Ingrid Neckermann

Requirements Engineering – ein Einstieg in die Informatik für Mädchen

Requirements Engineering ist ein Teilbereich der Informationstechnologie bzw. der Geschäftsprozessanalyse. Ziel ist es, die Anforderungen des Auftraggebers an das Zielsystem zu erheben, zu analysieren, zu bewerten und über den ganzen Software-Lebenszyklus effizient zu verwalten. Das erfordert neben hohem Abstraktionsvermögen insbesondere auch eine Vielzahl von sprachlichen und sozialen Kompetenzen. Gerade Mädchen, für die die Möglichkeit zu Kommunikation und Teamarbeit einen wichtigen Aspekt der Berufswahl darstellt, finden hier eine erfüllende Aufgabe: Dieser Beruf ermöglicht ihnen ganzheitliches Arbeiten, also Kombinieren logischer und empathischer Fähigkeiten, zudem in einem gut bezahlten Umfeld. Dieser Aufsatz motiviert die Idee, seine Methodiken und Techniken im Unterricht zu nutzen, gibt Beispiele dazu und fasst Empfehlungen aufgrund bisheriger Erfahrungen mit einem solchen Unterrichtskonzept zusammen, die auch die Auswahl von Lehrplaninhalten betreffen.

1. Vorbemerkung

„Oh, der Computer ist ja richtig menschlich! Das macht sogar Spaß!", war ausgerechnet von einem Mädchenteam der 7. Klasse zu hören, das dem Computer anfangs recht reserviert gegenüber getreten war. Die Lehrkraft hatte aufgrund dieser wahrgenommenen Abneigung bewusst eine personifizierende Sprache für Technik gewählt, in der Hoffnung, so Berührungsängste zu reduzieren. „Mein Bildschirm hat Migräne, er flimmert", lachte ein Mädchen, fuhr dann aber sachlich fort: „Der Bildschirm ist kaputt, Sie müssen ihn austauschen lassen."

„Als Fachdidaktiker richten wir unseren Fokus auf die Schule und fragen, weshalb so wenig Mädchen am Informatikunterricht teilnehmen. [...] Die Gründe liegen vorrangig in den unterschiedlichen Interessen, Herangehensweisen und Verhaltensweisen von Jungen und Mädchen im Unterricht. [...]" [SK08]

Eigene Beobachtungen im Unterricht, auch der eigene berufliche Werdegang, der eher unfreiwillig in die IT-Branche führte, bestätigen dies. Motiviert durch die selbst erlebte Kluft zwischen dem – für Mädchen wenig ansprechenden – Bild der Programmiersprachen-Informatik, das Lehrkräfte noch immer vermitteln, und der tatsächlichen Vielfalt in den IT-Berufen, entstand bereits 2006 mein Wunsch, mit alternativen Wegen im Unterricht MINT-Fächer für Mädchen attraktiver zu machen.

Für diese Absicht boten sich im Schulbereich bisher nicht wirklich wahrgenommene Gebiete der Informatik mit hoher Affinität zu anderen Wissenschaften an (beispielsweise Web-Design zu Kunst, IT-Projektmanagement zu Betriebswirtschaft und Psychologie, internationales IT-Recht zu Jura). Dass die Informatik zunehmend „menschen-zentrierter" und somit für Mädchen interessanter wird, zeigen auch Begriffe wie SOA (Service Oriented Architecture), MDD (Model Driven Development), Social Networks oder neuere Technologien wie Task- (Aufgaben-) zentrierte (statt wie früher: Betriebssystem-/Menü-/Programmzentrierte) Benutzungsoberflächen von Smartphones und Tablets.

Dieser Aufsatz greift sich Methodiken des Requirements Engineering heraus, einem erst seit wenigen Jahren international zertifizierten (zum aktuellen Lehrplan vgl. [IR12]) und im Schulbereich noch weitgehend unbekannten Gebiet, da mit Fächern wie Deutsch, Sozialkunde und Wahlfächern wie Psychologie kooperiert werden kann. Zudem gibt es hier bereits genügend, erfolgreiche IT-Freelancerinnen, Firmengründerinnen, sogar bekannte Buchautorinnen, die Mädchen sofort als Identifikationspersonen dienen können.

2. Requirements Engineering für die Schule

Der Beruf einer Anforderungsingenieurin zeichnet sich durch große Nähe zu Kommunikationswissenschaften und sehr hohe Kompetenzanforderungen an Software-Architekturwissen auf Programmiersprachen-unabhängiger Ebene aus (vgl. Anhang, A.1. Kurzeinstieg in das Requirements Engineering und zu Überblickswissen [Ru12]).

Die Unterrichtsidee, es Schülerinnen zu ermöglichen, sich als Anforderungsingenieurin zu erleben, lag nahe: Quasi als Dolmetscherinnen sollen sie von zuerst bewusst unklaren, vagen Aufgabenstellungen der Lehrkraft zu IT-Modellen gelangen, aus denen sie dann – dies ist nicht mehr Requirements Engineering, rundet jedoch das methodisch-didaktische Vorgehen ab – in der 10. Klasse auch noch Programmcode in Java generieren können.

