Fraunhofer Einrichtung Experimentelles Software Engineering

Seminar

“Werkzeuggestützte Modellierung des Tamagotchi”

Modellierung mittels SCR

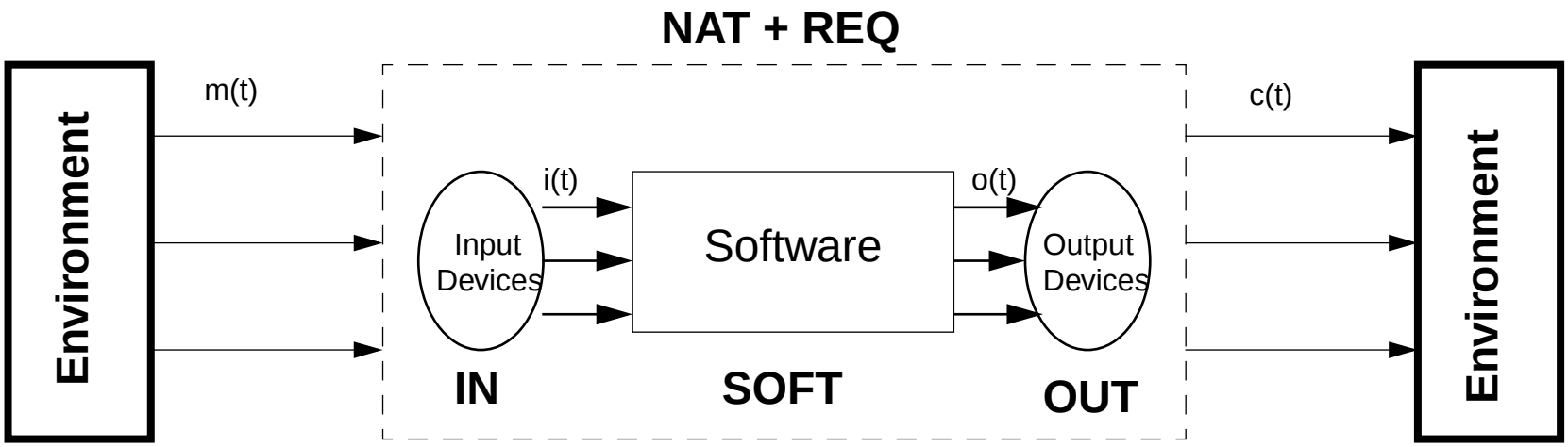
Ausgearbeitet von: Kizito Ssamula Mukasa und Ralf Hettesheimer

Einleitung

Charakterisierung der Technik

- Entwickelt 1978 am Naval Research Laboratory der US Navy
- Formale Beschreibungstechnik für System- und Softwarespezifikation
- Beschreibung von Verhaltensanforderungen basiert auf endlichem Automatenmodell
- Verhalten wird in Beziehung zur Systemumwelt als Black-Box beschrieben
- Tabellarische Repräsentation der Anforderungen
- Basiert auf dem Vier-Variablen-Modell von Parnas
- Senkung der Software-Entwicklungskosten durch einfache Modellierung
- Gedacht für reaktive eingebettete Echzeitsysteme

Einleitung



Das Vier-Variablen-Modell von Parnas

Aktuelle Version

- Tool ist Forschungsprototyp
- Einige Teilfunktionen noch nicht implementiert

Einleitung

Abstraktion der Umwelt

- Vereinfachtes Vier-Variablen-Modell
- Notation bietet drei Arten von Variablen:
  - Monitored
  - Controlled
  - Term
- Terme sind global (keine Kapselung)

Interne Modellierung

- Mode-Klassen (Modes)
- Terme

Einleitung

- Bedingungen (z.B. Knopf=gedrückt AND Druck < 50)
- Ereignisse (z.B. @T(Knopf=gedrückt))
- Tabellen
  - Ereignistabellen
  - Mode Transitionstabellen
  - Bedingungstabellen

