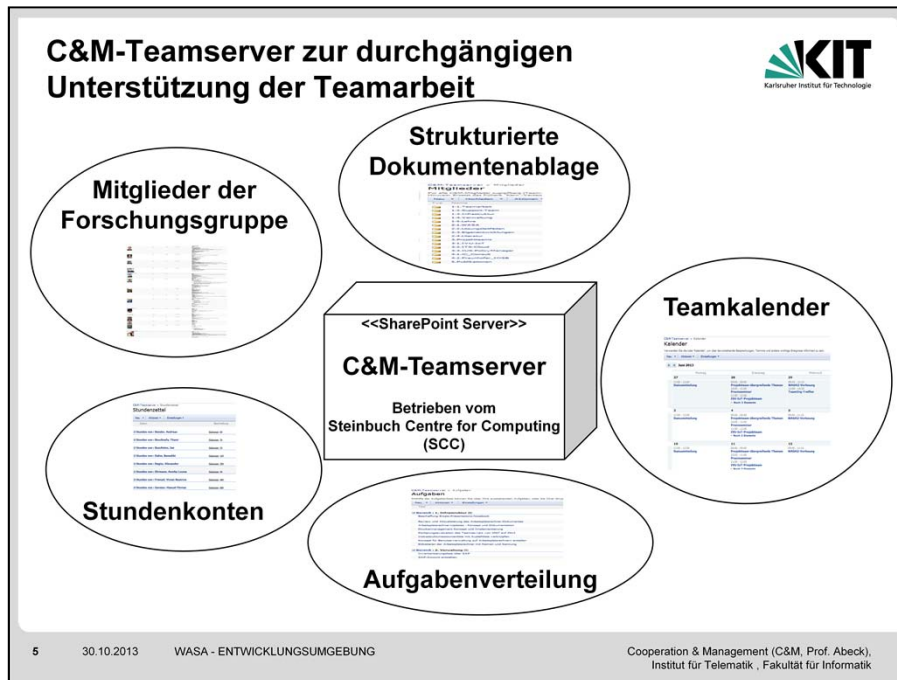
(Entwicklungs-IDE) Durch die Entwicklungs-IDE (Integrated Development Environment) wird die UML-Modellierung und insbesondere die Implementierung (Codierung) des Software-Systems unterstützt. Bei C&M werden Eclipse und VisualStudio eingesetzt.

Der C&M-TFS wird in der Implementierungsphase der Software-Entwicklung genutzt, um den im Team erstellten Code zu verwalten, indem die von den verschiedenen Entwicklern des Teams durchgeführten Änderungen synchronisiert werden und Versionen zu den einzelnen Software-Entwicklungsständen gebildet werden.

TFS Team Foundation Server

[CM-NCM] Cooperation & Management: Nutzung der C&M-Infrastruktur, C&M-Teamarbeitsdokument. *C&M-Teamserver* > *Mitglieder* > *1-3.Infrastruktur*

[CM-CMS] Cooperation & Management: C&M-SOFTWAREENTWICKLUNG, C&M-Teamarbeitsdokument. *C&M-Teamserver* > *Mitglieder* > *2-2.Lösungsleitfäden*



Der C&M-Teamserver ist der Dreh- und Angelpunkt der in der Forschungsgruppe durchgeführten wissenschaftlichen und projektbezogenen Arbeiten. Die Funktionen sind mit einem Standard-Web-Browser auch außerhalb des KIT-Netzes über eine VPN-Verbindung zugreifbar.

(Mitglieder der Forschungsgruppe) Hier stellen sich alle C&M-Mitglieder durch ein Profil mit einem Foto und einem kurzen Text vor.

(Strukturierte Dokumentenablage) Das ist der zentrale Informationsbereich von C&M, in dem alle internen und externen Dokumente der letzten zwei Jahre gehalten werden. Jeder Studierende hat hier seinen Arbeitsordner. Ältere Dokumente sind auf einem Datei-Server ausgelagert (siehe Beschreibung in [CM-NCM]).

(Teamkalender) Jedes bei C&M durchgeführte Projekttreffen ist hier mit der Liste der Teilnehmer und einer Agenda zum Treffen erfasst.

(Aufgabenverteilung) Ein bei C&M mitarbeitender Studierender trägt als ein vollwertiges C&M-Mitglied durch die Übernahme von Teamarbeiten (z.B. Erweiterungen des Teamservers, Wartungsarbeiten an der C&M-Infrastruktur) dazu bei, dass die Prozesse bei Forschungsgruppe gut funktionieren.

(Stundenkonten) Jeder Studierende protokolliert die von ihm im Rahmen seiner Arbeit investierten Stunden auf einem Stundenzettel.

VPN Virtual Private Network
 SCC Steinbuch Centre for Computing

[CM-NCM] Cooperation & Management: Nutzung der C&M-Infrastruktur, C&M-Teamarbeitsdokument. *C&M-Teamserver*
 > Mitglieder > 1-3. Infrastruktur

- (1) Mit der Entwicklung von Software verfolgt C&M die folgenden Ziele
 - (1) Erfüllung der Erwartung der Kunden
 - (2) Ausbildung von Studierenden
 - (3) Maximierung des Gewinns
 - (4) Tragfähigkeitsnachweise von wissenschaftlichen Arbeiten
- (2) Welche Arten von Software treten bei C&M auf?
- (3) Welche Systeme gehören zur C&M-Entwicklungsumgebung?
- (4) Welche Funktionen des C&M-Teamserver werden zu welchem Zweck für die Software-Entwicklung genutzt?

C&M-TFS UND SCRUM – Überblick



- (1) Webanwendung zur Projektplanung
- (2) Unterstützt den Scrum-Prozess
- (3) Bietet Code-Versionierung
- (4) Erreichbar unter:
<https://tfs.cm.tm.kit.edu>
- (5) Zugangsdaten: SCC-Konto (u-Kürzel)
- (6) Kann entsprechend des Lösungsleitfadens mit Visual Studio verbunden werden

7

30.10.2013

WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

(1) Bei dem Team Foundation Server handelt es sich um eine webbasierte Planungssoftware zur Softwareentwicklung.

(2) Er bildet alle Funktionen ab, die zur Durchführung von Scrum notwendig sind. Es lassen sich z.B. User Stories und Tasks anlegen, die Entwicklern zugewiesen können. Außerdem lassen sich die verschiedenen Backlogs pflegen und Releases planen. Das Monitoring erfolgt z.B. über Burndown-Charts.

(3) Quelltexte können direkt über den TFS versioniert werden.

(4) Achtung: Ohne Zugriff über SSL (<https://>) kann keine Verbindung aufgebaut werden.

(5) Es ist ggf. die Domäne (@kit.edu) mit anzugeben.

(6) Der Leitfaden findet sich unter [VS12]

[VS12] C&M-Teamserver > Mitglieder > 2-2.Lögunleitfäden > Windows_VisualStudioTFS > visualstudio_tfs.ppt

SCC	Steinbuch Centre for Computing
SSL	Secure Socket Layer
TFS	Team Foundation Server

C&M-TFS UND SCRUM – Überblick



- (1) Kollaborationsplattform, die Funktionen zur teambasierten Projektentwicklung bereitstellt
 - (1) Versionskontrolle
 - (2) Unterstützung der Anforderungserhebung
 - (3) Erstellung von Berichten
 - (4) Automatisierung der Builds
 - (5) Bereitstellung von Prozessvorlagen
- (2) Prozessvorlage definiert
 - (1) Typen von Arbeitsaufgabenobjekten für die Nachverfolgung
 - (2) Standardregeln zur Verwendung durch die Teammitglieder
- (3) Auf Basis der Prozessvorlagen werden verschiedene Entwicklungsverfahren unterstützt
 - (1) Bei C&M wird die Entwicklung gemäß der agilen Methode Scrum durchgeführt

8

30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

(1) Der Team Foundation Server (TFS) wurde von Microsoft als Kollaborationsplattform zur Unterstützung der Entwicklung von Softwareprojekten entwickelt. Er ist eine Sammlung von Werkzeugen und Technologien, die die Koordination eines Projektes im Team ermöglichen.

