Applying Formal Methods to a Telecommunications System in a Commercial Setting

Marsha Chechik and André Wong
Department of Computer Science
University of Toronto

December 8, 1998

Formal Methods

"The use of mathematical modeling, calculation and prediction in the specification, design, analysis, construction and assurance of computer systems and software."

- Engineering — partial differential equations to model variations in physical quantities over time or space.
- Software — model discrete quantities. Need concepts like sets, graphs, partial-orders, and finite-state machines.
- "Calculation" is in terms of formal logic rather than numerical computation.

FMs in Safety-Critical Systems

"Systems where compelling evidence is required that services are delivered in a manner that satisfies certain critical properties."

Examples:

- Military Systems
- Air Traffic Control Software
- Flight Control Systems
- Nuclear Power Plants
- Medical Systems
- Telephony Networks
- Transportation Systems

How much trust do you have?

CLARINGTON, Ont., April 24 /CNW/ - Ontario Hydro's Darlington Nuclear Generating Station... experienced a spill of moderator heavy water inside the plant...

Without the help of FMs, will the automated nuclear shutdown systems operate as expected?

FMs in Non-Safety Critical Systems

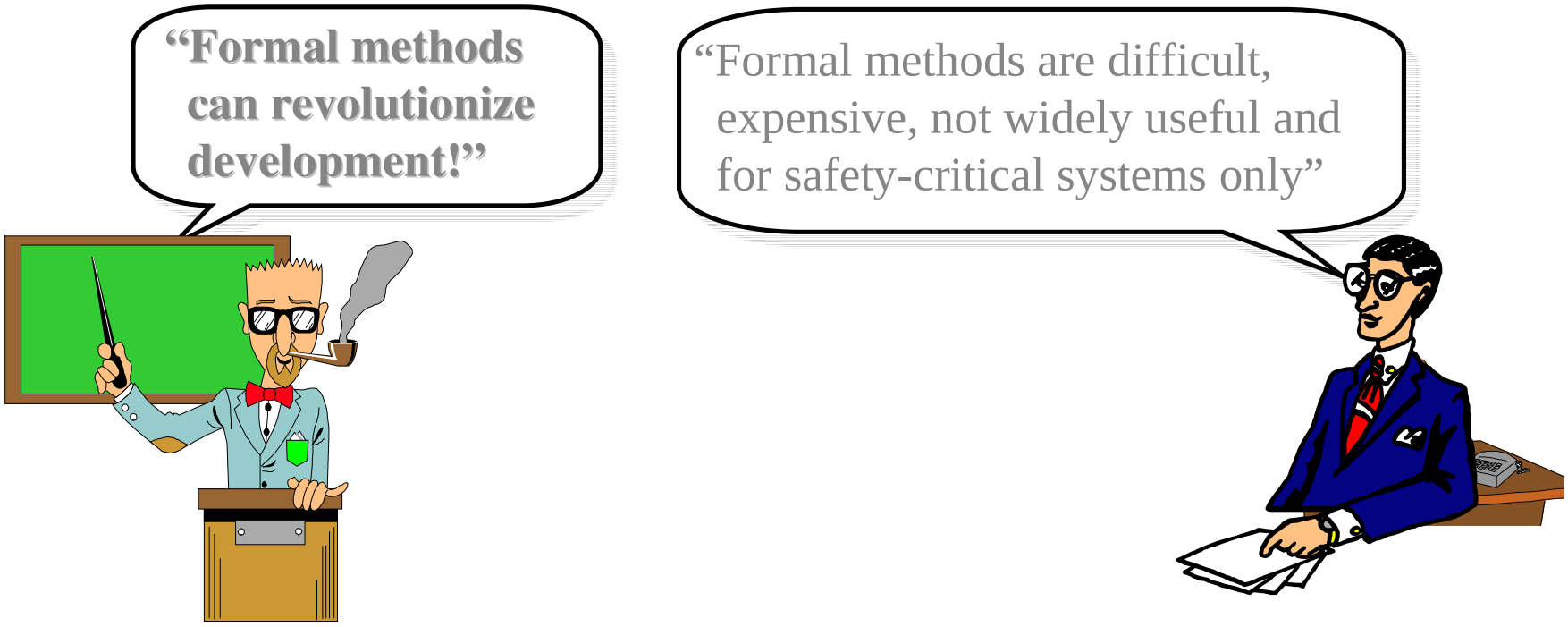
Examples:

- Transaction processing systems
- Databases
- Electronic devices
- CASE tools

Have less impact:

- Smaller scale
- Limited scope
- Adoption rate and % of penetration are much smaller

Why aren't FMs used more?



Why aren't FMs used more? (cont'd)

Myths?

- Require deep mathematical backgrounds
- Are all about proving correctness
- Benefit only the requirements phase
- Always require substantial investments and increase cost
- Are associated with steep learning curves
- Require radical changes to the development process

Incorrect expectations?

- Provide silver bullet for all problems
- Eliminate testing
- Guarantee correct software

Safety-Critical vs Non Safety-Critical

- Goals
  - Cost / Time / Quality
- Nature of the Project
  - Legacy code
  - Need for maintenance
  - Stability of requirements
- Staff
  - Educational background

The Questions

- What are the keys to successful applications of FMs?
  - What essential properties must those FMs have?
  - How should FMs be used?
- What are the improvements/costs?
- What can be done to amortize the cost of creating the formal specifications?
- Is it beneficial to apply FMs to commercial projects? Under what conditions?

The Project

Problem

- Manager of Nortel TorMAC testing team:

  "errors in the requirements specifications are often causing problems in the testing phase."

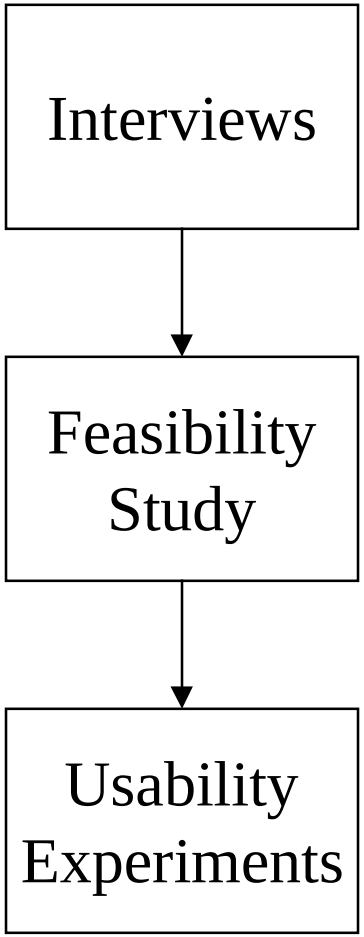
Project

- A collaboration between Nortel and the Formal Methods Laboratory of the University of Toronto

Goals

- Demonstrate feasibility of FMs in telecom systems
- Investigate benefits and problems with applying FMs in a commercial setting

