

Test Models in TEMA

Outline

- Model Semantics
- Models and Applications
- Handling Models

Model Semantics

Formalism

- Models are LSTS (labeled state transition system) state machines
- Functionality encoded into actions (transition labels)
- State labels used for auxiliary information

Parallel Composition

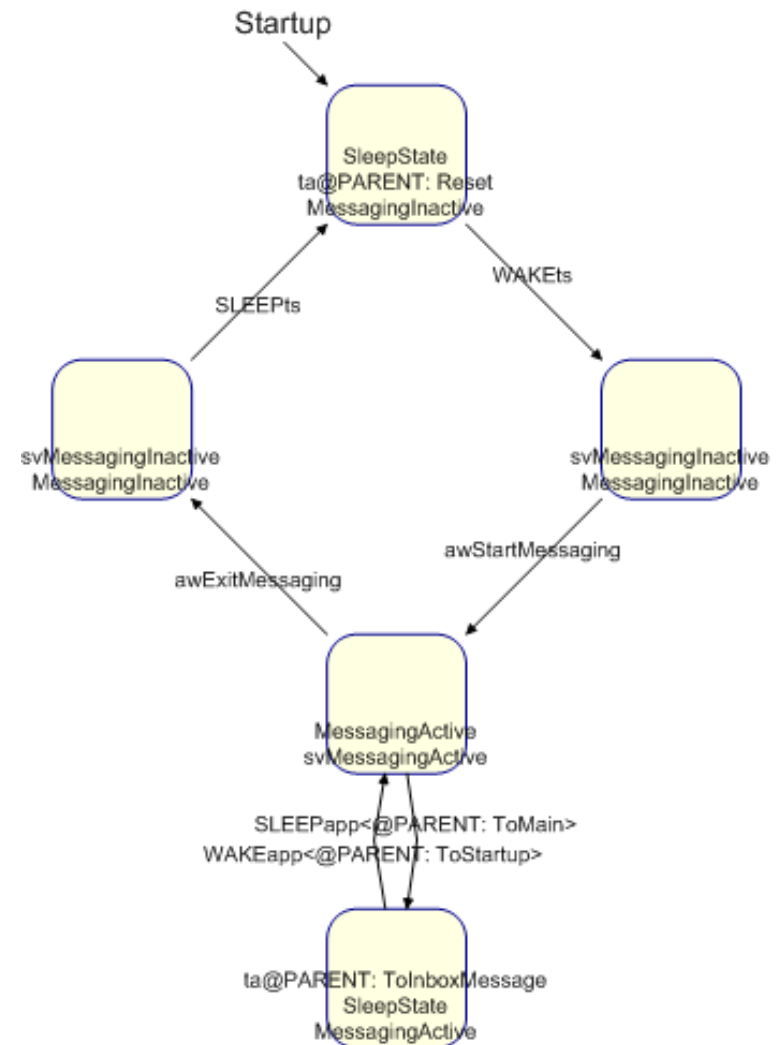
- Realistic systems are too large to model in a single state machine
- We create several smaller model components and combine them with parallel composition
- In parallel composition some actions of individual model components executed synchronously
- Rule-based composition: synchronizations defined explicitly

Model Architecture

- Single model component active at a time, others sleeping
- Active component can
 - Execute actions on SUT (system under test)
 - Query/alter the state of other components
 - Activate another component and go to sleep
- Models created in two tiers: action machines and refinement machines