(1.1) Der TFS ist jedoch nicht nur ein Werkzeug für die Verwaltung von Workflows, sondern bietet auch ein sicheres und stabiles Quellcode-Verwaltungssystem. Er integriert eine eigene Versionskontrolle für den Quellcode der verwalteten Projekte. Die Versionskontrolle unterstützt dabei Ein- und Auscheckverfahren zur Quellcodebearbeitung, das Zusammenführen von Quellcode und vieles mehr.

(1.2) Die Erstellung des Produktrückstands in der Projektplanungsphase wird komplett vom TFS unterstützt.

(1.3) In einer Datenbank auf dem TFS werden Berichte gespeichert. Somit können Status- und Trendinformationen während des Projekts eingesehen werden.

(1.4) Diese Funktionalität ermöglicht es dem Team, regelmäßige Builds für das Projekt zu erstellen und zu verwalten.

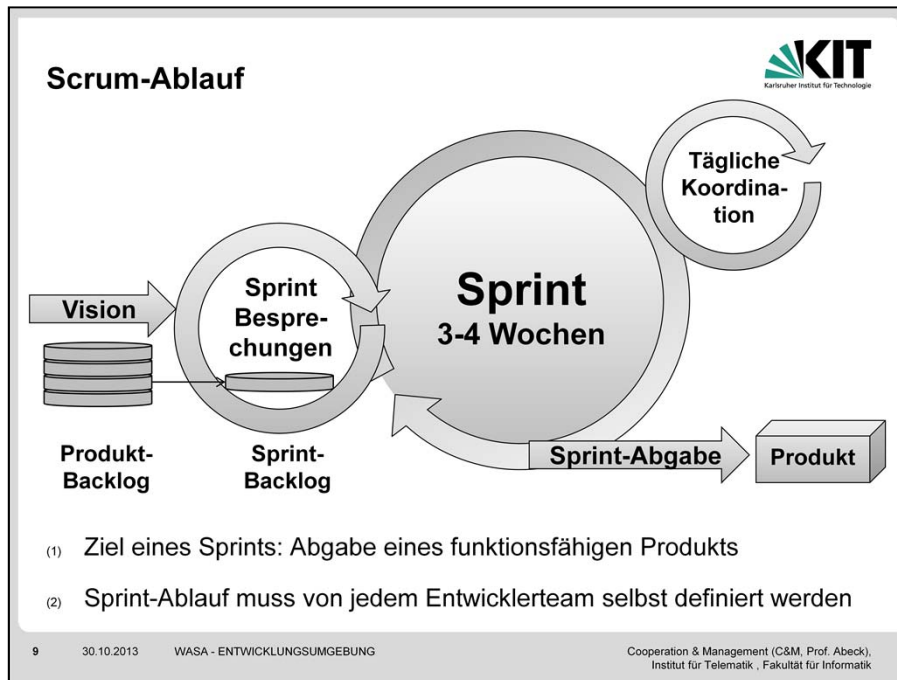
(1.5) Die standardmäßig unterstützten Projektvorlagen sind: "Microsoft Visual Studio Scrum", "MSF for Agile Software Development" und "MSF for CMMI Process Improvement". Weitere Vorlagen können heruntergeladen und angepasst werden.

(2) Eine Prozessvorlage ist eine Sammlung von Dateien, die zusammen verschiedene Prozesselemente eines Team-Projekts im TFS definieren. Bestandteile einer Prozessvorlage sind Arbeitsaufgaben (Work Items), Berichte (Reports), Teamaufgaben und diverse Dokumente.

(3.1) Für die Entwicklung der KIT-Smart-Campus-Anwendung wird entsprechend die die Scrum-Vorgehen unterstützende Prozessvorlage genutzt.

MSF
TFS

Microsoft Solutions Framework
Team Foundation Server



Scrum (deutsch: Gedränge) ist ein Softwareentwicklungsverfahren. Es basiert auf den agilen Grundsätzen und verfolgt einen empirischen, inkrementellen und iterativen Ansatz.

(Projektvorbereitung) Bevor mit der Entwicklung begonnen werden kann, wird aus der Vision des Kunden der Produktrückstand gebildet. In diesem werden die Anforderung an das Softwareprodukt zusammengefasst.

(Allgemeiner Ablauf) Ein Scrum-Projekt besteht aus einer Folge von 30-tägigen Iterationen, die Sprints genannt werden. Die 30 Tage sind ein empfohlener Richtwert; Sprints können aber auch eine Woche kürzer oder länger durchgeführt werden. Aus dem Produkt-Backlog (dt. Rückstand) wird derjenige Teil ausgesucht, der im nächsten Sprint entwickelt werden soll. Während des Sprints werden die ausgesuchten Features dann in Software umgesetzt. Bevor mit der Implementierung begonnen werden kann, müssen verschiedene Vorbereitungen getroffen werden.

(Sprint) Während eines Sprints werden die Aufgaben aus dem Sprint-Backlog vom Scrum-Team bearbeitet.


(Besprechungen) Während der Scrum-getriebenen Entwicklung wird eine Reihe von verschiedenen Besprechungen durchgeführt. Vor jeder Entwicklungsiteration steht eine Planungsphase an und während des Sprint-Ablaufs finden tägliche kurze Koordinationstreffen (Daily Scrum) statt. Am Ende jedes Sprints wird das entstandene Teilprodukt präsentiert und der letzte Sprint-Ablauf noch einmal evaluiert.

(1) Das funktionsfähige (Teil-) Produkt wird auch als Produktinkrement bezeichnet.

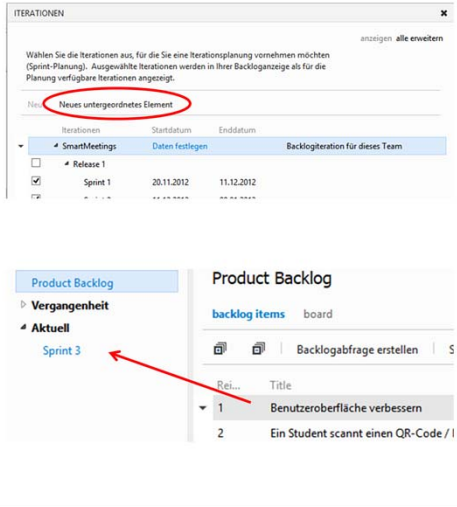
(2) Das Entwicklerteam ist selbstorganisiert und trägt eine hohe Verantwortung für den Erfolg des Projekts.

[CM-Scr] Cooperation & Management: Scrum, iCC-C&M-Workshop-Beitrag, Stuttgart, August, 2012. C&M-Teamserver > Mitglieder > 4-1.iC_Consult > 2.Beiträge_Treffen > 2012 > 12-08-09_Stuttgart

Beispiel einer TFS-Funktion: Konfiguration der Sprints



- (1) Anlegen über Startseite
 - (1) Menüpunkt "Zeitplan und Iteration konfigurieren..."
- (2) Neues Element anlegen
 - (1) Start- und Enddatum können konfiguriert werden
 - (2) Sprint muss aktiviert werden
- (3) Unter Arbeit können Backlog-Einträge Sprints zugeordnet werden
 - (1) Drag & Drop



10 30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

(1) Auf der Startseite des Projekts können Sprints konfiguriert werden.

(1.1) In der rechten unteren Ecke findet sich der Menüpunkt "Zeitplan und Iteration konfigurieren...", über den man zu dem oben gezeigten Menü gelangt.

(2) In der Liste befinden sich alle angelegten Sprints. Über den Punkt "Neues untergeordnetes Element" kann ein neuer Sprint angelegt werden. Sprints können verschachtelt werden.

(2.1) Der Sprint kann benannt werden. Als Name bietet sich allerdings "Sprint" und eine Nummer an. Ebenfalls kann der Start- und Endzeitpunkt des Sprints konfiguriert werden. Über einen Doppelklick auf einen vorhandenen Sprint lässt sich das Menü erneut zum Vorschein bringen, so dass Name und Zeitspanne geändert werden kann.

(2.2) Nachdem ein Sprint angelegt wurde, muss er aktiviert werden, indem der Haken links neben dem Element gesetzt wird. Nur aktiven Sprints können Aufgaben zugeordnet werden. Falls mehrere Sprints aktiviert sind, wird der erste unter den aktivierten Sprints als der aktuelle betrachtet.