Beispiele für unterrichtsgeeignete Aktivitäten aus dem Requirements Engineering, die teilweise auch schon in der Unterstufe nutzbar sind, sind:

- Befragen von Stakeholdern, beispielsweise von Lehrern und anderen SchülerInnen (Interview-Techniken, User Storys),
- Systematisieren, Strukturieren, Klassifizieren von Anforderungen (Mind Mapping auf Papier und/oder mit Free Mind [SW01]),
- Ermitteln von Systemgrenzen (Brainstorming paradox, was soll *nicht* Systembestandteil sein?),
- Erfassen des wesentlichen Systeminhalts unter Nutzung unterschiedlicher Perspektiven (CRC-Karten, 6-Hüte-Perspektivenwechsel),
- Erstellen von Lasten- und Pflichtenheften (Dokumentenvorlagen) mit textuellen und konzeptionellen Modellen (Satzschablone, Funktionsmodell: Use Cases, Verhaltensmodell: Zustands- und Sequenzdiagramme, Strukturmodell: Klassendiagramm auf Papier und/oder mit der Software Diagram Designer [SW03], UMLPad [SW02] oder auch online im Internet mit Gliffy [SW07]),

- Generieren von Programmcode (z. B. für Java mit dem JavaEditor [SW09]).

3. Gender[1]-Beobachtungen

Ausgangsbasis der Überlegungen waren Beobachtungen aus der IT-Branche und aus dem Informatik-Unterricht, die deutlich den Eindruck hinterließen, Mädchen verlören das Interesse an Informatik, noch bevor sie eine Vorstellung von der Berufsvielfalt der IT-Branche entwickeln könnten, obgleich es nachweislich möglich ist, hier als Frau erfolgreich (auch im Sinne von Work-Life-Balance) zu sein.

3.1 IT-Branche

Informatik-Studentinnen und in Softwarehäusern programmierende Frauen stammen oft aus osteuropäischen oder überraschenderweise aus arabischen Ländern. Auf die Frage, warum sie sich als Frau in die Männerdomäne IT-Branche gewagt hätte, antwortet eine Russin: Programmieren ist daheim keine gut bezahlte Tätigkeit, also weiblich.

Softwarehäuser, die gemischte IT-Projekt-Teams bevorzugen, berichten über gute Erfahrungen mit Qualität, Termintreue und Kosten. Es ist noch zu erforschen, ob Frauen bereits im Vorfeld Risiken konsequenter reduzieren oder Männer in solchen Teams weniger riskante Entscheidungen treffen.

Softskills werden immer häufiger als ausschlaggebend für den Projekterfolg angesehen. Dies liegt auch daran, dass reine Programmiertätigkeit, die ohne große Sozialkompetenz auskommt, weitgehend in billigere Länder ausgelagert wird. Der dort entwickelte Programmcode hat jedoch nur dann die erforderliche Qualität, wenn die zugrunde liegende Anforderungsdokumentation eindeutig interpretierbar ist. Dementsprechend weist die IT-Branche in Deutschland weiterhin steigenden Bedarf an Menschen auf, die nicht nur Wissen zu Technik mitbringen, sondern auch als Vermittler/Übersetzer zwischen Auftraggeber und Auftragnehmer wirken können.

3.2. Informatik-Unterricht

Jungenteams, die sich für sehr kompetent halten, beginnen zu arbeiten, noch bevor die Aufgabenstellung vollständig geklärt ist. Sie vertiefen sich sofort in Aspekte, die ihnen als besonders reizvoll erscheinen. Der Junge arbeitet allein am eigenen Rechner, fragt andere Teammitglieder nur, um auszuloten, welche Lösung besser ist. Zum Schluss wird stolz eine lose Zusammenstellung von Einzelarbeiten präsentiert. Das Ergebnis ist technisch ansprechend, passt nur leider häufig nicht zur Aufgabe.

Mädchenteams bezweifeln bis zuletzt ihre Fähigkeit, die Aufgabe lösen zu können, wollen wissen, wozu diese gut ist, und müssen erst von der Lehrkraft zum Arbeiten ermutigt werden. Dann stapeln sie sich vor einem Rechner. Ihr Abstimmen im Team erfordert erheblichen Zeitaufwand. Auch das letzte Detail wird durchdiskutiert und mehrheitlich entschieden. Das so entstandene Lösungskonzept ist ausgereift, passt zur Aufgabe, wird jedoch nicht fertig. Präsentiert werden eher Probleme mit der Aufgabe als Ergebnisse. Zudem sind ihre Vorträge kürzer (vgl. zu Sprachmustern [Bo99]).

