Fraunhofer

Einrichtung
Experimentelles
Software Engineering

Tamagotchi-Spezifikation mit OCTOPUS

Gerhard Landeck, Elena Torres

Uberblick

1. Charakterisierung
2. Prozeßmodell
3. Vorgehensweise
4. Das Werkzeug: StP/OMT und UseCase-Diagrams
5. Ausschnitt aus der Spezifikation.
6. Erfahrungen
7. Abschließende Bewertung

Charakterisierung von OCTOPUS (1)

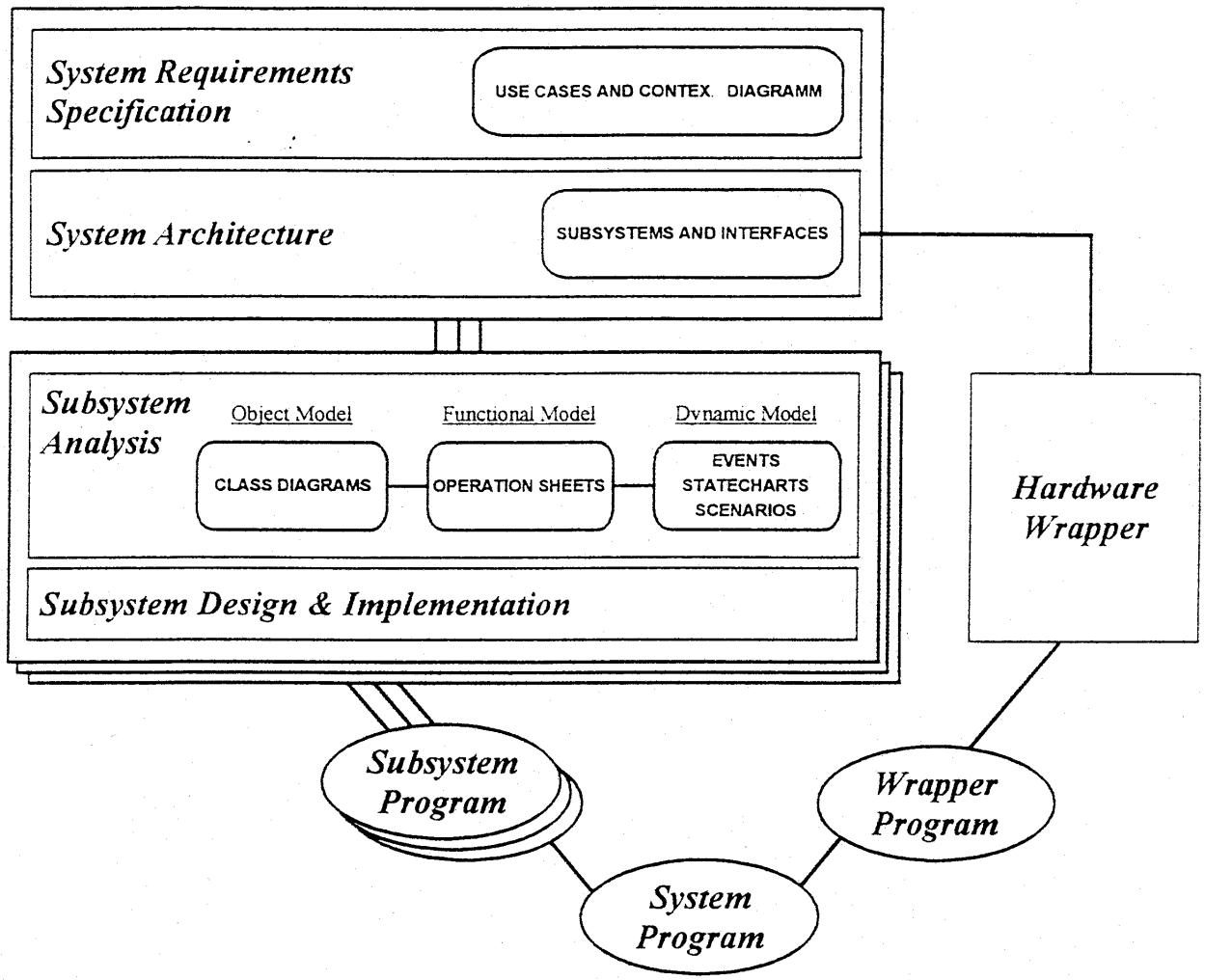
- Autoren: Maher Awad, Juha Kuusela, Jürgen Ziegier ('96) von Software Technology Laboratory in Nokia Research Center
- Anwendungsbereich: Telekommunikationsindustrie
- Basiert auf OMT, von Rumbaugh ('91), Fusion von Coleman ('93) und andere Features
- Methode zur Systementwicklung verwenden:
  - Konzeptuelle Modelle, 00-Modellierung
  - Verhalten: Kommunikationsaspekte