(3) In der Perspektive "Arbeit", genauer im Backlog, kann ein Backlog-Eintrag einem Sprint zugeordnet werden.

(3.1) Dazu zieht man das entsprechende Element einfach auf den Sprint in der linken Seitenleiste. Nur Sprints, die zuvor aktiviert wurden, erscheinen in der Liste.

[CM-CMS] Cooperation & Management: C&M-SOFTWARE-ENTWICKLUNG, C&M-Teamarbeitsdokument. C&M-Teamserver > Mitglieder > 2-2.Lösungsleitfäden

Scrum-Rollen

- (1) Scrum-Team ist selbstorganisierend
 - (1) Gutes Zusammenspiel durch Kommunikation und Vertrauen
 - (2) Ziele: Planung, Durchführung und Überprüfung der Sprints
- (2) ScrumMaster koordiniert den Scrum-Prozess
 - (1) Stellt ein funktionierendes Scrum-Team zusammen
 - (2) Erkennt Probleme und löst diese schnellstmöglich
 - (3) Ziel: Produktivität des Teams und des Produktbesitzers sicherzustellen
- (3) Produktbesitzer bildet die Schnittstelle zwischen Kunde und Team
 - (1) Erstellen des Produktrückstand und eines Zeitplans
 - (2) Fundierte Sachkenntnisse im Entwicklungsbereich
 - (3) Ziel: dem Team die Anforderung nahezubringen

(1) Die Aufgabe des Scrum-Teams besteht darin, eigenverantwortlich die einzelnen Sprints zu planen, die Programmierung durchzuführen und den aktuellen Stand zu testen.

(1.1) Gute Scrum-Teams zeichnen sich dabei durch gute Zusammenarbeit, häufige Kommunikation, Vertrauenswürdigkeit und Offenheit gegenüber den Mitgliedern aus.

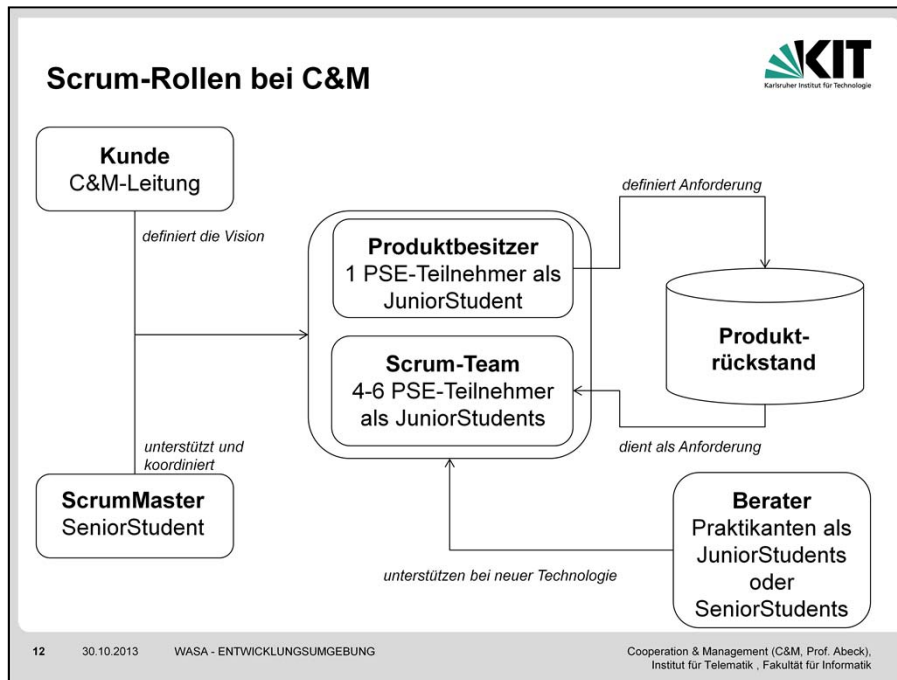
(2) Der ScrumMaster ist weder als Teammitglied noch als Teamleiter zu sehen. Er befindet sich auf Augenhöhe mit dem Scrum-Team, nimmt aber in seinem Aufgabenbereich eine komplett andere Stellung ein. Er zeichnet sich durch ausgezeichnete Kommunikations-, Verhandlungs- und Konfliktlösungsfähigkeiten aus. Diese setzt er ein, um ein Scrum-Team funktionsfähig zu halten.

(2.2) Auch Probleme innerhalb des Scrum-Teams sieht er rechtzeitig und versucht diese gleich zu lösen. Darüber hinaus ist er bemüht, den alltäglichen Scrum-Ablauf am Laufen zu halten. Probleme, die das Scrum-Team oder das Projekt von außen bedrohen, beispielsweise das Fehlen wichtiger Entwicklungsumgebungen oder nicht durchsetzbare Wünsche des Kunden, fallen in seinen Aufgabenbereich.

(3) Der Produktbesitzer (engl. Product Owner) erstellt aus der Vision des Kunden die Anforderungen an das Scrum-Team. Indem er bei Änderungswünschen des Kunden regelmäßig den Produktrückstand aktualisiert, kann das Produkt dessen Vorstellung angenähert werden.

(3.1) Dabei muss er den Produktrückstand erstellen und die einzelnen Anforderungen in entsprechender Reihenfolge priorisieren, sowie einen Zeitplan für das Scrum-Team ausarbeiten.

(3.2) Darüber hinaus muss er ein sehr tiefgreifendes Verständnis für den Fachbereich des Produktes mitbringen, um als Ansprechpartner alle Fragen des Scrum-Teams beantworten zu können.



(Kunde) Die Rolle des Kunden wird im C&M-Scrum-Ablauf von der C&M-Leitung übernommen. Deren Aufgabe besteht darin, den PSE-Teilnehmern die Vision des zu erstellenden Produkts näherzubringen. Darüber hinaus muss die C&M-Leitung als Lehrverantwortliche die Endnote der PSE-Teilnehmer, auf Basis der Dokumentation und der entstandenen Software, bestimmen.

(ScrumMaster) Als ScrumMaster fungiert im C&M-Umfeld ein SeniorStudent, welcher bereits Erfahrung in der C&M-Arbeitsweise hat. Zu seinen Hauptaufgaben zählt neben der Koordination des Teams auch die Gewährleistung, dass die Scrum-Regeln innerhalb des Teams eingehalten werden. Darüber hinaus ist er für die Installation der Koordinations- und Entwicklerwerkzeuge sowie deren korrekte Ausführung zuständig. Im Fall des KCM-Projekts übernimmt ein Student, der seine Bachelorarbeit in der Forschungsgruppe schreibt, die Rolle des ScrumMasters.

(Produktbesitzer) Ein PSE-Teilnehmer wird als Produktbesitzer ausgewählt. Zu seinen Hauptaufgaben gehört es, aus der Vision des Kunden die Scrum-Dokumentation durchzuführen und regelmäßig zu aktualisieren. Da die C&M-Leitung bei den wöchentlichen Meetings als Kunde teilnimmt, muss der Produktbesitzer nicht unbedingt, wie im literarischen Scrum beschrieben, als Schnittstelle zwischen Kunde und Scrum-Team fungieren.

(Scrum-Team) Das Team besteht aus den Teilnehmern des PSE-Praktikums. Sie werden als JuniorStudents in die Forschungsgruppe integriert und dem oben genannten SeniorStudent zugewiesen. Das Scrum-Team soll laut den Scrum-Richtlinien möglichst selbstorganisierend sein, d.h., die PSE-Teilnehmer sind für die Koordination des gemeinsamen Programmiervorgangs oder für die Festlegung der Aufgabenverteilung selbst verantwortlich. Der ScrumMaster soll aus pädagogischen Gründen an dieser Stelle explizit außen vor gelassen werden.

(Berater) JuniorStudents aus anderen Praktika können dem Projekt als Berater zugewiesen werden. Beispielsweise können sie den PSE-Teilnehmern mittels Workshops die Technologien nahebringen.