Zusammengefasst fällt für Mädchen auf: Sie

- trauen sich keine natürliche Kompetenz zu Technik zu,
- brauchen Ermutigung, um selbst aktiv mit dem PC-Arbeiten zu beginnen (und nicht nur Beifahrerin eines Jungen zu sein, die selbst das Fahren nicht lernt),
- wünschen Unterstützung, um den richtigen Einstieg in eine Aufgabe zu finden,
- bevorzugen konkrete Aufgaben, die Wert auf Ästhetik und Perfektion legen,
- hinterfragen Aufgaben nach ihrem Sinn, suchen nach nützlichen Lösungen,
- können eigene Gefühle leichter formulieren als Systemgrenzen,
- finden Lösungsideen eher konzeptionell im Dialog als experimentell allein,
- empfinden „heiße Tipps und Tricks“ zu einer Software oft als demotivierend,
- halten ihre Lösungen stets für unzureichend, finden daher schwer ein Ende,
- gelangen, wenn auch auf anderem Weg, zu nahezu denselben Lösungen wie Jungen, wenn sie genügend Selbstsicherheit gewinnen konnten und Zeit zum Trainieren hatten.

4. Folgerungen, Unterrichtsbeispiele

Naheliegend sind somit folgende Fragen (Analoges lässt sich übrigens auch für Jungen bei sprachlichen Fächern erfragen):

- Berücksichtigen Lehrplan und Schulbücher die unterschiedlichen methodischen und thematischen Bedürfnisse beider Gender in gleicher Weise?
- Wie lassen sich anfängliche Schwierigkeiten von Mädchen ausgleichen?
- Wie kann so unterrichtet werden, dass zusätzlich für Mädchen attraktive IT-Berufsbilder (mit und auch ohne Informatik-Studium) vermittelt werden?

Der bayerische Informatik-Lehrplan [LPBay] bietet nur gelegentlich Ansätze zu konzeptionellem Arbeiten. Schulbücher enthalten eher Jungen interessierende Aufgaben wie Roboter, LWK, Aufzug oder Fußball, jedoch kaum für Mädchen angemessene wie soziale Themen, Tiere, Chat oder Facebook. Ihre Fragen der Art „Wie schminke ich mich gut?“, „Wie fühle ich mich?“, „Wie spreche ich mit meiner Freundin, wie im Vergleich dazu mit der Lehrkraft?“ lassen sich als Algorithmus, mit Zustands- oder Sequenzdiagrammen lebensnah modellieren.

Gegen anfängliche Schwierigkeiten erwies sich bewusstes Gegensteuern der Lehrkraft zu Schuljahresbeginn als hilfreich. Die stolz vortragenden Jungen erhielten nur sparsam Lob, wurden auf ihre Themenverfehlung deutlich hingewiesen: „Stell' Dir vor, Du wünschst Dir von Deinen Eltern ein Snowboard um Snowboarden zu gehen, und bekommst eine perfekte Computersimulation mit Snowboard!" Die Mädchen wurden für gute Gedanken gelobt, ihr Nichtfertigstellen als völlig normal eingeordnet. Lehrplaninhalte zu Schuljahresbeginn wurden bewusst dem konzeptionellen Bereich entnommen, die Themenwahl zum Umsetzen jedoch den einzelnen Teams selbst überlassen. (In ersten Leistungsnachweisen schnitten die Mädchen in den Klassen stets besser als die Jungen ab, was sie dazu motivierte am Ball zu bleiben, und die Jungen dazu, den vermeintlichen Vorsprung in ihrer Domäne wieder einzuholen. Zum Halbjahreswechsel waren weit weniger Verhaltens- und Arbeitsergebnisunterschiede als zu Beginn zu beobachten.)

Zum Unterrichten verdeutlicht die folgende Auswahl an Beispielen, wie „konzeptionell" und „Requirements Engineering à la Schule" (REQ) in diesem Aufsatz aufzufassen ist (siehe auch Anhang, A.2. Ergänzungen zum Kapitel 4):

- 6. Jg. – LP: *objektorientiertes Modellieren, Beziehungen, Objektdiagramm*
  REQ: Anforderungen strukturieren, CRC-Karten, Interview und Kreativitätstechniken nutzen, das System abgrenzen und bestimmen
  Aufgabe: Die SchülerInnen erhalten Kärtchen mit Begriffen. Diese sollen sie im Dialog mit einem Partner strukturieren und – ergänzt um wenige eigene Kärtchen mit ihren Begriffen – grafisch zu einer sinnvollen Geschichte zusammenbauen.

- 6. Jg. – LP: *Umgang mit Textverarbeitungsprogrammen, Vorlagen*
  REQ: Brainstorming, User Storys, Text-Templates, Reuse, Satzschablone
  Aufgabe: Die Schüler sollen eine eigene Geschichte erfinden und diese am Computer in einer standardisierten Form als Text eingeben.