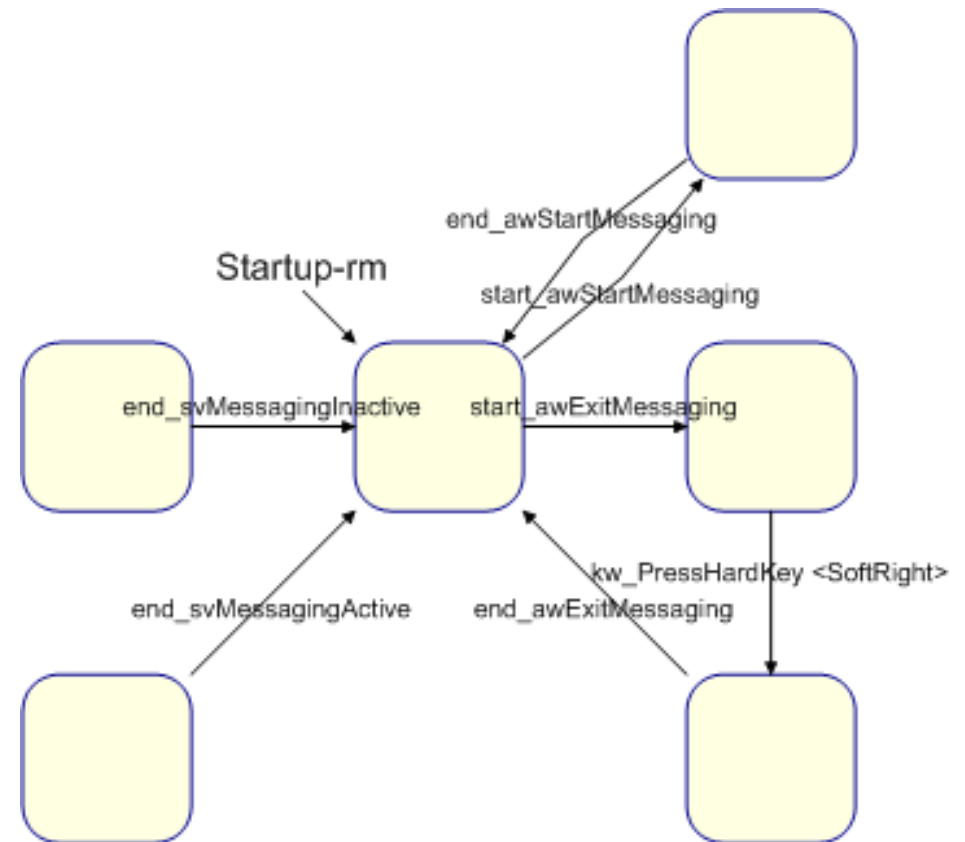
Action Machines

- Describe the abstract functionality of the SUT
- Can be used across different products



Refinement Machines

- Describe the implementation of the functionality on UI
- Specific to UI and product



Generated Components

- Task switcher: manages control switches within a single target (phone)
- Target switcher: manages control switches between targets
- Synchronizer: used to form connections between specific targets

Data

- Localization tables used to store simple product-specific information
- Data tables used to store structured information
- Data in data tables accessed through data statements, which can execute Python code