LZ C&M-TFS UND SCRUM – ÜA TFS-Funktionen, Sprint, ScrumMaster



- (1) Welche Art von System ist der Team Foundation Server (TFS)
- (2) Welche Funktionen stellt der TFS bereit?
- (3) Was ist sind Eingabe und Ausgabe eines Sprints?
- (4) Was ist die Aufgabe des ScrumMasters?
- (5) Wer übernimmt die Rolle des ScrumMasters bei C&M?

ECLIPSE UND JAVA – Überblick über Eclipse



- (1) Open-Source-Projekt, das im Jahr 2001 von IBM und sieben anderen Firmen gestartet wurde
- (2) Plattform für Anwendungsentwicklungswerkzeuge
 - (1) Zunahme der Programmierartefakte und dafür erforderliche Editoren
 - (2) Erfüllung der Anforderungen der Open-Source-Bewegung
- (3) Plattform für beliebige Desktop-zentrierte Anwendungen
 - (1) Gleiche Grundfunktionen mit einer flexibleren Benutzeroberfläche
- (4) Unterstützung von Projekten, die Werkzeuge und Anwendungen auf der Eclipse-Plattform entwickeln
- (5) Starke Unterstützung im kommerziellen, Open-Source- und akademischen Bereich

14

30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[DF+04:1]

(1) Hierzu wurden von IBM die bis dahin geleisteten Entwicklungsarbeiten als Schenkung (im Wert von 40 Mio. Dollar) an die Softwaregemeinschaft übergeben.

Aus der Eclipse-Organisation ist die Nonprofit-Organisation "Eclipse Foundation" geworden, die heute aus sehr viel mehr Mitgliedern als den 8 Gründerfirmen besteht [:1].

(2) Eclipse war ursprünglich nur als Integrationsplattform für Softwareentwicklungswerkzeuge gedacht.

(2.1) Engl. programming artifacts; Beispiele sind Java, Perl, HTML, digitales Audio und Video, Web-Services, Enterprise Java Beans (EJB) [:2].

Integration der Werkzeuge liefert einen besseren Austausch und eine verbesserte Lernkurve [:3].

(2.2) Die Programmierer benötigten eine herstellerneutrale, offene und flexible Umgebung für die Integration beliebiger Werkzeuge [:3].

(3) Eines der zentralen Themen von Eclipse 3.0 war die Weiterentwicklung von einer Integrated Development Environment (IDE) zu einer Rich Client Platform (RCP).

(3.1) Umfasst u.a. eine flexiblere Laufzeitumgebung, durch die Funktionalität dynamisch hinzugefügt und beseitigt werden kann.

(4) Open-Source-Projekte, die unter der "Eclipse.org"-Gemeinschaft (<http://www.eclipse.org>) laufen.

(5) Universitäten setzen Eclipse bei der Java-Ausbildung und in der Forschung ein

EJB Enterprise Java Beans

RCP Rich Client Platform

[DF+04] Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy: The Java Developer's Guide to Eclipse. Addison-Wesley, 2004.

Organisatorische Struktur und Projekte



- (1) Das Eclipse-Konsortium (Eclipse.org) hat sich von einer lose gekoppelten Gruppe von Unternehmen zu einer formal strukturierten Organisation entwickelt
- (2) Die Arbeiten werden in Projekten und Unterprojekten organisiert
 - (1) Eclipse selbst wird als Eclipse-Projekt bezeichnet
 - (2) Eclipse Tools Project
 - (1) Unterprojekte: C/C++, GEF, EMF, UML2
 - (3) Eclipse Web Tools Platform (WTP)
 - (4) Eclipse Technology Project
- (3) Lizenziert unter der Eclipse Public License (EPL)
 - (1) Ermöglicht einen effektiven kommerziellen Nutzen der lizenzierten Software

15

30.10.2013 WASA - ENTWICKLUNGsumGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[DF+04:7]

(1) Diese Organisation arbeitet ohne Profit und ist als die Eclipse Foundation bekannt.
Die formale Struktur besteht aus einem Board of Directors [7].

(2.1) JDT (Java Development Toolkit) ist ein Unterprojekt des Eclipse-Projekts.

(2.2) Fördert Best-of-Breed-Werkzeuge

- IDE für C/C++

- Graphical Editing Framework (GEF) zur Entwicklung graphischer Editoren.

- Eclipse Modeling Framework (EMF) zur Beherrschung (engl. hosting) von Anwendungsmodellen jeder Sorte.

- UML2 als ein Rahmenwerk für Modellierungswerkzeuge basierend auf der Unified Modeling Language (UML).

(2.3) Generische, erweiterbare und Standard-basierte Werkzeug-Plattform zur Entwicklung von J2EE- und Web-zentrierten Anwendungen.

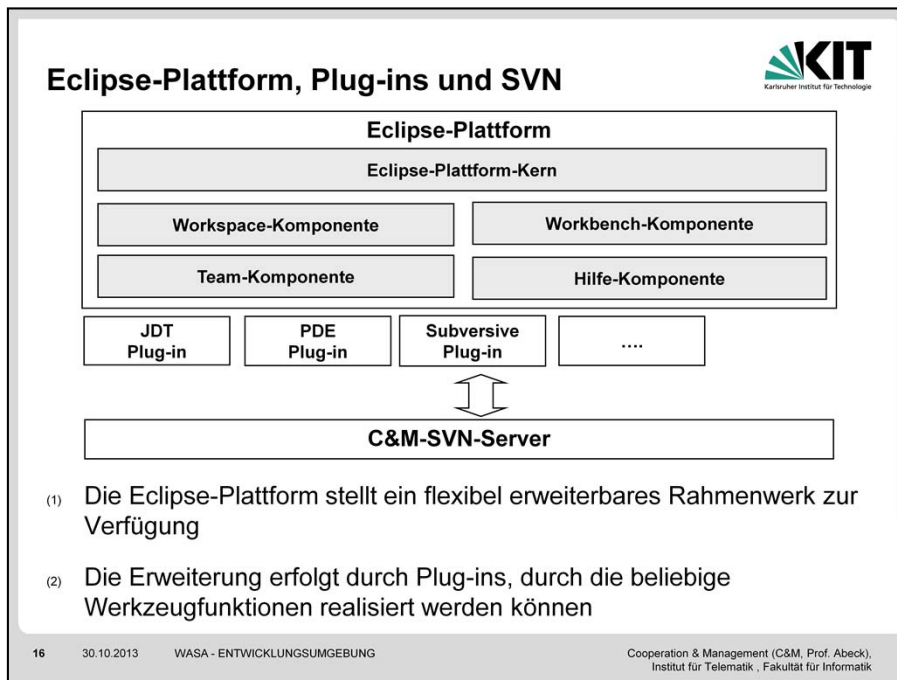
(2.4) Unterstützung von Forschungsarbeiten, Technologieuntersuchungen und Ausbildung (engl. education).

(3) EPL wird von der Open Source Initiative (OSI) anerkannt und stellt eine leicht abgeänderte Version der Common Public License (CPL) dar.

(3.1) Nicht jedes auf eine EPL-basierenden Software muss unter die EPL gestellt werden. Allerdings muss jedes unter der EPL stehende Modul, das verändert wurde wiederum unter die EPL gestellt werden. Hierdurch wird eine größtmögliche Flexibilität und Kontrolle über die Nutzung der Open-Source-Software ermöglicht.

CPL	Common Public License
EPL	Eclipse Public License
EMF	Eclipse Modeling Framework
GEF	Graphical Editing Framework
JDT	Java Development Toolkit
OSI	Open Source Initiative
UML	Unified Modeling Language
WTP	Web Tools Platform

[DF+04] Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy: The Java Developer's Guide to Eclipse. Addison-Wesley, 2004.



Eclipse selbst ist in Java geschrieben und benötigt daher zur Ausführung ein "Java Runtime Environment" (JRE). Das Werkzeug kann durch das in der Grundversion mit ausgelieferte Plug-in "Java Development Toolkit" (JDT), das zur Ausführung das "Java Development Kit" (JDK) benötigt, als Java-Entwicklungsplattform genutzt werden. Durch ein weiteres grundsätzlich in einer Eclipse-Installation vorhandenes Plug-in, das "Plug-in Development Environment" (PDE), lassen sich eigene Erweiterungen für Eclipse entwickeln. Die Abbildung zeigt die Architektur von Eclipse [Ho04:7], deren Bestandteile im Folgenden näher beschrieben werden.