- 7. Jg. – LP: *Punktnotation, (nicht im LP) Nutzen der Programmiersprache EOS*
  REQ: Gewichten von Anforderungen, Satzschablone, Übersetzen von Gesprächsnotizen in standardisierte Satzform
  Aufgabe: Die Schüler sollen einfache Sätze mit Subjekt, Objekt und Prädikat in Punktnotation umformen, danach schwerere erst in einfache und dann in Punktnotation umformen, für geometrische Objekte dies auch programmieren.

- 7. Jg. – LP: *algorithmische und objektoriente (Grundwissen-Wiederholung) Modellierung*
  REQ: Anforderungen strukturieren, unscharfe Anforderungen präzisieren, dynamische (Algorithmus) und statische (Objekte) Sicht einnehmen
  Aufgabe: Die SchülerInnen erhalten Sätze, aus denen sie mit möglichst wenig zusätzlichen Worten eine Anleitung erstellen sollen. Sie sollen aufschreiben, welche Objekte vorkommen, und in welcher Beziehung diese zueinander stehen. (Wird die Aufgabe durch Vertauschen von Buchstaben z. B. statt „Stall" z. B. „Stlla" erschwert, sind Mädchen dann oft schneller als Jungen.)

- 9. Jg. – LP: *EVA-Prinzip, prozessorientiertes Modellieren, Prozessdiagramm*
  REQ: muss/kann/soll unterscheiden, dynamische Sicht (Prozesse), statische Sicht (Objekte), Persistenz, Teilanforderungen ermitteln
  Aufgabe: Die SchülerInnen sollen das Zusammenbauen eines selbst gewählten Gegenstands beschreiben, dann mittels eines Prozessdiagramms darstellen, dabei klar zwischen Materialen und Aktivitäten unterscheiden.

- 10. Jg. – LP: *UML-Modelle (Zustands-, Sequenz-, Klassendiagramm), Java*
  REQ: Systemgrenzen bestimmen, User Storys, Funktionsperspektive (Use Cases), Verhaltensperspektive, Strukturperspektive einnehmen
  Aufgabe: Die SchülerInnen sollen textuell, dann mit Use Cases ihre Projektidee beschreiben. Ausgehend von einer wichtigen Klasse sollen sie unterschiedliche Perspektiven einnehmen, passende Modelle, z. B. Zustandsdiagramme iterativ erstellen, zuletzt aus diesen Modellen Java-Code generieren.

Ingrid Neckermann

Ingrid Neckermann blickt auf eine langjährige IT-Erfahrung in den Sektoren Banken, Versicherungen und Versorgungsindustrie zurück. Nach ihrer Tätigkeit am Mainframe als Entwicklerin befasste sie sich als Software-Architektin und Projektleiterin vor allem mit strategischen Technologiefragen wie Einführung eines Firmen-Intranets, Neuentwicklung eines Unternehmensframeworks in J2EE oder Marktfähigkeit neuer Hypes wie JavaScript, XML, JUnit, OLAP, Data Warehouse oder BPMN. Auch in ihrer Tätigkeit als Gymnasiallehrerin strebte sie stets nach innovativen Ansätzen. Ihr besonderer Augenmerk galt der Mädchenförderung für MINT-Fächer. Das abgedruckte Paper ist im Tagungsband des 5. Münsteraner Workshops zur Schulinformatik, Leitung Hr. Prof. Thomas Marco und Hr. Weigend, ISBN 978-3848201815 am 25.04.2012 erschienen.

5. Empfehlungen

Die in diesem Aufsatz aufgeführten Beobachtungen und Schlüsse gilt es noch wissenschaftlich tiefer zu erforschen. Überdenkenswert sind hierbei folgende Empfehlungen:

1. Weiterentwickeln des Lehrplans (auch in Form von Kooperationsvorschlägen mit Fächern wie Deutsch, Englisch, Sozialkunde, Psychologie u. a.):

   - Aufnehmen von in der IT eingesetzten Befragungstechniken (z. B. Interview, Fragebogen) und Kreativitätstechniken (z. B. Brainstorming, Perspektivenwechsel, Analogietechniken) aus kundennahen Gebieten der Informatik [RU12],
   - Hinzufügen von IT-typischen Methodiken wie iterativem Entwickeln von IT-Systemen, Annähern an ein System aus unterschiedlichen Sichten und Perspektiven,
   - Hervorheben des Modellierens als wesentliches Thema der Informatik,
   - Überarbeiten nicht altersgemäßer Inhalte: Klassendiagramme überfordern wie das Variablenkonzept in der Mathematik oft noch in der 6. Jg. Statt in der 9. Jg. SQL und in der 10. Jg. Java als Block zu unterrichten, wäre es besser, in der 9. Jg. einfaches SQL und einfaches Java zu vermitteln, beides in der 10. Jg., auch miteinander kombiniert, zu vertiefen.