Methodik

- Methodik nur sehr grob vorgegeben (Identifizierung von Monitored- und Controlled-Variablen)
- Abbildung der informellen Anforderungen in Modell nicht definiert

Einleitung

Features des Tools

- Tools bietet umfangreichen Konsistenz-Checker. Beispielsweise Prüfung auf Eindeutigkeit der Zustandsübergänge und Variablenzuweisungen
- Dependency Graph Browser:
  - Darstellung der Abhängigkeiten sämtlicher Variablen und Modeclassas
  - Darstellung von Zyklen in der Spezifikation
  - Vereinfachte Navigation durch die Spezifikation
- Simulator
  - Simulation durch direkte Manipulation der Monitored-Variablen
  - Darstellung aller vorhandenen Variablen und States der einzelnen Modeclasses
  - ModelChecker anschließbar (SPIN)

Modellierung im Team

Aufteilung

- Aufteilung in Backend (Entwicklung des Tamagotchi, Zeitgesteuertes Verhalten etc.) und Frontend (Interaktion, Hauptmenü etc.)
- Aufteilung erschien sinnvoll, da die beiden Teile nur an wenigen Stellen zusammenhängen (wenig Schnittstellen zu besprechen)
- Unterstützung durch das Tool ist nicht besonders gut.
  - Durch Term-Variablen zusammenhängende Modeclasses lassen sich nur schwierig getrennt entwickeln
  - Da alle Terme global sind, können beim zusammenfügen leicht mehrfache Deklarationen entstehen
  - Kein Import von Teil-Spezifikationen möglich (Neueingabe)
- Simulation der beiden Teile war schon während des Entwurfs möglich

Modellierung im Team

Umfang

- Die erstellte Spezifikation umfasst *fast* die gesamte Liste der informellen Anforderungen
- Teilweise Tricks nötig, da das Tool keine Timer oder Zufallsgeneratoren enthält
- Die Spezifikation umfaßt 71 Tabellen.

Modellierung des Spiels

- ❑ Spielablauf BA-F-23 - 25
- Tamagotchi schaut abwechselnd nach rechts und links.
- Benutzer wählt rechts oder links.
- Tamagotchi wählt zufällig rechts oder links.
- stimmen die beiden Richtungen überein, ist die Runde gewonnen.
- Ein Spiel besteht aus fünf Runden. Sind mehr als zwei Runden gewonnen, steigt die Glücklichkeit um eine Einheit.
- Pro Spiel nimmt Tamagotchi 1 oz ab.
- Ein Spiel wird mit der R-Taste abgebrochen, oder wenn Tamagotchi ins Bett muß.
- Sonst erfolgt nach jedem Spiel ein weiteres.

Modellierung des Spiels

❑ Das Modell

- Das ganze ist Event-gesteuert.
- Ein Event ist entweder @T(Ausdruck), @F(Ausdruck), @C(Ausdruck) oder @A
- Es gibt zwei Spielzustände:

  Zustand “saPlaying” ist für das tatsächliche Spielen.

  Zustand “saReset” ist für das rücksetzen der Variablen.
- Ein Wechsel von einem Zustand zum anderen erfolgt, wenn ein entsprechendes Event eintritt.
- Die entsprechenden Events werden in einer Zustandsübergangstabelle festgelegt.

  (s. Tabelle Nr. 35)

Modellierung des Spiels

- Für die Ausdrücke braucht man Variablen.
- Diese können Monitored (M), Controlled (C) oder Term (T).
- Variablen werden in einer Tabelle deklariert.