(Eclipse-Plattform-Kern) Wird beim Start von Eclipse aufgerufen und übernimmt die zentrale Aufgabe des Ladens und Ausführens von Plug-ins. Beim Start werden alle Plug-ins geprüft, aber aus Performanzgründen (CPU, Speicher) erst dann geladen, wenn sie tatsächlich benötigt werden.

(1) Die vier Plattform-Komponenten sind ebenfalls als Plug-ins realisiert.

(Workbench-Komponente) Die grundlegende Grafikschnittstelle bestehend aus verschiedenen Typen von Toolbars und Menüs. Eclipse nutzt hierzu eine eigene Grafikbibliothek, das "Standard Widget Toolkit (SWT)", welches Betriebssystem-spezifische Funktionalität kapselt, und das auf SWT aufsetzende JFace. Um Eclipse auf verschiedenen Betriebssystemen (z.B. Windows, Linux/Motif, Linux/GTK2, HP-UX, Mac OS X) ausführen zu können, existieren entsprechende Portierungen von SWT auf die jeweilige Plattform.

(Workspace-Komponente) Ist verantwortlich für das Managen aller Ressourcen, die innerhalb von Eclipse in Projekten organisiert sind. Ein Projekt erhält jeweils einen Ordner (Folder) mit Unterordnern im Workspace, so dass man die Ablagestruktur einfach überblicken kann.

(Team-Komponente) Plug-in, das die Versionskontrolle in der Form eines CVS-Client (Concurrent Version System) unterstützt.

(Hilfe-Komponente) Erweiterbares Dokumentationssystem, das HTML-Dokumente mit XML-formatierter Navigationsinformation aufzunehmen gestattet.

(2) Auch wenn JDT und PDE zwar immer mit Eclipse ausgeliefert werden, stellen diese keine Grundfunktionalität dar.

CVS	Concurrent Version System
JDK	Java Development Kit
JDT	Java Development Toolkit
JRE	Java Runtime Environment
PDE	Plug-in Development Environment

SWT Standard Widget Toolkit

[Ho04] Steve Holzer: Eclipse, O'Reilly, 2004.

Arbeiten mit Eclipse



- (1) Die Oberfläche setzt sich aus drei Arten von Elementen zusammen
 - (1) Editoren
 - (2) Sichten
 - (3) Perspektiven
- (2) Die Auswahl eines Befehls erfolgt alternativ über die Menü- oder Werkzeugleiste
- (3) Zur Verwaltung der während des Softwareentwicklungsprozesses im Team entwickelten Ressourcen bietet sich die Verwendung eines Versionskontroll-Werkzeugs an
 - (1) Beispiel: Subversion (SVN)

17

30.10.2013 WASA - ENTWICKLUNGsumGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[DF+04:18]

(1) Eine Eclipse-Oberfläche setzt sich aus Editoren, Sichten (Views) und Perspektiven (Perspective) zusammen [18].

(1.1) Ein Editor dient zum Erzeugen und Ändern von Ressourcen. Je nach Ressourcentyp werden unterschiedliche Editoren eingesetzt. Ein Beispiel ist ein BPMN-Editor zur Beschreibung eines Geschäftsprozesses [19].

(1.2) Eine Sicht stellt einen Weg zur Navigation durch Ressourcen oder anderen Informationen in Eclipse dar. Sichten können u.a. in ihrer Größe geändert werden oder gestapelt werden.

(1.3) Eine Perspektive definiert eine Menge von Editoren und Sichten, die für eine bestimmte Aufgabe benötigt werden.

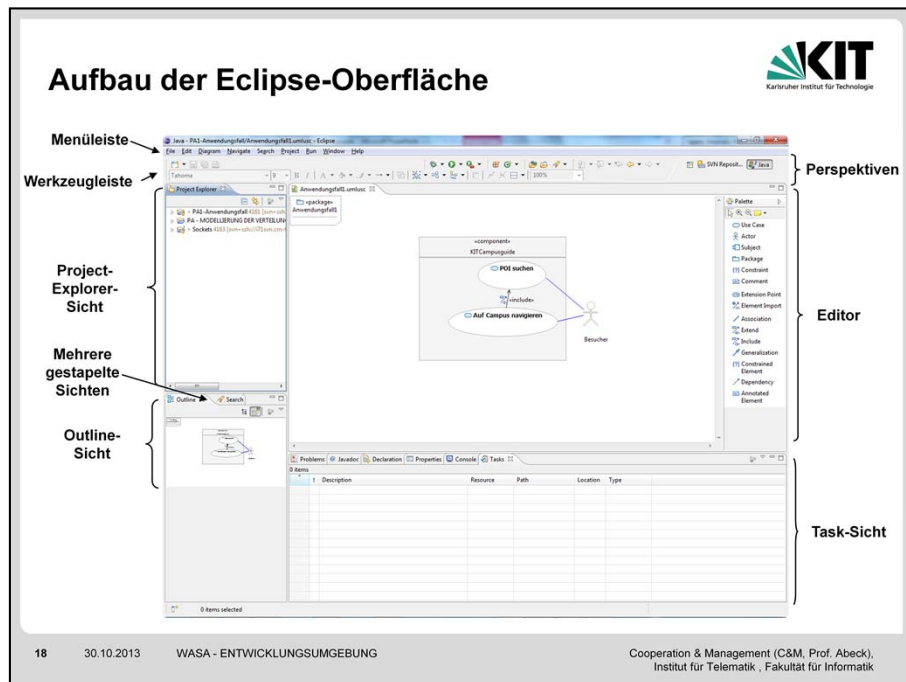
(2) Alle Benutzerschnittstellenaktionen sind über die Menüleiste ausführbar. Eine Menge von Icons, über die Aktionen ausgeführt werden können (diese Aktionen können alternativ auch über die Menüleiste in Anspruch genommen werden).

(3) In der WASA-Veranstaltung wird keine Software im Team entwickelt, weshalb auf den Einsatz eines Versionskontrollwerkzeugs aus Komplexitätsgründen bewusst verzichtet wird.

(3.1) Zu Subversion besteht ein entsprechendes Eclipse-Plug-in, das auch in der Forschungsgruppe C&M genutzt wird [CM-O-CTA].

[CM-O-CTA] Cooperation & Management: Organisationsdokument C&M-TEAMARBEIT, C&M-Teamserver > Mitglieder > 0-1.Teamarbeit.

[DF+04] Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy: The Java Developer's Guide to Eclipse. Addison-Wesley, 2004.



[DF+04:21]

(Menüleiste) Alle Benutzerschnittstellenaktionen sind über die Menüleiste ausführbar.

(Werkzeuggeste) Eine Menge von Icons, über die Aktionen ausgeführt werden können (diese Aktionen können alternativ auch über die Menüleiste in Anspruch genommen werden).

(Perspektive) Definiert eine Menge von Editoren und Sichten, die für eine bestimmte Aufgabe benötigt werden.

(Editor) Dient zum Erzeugen und Ändern von Ressourcen. Je nach Ressourcentyp werden unterschiedliche Editoren eingesetzt. Im obigen Beispiel wird ein BPMN-Editor zur Beschreibung eines Geschäftsprozesses verwendet.

(Sicht) Stellt einen Weg zur Navigation durch Ressourcen oder anderen Informationen in Eclipse dar. Neben der reinen Ansicht bestehen auch Möglichkeiten zur Änderung einer Ressource (z.B. Namensänderung).

(Mehrere gestapelte Sichten) Sichten können u.a. in ihrer Größe geändert werden oder gestapelt werden.

[DF+04] Jim D'Anjou, Scott Fairbrother, Dan Kehn, John Kellerman, Pat McCarthy: The Java Developer's Guide to Eclipse. Addison-Wesley, 2004.

- (1) Richtig oder falsch?
 - (1) Die gesamte Eclipse-Plattform besteht ausschließlich aus Plug-ins
 - (2) Die Eclipse-Plattform umfasst insgesamt 4 Plug-ins
 - (3) Die Eclipse-Oberfläche bietet Perspektiven an, die eine Menge von Editoren und Sichten umfassen
- (2) In welche Projekte sind die Arbeiten an der Eclipse-Software eingeteilt?
- (3) Wie sind die Lizenzierungsvereinbarungen zur Eclipse-Software geregelt?

