Ubiquitous computing simulator

P. Rajeeva
Protocol Engineering and Technology
Electrical Communication Engineering,
Indian Institute of Science,
Bangalore – 560012, India
Organization of the talk

- Introduction to simulation
- Development cycle of simulation tool & different types of simulators
- WISE – A Simulator Tool kit for Ubiquitous computing
- UBIWISE - Ubiquitous Wireless Infrastructure Simulation Environment
- TATUS – Ubiquitous simulator
- Development work for TATUS
- conclusion
Simulation

Definition:

A simulation is an imitation of some real thing, state of affairs, or process. The act of simulating something generally entails representing certain key characteristics or behaviours of a selected physical or abstract system.
Simulation

• Simulation is used in many contexts, including the modelling of natural systems or human systems in order to gain insight into their functioning.

  - simulation of technology for performance optimization, safety engineering, testing, training and education.
Development of Simulation

- Problem Formulation
- Data Collection and Analysis
- Simulation Model Development
- Model Validation, Verification and Calibration
- Input and Output Analysis
- Performance Evaluation and What-If Analysis
- Sensitivity Estimation
- Optimization
- Gradient Estimation Applications
- Report Generating
Classification

- Physical and interactive simulation.
- Computer simulation.
- Computer Science simulation.
- Simulation in education and training.
Examples in different areas

• Truck simulator:
  - provides an opportunity to reproduce the characteristics of real vehicles in a virtual environment.

• Healthcare simulators:
  - diagnostic procedures as well as medical concepts and decision making to personnel in the health professions.

• Military simulations.

• City simulators / urban simulation.

• Flight simulators.

• Robotics simulators.

• Marine simulators.

• Simulation and games.
Types of Simulator Tools

• Special purpose simulators.

• general purpose framework simulators:
  - Discrete Event Simulators
  - Agent-Based Simulators
  - Continuous Simulators
  - Hybrid Simulators
Network simulator

• A network simulator is a piece of software that simulates a network, without an actual network being present.

• It implements network protocols such as TCP and UDP, traffic source behaviour such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more.
WISE – A Simulator Toolkit for Ubiquitous computing

• WISE - Wireless Infrastructure Simulation Environment.
  - toolkit provide a means for comparing wireless protocols.
  - motivated by a vision of digital “appliances” interacting directly with Internet services.
  - a scientific basis for judging the effectiveness of the protocols.
  - By developing a simulator for the client side, enable service creation and testing, to move forward and further development of digital appliances.
WISE - simulator properties

- Creation and usage of devices.
- Multiple scenarios.
- Flexible User Interface and Form Factor.
- Data Manipulation.
- Communication Protocols.
- Ease of use.
- Fidelity of the simulation.
The WISE model:

- The WISE model presents a layered abstracted view of the environment to the end-user.
- The top layer is the World and the next layer includes the Devices. Below devices live layers for data collection and communication protocols.

- Layers: 
  - The world
  - The Devices
  - The Data API
  - The Communication Protocol API
  - Connectivity
Characteristics of wise

- Multiple views of the device.
- Grouping of Devices.
- Context-sensitive devices.
- Service models & security.
- Multiple access point.
- Scripted scenario.
- Multi-modal devices and multi-modal scenarios.
UBIWISE - Ubiquitous Wireless Infrastructure Simulation Environment

- UbiWise
  - simulator for ubiquitous computing.
  - computation and communications devices, either integrated with physical environments.

- The design of suitable hardware depends upon understanding how the hardware fits in to ubiquitous computing applications.

- Ubiquitous computing applications cannot be developed without suitable hardware.
Ubiwise Ubicomp Simulator

- Construction Difficulties
  - 1 Device per on research group per year
  - Rarely iterate on design
- Little work on communications
  - Device/device or device/service
- Irreproducible
  - Demo hardware vanishes
- Many issues ignored or poorly studied
  - Context, services, environment, etc
Ubiwise Design Goals

• Ease of Design
  – Java Toolkit, Scenario Editors
• Communications focus
  – Protocol development framework
• Reproducible
  – Record/playback; update strategy
• Open research tool
  – Available, open source (LGPL)
Motivation:

- the design of suitable hardware depends upon understanding how the hardware fits in to ubiquitous computing applications.
- ubiquitous computing applications cannot be developed without suitable hardware.
- experiment with new sensors, whether handheld or environmental, without building and deploying them.
- aggregate device functions without connecting real devices.
- observe how users might react to new devices and services before we can fully realize them.
- explore the integration of hand held devices and Internet services without the huge cost and large teams need to realize this integration.
Requirements and challenges

• explore the integration of handheld devices and Internet services without the cost and large teams need to realize this integration.
• aid rapid prototyping of new devices and protocols.
• test robustness and interoperability of subsystems.
• reproduce experiments and results of other researchers.
• mix simulated and prototype devices and services
Requirements and challenges

