Software Entwicklung in der Wissenschaft

Ein Dialog zwischen Theorie und Praxis einiger Softwaretechnikelemente anhand eines Interviews

Gliederung

1. Motivation
2. Interviewpartner
3. Vorstellung des Forschungsgebiets und der Software
4. Ausgewählte Hauptkonzepte der Softwaretechnik und die Praxis
5. Andere relevante Aspekte der Entwicklung und Besonderheiten
6. Fazit

Motivation

- Warum ein Interview?
  - Interesse
  - Erwartungen
  - Unterschiede

Quelle: http://www.stemdigitalvillage.com/stem/images/inter.png

Interviewpartner

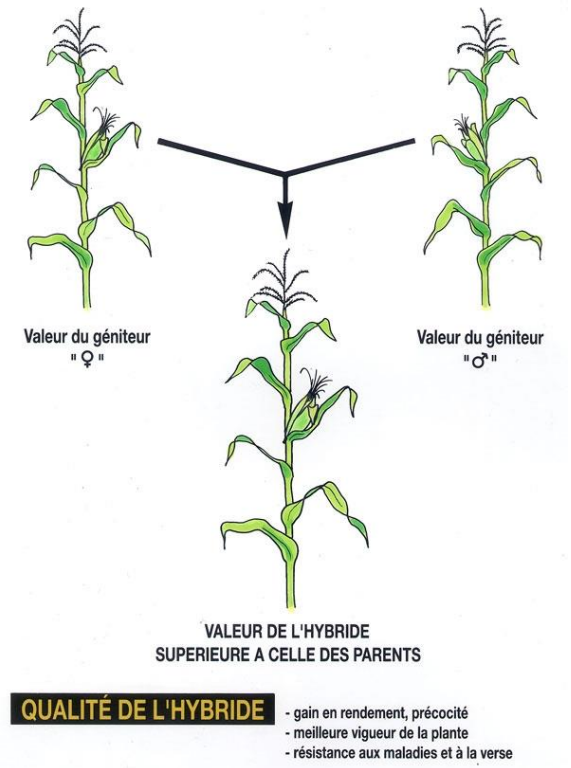
- **Wissenschaftler**: Dr. Felix Seifert PhD in Bioinformatik
- **Institution**: Universität Hamburg
- **Formation**
- **Wissenschaftliche Software**: Vorhersage von hybridischen Kreuzungen

Quelle: https://www.biologie.uni-hamburg.de/biozentrum-klein-flottbek/forschung/eb-schnittger/mitarbeiter/m-scholten/fseifert.html

Forschungsgebiet und Software

Forschung

- **Schwerpunkt**: Kreuzung von Maispflanzen
- **Problematik der Arbeit**: Wie kann man die besten Pflanzen für die Kreuzung auswählen
- **Methodik**: Vorhersagemethode mit RNA-Markierung
- **Ziel**: Probekreuzung vermeiden, Zeit Gewinnen, Kosten sparen



Quelle: http://www.gnis-pedagogie.org/photos/mais---effet-dheterosis.jpg

Software

- **Funktion**: Auswahl der besten Pflanzen für die praktische Kreuzung in großen Zuchtlinien von Maispflanzen
- **Berechnungsdauer**: Über eine Stunde
- **Hardwareanforderungen**: Linux-Maschine mit 4Ghz Prozessor

- **Historie**
  - **Konzipierung**: Ende 2012
  - **Aufwand**: 6 Monate
  - **Formulierung**: Scratch
- **Entwickler**
  - Dr. Felix Seifert
- **Benutzer**
  - Eigener Endbenutzer

- **Struktur**
  - **Sprache**: Java
  - **Zusatzsoftware**: MySQL
  - **Entwicklungswerkzeug**: Netbeans
  - **Größe der Software**: Ca. 2 MB
  - **Interne Klassen**: 10-15
  - **Strukturierung**: Packages für Cluster von Klassen vorhanden
  - **Klassengröße**: Im Schnitt 200 Zeilen
  - **Kommentaren**: Teilweise vorhanden
  - **GUI**: Nicht vorhanden
  - **Test-Klassen**: Nicht vorhanden