2. Überdenken der Benachteiligung des Fachs Informatik in der Abiturprüfungsordnung: Informatik, eine Ingenieurwissenschaft, gilt nicht als Naturwissenschaft, reduziert daher die Anzahl der noch frei wählbaren Profileinbringungen und führt so bei vielseitig begabten Mädchen zur Wahl risikoärmerer Fächerkombinationen.

3. Weiterentwickeln der Schulbücher:

   - Erweitern um konzeptionelle, strukturelle, Gendersensible Aufgaben,
   - Aufnehmen von Rollenspielen, die die Vielfalt der IT-Berufe bekannt machen,
   - Verbessern von „Unschärfen“: Prozessdiagramme sollten nicht unter dem Titel „Datenflussdiagramm“, prozeduraler Programmcode unter der Überschrift „Objektorientiertes Programmieren“ abgedruckt werden u. ä.

Überlegenswert wäre es, ein eigenes Schulfach *Kommunikation und Information* zu entwickeln. Dieses könnte bewusst Parallelen zwischen menschlichem Verhalten und technischer Umsetzung ziehen (vergleiche hierzu: Bionik, Bisoziation). In diesem zwei- bis dreistündigen Fach ist eine Stunde neigungsspezifisch vertiefend gestaltet. Zu demselben Thema arbeitet die eine Gruppe konzeptionell (Diskutieren, Modellieren, organisatorisch Managen), die andere technisch vertiefend (Experimentieren, Programmieren, technisch Managen). Austausch zu Ergebnissen erfolgt im Klassenverband.

Anmerkung

1 Der Begriff Gender wird verwendet, um auszudrücken, dass die Unterscheidung nach Sex nicht getroffen werden kann.

Literaturverzeichnis

[Bo99] Bown, Geraldine; Brady Catherine: Klartext sprechen, mehr Erfolg im Beruf. Walhalla U. Praetoria; 1. Aufl., 1999.

[IR12] CPRE Zertifizierung (CPRE Foundation Level Lehrplan 2.1. (deutsch) ist seit 1. September 2012 in Kraft). http://www.certified-re.de,

[LPBay] http://www.isb.bayern.de, Navigation: Lehrpläne/Standards, G8-Lehrplan, Natur- und Technik, 6.-7. Jg. und Informatik, 9. – 12. (nur am naturwissenschaftlichen Gymnasium, am nicht-naturwissenschaftlichen Gymnasium wird die 9.-10. Jg. für die Oberstufe als Wahlpflichtfach empfohlen).

[Po96] Pohl, Klaus: Process Centred Requirements Engineering. Research Study Press, 1996.

[PR11] Pohl, Klaus; Rupp, Chris: Basiswissen Requirements Engineering, Aus- und Weiterbildung zum „CPRE (Certified Professional for Requirements Engineerings)“. dpunkt.verlag, Heidelberg, 3. Aufl., 2011; S. 11-14.

[Ru12] Rupp, Chris: Requirements Engineering, Ein Überblick. dpunkt. verlag, Heidelberg, 3. Aufl., 2012.

[SK08] Schulte, Carsten; Knobelsdorf, Maria: »Jungen können das eben besser« – Wie Computernutzungserfahrungen Vorstellungen über Informatik prägen. In Mechthild Koreuber, Hrsg.: Struktur und Geschlecht. Über Frauen und Männer, Mathematik und Informatik, Baden-Baden, 2008. Nomos Verlagsgesellschaft.

[SW00] Software für den Mathematikunterricht, Geogebra. http://www.geogebra.org/cms/

[SW01] Software zum Erstellen von Bildern wie Objekt-, Klassendiagramme, Benutzeroberflächen, Schaltkreise … DiagramDesigner: http://meesoft.logicnet.dk

[SW02] Software zum Erstellen von UML-Diagrammen. Luigi Bignami, UMLPad. http://web.tiscali.it/ggbhome/

[SW03] Software zum Erstellen von Mind Map‘s, FreeMind. http://freemind.sourceforge.ne

[SW04] Software zum objektorientierten Programmieren, EOS. http://www.pabst-software.de

[SW05] Software zum Erstellen von Webseiten, PSPad. http://www.pspad.com

[SW06] HTML-Seiten für Kinder, außerdem für HTML4.0: http://de.selfhtml.org und für CSS 2.1.: http://www.css4you.de

[SW07] Software (im Web) zum Erstellen von z. B. UML, Gliffy. http://www.gliffy.com

[SW08] Software zum Programmieren in Java, BlueJ. http://www.bluej.org

[SW09] Software zum Generieren von Java-Code, JavaEditor. http://www.javaeditor.org

[SW09] Software zum Generieren von Java-Code, JavaEditor. http://www.javaeditor.org

Anhang

A.1. Kurzeinstieg in das Requirements Engineering

„Requirements Engineering ist kaum noch wegzudenken, wenn es darum geht, für den Kunden zufriedenstellende Systeme zu entwickeln und dabei Zeit- und Budgetpläne einzuhalten." [Ru12] Dieser systematische, kooperative und iterative Ansatz zur Spezifikation und zum Management von Anforderungen versucht, [...] alle relevanten Anforderungen aller Stakeholder [...] (Erg.: Diese sind Personen und Systeme mit Einfluss auf das System) [...] zu ermitteln, ausreichende Übereinstimmung unter diesen zu erzielen und diese Anforderungen so zu managen, dass Risiken [PR09], insbesondere auch Kosten, minimiert werden.