Table 1:
Table 2: Monitored Variable Dictionary(Abschnitt 1 von 2)

Name	Type	Initial Value	Accuracy	Comment
m_cardboardswitch	switch	Yes		Pappstreifen (Yes=drinnen, No=draussen)
m_critical	Boolean	FALSE		
m_development	state	dev_initial		
m_exit	switch	No		
m_flying	Boolean	FALSE		
m_happy	Integer	4		
m_hunger	Integer	4		

Table 1:

Table 2: Monitored Variable Dictionary(Abschnitt 2 von 2)

Name	Type	Initial Value	Accuracy	Comment
m_lbutton	button	notpressed		linker Button
m_lifeexpect	Integer	30		
m_mametchi	Boolean	TRUE		controlled/monitored-Verfahren
m_mbutton	button	notpressed		
m_randexpect	Float	1.0		Zufallswert fuer Lebenserwartung (in Prozent)
m_rbutton	button	notpressed		
m_rhappytimer	Integer	0		Externer Timer
m_rhungertimer	Integer	0		Externer Timer
m_sleep	Integer	1500		
m_tamdirection	direction	middle		
m_timeregghatch	Integer	0		Externer Timer zum Eiausbrueten
m_wakeup	Integer	1500		
time	Integer	0		Globale Zeit

Modellierung des Spiels

- Den Wert einer M-Variablen muß der Benutzer selbst setzen.
- C- und T- werden intern gesetzt . Allerdings müssen die Regeln vorher beschrieben werden.
- Für jede C- oder T- Variable gibt es genau eine Event- oder Conditiontabelle, die die Regel für den Wert der entsprechenden Variablen festlegt, und zwar in allen möglichen Zuständen einer Modeclass.

Table 3: Condition Table for c_starttimeregghatch

Name		Mode Class
c_starttimeregghatch		development
Modes	**Conditions**	
dev_egg	TRUE	FALSE
dev_initial, dev_babytchi, dev_marutchi, dev_tamatchi, dev_kuchitamatchi, dev_mametchi, dev_kuchipatchi, dev_masktchi	FALSE	TRUE
c_starttimeregghatch =	TRUE	FALSE

Modellierung des Spiels

Table 4: Event Table for c_scores

Name		Mode Class	
c_scores		selectedAction	
Modes	**Events**		
saPlaying	@T(m_tamdirection = left) WHEN (m_lbutton = pressed)	@T(m_tamdirection = right) WHEN (m_mbutton = pressed)	NEVER
saReset	NEVER	NEVER	@T(m_mbutton = pressed OR m_lbutton = pressed OR m_rbutton = pressed)
saSleep,saNoAction,saGames,saFood,saSnacks,saFoodAndSnacks,saInformations,saHunger,saHappyness,saWeightAndAge,saTime	NEVER	NEVER	@A
c_scores' =	c_scores + 1	c_scores + 1	0

Erfahrungen

Einarbeitung

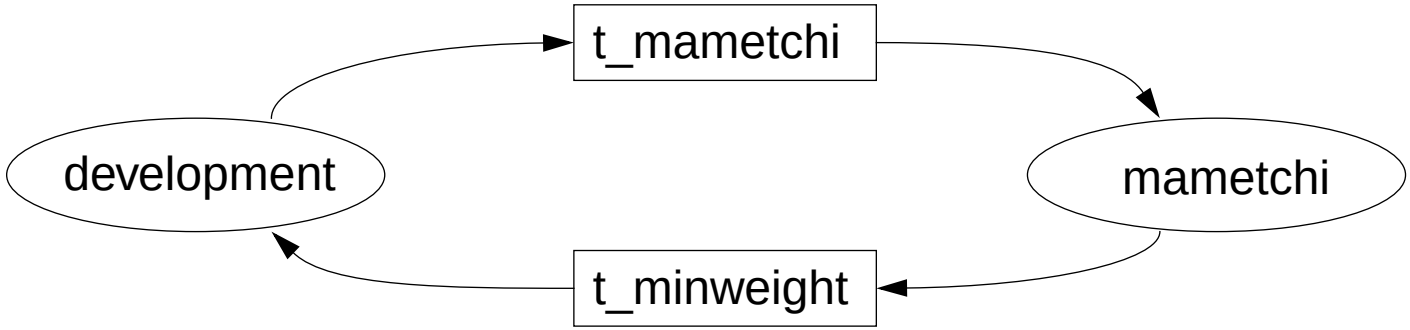
- Aufwand: ca. 40 Stunden
  - Lesen des Tool Guides / Kennenlernen des Tools
  - Modellierung des Safety Injection Device
- Erlernbarkeit
  - Die Strukturen sind nicht besonders umfangreich -> Leichte Erlernbarkeit
  - Teile der Notation sind im Tool-Guide nicht beschrieben (Duration-Funktionalität zur Zeitüberwachung, @A-Event)