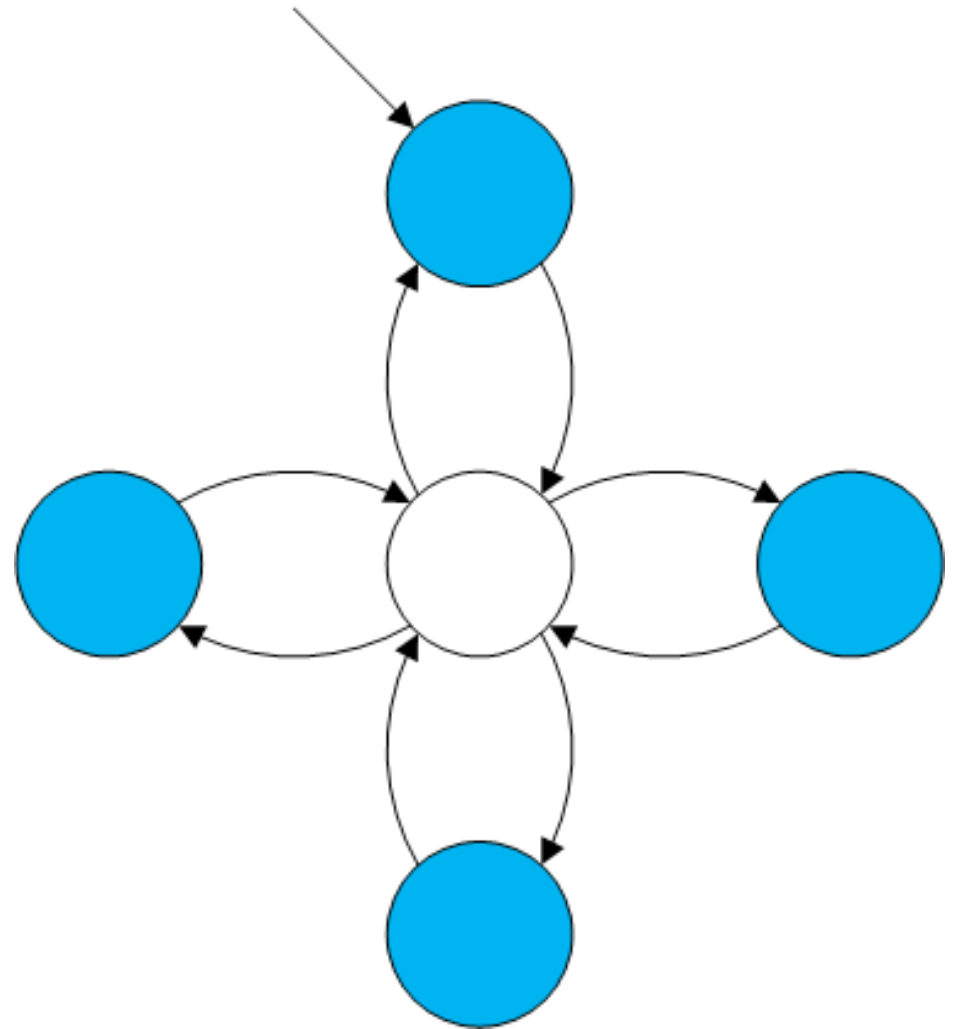
Models and Applications

Model Components in Applications

- Applications divided into smaller model components to simplify modeling
- Separate components e.g. for
 - Each view
 - Complicated functionality
 - Important variables, memory