Requirements Engineering (Abb.1) führt insbesondere in den Phasen „Initialisierung" und „Voranalyse" eines IT-Projektes mit vielen kleinen Bausteinen, d. h. Methodiken aus den Kommunikationswissenschaften und Techniken aus den Ingenieurwissenschaften, zu Anforderungen und letztendlich zu standardisiert formulierten Dokumenten, die Modelle enthalten. Bis zum Ende eines IT-Systems bleibt Requirements Engineering begleitend, da es noch in der Projektphase den Grundstein auch für das zukünftige Managen von Anforderungen legt:

Requirements Engineering hat den großen Vorteil, nur bedingt automatisierbar und somit ungeeignet für Outsourcing bzw. Offshoring zu sein.

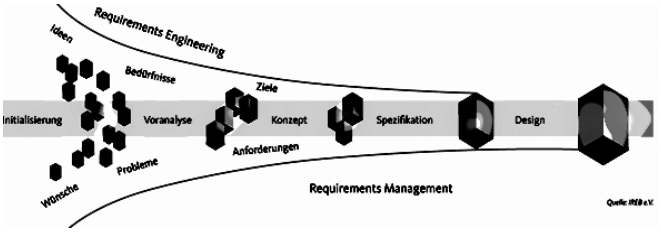


Abbildung 1: Requirements Management (Anforderungsmanagement), Quelle: IREB e.V

Als Anregung zum Entwickeln eigener Unterrichtsideen sei noch auf seine vielfältigen Schnittstellen wie IT-Projektmanagement, Controlling, Software-Architektur, System-Design, Test- und Schulungsmanagement hingewiesen.

A.2. Ergänzungen zum Kapitel 4

A.2.1. Typische Themen in SchülerInnen-Teams (Beispiele)

Abbildung 2: 5. Jg. LFG Biene, 7. Jg. LSG Handtaschen, 10. Jg. LSG ein Modell, das zum Friseur muss – jeweils von Schülerinnen-Teams erstellt

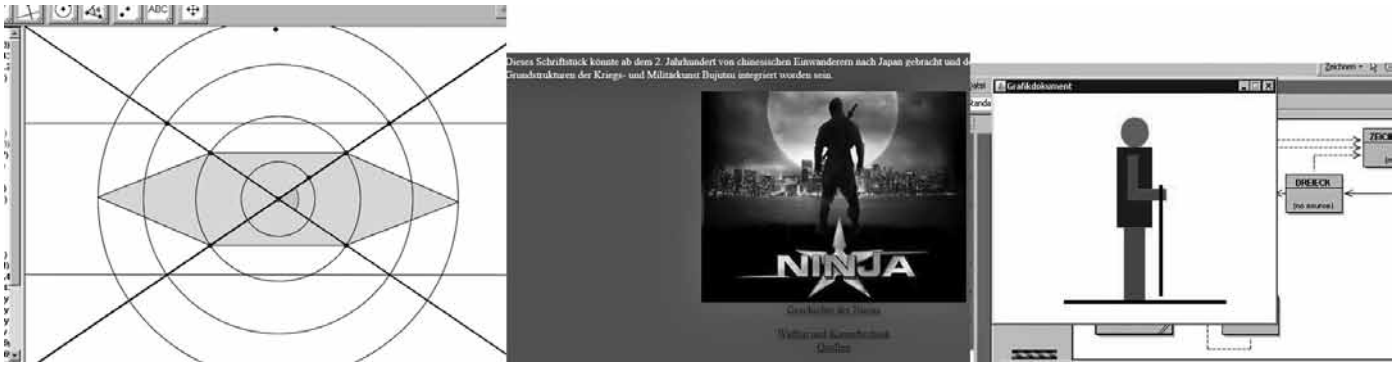


Abbildung 3: 5. Jg. LFG Zielscheibe, 7. Jg. LSG Ninjas und Samurei, 10. Jg. LSG ein Skifahrer – jeweils von Schüler-Teams erstellt (5. Jg. mit der Mathematik-Software-Geogebra [SW00] erstellt, 7. Jg. mit der Software PSPad [SW05] für HTML-Web-Seiten, 10. Jg. mit der Software BlueJ für Java [SW08]).