Überblick über Java



- (1) James Gosling von Sun Microsystems ist der Erfinder von Java
 - (1) Erste Arbeiten ab dem Jahr 1991
- (2) Wird heute zur Entwicklung
 - (1) sowohl von verteilten Web-Anwendungen
 - (2) als auch von Einzelanwendungen
 - (3) auf Servern, Arbeitsplatzrechnern und mobilen Geräten genutzt
- (3) Mittels Applets lassen sich Java-Programme im Web-Browser ausführen
- (4) Mittels Java Servlets und Java Server Pages lassen sich serverseitige Anwendungen entwickeln

20

30.10.2013

WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[Li05:8]

Java gehört neben

- COBOL (COmmon Business Oriented Language, zur Verarbeitung von Geschäftsdaten),
- FORTRAN (FORmula TRANSLator, für mathematische Berechnungen),
- C (Nachfolger von B, maschinennahe Programmierung) u.a.

zu den wichtigsten und verbreitetsten höheren Programmiersprachen.

(1) Die Sprache hieß ursprünglich Oak und diente der Entwicklung eingebetteter Verbraucherelektronik-Anwendungen (engl. appliances).

(1.1) Den Namen Java erhielt die Sprache im Jahr 1995.

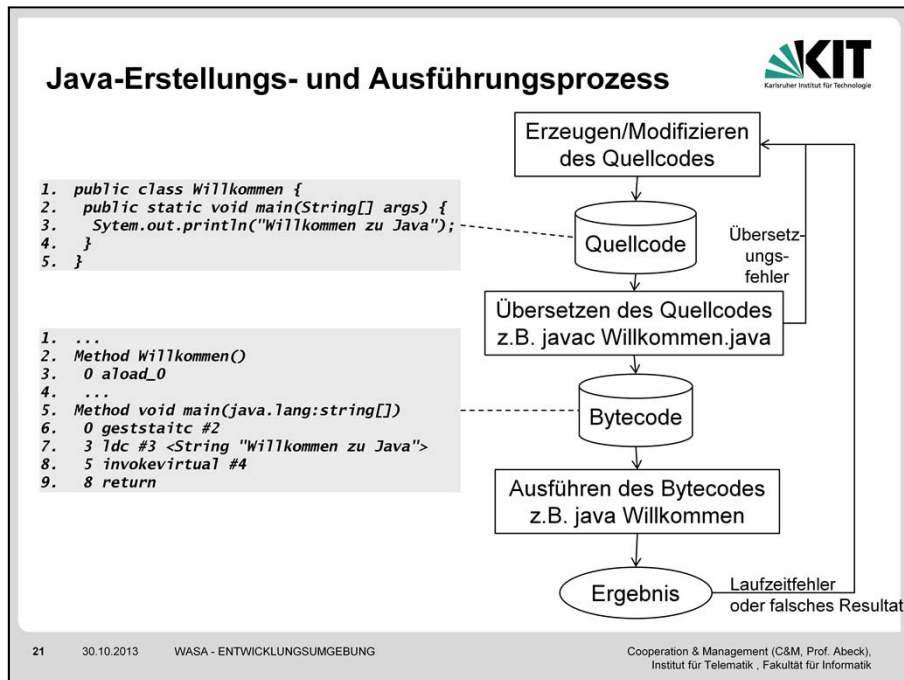
(2) Java wurde z.B. zur Steuerung des auf dem Mars eingesetzten Roboter-Rover genutzt.

(3) Aus dem Applet-Konzept, das moderne graphische Benutzeroberflächen verwendet, resultierte eine hohe initiale Attraktivität von Java [:15]. Applets werden im Browser und damit clientseitig ausgeführt

(4) Hierdurch können Webseiten dynamisch erzeugt werden.

COBOL	Common Business Oriented Language
FORTRAN	Formula Translator

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.



[Li05:21]

(Erzeugen/Modifizieren) Hierzu kann ein beliebiger Texteditor (z.B. Windows NotePad) oder eine IDE (z.B. Eclipse, NetBeans) verwendet werden.

(Quellcode) Wird vom Programmierer erstellt.

(Bytecode) Wird vom Übersetzer erzeugt und durch die Java Virtual Machine (JVM) interpretiert.

(Übersetzungsfehler, Laufzeitfehler) Im Falle des Auftretens von Fehlern ist der Prozess iterativ, d.h. es wird an den Anfang des Prozesses gesprungen, um das Programm zu modifizieren.

(Willkommen.java) Die Endung ".java" ist vorgeschrieben und der Name der Datei muss mit dem öffentlichen Klassennamen (hier: "Willkommen") übereinstimmen.

Zu beachten: Java-Quellprogramme unterscheiden Groß-/Kleinschreibung.

IDE Integrated Development Environment

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

Java-Sprachspezifikation, APIs und Entwicklungswerkzeuge



- (1) Die Sprachspezifikation und die API definieren den Java-Standard
 - (1) Syntax und Semantik der Sprache werden durch die Sprachspezifikation in Form einer technischen Definition festgelegt
 - (2) Die Java API enthält vordefinierte Klassen und Interfaces
- (2) Es werden verschiedene Editionen der Java API unterschieden
 - (1) Standard Edition, Enterprise Edition, Micro Edition
- (3) Zu jeder Standard-Edition wird ein Java Development Toolkit ausgeliefert
 - (1) Beispiel: Das Toolkit zu J2SE 5.0 ist JDK 5.0
- (4) Integrierte Entwicklungsumgebungen (Integrated Development Environment, IDE) unterstützen eine effiziente Entwicklung von Java-Programmen
 - (1) Neben Eclipse existieren noch zahlreiche weitere IDEs

22

30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[Li05:19]

(1) Die Weiterentwicklung befindet sich in der Hand von SUN, wodurch ein einheitlicher Standard von Java sichergestellt wird [Li05:19].

(1.1) Das Dokument befindet sich unter "java.sun.com/docs/books/jls".

(1.2) Die aktuelle Version und Updates können von der Web-Seite "java.sun.com" heruntergeladen werden.

Hinweise hierzu finden sich in der zu einer JDK-Download mitgelieferten "readme-Datei" (...\\Java\\jdk1.5.0\\README.html).

(2) Jede Edition unterstützt die Entwicklung einer bestimmten Typs von Java-Anwendung.

(2.1) Standard Edition (J2SE): Client-seitige lokale Anwendungen oder Applets; Enterprise Edition (J2EE, JEE): Server-seitige verteilte Anwendungen; Micro Edition (J2ME): Mobile Geräte

(3) Das Java Development Toolkit (JDK) besteht aus einzelnen Programmen zur Entwicklung und zum Testen von Java-Programmen, die jeweils von der Kommandozeile aufgerufen werden.

(4.1) Beispiele sind: NetBeans (SUN) oder JBuilder (Borland)

J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

- (1) Die von der Java-Plattform angebotene Bibliothek (eng. library) ist in Pakete organisiert
 - (1) Eindeutigkeit des Paketnamens lässt sich durch die Verwendung des Internet-Domänen-Namens erreichen
 - (2) Eine Klasse wird durch das Schlüsselwort "package" einem Paket zugeordnet
 - (3) Verwendung von Klassen aus einem Paket mittels
 - (1) Angabe des voll qualifizierenden Namen oder
 - (2) Importieren einzelner Klassen oder aller Klassen eines Pakets
- (2) Wichtige Pakete
 - (1) java.lang (language), java.util (utility), java.io (input/output), java.net (network), java.awt und java.awt.event, javax.swing und javax.swing.event

[Li05:159]

Durch eine gezielte Verwendung von Bibliotheksklassen können Java-Programme einfacher, effizienter und robuster gestaltet werden. Daher ist es wichtig, sich als Java-Programmierer mit diesen Paketen vertraut zu machen [BK09:585].