Three Phases of the Project



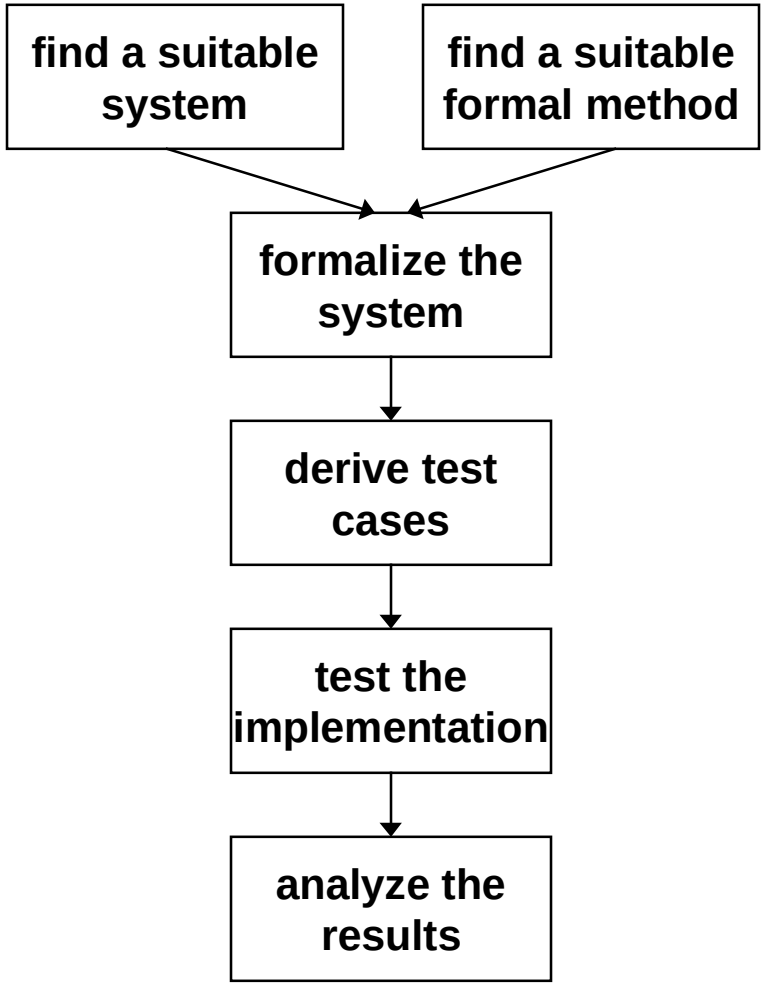
- Conduct interviews to identify problems with development process
- Agree on features of FM
- Model a telecom system using the chosen FM
- Evaluate *effectiveness* of FM vs existing development method
- Evaluate *usability* of the chosen FM by conducting small controlled experiments

Results of Interviews

- Use of prose often causes problems
- Still many errors in the software requirements specifications (SRSs) after several manual inspections
- Errors in the SRSs caused unavoidable design and implementation changes and affected the quality of the test cases created for acceptance testing
- Late changes to the designs and implementations