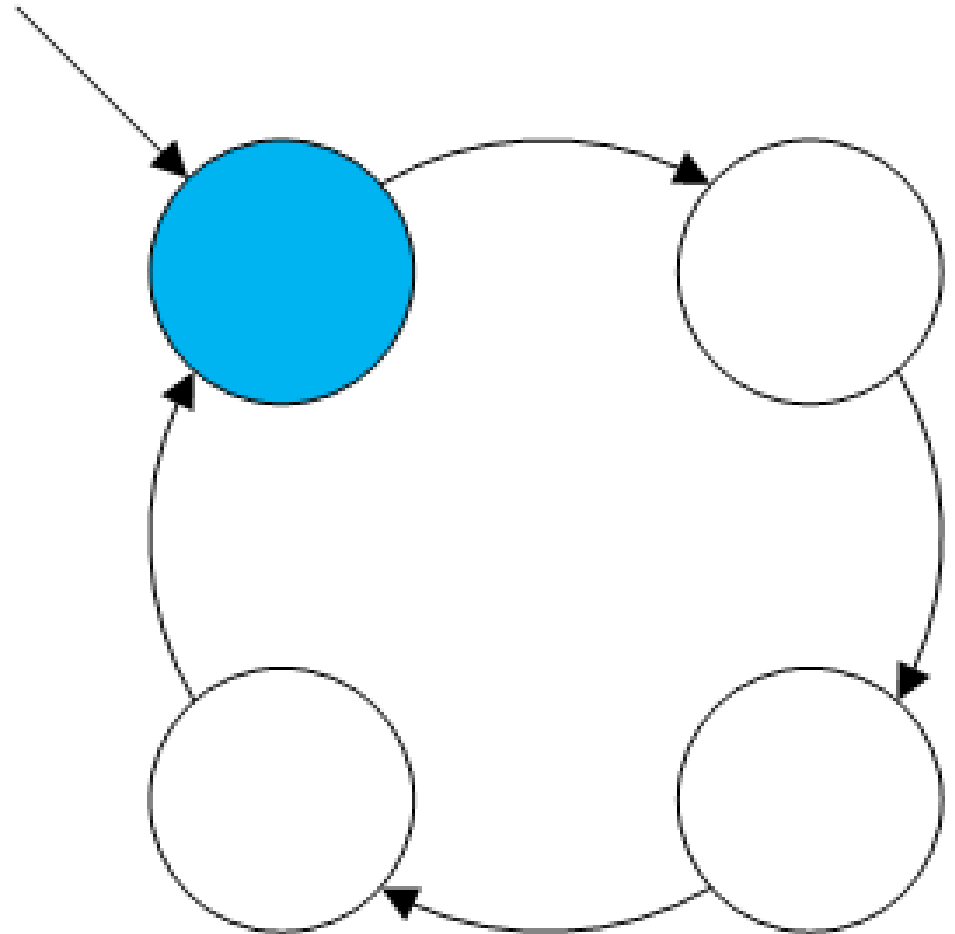
Models for Views

- Describe a single view of application
- Offer means to switch to other views
- May contain simple functionality



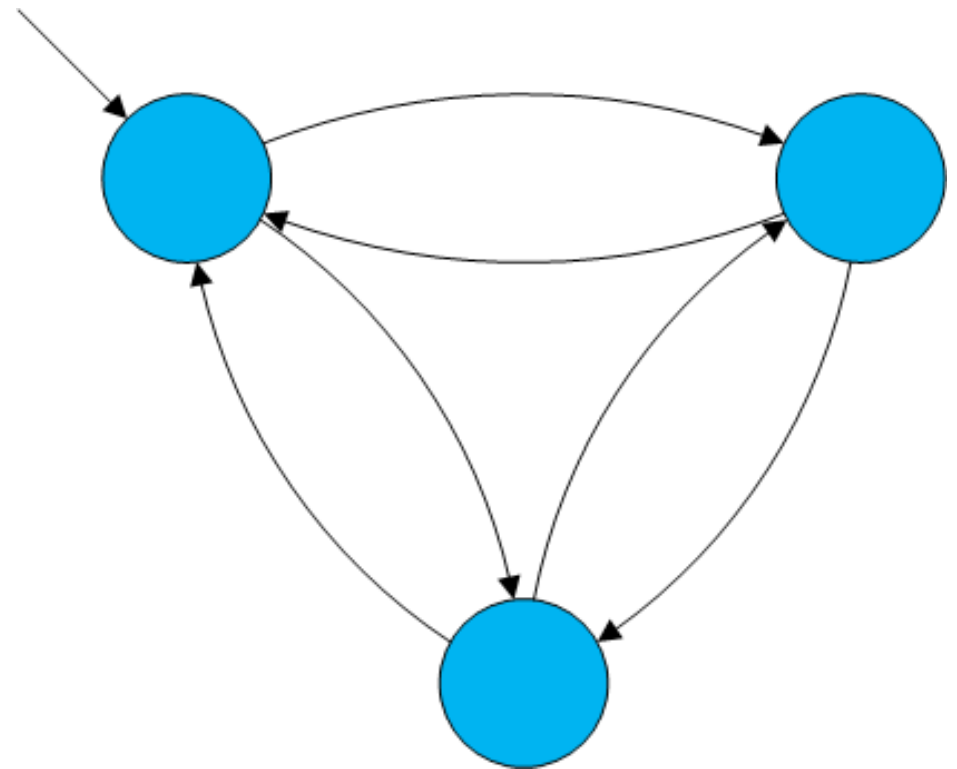
Models for Functionality

- Describe a single, often linear process
- Useful for complex action or synchronization sequences
- Usable in multiple views