Modellierung des Tamagotchi

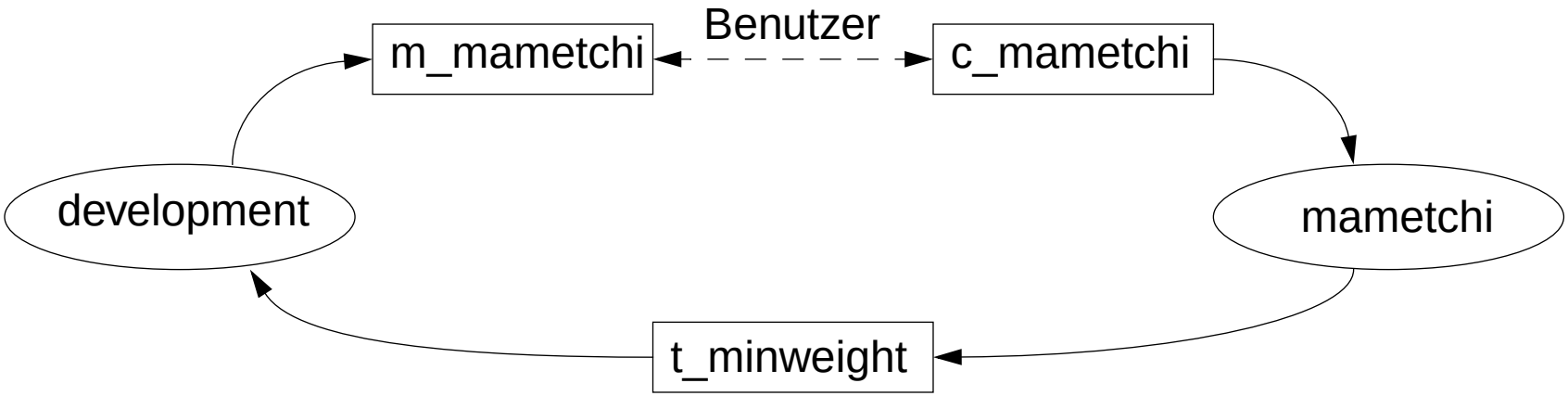
- Aufwand: ca. 100 Stunden

Erfahrungen

- Das Zyklus-Problem



- Lösung durch Auftrennung



Erfahrungen

- Wir haben unabhängig voneinander in unseren ersten Modellen (auf Papier) solche Zyklen eingebaut
- Zwei Herangehensweisen
  - Von vornherein sämtliche koppelnden Variablen auftrennen (Nachteil: sehr aufwendige Spezifikation)
  - Modell mit Zyklen eingeben und anschließend auftrennen
- Exklusiver Zugriff auf Term-Variablen
  - Term-Variablen nur von einer Modeclass beeinflußbar
  - Unser Problem: Frontend und Backend müssen beide Gewicht, Sättigung und Glücklichkeit beeinflußen.
  - Lösung: Offsets, die von einer dritten Modeclass zu den endgültigen Werten verarbeitet werden

Erfahrungen

Review / Simulation

- Im Vergleich zur reviewten Spezifikation (UML / Rhapsody) unübersichtlich
- Simulation kompliziert, Benutzer muß sich um viele Dinge kümmern (Zyklus-Problem mit Auftrennung)
- Grafische Darstellung der Zusammenhänge in Rhapsody wesentlich besser
- In unserer Spezifikation nur mit *Tricks* lösbare Probleme in Rhapsody einfacher umsetzbar (Timer, Zufallsgenerator)