  → the SRSs and test cases were not always updated

Overview of the Feasibility Study



System Selection

Properties of the Ideal System

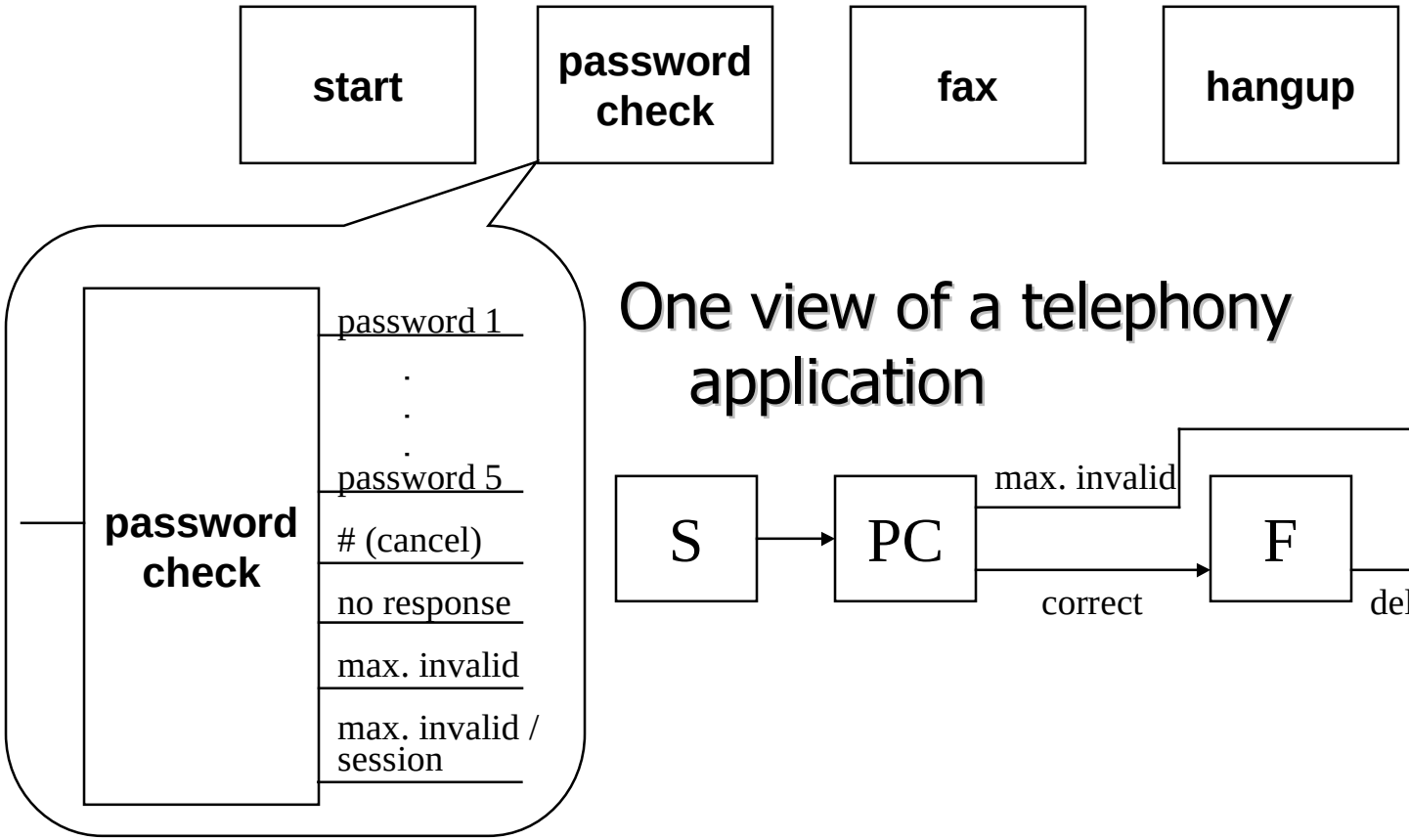
- worth engineering
- self-contained / loosely coupled with the rest of system
- relatively stable requirements
- not too broad in scope but contain complex parts
- reactive

Choose *Service Creator* (SC)

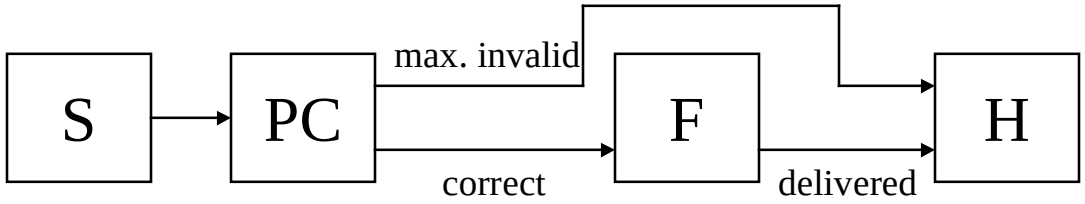
- Voice-service creation environment
- Used for constructing telephony applications through a GUI

System Selection (cont)

Block Run-time Modules (BRMs)



One view of a telephony application



Formal Method Selection

A broad survey to look for FMs that

- have a simple notation
- allow users to produce readable and reviewable artifacts
- provide support for both modeling and testing

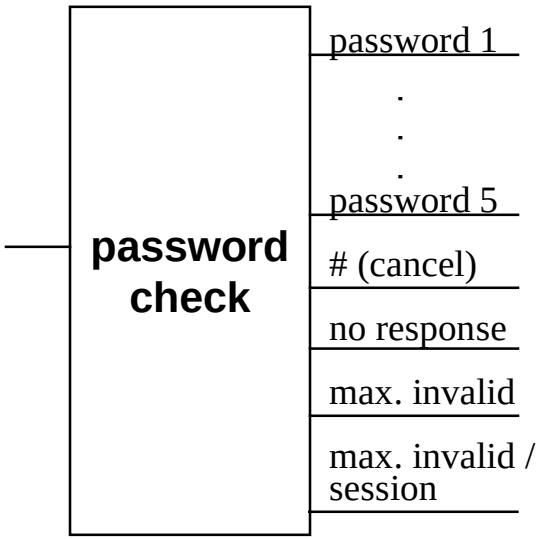
  (for amortizing the cost of creating the formal specification)

