**The Problem**

### Web Applications’ Complexity
- Web applications exhibit complex transactional behavior:
  - Hierarchical structure of transactions satisfying user (sub) goals
  - Multiple resource managers, with diverse semantics and characteristics, are accessed in the scope of the same transaction
  - Pre-existing logic is utilized (e.g. Legacy Systems, Web Services)
  - Not all user activities are strict ACID transactions
- Navigation actions may mislead user regarding transaction status

### Ubiquity Issues
- Ubiquity introduces new issues:
  - Implementers “would like” application to be written once independently of delivery channel, device, etc.
  - Asynchronous transaction execution is needed; how is it supported? What’s now the application’s behavior?
- Design and documentation for such applications is important. No such mechanisms exist.
- A modeling language for analyzing, designing and documenting their transactional behavior would be valuable
An Execution Environment

Characteristics

- A single access point of the application
- Multiple Resources with diverse semantics and interfaces
- Use of pre-existing logic (legacy systems, web services, etc.)
- Ubiquity
  The same application logic is delivered
  - through different channels
  - at different devices
  - in different user profiles

Our Goal

- Design the transactional properties of the application logic in advance
- Enable the design of web applications in both top-down and bottom-up fashion
- Document the application behavior enabling easy derivations of new implementations (or transformations) for new devices, user profiles, etc.
## Objectives Set for UTML

- Description of both **static structure** and **execution flow** of transactions
- Modeling of transactions including most of known transaction models
- **Extensibility** for designing new transaction models according to the application’s requirements
- Description of **diverse decomposition semantics** and **behavior** into the same structured transaction
- Support for **weak transactions** (weaker than ACID)
- Description of **long-lived transactions**
- Provision for modeling **asynchronous execution** of transactions

## Followed Methodology

- **Built on top of UML**
- Use of UML class diagrams for modeling the **static structure** of transactions and UML state charts for modeling their **dynamic behavior**
- Provide a **flexible and extensible meta-model** capable to describe transactions following most known transaction models
- Give appropriate **well-formedness** rules to formalize and automate the transaction modeling process
- Provide a **complete notation system** to visualize the transaction modeling process
- Provide **Documentation** for the designed applications in appropriate format; Important for different implementations of the same applications
The UTML Transaction Meta-Model

Characteristics
- Activities and Operations as main modeling concepts
- Distinction between management and logic of activities
- Definition of execution contracts (subsets of ACID) for activities
- Separate modeling of activity decomposition semantics
- Modeling of Compensations
- Well-Formedness rules are used to formalize the user-defined models

Extensibility Mechanism
- 1st Part: Definition of new management operations for the custom model
- 2nd Part: Definition of appropriate well-formedness rules formalizing model's behavior
Also, **Well-Formedness Rules** may be attached on activities formalizing their behavior and co-ordination with parent/sub activities (2nd part of meta-model’s **extensibility mechanism**).
Conclusions

It has the ability to:

- Describe transactions in a high level and declarative way
- Support design in both top-down and bottom-up approach
- Model weak transactions – weaker than ACID
- Describe transactions conforming to the most of known transaction models
- Incorporate different semantics and behaviors into the same structured transaction
- Describe transaction models from scratch by using its extensibility mechanism (management operations & well-formedness rules)
- Model the execution flow of transactions, defining a primitive user navigation model

Future Work

- Better formalization of UTML
- Extension to directions of:
  - Describing asynchronous execution of transactions (replication, allotment, virtual executions, synchronization), enabling the design of mobile applications
  - Modeling data flow dependencies between transaction and compensation strategies
  - Modeling persistent activities (recoverability of activities; not only databases)

A work financed by the EU Project Ubiquitous Web Applications (IST-2000-25131) www.uwaproject.org