Abschließende Bewertung

- Aufteilung der Spezifikation gut möglich
- Schwierigkeiten beim Zusammenfügen der beiden Teile
- Das Tool ist zur Modellierung kleinerer Systeme (Safety Injection Device) gut geeignet
- Modellierung größerer Systeme (Tamagotchi) wird aufgrund der vorher beschriebenen Schwächen unnötig kompliziert und unübersichtlich
- Das Tool sollte im Hinblick auf Import und Simulation von Teilspezifikationen erweitert werden
- Austausch von Teilkomponenten schwierig
- Der Tool-Guide verdient eine Überarbeitung mit Aufnahme der bereits implementierten Notation
- Modellierung des Tamagotchi war sinnvoll, zum besseren Verständnis der Abhängigkeiten des Systems.

Abschließende Bewertung

- Erstellte Spezifikation nicht besonders geeignet für Umsetzung in lauffähiges Produkt
- Eine erneute Modellierung mittels SCR würde schneller ein besseres Ergebnis bringen (Erfahrung aus der erstellten Spezifikation)
- Für kleine Systeme oder bei keiner Verfügbarkeit eines anderen Tools würden wir das Tool erneut verwenden
- Für größere Systeme wäre ein Tool mit einer größeren Nähe zu einer späteren Implementierung vorzuziehen

Anhang

Table 1: Mode Transition Table for selectedAction (Sheet 1 of 2)

Source Mode	Events	Destination Mode
saNoAction	@T(m_lbutton = pressed)	saFoodAndSnacks
saNoAction	@T(t_awake = FALSE)	saSleep
saFoodAndSnacks	@T(m_mbutton = pressed)	saFood
saFoodAndSnacks	@T(t_awake = FALSE)	saSleep
saFoodAndSnacks	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saFood	@T(m_mbutton = pressed OR m_rbutton = pressed OR m_exit = Yes)	saNoAction
saFood	@T(t_awake = FALSE)	saSleep
saFood	@T(m_lbutton = pressed)	saSnacks
saSnacks	@T(m_mbutton = pressed OR m_rbutton = pressed OR m_exit = Yes)	saNoAction
saSnacks	@T(t_awake = FALSE)	saSleep
saFoodAndSnacks	@T(m_lbutton = pressed)	saGames
saGames	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saGames	@T(t_awake = FALSE)	saSleep
saGames	@T(m_mbutton = pressed)	saPlaying
saPlaying	@T((m_mbutton = pressed OR m_lbutton = pressed) AND (c_playedrounds < 5))	saPlaying
saPlaying	@T(m_rbutton = pressed)	saNoAction
saPlaying	@T(t_awake = FALSE)	saSleep
saGames	@T(m_lbutton = pressed)	saInformations
saInformations	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saInformations	@T(t_awake = FALSE)	saSleep

Anhang

Table 1: Mode Transition Table for selectedAction (Sheet 2 of 2)

Source Mode	Events	Destination Mode
saInformations	@T(m_mbutton = pressed)	saWeightAndAge
saWeightAndAge	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saWeightAndAge	@T(t_awake = FALSE)	saSleep
saWeightAndAge	@T(m_mbutton = pressed)	saHunger
saHunger	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saHunger	@T(t_awake = FALSE)	saSleep
saHunger	@T(m_mbutton = pressed)	saHappyness
saHappyness	@T(m_rbutton = pressed OR m_exit = Yes)	saNoAction
saHappyness	@T(t_awake = FALSE)	saSleep
saSleep	@F(t_awake = FALSE)	saNoAction
saPlaying	@T(c_playedrounds >= 5 AND (m_mbutton = pressed OR m_lbutton = pressed))	saReset
saReset	@T(m_lbutton = pressed or m_mbutton = pressed)	saPlaying
saReset	@T(t_awake = FALSE)	saSleep
saReset	@T(m_rbutton = pressed)	saNoAction
saNoAction	@T(m_mbutton = pressed)	saTime
saTime	@T(m_exit = Yes)	saNoAction
saTime	@T(t_awake = FALSE)	saSleep