(1) Pakete gruppieren Klassen und dienen (i) zur einfachen Auffindung (ii) zur Verhinderung von Namenskonflikten (iii) zur angemessenen Verteilung von Software und (iv) zum Schutz vor dem Zugriff von außerhalb des Pakets. Wird als Sichtbarkeit (engl. visibility modifier) weder "private" (Zugriff nur in der Klasse) noch "public" (unbeschränkter Zugriff), sondern stattdessen "protected" oder keine Angabe in Java gemacht, so entspricht das dem Zugriff innerhalb des Pakets [Li05:224]. In UML wird dieser Zugriffsmodifikator durch "#" ausgedrückt.

(1.1) Paketnamen werden grundsätzlich klein geschrieben. Der Internet-Domänen-Name wird üblicherweise in umgekehrter Reihenfolge angegeben (z.B. "edu.wasa.mypackage"). Es besteht eine 1:1-Abbildung des hierarchisch aufgebauten Namens auf das Dateiverzeichnis. Bezogen auf das Beispiel liegen die Klassen-Dateien im Verzeichnis "\edu\wasa\mypackage"

(1.2) Fehlt die Angabe, wird die Klasse dem aktuellen Verzeichnis zugeordnet, was dem "Default Package" entspricht. Der "package"-Befehl muss vor jedem Kommentar oder Nicht-Leerzeichen-Kommando erfolgen (Beispiel: "package edu.wasa.mypackage;").

(1.4.1) Voll qualifizierter Name: Falls eine Klasse "Test" im Beispiel-Paket besteht, der voll qualifizierte Name lautet "edu.wasa.mypackage.Test". Dieses Vorgehen ist nur für den Fall zu akzeptieren, dass die Klasse selten im Programm auftritt.

(1.4.2) Importieren: Hierzu stellt Java das Schlüsselwort "import" bereit. Durch "*" werden alle Klassen eines Pakets bei Bedarf (on-demand declaration) importiert.

Beispiel: "import edu.wasa.mypackage.*";

"import edu.wasa.mypackage.Test"; // Only the class Test is imported

(2.1)

java.lang: Für die Sprache Java grundlegende Klassen und Interfaces, weshalb dieses Paket automatisch in jeder Klassendefinition importiert ist.

Wichtige Klassen sind u.a. (i) "Class Object": Wurzelklasse, von der alle Klassen erben; beinhaltet als Methoden u.a. "equals", "toString", "clone", "hashCode" oder (ii) "Class String": unveränderliche Objekte mit den zentralen Methode "charAt", "equals", "indexOf", "length", "split", "substring", (iii) "Class StringBuilder" (effiziente Alternative zu "String" mit den die Zeichenkette ändernden Methoden "append", "insert" und der Konversionsmethode "toString").

java.util: Relativ zusammenhanglose Sammlung von nützlichen (utility: Nützlichkeit) Klassen und Interfaces, wie z.B. "Interface Collection" (mit den Methoden "add" und "iterator"), "Interface Iterator" (mit den Methoden "next", "hasNext"), Klasse "Collections" (mit den statischen Methoden "binarySearch", "fill" und "sort"), "Class Scanner" (bietet Unterstützung beim Einlesen und Zerlegen von Eingaben).

java.io: Unterstützung der Ein- und Ausgabe durch zwei Arten von Klassen und Interfaces: (i) Verarbeitung von binären Daten bzw. Datenströmen (Streams) und (ii) "Reader"- und "Writer"-Klassen, die auf menschenlesbaren Zeichen arbeiten. Ein Beispiel für (i) ist "Interface Serializable" (leeres Interface, durch das Objekte als Ganzes aus Eingabequellen gelesen und in Ausgabekanäle geschrieben werden können), "Class ; Beispiele für (ii) sind "Class BufferedReader" und "Class BufferedWriter".

java.net: Unterstützung der Netzwerkprogrammierung. Ein Beispiel einer Klasse ist "Class URL", die einen Uniform Resource Locator repräsentiert. URLs beschreiben Ressourcen im Internet und werden auch zur Beschreibung bzw. Addressierung von Ressourcen in einem lokalen Dateisystem (-> java.io) und auf einer GUI (-> javax.swing) genutzt.

java.awt, java.awt.event: Klassen zur Erstellung graphischer Oberflächen.

javax.swing, javax.swing.event: Für die Swing-Klassen wurde die Bezeichnung javax ("x" für extended?) gewählt, wofür es keine logische Erklärung gibt [BK09:404].

[BK09] David J. Barnes, Michael Kölling: Java lernen mit BlueJ, Pearson Studium, 2009.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

Programmierstil und Dokumentation



- (1) Angemessene Kommentare
 - (1) Am Anfang des Programms und vor jedem größeren Schritt
 - (2) javadoc-Kommentare (`/** ... */`) vor einer Klasse und einem Methodenkopf
- (2) Namenskonventionen
 - (1) Variablen und Methoden: erster Buchstabe kleingeschrieben; erster Buchstabe jedes neuen Wortes großgeschrieben
 - (2) Klassen: erster Buchstabe großgeschrieben
 - (3) Konstanten: Alle Buchstaben großgeschrieben und Worte durch einen Unterstrich getrennt
- (3) Einrückung und Leerräume
- (4) Blockstile
 - (1) Next-line-Stil und End-of-line-Stil

24 30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

[Li05:66]

Der Programmierstil betrifft das Aussehen des Programms. Ein in einer Zeile geschriebenes Programm ist ein (extremes) Beispiel für schlechten Programmierstil.

Dokumentation ist die Sammlung von erklärenden Bemerkungen und Kommentaren zu den Programmbefehlen.

(1.1) Hierzu gehören die Schlüsselfeatures, unterstützende Datenstrukturen und jede einzigartige Technik.

Kommentare sollten möglichst kurz und bündig (engl. concise) sein, um das Programm nicht zu überfrachten und damit schwer lesbar zu machen.

(1.2) Diese Kommentare können automatisch in eine javadoc-HTML-Datei zusammengeführt werden.

(2) Die Namen sollten beschreibend sein und geradlinige (engl. straightforward) Bedeutungen aufweisen.

(2.1) Beispiele: `campusFlaeche`, `poiIdentifier`, `printPoi()`

(2.2) Beispiele: `Poi`, `Person`

(2.3) Beispiele: `CAMPUS_FLAECH`

(3.1) Einrückungen (durch zwei Leerzeichen) sollen die strukturellen Beziehungen zwischen den Komponenten oder Anweisungen verdeutlichen [Li05:67].

(3.2) Jeweils ein Leerzeichen vor und nach einem binären Operator; Beispiel: `boolean b = 3 + 4 * 4 > 5 * (4 + 3) - ++i`

(4.1) Next-line-Stil: vertikale Ausrichtung von Klammern (d.h., neue Zeile vor den Klammern, so dass diese jeweils am Anfang der eingerückten Zeile stehen)

End-of-line-Stil: Öffnende Klammer ist das letzte Zeichen in der Zeile

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

Richtig oder falsch?

- (1) Java-Programme werden grundsätzlich serverseitig ausgeführt
- (2) Der Bytecode ist unabhängig von einer bestimmten Rechnerarchitektur
- (3) Der Java-Standard wird durch die Sprachspezifikation definiert
- (4) Die Variable mit dem Namen "KundenNummer" entspricht der Namenskonvention

JAVASCRIPT – Historie



- (1) Praktisch jede Website nutzt JavaScript im Browser
- (2) JavaScript geht aus der für den Netscape-Browser entwickelten Sprache LiveScript hervor
 - (1) Viele Schwächen hinsichtlich des Sprachdesigns
- (3) Die Sprache Self ist ein Vorbild von JavaScript
 - (1) Geht aus einer Vereinfachung von Smalltalk hervor
 - (2) Prototypische Vererbung statt klassenbasierter Vererbung
 - (3) Fehlendes Klassenkonzept
- (4) Die Sprache Scheme ist ein weiteres Vorbild
 - (1) Paradigmen der funktionalen Programmierung, die durch weitere Paradigmen ergänzt werden
 - (2) Programmierbare Programmiersprache

26

30.10.2013 WASA - ENTWICKLUNGSUMGEBUNG

Cooperation & Management (C&M, Prof. Abeck),
Institut für Telematik, Fakultät für Informatik

Durch die Betrachtung der Historie wird u.a. ermöglicht, JavaScript besser von anderen objektorientierten Sprachen unterscheiden zu können [Oc12:1].