Models for Memory

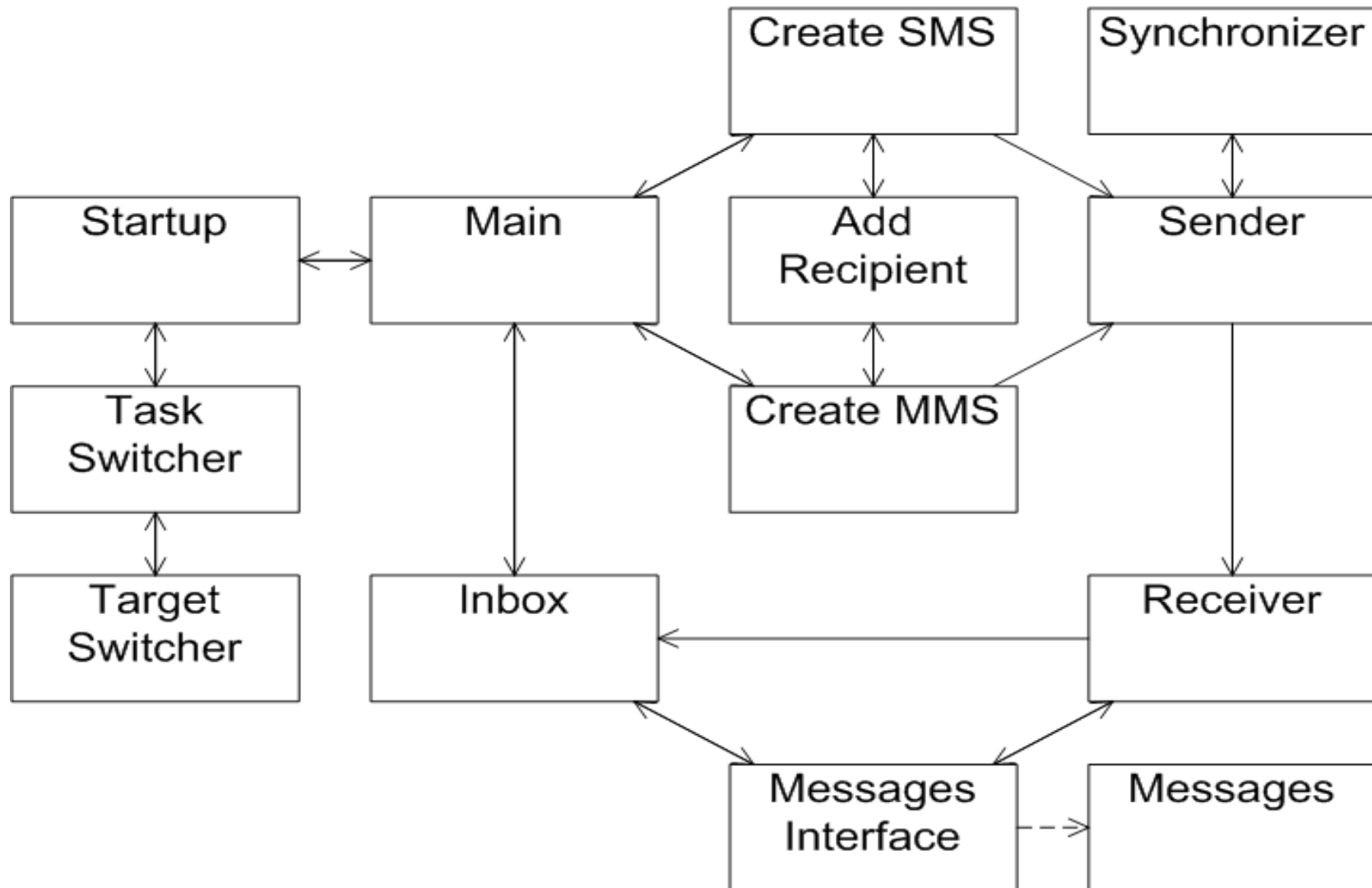
- Have a state for each possible value
- Retain their states while other components are active



Example: Messaging

- Models for views: Main, Inbox, Create SMS, Create MMS
- Models for functionality: Add Recipient, Sender, Receiver
- Models for variables: Messages, Messages Interface
- Other: Startup

Messaging on Single Phone



Handling Models

Model Library

- Collection of model components designed to work together
- Contains information on how the components are related to each other
- A number of components can be selected for composition into test model

Synthesizing Models

- Test models can be synthesized from test cases
- Based on finding identical action sequences and corresponding states
- Resulting model can generate original test cases and their combinations
- Currently lacks sufficient tool support

Filtering

- Incomplete and/or bugged SUT cannot execute all tests model can generate
- Test generation must avoid unexecutable features
- Such features can be filtered out of the model