Charakterisierung von OCTOPUS (2)

Objektorientierte Technologie für eingebettete Realzeit-Systeme

- OMT -> 00-Technologie:
  - Notation der Objektmodelle
  - Trennung in strukturelle, funktionale und dynamische Aspekte
- Fusion -> Eingebettete Realzeit-Systeme
  - Analyse-Phase externes Verhalten
  - Design-Phaseinternes Verhalten

Prozeßmodell von Octopus



OCTOPUS - Vorgehensweise (1)

Große Systeme werden in fünf Phasen geteilt:

1. Spezifikation von Systemanforderungen

1.1 Definition von Use Cases --> funktionales und dynamisches Modell

1.2 Erstellung von System Context Diagram --> Objektmodell

2. Systemarchitektur

3. Subsystem-Analyse
    - Schnittstellen klar definieren
    - Systemarchitektur verifizieren

OCTOPUS - Vorgehensweise (2)

3.1 Strukturellesmodell -> orientiert an OMT Methode:

Object diagrams, class description table und Class Diagrams

3.2 Funktionales Modell -> funktionelle Schnittstellen der Subsysteme

Operation Sheets, informelle Beschreibung von Operationen

3.3 Dynamisches Modell -> Operationen des Subsystems unter Berücksichtigung von Realzeit und reaktiven Aspekten

- Ereignis-Analyse: event list, event diagrams, event sheets
- Zustands-Analyse: concurrent statecharts, action table

