Requirement Traceability: A Model-Based Approach
Traceability in Software Engineering

- Tractability is about:
  - The ability to **interrelate** any uniquely identifiable artifact to any other;
  - To **maintain** links over time; and
  - To **use** the resulting network to answer questions of both the software product and its development process.
Why Trace Requirements

- Verify Coverage
- Avoid Redundancy
- Assess Impact of Changes
- Requirements Traces must be
  - maintained as the system evolves
  - Available at run time (maybe?)
  - Accessible at variable levels of abstractions
Current Practice

- Requirements activities are **separate** from development activities
- Links between requirements and development artifacts **must be maintained**
- MDA Approach
  - Focus on system entities and behaviour
  - **Less emphasis** on system goals, actors, and non functional requirements.
Model Oriented Programming

```java
class Person {
}

class Student {
    isA Person;
    Integer stNum;
    status {
        Applied {
            quit -> Quit;
            enroll ![hold] -> Enrolled;
        }
        Enrolled {
            quit -> Quit;
            graduate -> Graduated;
        }
        Graduated {
        }
        Quit {
    }
    }
    * -- 0..1 Supervisor;
}

class Supervisor {
    isA Person;
}
```
namespace BankingSystem;

// Namespace for core of the system.
namespace BankingSystem.core.humanResources;
class PersonRole{
    name;
    address;
    phoneNumber;
    1 -- * PersonRole;
}
class Person{
    name;
    address;
    phoneNumber;
    1 -- * PersonRole;
    
    class Employee{
        isA PersonRole;
    }
}
class Client{
    isA PersonRole;
    name;
    address;
    phoneNumber;
    1..2 -- 1..* Account;
}
class Manager{
    isA Employee;
    0..1 -- * Employee;
}

// Accounts, privileges, etc.
namespace BankingSystem.core.intangibleResources;
class Account{
    balance accountNumber;
    ...
Overview of Research Direction

- Umple, a Model Oriented Programming Platform
- Enhance OO code with modeling abstractions
- Associations, State Machine, Model Based Tracing, OCL like constraints.
- Visual and Textual views are automatically synchronized
- No need to edit the generated code
Adding Key Requirements Entities

- We propose to incorporate textual representation of key requirements entities into the "Model Oriented" code.

- Thus, one can interplay between requirements, models, and code.

- As a results, we eliminate or reduce the need for creating or maintaining requirements links.
Requirement-Oriented Model and Programming Language (ROMPL)

- Language components
  - OO code
  - Modeling Abstractions (state machines, Associations, etc.)
- Requirements Entities
  - Goals, KPIs, Business Rules, ..
Forms & Users

Form Registration {
14 Patient.age mandatory;
15 Patient.name mandatory;
16 symptom optional;
}

Actor Nurse { .. }
Actor Clinician { .. }
UserGroup Accountants { .. }
State Machine Modeling

Form Registration {
    Patient.age mandatory;
    Patient.name mandatory;
    symptom optional;

    //state machine to define
    //behavior of the form.
    status {
        Open {
            submit [complete] -> Submitted
            close -> Closed; }

        Closed {
            entry/ {saveFormData();} }

        Submitted {
            reOpen -> Open; } }

    calculatePriority {
        // Algorithmic code to calculate
        // priority of patients. } }
UML Attributes and Associations
Tasks & Business Rules

- Performed by Actors
- May involve completing Forms
- Measured by KPIs
- Must conform to Business Rules

```c
39 Task PatientRegistration {
40    Actor Nurse;
41    Form Registration;
42    KPI patientWaitTime;
43    BusinessRules CostReimbursement;
44
45 BusinessRule CostLimit {
46    // Definition of Business Rule...
47}
```

```
Scenarios

- Scenarios are a sequence of tasks
- Support for Forks and Joins
Goals and SoftGoals

- Support for “AND” and “OR” decompositions.
- Goals are measured by KPIs
ROMPL Key Benefits

• Requirements are integrated within executable artifacts (no longer a separate artifact)

• Reduce or eliminate the need for requirements links.

• Broaden participation to include Business Analysts.

• Other?
Challenges in Adopting ROMPL

• ROMPL is in its incubation phase and requires further refinements

• Using different abstractions may introduce some problems

• Evaluation
  
  • Nurse on-Boarding Process (healthcare domain)
Thank You
Literature Overview

- Connecting requirements to code has been attempted [1, 15].
- Attempts to link MDA and Requirements [18, 19]
- Probabilistic modeling of requirement traces [14]
- Computational reflection: the software system’s ability to dynamically observe and possibly modify its behavior [17]
References


[6] Derivation of Event-Based State Machines from Business Processes