(1) Das macht JavaScript zu einer der am häufigsten verwendeten Programmiersprachen der Welt und zur Lingua franca des WWW.

(2) LiveScript hieß ursprünglich Mocha und wurde 1995 von Brendan Eich entwickelt.

Es wurde der Name JavaScript gewählt, um von dem Hype der gerade von Sun entwickelten Sprache zu profitieren.

(2.1) Gleichzeitig ist JavaScript die am meisten missverstandene Programmiersprache.

(3) Self wurde 1980 von Xerox und später von SUN entwickelt.

(3.1) Die Vereinfachung hat zum Ziel, dem Entwickler mehr Freiheiten im objektorientierten Programmieren zu geben.

(3.2) Die prototypische Vererbung erfolgt durch Kopieren eines anderen Objekts, wobei das Quellobjekt als Prototyp bezeichnet wird.

(3.3) Es wird nicht zwischen dem Verhalten eines Objekts (= Methoden der Klasse) und dem Zustand des Objekts (= Variablen der Klasse) unterschieden.

(4) Scheme ist ein Lisp-Dialekt.

(4.1) Ein im Widerspruch zum funktionalen Paradigma stehendes ergänzendes Paradigma ist die imperative Programmierung.

(4.2) Durch Hinzufügen von Features können sogar Paradigmen der objektorientierten Programmierung unterstützt werden.

[Oc12] Oliver Ochs: JavaScript für Enterprise-Entwickler, dpunkt.verlag, 2012.

Skriptsprachen

- (1) Skriptsprachen nehmen bestehende Funktionen eines Systems über eine Schnittstelle in Anspruch
- (2) JavaScript nutzt die vom Browser über eine Schnittstelle angebotenen Funktionen, um eine HTML-Seite im Browser zu verändern
 - (1) Die Sprache ist aber selbst nicht abhängig vom Browser
- (3) Kernkonzepte von JavaScript werden als Core JavaScript bezeichnet
- (4) JavaScript unterstützt viele Programmierparadigmen

Es werden Kernfeatures von JavaScript vorgestellt [Oc12:17].

(1) Eine über eine programmierbare Schnittstelle angebotene Funktion heißt skriptbar oder skriptfähig.

Als Schnittstellen lassen sich Benutzer- und Befehlsschnittstellen unterscheiden.

(2) Beispiele solcher vom Browser bereitgestellte Funktionen betreffen den Dokumentenbaum oder das Netzwerkprotokoll.

(2.1) Daher kann JavaScript auch außerhalb des Browsers verwendet werden.

(3) Core JavaScript wird von allen Laufzeitumgebungen implementiert (auch z.B. von ActionScript von Adobe).

(4) Hierzu gehören die folgenden Paradigmen: imperativ, funktional, prototypisch, objektorientiert

Prototypische Programmier-Elemente

- (1) Prototypisch bezeichnet eine spezielle Ausprägung der objektorientierten Programmierung
- (2) Alle Typen mit Ausnahme der Basistypen und Ausnahmewerte sind Objekte
- (3) Ein Objekt ist ein Container für Schlüssel-Wert-Paare
 - (1) Schlüssel ist ein Name
 - (2) Wert ist eine Variable (Property), eine Funktion (Methode) oder ein weiteres Objekt
 - (3) Das Objekt wird durch "var <Objektname> = ..." gebildet
 - (4) Der Zugriff auf die Werte eines Objekts erfolgt durch eine Punktschreibweise oder eine Klammerschreibweise
 - (5) Eigenschaften eines Objekts lassen sich zur Laufzeit hinzufügen und löschen

(1) Die prototypische Programmierung ist die Art der Objektorientierung, die JavaScript verwendet [Oc12:51].

(2) Selbst für die Basistypen gibt es Objekte, die sich auf diese Basistypen abbilden lassen. Somit können alle Typen als Objekte in JavaScript angesehen werden.

(3.1) Der Name kann ein beliebiger String oder ein gültiger JavaScript-Name sein.

(3.2) Ist der Wert eine Variable, nennt man diese Variable auch Eigenschaft des Objekts.

Ist der Wert eine Funktion, nennt man diese Funktion auch Methode des Objekts.

(3.3) (3.4) (3.5) Diese Aspekte werden anhand des nachfolgenden Beispiels verdeutlicht.

Beispiel eines Objekts "myC64"

(1) Das Objekt beschreibt den Homecomputer der 80er Jahre

```
1. var myC64 = {  
2.   "producer" : "Commodore",  
3.   "model" : "C 64",  
4.   "memory" : {  
5.     "ram" : "64 kB",  
6.     "rom" : "20 kB"  
7.   }  
8. }  
9.  
10. print(myC64.model); // C 64  
11. print(myC64.memory.ram); // 64 kB  
12. print(myC64["memory"]["ram"]); // 64 kB  
13.  
14. print(myC64.processor); // undefined  
15. myC64.processor = {};  
16. myC64.processor.producer = "MOS Technology";  
17. myC64.processor.model = "6510";  
18. print(myC64.processor.producer); // MOS Technology  
  
19. delete(myC64.producer);  
20. print(myC64.producer); // undefined
```

Anhand eines einfachen Beispiels wird die Deklaration eines Objekts, der Zugriff auf die Eigenschaften und die dynamische Änderung von Eigenschaften erklärt [Oc12:52].

(1) Das Objekt wird zunächst mit einigen, den Homecomputer beschreibenden Eigenschaften definiert und nachfolgend verändert.

(11. ... myC64.memory ...) ("12. ... myC64["memory"] ...) Die eckigen Klammern werden gewählt, wenn der Bezeichnername dynamisch (über String-Operationen) ermittelt wird.

(14. ... processor) Die Eigenschaft kennt das Objekt zu diesem Zeitpunkt nicht, weshalb "undefined" zurückgegeben wird.

(15. – 17.) Das Objekt wird um ein Objekt "processor" mit den Eigenschaften "producer" und "model" ergänzt.

(19.) Die Eigenschaft "producer" wird vom Objekt entfernt.

Prototypen

- (1) Ein Prototyp ist eine Blaupause, anhand dessen sich Objekte zur Laufzeit erzeugen lassen
 - (1) Die "prototype"-Eigenschaft des Konstruktors liefert eine Referenz
- (2) Ein Prototyp kann selbst wieder einen Prototypen haben
 - (1) Das Ende der Prototyp-Kette ist das Objekt "Object.prototype"
- (3) Ein Objekt erbt die Eigenschaften aller Prototypen aus der Prototypen-Kette
 - (1) Methode "hasOwnProperty" überprüft, ob die angegebene Eigenschaft zum Objekt gehört

```
1. var hello = new String("Hello");
2. hello.myProperty = "world";
3.
4. if (hello.hasOwnProperty("myProperty")) {
5.   print(hello + " " + hello["myProperty"]); // Hello world
```

Da JavaScript kein Klassenkonzept kennt, lassen sich Objekte zur Laufzeit verändern [Oc12:53].

(1) Die Vererbung wird in JavaScript durch die Prototypen realisiert.

(1.1) Der Konstruktor ist eine Funktion, die das Objekt erzeugt. Hierüber kann ein Objekt auf den Prototyp (bzw. genauer eine diesen repräsentierende Variable) zugreifen.

(2) Hierdurch entsteht eine Prototypen-Kette.

(2.1) "Object.prototype" ist der Prototyp eines normalen Objekts.

(3) Falls das Objekt die gleiche (gleichnamige) Eigenschaft besitzt, wird die vererbte Eigenschaft verdeckt.

(3.1) Die Herkunft kann sein: (i) aus dem Objekt selbst oder (ii) aus der Prototyp-Kette.

(4. if ...) Die Methode liefert "true" zurück, weil mit der Anweisung in Zeile 2 ("hello.myProperty = "World";) die Eigenschaft "myProperty" zum Objekt "Hello" hinzugefügt wurde.

- (1) Welche Sprachen sind Vorbilder von JavaScript und wie haben diese Sprachen JavaScript beeinflusst?
- (2) Was ist ein Objekt in JavaScript und was ermöglicht dieses zur Laufzeit?
- (3) Was leistet die Methode "hasOwnProperty"?