Das Werkzeug: StP/OMT und UseCasesDiagrams

Software through Pictures / Object Modeling Technique, Version 3.4 - Aonix 1997

1. OMT modelliert durch Diagramme

→ Objektmodell

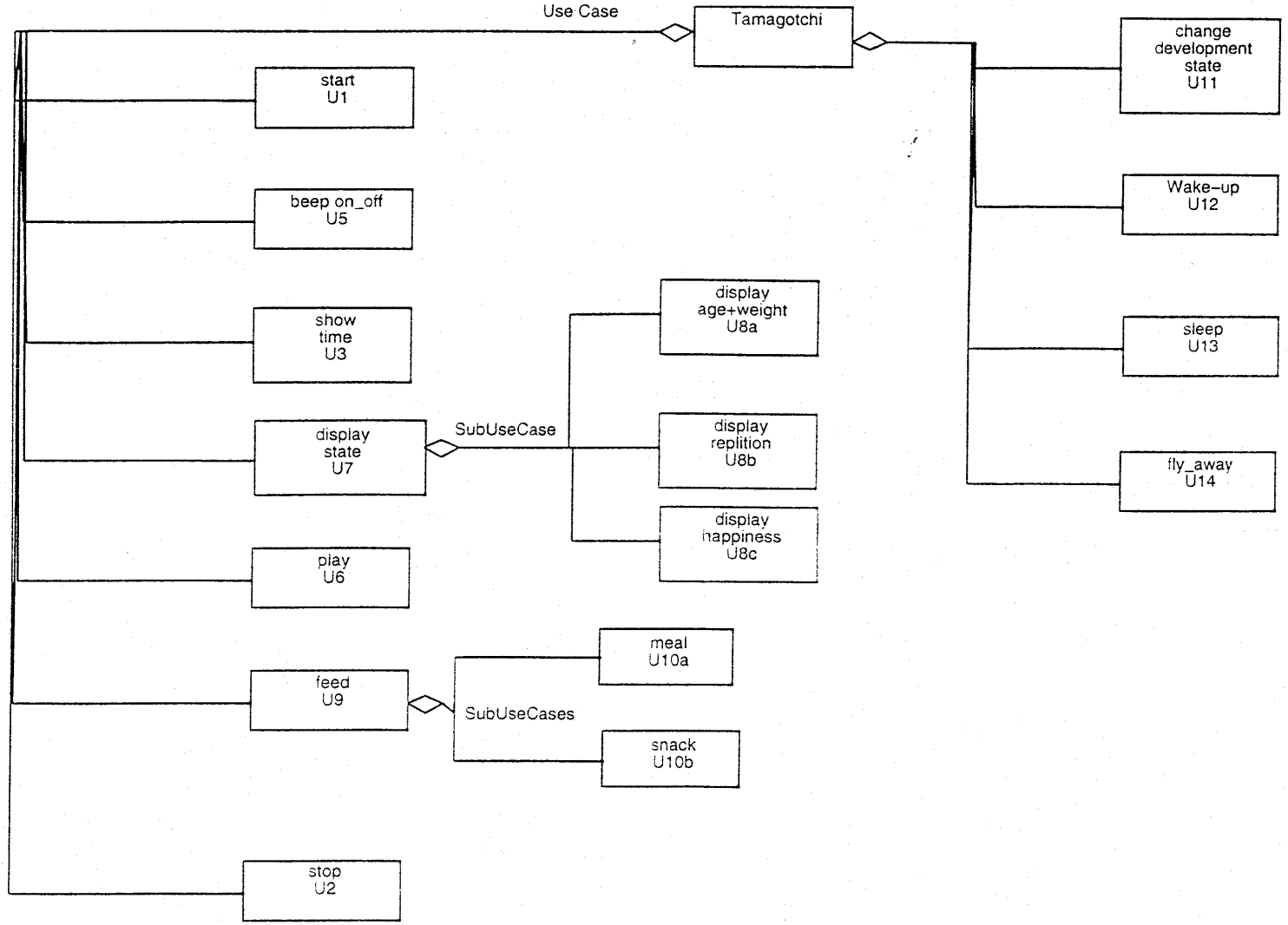
→ Dynamisches Modell

→ Funktionales Modell

2. Use Cases Diagrams.

→ Übersicht von B-Anforderungen

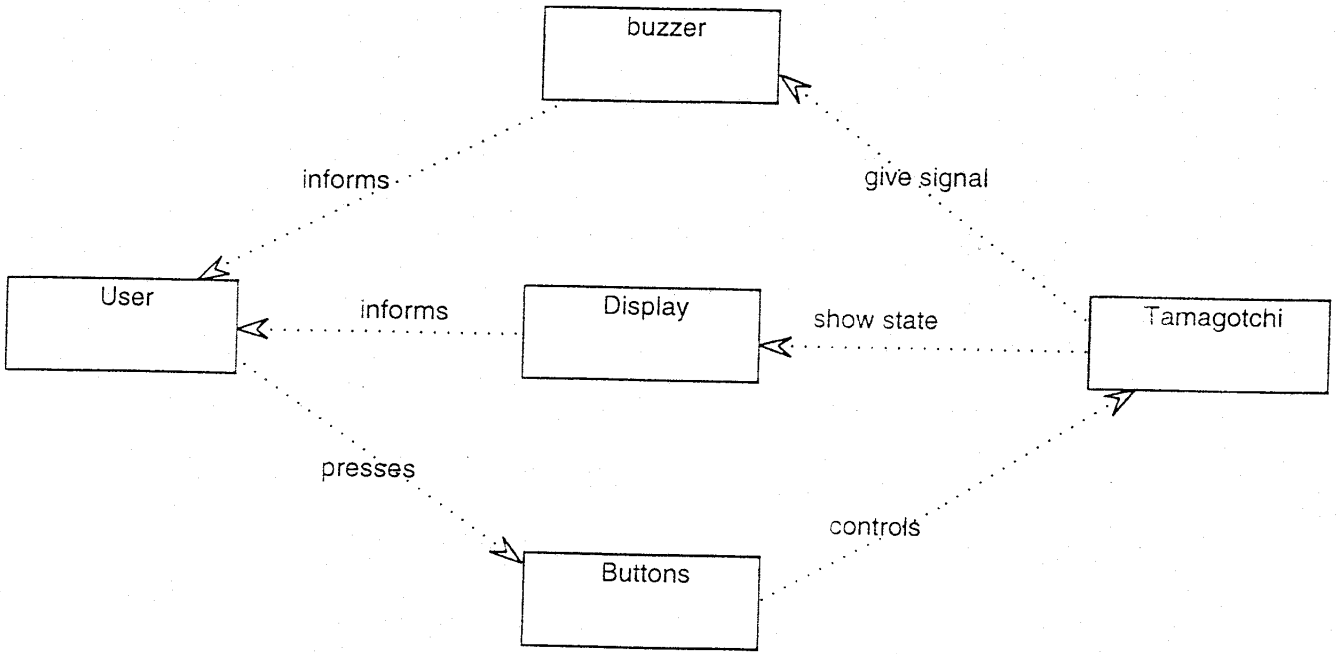
Use Case Diagramm fuer Tamagotchi



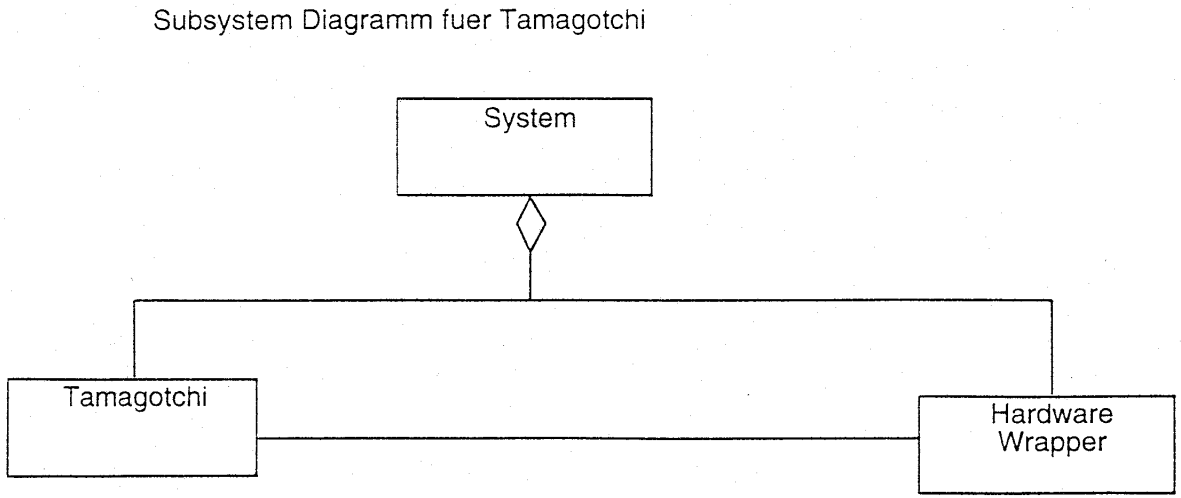