Bzgl. Technik bevorzugten Mädchen zuerst Farbauswahl. Für ihre Web-Seiten die passende RGB-Farbe entsprechend ihres Lieblingsstars abzumischen, erforderte mehr als eine Unterrichtstunde [SW05]. Fading-Effekte mit CSS [SW06] waren bei ihnen sehr beliebt. Jungen dagegen untersuchten zuerst den von der Lehrkraft angegebenen Link zu SelfHTML [SW06]. Als die Mädchen bei Jungen Laufschriften entdeckten, übernahmen sie diese und färbten die Texte. Umgekehrt übernahmen Jungen später Farbtricks der Mädchen. Nach fast drei Monaten Projektarbeit glich sich das Wissen aneinander an.

A.2.2. Anfangsunterschiede im Abstraktionsgrad (Beispiele)

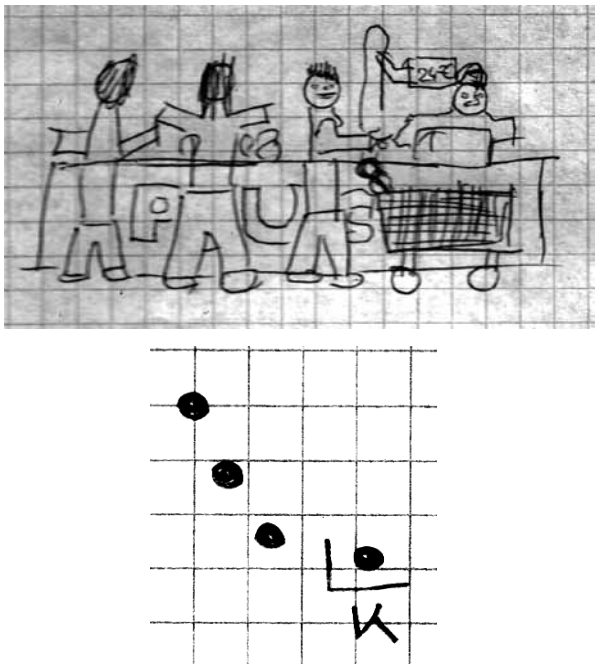


Abbildung 4: Aufgabenart „Zeichne für einen Computer ‚umsetzbar' mit möglichst wenigen Strichen das Geschehen an einer Supermarktkasse". 9. Jg. LFG
oben: (vermutlich) ungeübte Mädchen – unten: abstrakte Jungenlösung

Wurde diese Aufgabenart wiederholt, stieg der Abstraktionsgrad, und es waren keine Unterschiede zwischen den Gender mehr zu erkennen. Eventuell haben Jungen aufgrund von Computerspielen einfach mehr Vorerfahrungen?

A.2.3. Satzschablone (Mittelstufe, vereinfacht auch für Unterstufe)

SchülerInnenbeispiel:

- 1. Schritt (Alltagstext): „Heute ist es sehr heiß, hat wohl 31°. Da ist eigentlich Hitzefrei. Das muss der Schulleiter in der Pause noch durchsagen".
- 2. Schritt (Satzschablone): „Bei über 30° MUSS der Schulleiter uns hitzefrei geben".
- 3. Schritt (Punktnotation): *SchulleiterIn.durchsagen(Hitzefrei).*

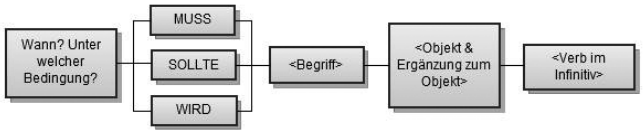


Abbildung 5: Satzschablonen-Vorlage für den Unterricht (zur Satzschablone siehe [PE11])

A.2.4. Projekt-gesteuertes Vermitteln der Lehrplaninhalte (10. Jg.)

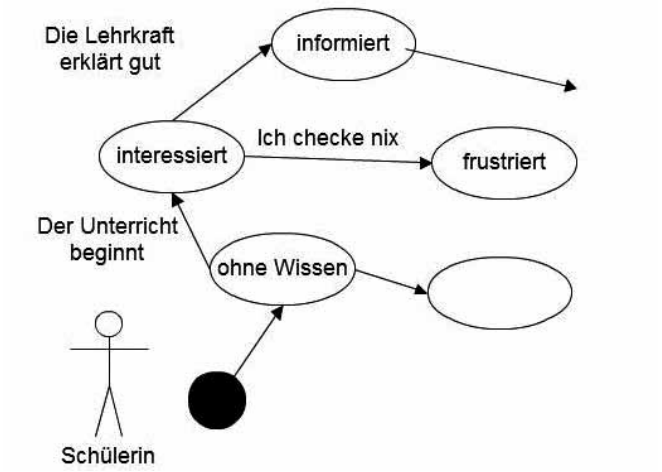


Abbildung 6: Vorübung: Erstelle ein Zustandsdiagramm zum Thema „Wie fühle ich mich im Unterricht?", Nachzeichnung einer Schülerin-Zeichnung mit [SW07], 10. Jg. LSG

Projekt-Ablauf (Ziel: Vermittlung des Lehrplans entsprechend aktueller Bedürfnisse im Projekt):