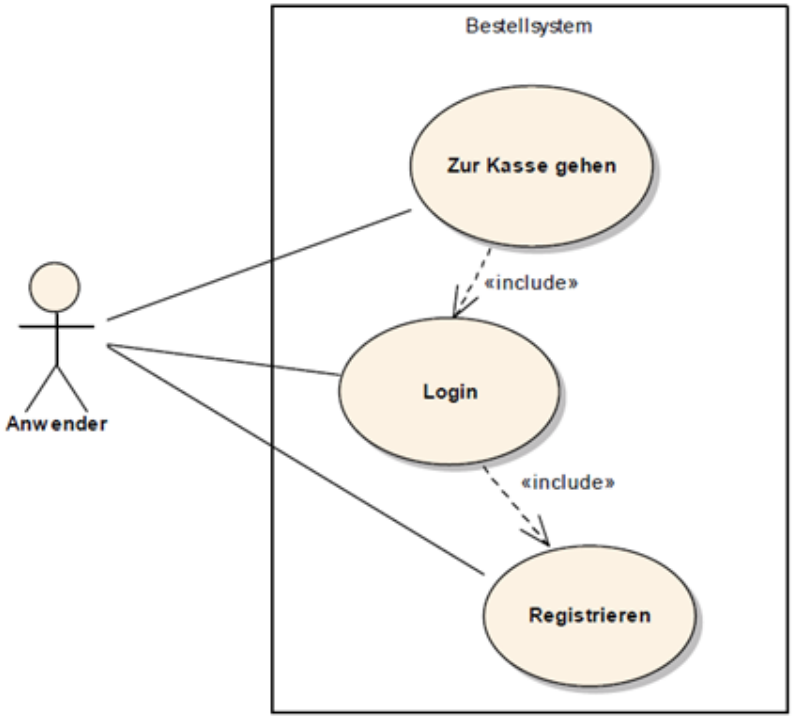
- **Einsatz**
    - **Eingabedaten**: Zur Zeit sind es 150 GigaBytes in der DB
    - 9.6 Gig 25 Linien Pflanzen werden bei jedem neuen Lauf der Software eingespielt

- **Ergebnis**
    - **Ausgabedaten**: Csv Datei
    - **Datenmanagement**: Mustererkennen. Andere Software angewendet. Selbe Festplatte. Nicht gebrauchte Daten werden einfach ignoriert.