Ausschnitt aus Use Case Sheets

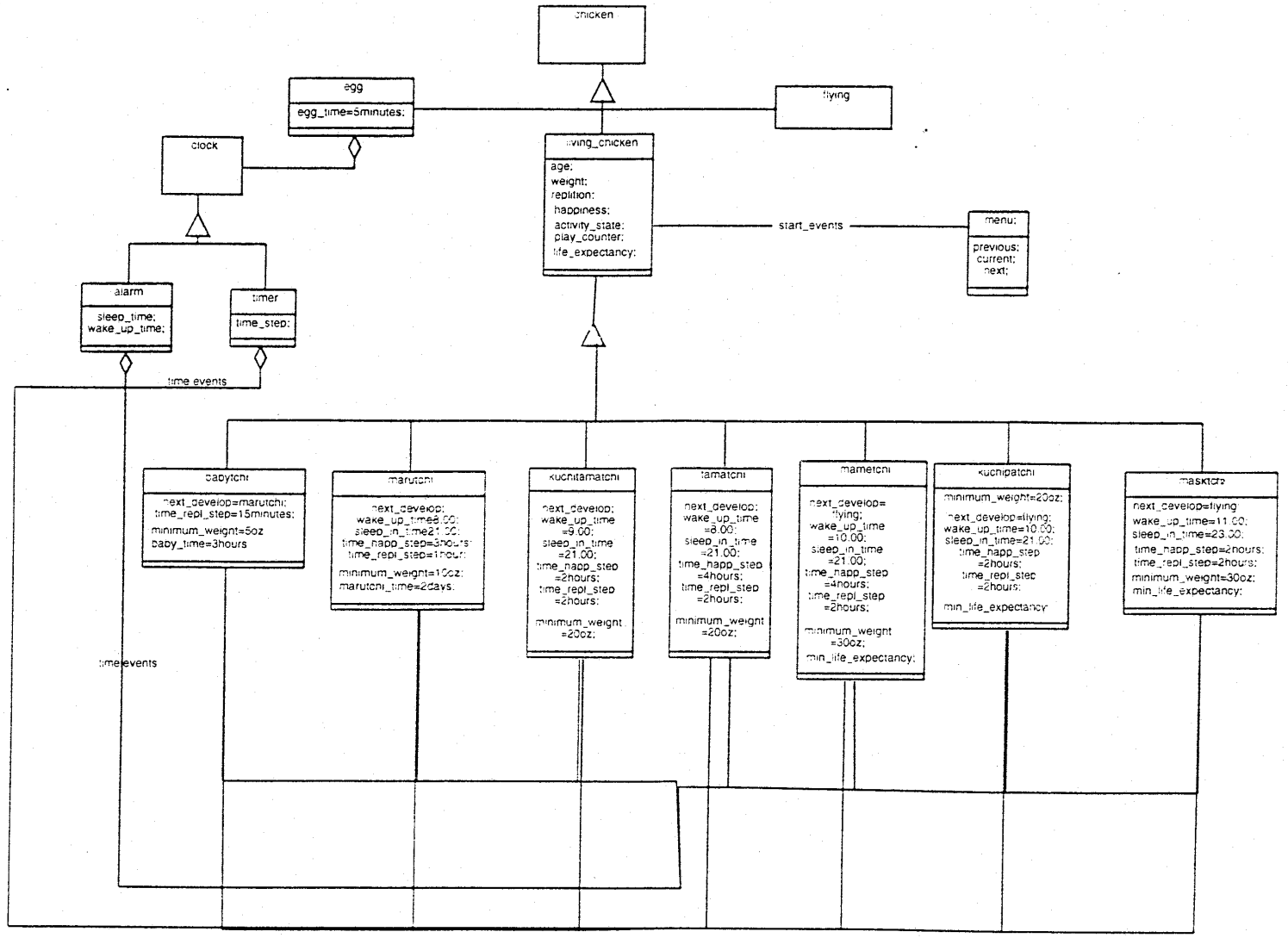
Use Case (U6)	**play**
Actor	Benutzer
Preconditions	Das Tamagotchi ist wach und befindet sich nicht , im Zustand Ei oder fliegendes T.
Description	Das Display zeigt ein spielendes Küken des aktuellen Entwicklungsstadiums. der Piezosummer piept penodisch. 5 mal wählt der Benutzer mit der linken oder mittleren Taste eine Richtung aus, das Küken ebenso per Zufallsgenerator. Stimmt die Richtung überein, zeigt das Display ein lachendes, sonst ein weinendes Küken. Nach 5 Runden wird der Spielstand angezeigt.
Sub Use Cases	---
Exceptions	end (rechteTaste), stop (Streifen rein), sleep_in (Einschlafzeitpunkt erreicht);
Activities	Change weight und falls das Spiel gewonnen wurde change happiness.
Postconditions	Das Display zeigt ein Küken des aktuellen Entwicklungsstadiums

