MoDRE '18
Modelling and Testing Requirements via Executable Abstract State Machines
Research Contributions

- Metamodel: Language for Specifying User Interfaces
- Mathmodels: Language for Specifying Business Logic as Abstract State Machines
- Architecturally Structured, Executable Code

Requirements Document

EHealth UI grammar

EHealth acceptance tests

Requirements Engineers

Customers

dependency

data flow
system ehealth
-- semantics types
type MEDICATION = STRING
type PATIENT = STRING
-- events
add_patient (p: PATIENT)
add_medication (m: MEDICATION)
add_interaction (m1: MEDICATION; m2: MEDICATION)
add_prescription (p: PATIENT; m: MEDICATION)
remove_interaction (m1: MEDICATION; m2: MEDICATION)
remove_prescription (p: PATIENT; m: MEDICATION)

patients \in \mathbb{P} PATIENT
medications \in \mathbb{P} MEDICATION
interactions \in MEDICATION \leftrightarrow MEDICATION
prescriptions \in PATIENT \leftrightarrow MEDICATION

Acceptance Test

... state 16
patients: \{p_1, p_2, p_3\}
medications: \{m_1, m_2, m_3, m_4\}
interactions: \{m_1 \rightarrow m_2, m_2 \rightarrow m_1, m_2 \rightarrow m_4, m_4 \rightarrow m_2\}
prescriptions: \{p_1 \rightarrow m_1, m_3; p_3 \rightarrow m_2\}
->add_prescription("p_3","m_4")

state 17 Error e_4: this prescription dangerous
->remove_interaction("m_2","m_4")

state 18
patients: \{p_1, p_2, p_3\}
medications: \{m_1, m_2, m_3, m_4\}
interactions: \{m_1 \rightarrow m_2, m_2 \rightarrow m_1\}
prescriptions: \{p_1 \rightarrow m_1, m_3; p_3 \rightarrow m_2\}
->add_prescription("p_3","m_4")

state 19
patients: \{p_1, p_2, p_3\}
medications: \{m_1, m_2, m_3, m_4\}
interactions: \{m_1 \rightarrow m_2, m_2 \rightarrow m_1\}
prescriptions: \{p_1 \rightarrow m_1, m_3; p_3 \rightarrow m_2, m_4\}
If one *medication* interacts with another, then the reverse also applies (Symmetry).

A medication does not interact with itself (Irreflexivity).

The system maintains records of *patient prescriptions*. No prescription may have a *dangerous interaction*.

```plaintext
class HEALTH_SYSTEM
feature -- abstract state
patients: SET [PATIENT]
medications: SET [MEDICATION]
prescriptions: REL [PATIENT, MEDICATION]
interactions: SET [INTERACTION]
invariant
  symmetry_ENV3:
    across medications as m1 all
    across medications as m2 all
      interactions.has ( [m1.item, m2.item] ) = interactions.has ( [m2.item, m1.item] )
  end end
irreflexivity_ENV4:
  across medications as m1 all not interactions.has ( [m1.item, m1.item] ) end
no_dangerous_interactions_REQ6:
  across prescriptions.domain as p all
  across prescriptions [p.item] as m1 all
  across prescriptions [p.item] as m2 all
  interactions.has ( [m1.item, m2.item] )
    implies not( prescriptions.has( [p.item,m1.item] ) and prescriptions.has( [p.item,m2.item] ) )
  end end end
consistent domain:
  prescriptions.domain ⊆ patients
end
```
Physicians shall be allowed to add a medication to a patient's prescription, provided it does not result in a dangerous interaction.

```matlab
class
  ADD_PRESCRIPTION
inherit
  HEALTH_SYSTEM -- inherits all system invariants
feature -- commands
  add_prescription (p: PATIENT; m: MEDICATION)
    -- Add a prescription of 'm1' to 'p1'.
    require
      -- p ∈ patients
      patients.has (p)
      -- m ∉ prescriptions[p]
      not prescriptions[p].has (m)
      -- cannot cause a dangerous interaction
      ∀ med ∈ prescriptions[p] : (med, m) ∉ interaction
      across prescriptions[p] as med all not interactions.has([med.item, m]) end
    do
      prescriptions.extend([p, m])
    ensure
      prescriptions ~ old prescriptions + [p, m]
      -- UNCHANGED (patients, medications, interactions)
  end
end
```