Ausgewählte Konzepte der Softwaretechnik und die Praxis

- Use Cases
- Aktivitätsdiagramm
- Mock-ups
- Testen
- Code-Review

4.1 **Use Cases**

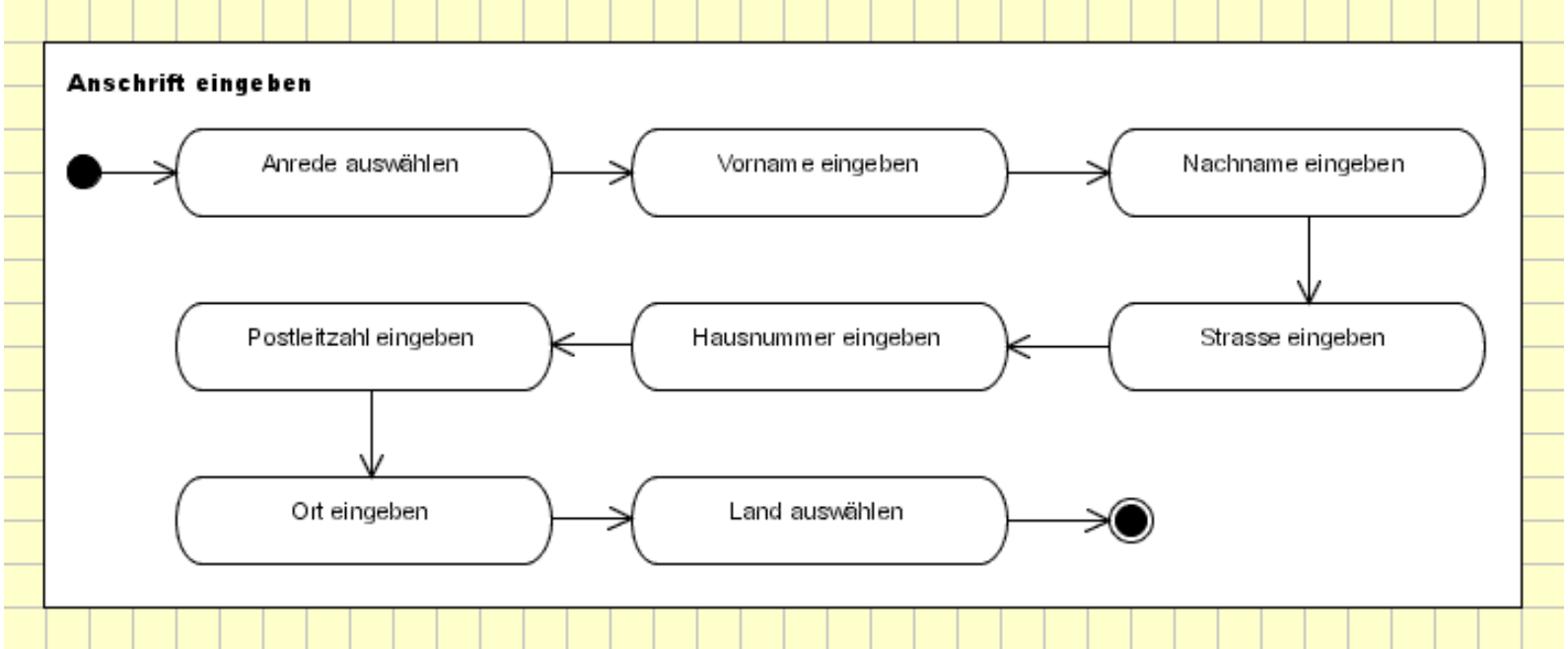


Quelle: https://www.codecentric.de/files/2011/07/web-shop-beispiel-user-cases.png

Praktische gefundene Form

Use Cases: Keine

4.2 **Aktivitätsdiagramm**

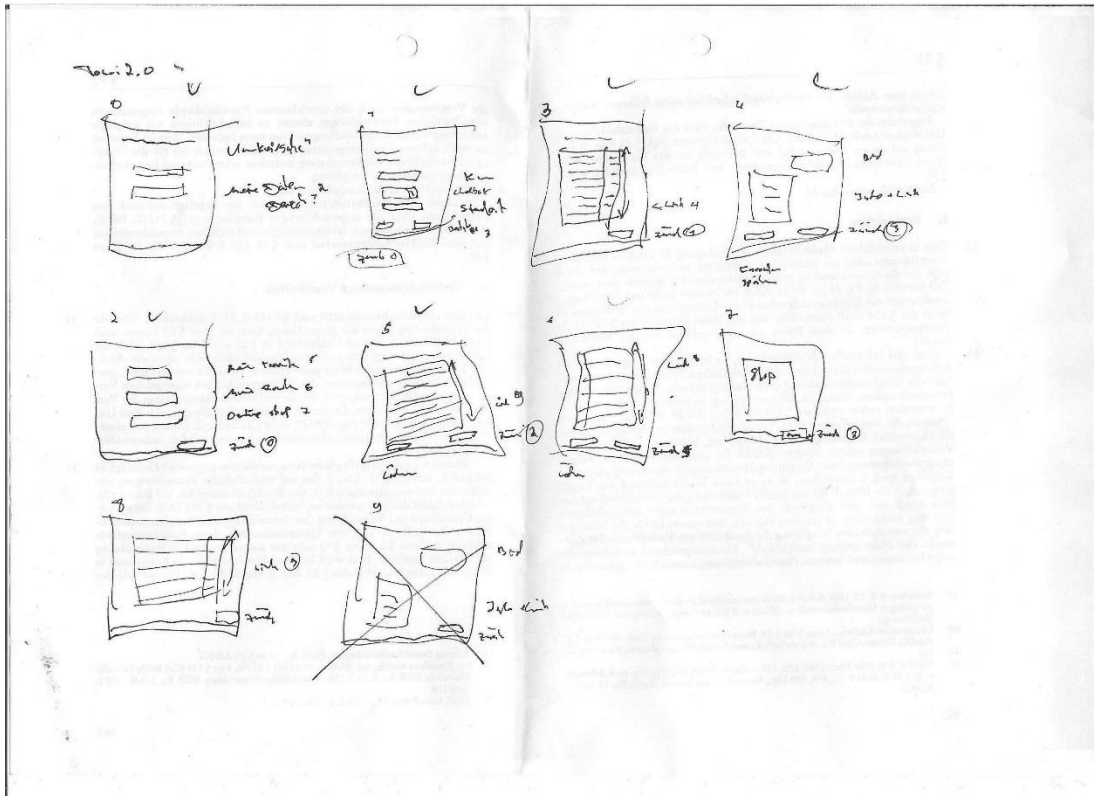


Quelle: http://www.highscore.de/uml/img/aktivitaet_anschrifteingeben.gif

Praktische gefundene Form

Aktivitätsdiagramm: Keine

4.3 **Mock-up**



Quelle: http://www.highscore.de/uml/img/aktivitaet_anschrifteingeben.gif

Praktische gefundene Form

Mock-up: Keine

4.4 **Testen**

Quelle: Links http://www.softwaretestingclass.com/wp-content/uploads/2013/10/negative-testing.jpg
Rechts http://www.softwaretestingclass.com/wp-content/uploads/2013/10/postitive-testing.jpg

Praktische gefundene Form

- **Testen**
  - **Debuggen**: Nur bei Fehlern. Wenn das Programm „nicht weiter berechnet“
  - Testen im theoretischen Sinne nicht vorhanden
  - Bei Abwesenheit von Fehlern werden die Ausgabedaten mit Erwartungsdaten (Probe) verglichen und damit weiter gearbeitet