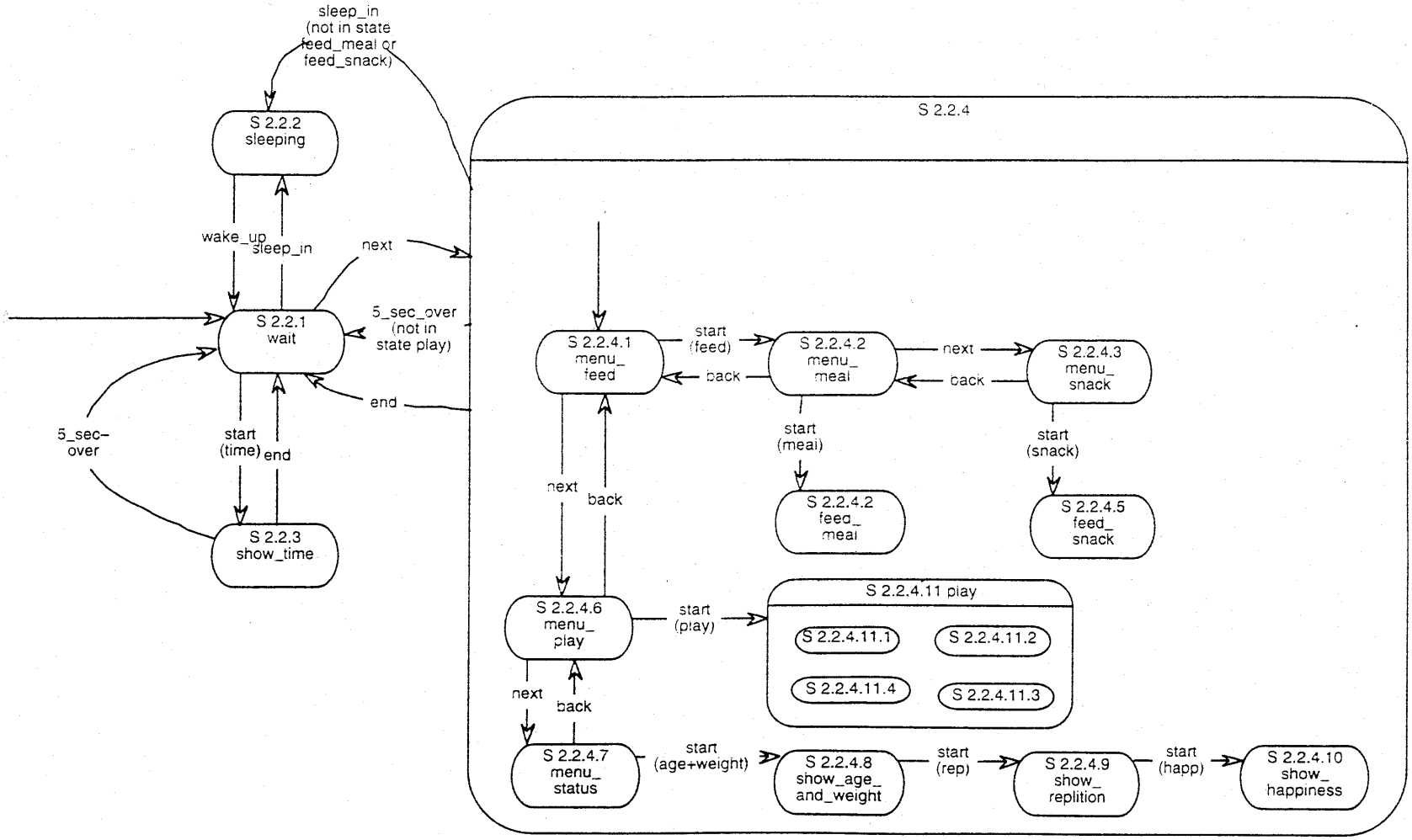
System Context Diagram für Tamagotchi



Subsystem Diagramm für Tamagotchi




chicken
egg
egg_time=5minutes;
flying
clock
living_chicken
age;
weight;
repition;
happiness;
activity_state;
play_counter;
life_expectancy;
start_events
menu;
previous;
current;
next;
alarm
sleep_time;
wake_up_time;
timer
time_step;
time events
babytchi
next_develop=marutchi;
time_repi_step=15minutes;
minimum_weight=5oz
baby_time=3hours
marutchi
next_develop;
wake_up_time6.00;
sleep_in_time21.00;
time_happ_step=3hours;
time_repi_step=1hour;
minimum_weight=10oz;
marutchi_time=2days;
kuchitamatchi
next_develop;
wake_up_time
=9.00;
sleep_in_time
=21.00;
time_happ_step
=2hours;
time_repi_step
=2hours;
minimum_weight
=20oz;
tamatchi
next_develop;
wake_up_time
=8.00;
sleep_in_time
=21.00;
time_happ_step
=4hours;
time_repi_step
=2hours;
minimum_weight
=20oz;
mametchi
next_develop=
flying;
wake_up_time
=10.00;
sleep_in_time
=21.00;
time_happ_step
=4hours;
time_repi_step
=2hours;
minimum_weight
=30oz;
min_life_expectancy;
kuchipatchi
minimum_weight=20oz;
next_develop=flying;
wake_up_time=10.00;
sleep_in_time=21.00;
time_happ_step
=2hours;
time_repi_step
=2hours;
min_life_expectancy
masktchi
next_develop=flying;
wake_up_time=11.00;
sleep_in_time=23.00;
time_happ_step=2hours;
time_repi_step=2hours;
minimum_weight=30oz;
min_life_expectancy;
time events