Major problems:

- Only a few tools satisfy the last point
- Most research tools are
  - Overkill
  - Too complicated / time consuming

Formal Method Selection (cont)

- Three runners-up:
  - Telelogic SDT
  - Aonix Validator/Req (V/Q)
  - Teradyne TestMaster (T/M)
- Used to model a small part of SC (the password-check component)



Quantitative Feature Analysis

Criteria	Weighting	SDT	V/Q	T/M
has modeling support	5	4	1	3
has testing support	5	3	4	5
has a gentle learning curve	4	3	5	3
produces easy-to-understand artifacts	3	3	5	4
enhances understanding of the system	3	5	3	4
integrates smoothly with the existing development process and tools	4	5	3	3
is scalable	3	4	2	3
has strong industrial support	3	5	4	5
performs completeness or consistency checks	2	3	3	3
provides features such as multi-user support, requirements traceability and document generation	1	3	4	2
Score	**N/A**	**136**	**112**	**124**

SDL

Specification and Description Language (SDL)

- Based on extended finite state machines
- A lightweight formal method (International Telecommunications Union standard Z.100)
- Used in the telecom field (e.g., PBXs and network protocols); now being applied in areas such as aerospace and train control

SDL Tool Support

Features of SDT, the SDL modeling environment

→ Derive MSCs from the model

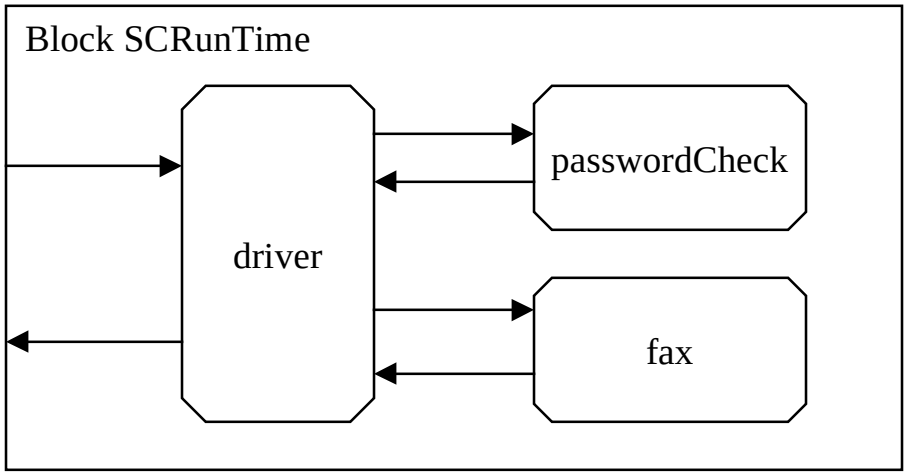
→ Verify MSCs against the model

- Create TTCN conformance test cases (not in our project)
- Check invariant
- Generate code

Modeling the System

- 80 pages of SC specification as a 70-page SDL model
- 10 of the more complicated BRMs → 10 SDL processes
- A 'driver' process administrates the control of the telephony application
- 23 SDL signals (8 external)

The whole SDL system mimics a high-level behavior of a telephony application built by SC.