➢ experiment with new sensor concepts, whether handheld or environmental, without building and deploying them.
➢ aggregate device functions without connecting real devices.
➢ develop new service and device discovery protocols without implementing protocols in multiple mobile devices.
The Simulator description

• a user – a person running the simulation
• a researcher – a person configuring the simulation
• a developer – a person extending the simulator itself
Users view:

- multi-user client-server system.
- device interaction view.
- physical-environment view.
Device-interaction View for the camera user scenario:
Researcher’s View of UbiWise

• Inside the Device-Interaction View.
• Inside the Physical-Environment View.
• Connectivity.
• Protocols.
• Scenarios.
• Mixing in reality.
Device- Interaction programming model

Screen regions match Java methods

Screen region managed by WebBrowser
• creating the objects which will be present in the environment.
• creating the surrounding environment.
• adding and compiling physical interaction code into a dynamic link library

• Physical environmental view
Developers View of UbiWise

External services

Device interaction events

Physical environment events

UbiSim server

WISE

UbiSim

User #1

WISE

UbiSim

User #2
UbiWise system overview:

- **WISE server**
- **UbiSim server**
  - **HTTP events**
  - **Q3A events**
  - **Phys/Virtual events**

Connections:
- WISE server to UbiSim server via **Phys/Virtual events**
- WISE server to Client #1
- UbiSim server to Client #2
- WISE server to UbiSim

**Tools and Applications:**
- **Petstore**
- **petstore**
- **client #1**
- **Client #2**
Device design view for the scenario.
Snapshot of a Ubisim environment:
TATUS – Ubiquitous simulator
High-level simulator overview

- Software under test (SUT) holds the environment
  - Decisions making in user movement and behaviour
  - SUT controls the TATUS virtual environment
  - Testing adaptive softwares
• Flexible experiments because multi-play game allow multiple experimenters to in a single experiment while non player-character Artificial Intelligence allows single researcher to work control multi-person scenarios.

• Replication of experiments using saved settings, both of test subjects control of the simulated environment and the resulting simulated sensor events and adaptive software responses.
The simulator must be flexible to handle diversity in research projects

- Provide researchers with a mechanism to allow the SUT to control actions within the simulator
- Allow researchers to select simulator events of interest for notification to the SUT
- The simulator must support connection to multiple SUT connections in simultaneously
Introduction to Half-Life

• Half-life uses client–server architecture.

• Built – in artificial intelligence.

• Correcting the true picture.

• Conversion of HL - creating new maps, weapons, characters, physics and rules.
Ubiquitous computing simulators

High Level Test - Bed
To allow researchers/testers to readily connect an application to the simulator.

To readily reflect the changing state of ubiquitous computing technology especially new types of sensors and actuators.

To replicate experiments using saved settings, both of test subjects control the simulated environment and the resulting simulated sensor events and adaptive software responses.

To review experiments by rerunning recorded and logged experiments
• To simulate conditions in a physical environment and to simulate corresponding sensor events.

• To allow researchers to define simulator events of interest for notification.

• To provide researchers with a mechanism to allow to control action within the simulator.

• To be flexible in handling a diversity of test scenario's.
Architecture of the TATUS SIMULATOR

Ubiquitous computing
An overview of TATUS
TATUS simulator

1. Core 3D Simulator
2. Proxy
3. Message Definition Tool
4. Map Editor
• The Core 3D simulator provides an interactive simulated ubiquitous computing environment

• The Proxy hosts the network connection to the simulator so that the connection setup procedure between SUT and TATUS is minimised. The Proxy’s classes provide an API

• The Message Definition Tool generates and saves the XML Message Format File. The contents of the file are defined by the researcher using the tool’s GUI

• The Map Editor offers a toolset for creating maps and saves them in a BSP/MAP format so they can be loaded into the simulator
Typical simulation scenario description

Screen shots from TATUS simulator

Ubiquitous computing
Operation of integrated simulators

Ubiquitous computing

Diagram:

- Ubicomp environment editor
  - Maps, sensor, actuator and NPC configuration
  - Maps and access point placement

- TATUS
  - Player/character movement
  - Sensor data

- Wireless Network Simulator
  - Accuracy adjusted location tracking

- SUT
  - UbiComp Service Interactions
  - Wireless throughput values

- Throughput Throttle

- Human test subject

- PDA

Simulator integration

Ubiquitous computing
Development work for TATUS

- Extension to Proxy
- Network connectivity modelling
- XML file logging for Message Definition Tool
- Logging and recording experiments
- Sensor/Actuator library
conclusion

• From the ubiquitous computing simulator testing of the adaptive system is done.
• The Java Proxy hosts a network connection and protocol interpretation scheme in order to connect the SUT to the simulator.
• Adaptable message content, combined with both event-driven and polled state extraction provides a flexible access route to Half-Life as data.
  • TATUS also has built-in provisions as part of its socket control to accept more than one SUT as a client.
THANK YOU