sleep_in (not in state feed_meal or feed_snack)
S 2.2.2 sleeping
wake_up
sleep_in
next
S 2.2.1 wait
5_sec_over (not in state play)
end
5_sec_over
start (time)
end
S 2.2.3 show_time
S 2.2.4
S 2.2.4.1 menu_ feed
start (feed)
back
S 2.2.4.2 menu_ meal
next
back
S 2.2.4.3 menu_ snack
start (meal)
start (snack)
S 2.2.4.2 feed_ meal
S 2.2.4.5 feed_ snack
next
back
S 2.2.4.6 menu_ play
start (play)
S 2.2.4.11 play
S 2.2.4.11.1
S 2.2.4.11.2
S 2.2.4.11.4
S 2.2.4.11.3
next
back
S 2.2.4.7 menu_ status
start (age+weight)
S 2.2.4.8 show_age_ and_weight
start (rep)
S 2.2.4.9 show_ replition
start (happ)
S 2.2.4.10 show_ happiness

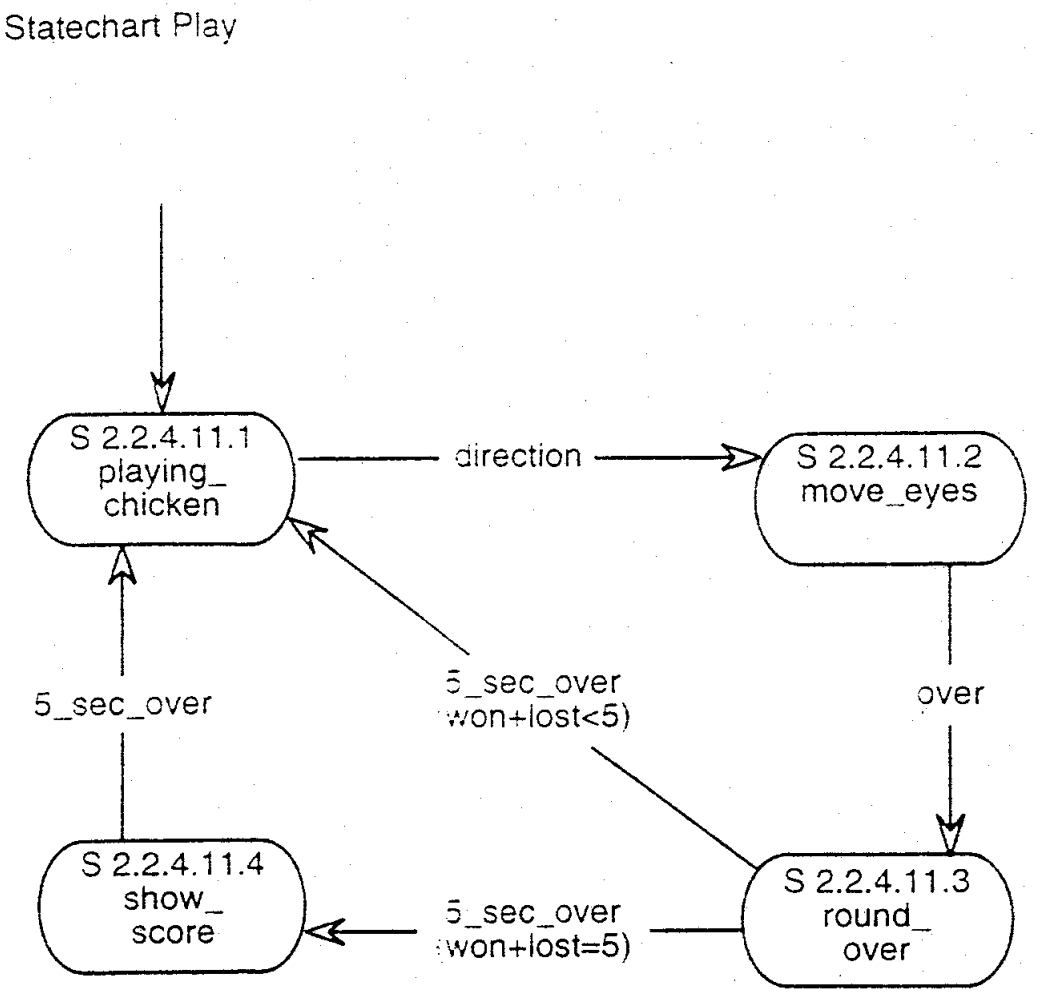
Ausschnitt aus Event Sheets (dynamisches Modell)