Level of Abstraction

Modeling is mostly straightforward except for determining the *level of abstraction*

high level of abstraction for simplicity

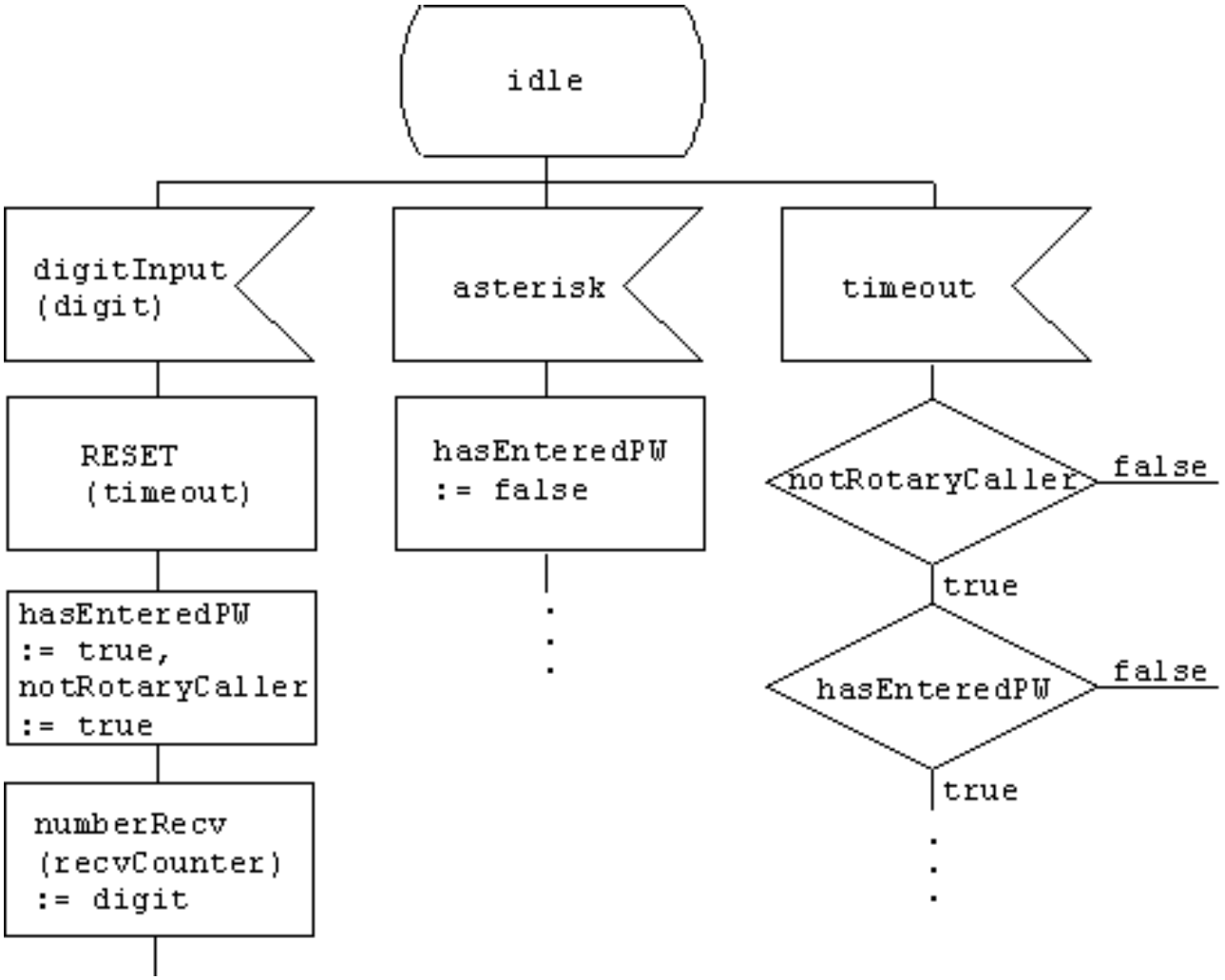
vs

detailed behavioral modeling for deriving sufficiently accurate Message Sequence Charts (MSCs)