4.5 **Code-Review**

- **Quelltext**: Abschnitt vom Code auf Fehlern, Korrektur, Vereinfachung prüfen
- Wird selbst oder durch Dritten durchgeführt

Qualitätsziele / Qualitätsmerkmale / Checkliste	Wartbarkeit					
	Analysierbarkeit	Änderbarkeit	Stabilität	Testbarkeit	Konformität (W.)	
SE2-Konventionen						
2.1 Code-Reihenfolge						
2.2 Einrückungen						
2.3 explizite Import-Anweisungen						
3.1 Schnittstellenkommentare („Was“)						
3.2 Implementationskommentare („Wie“)						
3.7 Methoden-Doku (“Javadoc-Tags“)						
4.2 Variableninitialisierung						
4.3 Klammerung von Blöcken(„{ ... }“)						
4.4 Klammerung von Bool-Ausdrücken						
5.1 Benennung von Klassen („Nomen“)						
5.2 Benennung von Methoden („Verben“)						
5.3 „Sprechende“ Variablen-/Methodennamen						
5.3 Konvention für Exemplarvariablen						
5.5 Namenskonvention für technische/fachliche Begriffe						
6.1 „private“ Exemplarvariablen						
6.1 Methoden mit einer „Return“-Anweisung						
Fowler: Bad Smells						
Comments: clarify "why" not "what"						
Klassengröße						
Methodengröße						

Praktische gefundene Form

Code Review: Teilweise. Während der Entwicklungsphase und nur bei Feststellung von Fehlern.

Andere relevante Aspekte

- Notwendigkeit der Software
- Warum Java?
- Zeitlicher Aufwand vs. Kosten
- Beziehung zwischen Softwaretechnik und Softwareentwicklung

- Entwickler und Benutzer sind dieselbe Person. Keine Software-Entwickler waren im Entwicklungsprozess benachteiligt.

- Erweiterungen, Pflege, Verbesserungen werden in der Software nicht durchgeführt. Auch eine Analyse von Änderungsnotwendigkeit findet nicht statt

- Freeware-Softwares verwendet (Netbeans, MySQL)

Fazit

- Software wird als Mittel gesehen
- Schnelle Leistung ist nicht Ziel der Software: Kosten werden priorisiert
- Theorie teilweise unbekannt oder bewusst nicht eingesetzt
- Gute Software obwohl keine intensive theoretische Auseinandersetzung

Literatur

Unterstanding the High-Performance-Computing Community: A Software Engineer‘s Perspective. Cruzes, Carver, Hochstein, Hollingsworth, Shull. IEEE 2008.

Vorlesung: Softwaretechnik. Gryczan, Riebisch. Sommersemester 2015.

Softwaretechnik in Wikipedia https://de.wikipedia.org/wiki/Softwaretechnik Letztes Mal aufgerufen am 06.07.2015

Vorstellung der Software

Anwendungsbereich

Die Pflanzenzucht der meisten wirtschaflich relevanten Pflanzen erfolgt über Hybridzucht bei der zwei unterschiedliche reinerbige Eltern gekreuzt werden um den dabei auftretenden Heterosiseffekt, welcher eine den Eltern in vielen Merkmalen überlegene Pflanze bewirkt, sowie die Uniformität dieser ersten Hybridgeneration zu nutzen. Da jedes Jahr tausende neue Inzuchtlinien erzeugt werden ist die Anzahl der Kreuzungskombinationen nicht in Feldversuchen zu testen, die optimalen Kreuzungspartner werden daher über markerbasierte Vorhersagemethoden bestimmt.
In unserer Arbeit werden kurze RNA (sRNA), deren Profile in Keimlingen von Inzuchtlinien über Tiefensequenzierungen bestimmt werden mit Hybridmerkmalen assoziiert. Hierfür wurde ein Programm geschrieben, welches die Signifikanz der Assoziation für jede einzelne sRNA bestimmt und diese Wahrscheinlichkeit korrigiert. Dieses Programm erhält sowohl die sRNA-Expressionsprofile als auch die bekannten Hybridmerkmalswerte der möglichen Inzuchtlinienkombinationen und liefert als Ergebnis eine Auflistung der assoziierten sRNA sowie deren Wahrscheinlichkeit. Diese Information wird zum einen für weitergehende Analysen zur Charakterisierung dieser Sequenzen zur Identifikation des Beitrags von sRNA zur Entstehung von Heterosis, als auch für eine markerbasierte Vorhersage von Hybridmerkmalen herangezogen und das Verfahren patentiert. In dem gegenwärtigen Projekt wird dieses Programm nach Bestimmung der optimalen Parameter einmalig zur Assoziation und anschließend für eine Bewertung der Vorhersagegenauigkeit über eine Kreuzvalidierung genutzt.