Event	**next**
Response	Display zeigt nächsten Mentipunkt;
Associations	buttons, menu;
Source	linker Knopf wurde gedrückt;
Contents	---
Response Time	sofort;
Raw	unregelmäßig;

Ausschnitt aus Operation Sheets (funktionales Modell)

O7	**show_next**
Description	Auf dern Display wird der nächste Menüpunk angezeigt;
Associations	menu, aktuelle Unterklasse von living_chicken
Preconditions	Das Tamagotchi ist wach, die linke Taste wurde gedrückt;
Inputs	linke Taste (vom Hardware wrapper);
Modifies	die Ausgabe des Displays, den Zustand des Menüs;
Outputs	Display-Ausgabe;
Postconditions	Display zeigt neuen Menüpunkt für 5 Sekunden;

Statechart Play



Ausschnitt aus Actions table (dynamisches Modell)

Elementary state	Event	Actions/activities
S 2.2.1 wait	<enter>	Display zeigt waches Küken entsprechenden Entwicklungsstadiums
	<wake_up>	Operation 05; Spielezähler:= 0; Gewicht wird um entwicklungsabhängigen Wert verringert;
S 2.2.4.1 menu feed	<enter>	Display zeigt "Nahrung und Süßigk."
S 2.2.4.6 menu_play	<enter>	Display zeigt"Spielen"
S 2.2.4.11.1 playing_chicken	<enter>	Display zeigt ein spielendes Küken des entsprechenden Stadium .
S 2.2.4.11.2 move_eyes	<enter>	T. wählt Richtung, auf Display bewegen sich die Augen, T. vergleicht, ändert Spielstand;
S 2.2.4.11.3 round_over	<enter>	Display zeigt lachendes oder weinendes Küken, zählt gewonnene und verlorene Runden zusammen
S 2.2.4.11.4 show score	<enter>	T. zeigt den Spielstand an
	<exit>	T. erhöht Spielezähler

Erfahrungen - Einarbeitung

- Hoher Zeitaufwand für Anfänger
  - Darin enthalten: Buch lesen, Methode verstehen, Tool-Einarbeitung
  - Beispiele im Buch manchmal unübersichtlich
- Aufwand ca. 8h pro Woche
  - Benutzeranforderungen: Software beschrieben, Hardware fehlt

Probleme

- Tool bezieht sich auf OMT, nicht OCTOPUS
- Beschriebene Vorgehensweise schwierig
- Tiefe der Use Cases
- Unterscheidung Events - Operationen
- Aufteilung der Subsysteme und Klassen
- Analysephase: besser zuerst dynamisches Modell

Erfahrungen - Review/Simulation

- Simulation in Analysephase nicht möglich
- Überblick über Spezifikation ohne Kenntnis der Methode schwer
- Abbildung zwischen Benutzeranforderungen und Diagrammen 1:n
- Überprüfen der Spezifikation nur von Hand möglich
- Review: Spezifikation nur dynamisch erscheint auf ersten Blick einfacher

Abschliessende Bewertung - Teamarbeit, Stärken, Schwächen

- Parallelisierung bei dieser Einteilung schwer möglich
- Stärken von OCTOPUS
  - Praktisch alles modellierbar
  - Reaktive und Realzeit-Aspekte gut darstellbar
- Schwächen
  - Detaillierungsgrad nicht scharf umrissen
  - Viele Tabellen und Diagramme erschweren Übersicht