Our approach

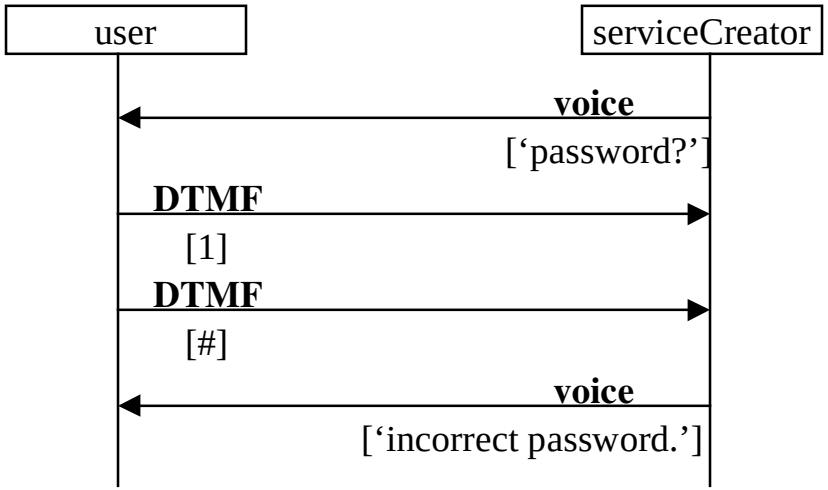
- start with a black-box view
- fill in only the necessary details

Level of Abstraction: Example



Test Case Derivation

- 120 MSCs created
- Creation is not fully automatic



Types of test cases

- Individual
- Component Interactions (more than 20 test cases)

Errors Found

Omissions in the specification (password-check component)

1. Which voice prompts are interruptible?
2. What are the lengths of various timeouts?
3. What are the minimum/maximum # of digits for valid passwords?

Implementation errors

- Some voice prompts are missing
- Rotary callers never get timeouts
- System crashes when passwords are too long

Duration of the Study

The entire study took 3 person-months

- 11 days for creating the models for the BRMs
- 9 days for deriving test cases
- 7 days for executing test cases
- Other activities
  - Choosing the formal method
  - Understanding the specification
  - Solving technical problems
  - Documenting the activities

Quantitative Results

Productivity Data	Study	Existing Process
Time to model (person-days)	11	N/A
# of spec. errors reported	56	N/A
# of test units	269	96
Test unit creation rate (test units/day)	30	21
Test unit execution rate (test units/day)	27	11
# of implementation errors identified	50	23

- Measure data in *test units*
- Some values were comparable
- 18 of 50 implementation errors from the SRS

Qualitative Observations

- Problems in the specification concentrated in a few areas
  - Most were avoidable
  - Most were omissions and vagueness
  - Some led to errors in the implementation
- Benefits of Creating a Model
  - Provide a better understanding of the system
  - More complete SRS → better test coverage
  - Remove errors in the SRS earlier in the development process
  - Aid the design and the testing processes

Changes Required

- Process
  - An add-on → no radical changes
  - Need slightly more time for requirements analysis
  - Require firm commitments from all stakeholders
- Staff
  - Training
  - Model parts of the SRSs in SDL
  - Derive test cases from the model
  - Keep the model and test cases up-to-date

Concerns about FMs

- Need careful application
- Are applicable only to some types of systems
- Require some changes to the existing development process
- Increase requirement cost
- Demand commitments from both the design and the testing teams
- How usable?

The Questions

- What are the keys to successful applications of FMs?
  - What essential properties must FMs have?
  - How should FMs be used?
- What are the improvements/costs?
- What can be done to amortize the cost of creating the formal specifications?
- Is it beneficial to apply FMs to commercial projects? Under what conditions?

Lessons from the Study

- FMs can be applied to commercial systems effectively
- A successful application of a FM depends on
  - Choice of the system to be modeled and a suitable formal method
  - Careful use of the FM and its tool
  - Established plan for a smooth integration
  - Commitments from stakeholders
  - Reasonable expectations from the FM
  - Availability of on-site "champions"

Current Work

- Usability of SDT/SDL
  - Experiment with Nortel engineers
    - Effective use of the SDL framework
    - Perceived usability problems
  - Surveys to collect opinions of Nortel engineers
    - Usefulness/impact of using the SDL framework
    - Integration problems

Future Work

- Pilot study by Nortel engineers
  - SDL integrated into the development process
  - More SDL modeling: a larger part of SC
  - Progress monitoring: time/cost/quality
  - Researchers act as observers and consultants