- 1. Schritt: Teambildung, Themenwahl in den Teams (z. B. Auto, Feuerzeug, Glücksschwein).
- 2. Schritt: Erklären des Ablaufs im Projekt: Das Thema der Lehrkraft hat den Titel „Sonnenaufgang". Sie zeigt anhand dieses Themas die nächsten Schritte, die die Teams für ihr Thema analog umsetzen sollen.
- 3. Schritt: (iterativ) Realität – Modell – Programmcode erzeugen und testen
  Lehrkraft: Foto mit Sonne, Abstrahieren der Sonne zu einem Kreis, Eigenschaften (Attribute) und Aktivitäten (Methoden) der Kreis-Sonne, Klassenkarte zur Kreis-Sonne, Programmcode (mit noch leeren Methoden) zur Kreis-Sonne, Systemgrenzen (gehört der Horizont noch zum System?), Zustandsdiagramm zum Farbwechsel der Kreis-Sonne, zugehöriger Programmcode (if-then-else, for, while), Sequenzdiagramm zum Lauf der Kreis-Sonne über den Bildschirm, Timer-gesteuert usw. bis zu Vererbung, Polymorphismus.
- 4. Schritt: Präsentieren der Team-Ergebnisse.
- 5. Schritt: Erstellen einer Programmbibliothek mit den Projekt-Ergebnissen, Austausch.

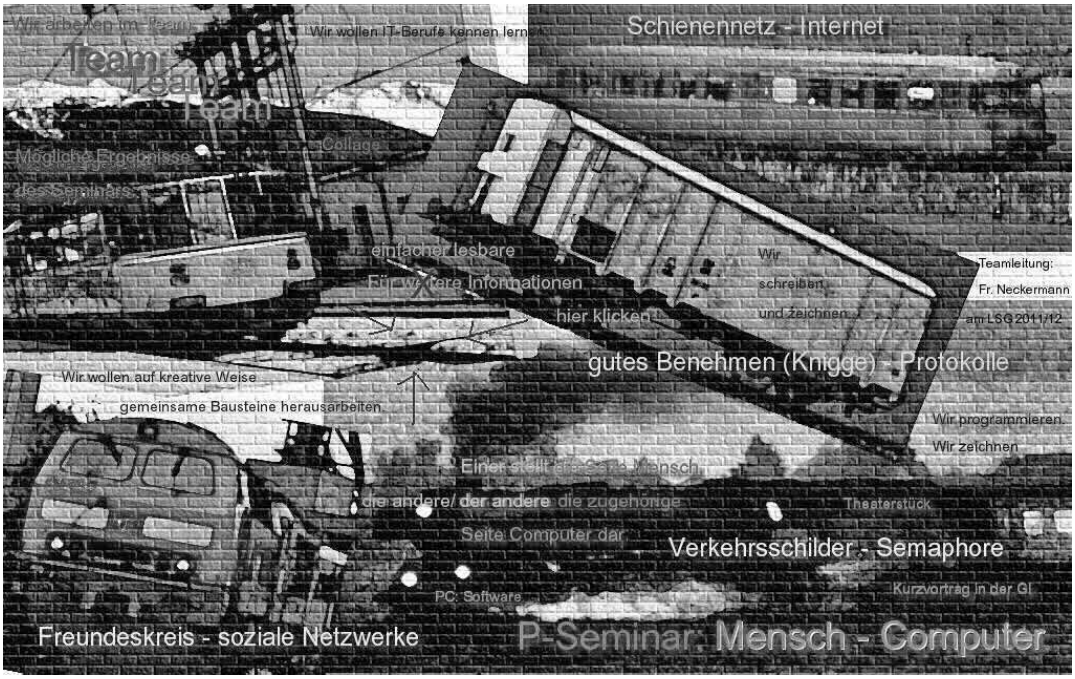
A.2.5. „Mensch und Maschine“ (Oberstufe, Seminarfachthema)

Das bekannte Modell „Die vier Seiten einer Nachricht“ hat strukturelle Ähnlichkeit mit der Projektion in der Mathematik, der Fluchtpunkt-Zeichnung in der Kunst und den unterschiedlichen Perspektiven „Funktion“, „Verhalten“ und „Struktur“ des Requirements Engineering.

Das Begrüßen zweier Menschen erfolgt nach einem Protokoll so wie dies bei Netzwerken auch üblich ist. Dies, auch ein Dialog zwischen Lehrkraft und Klasse, lässt sich als Sequenzdiagramm darstellen.

Probleme zwischen Männern und Frauen ähneln nicht kompatiblen IT-Systemen. Gelegentlich sind Ideen nötig, die Adapterlösungen gleichen oder Middleware erfordern.

Ein solches Seminarfach könnte noch in der Oberstufe Mädchen, die mit Informatik nur Computerspiele und Programmieren verbinden, auch eine andere Seite der Informatik zeigen.



erschienen in der FIfF-Kommunikation, herausgegeben von FIfF e.V. - ISSN 0938-3476 www.fiff.de