The Systems
&
Software Development Laboratory
**Goal:**
Improving the quality of systems and software

**Means:**
- Improved **methodology**
- Languages for programming and specifying
- Software that analyzes or helps develop other software (**tools**)
- Advanced **teaching** and training
Who are we?

- 6 faculty members
  
  Prof. Orna Grumberg
  verification of hardware and software

  Prof. Yossi Gil
  OO programming constructs and design

  Prof. Shmuel Katz
  program verification, distributed programming constructs

  Prof. Ron Pinter
  Software Engineering methodology and technology

  Dr. Eran Yahav
  Programming languages, program synthesis & analysis, verification

  Prof Eliezer Kantorowitz
  Software engineering, user interface, internet programming
Who are we?

- 6 faculty members
- system engineer
- 15 Post-docs and Grad students
- Over 500 undergrad students
SSDL: overview

Facilities

- Several servers
- 20 powerful workstations (Linux)
- Language support (C++, Java, …)
- CASE tools (design, track & debug)
- Supportive development environment
- Formal methods platform
SSDL: overview

Lab Activities

- MSc and PhD theses and research
- Advanced undergrad team projects
- Software Engineering courses
- Research projects in cooperation with industry
- International research in EU projects
SSDL: Research

Theses

- Verification using Convenient Computations. (scenario-based verification, reduces every computation to a convenient one)
- Superimpositions & Aspects (modularity for distributed systems, add functionality in stages that cut across usual object structure)
- 3D-Editor
- Efficient Algorithms for the Runtime Environment of Object Oriented Languages
Theses (cont.)

- Distributed model checking for large systems
- Model reduction using abstraction and symmetry
- Efficient model checking using abstraction, symmetry and compositionality
Software Engineering Track

- Methods in Software Engineering
- Programming Languages
- Formal Specification and Verification
- Program and System Design
- Security and Privacy
- Quality assurance for Software
SSDL: Teaching

Student projects

- Travel agency.
- Automated highway control.
- Home budget.
- Multi choice exams
- Software metrics
- Analysis of JAVA programs
- Brain exercise: games for seniors
Two Cars Driving at Minimal Distance
Aspects – A definition

An aspect is a concern (view) orthogonal to the way the rest of the system works.

Aspects include both changes to be made, and where to make them.

Languages: AspectJ, Compose*, ...
As a result of changes in the regulations of the central bank,

(1) the balance of an account can’t be negative.

(2) Key messages (and only those) have to be encoded

We have to update the banking software accordingly.
Aspects – the banking system:

The traditional solution

Change every place that updates the balance or sends key messages.
Aspects – the banking system:

The traditional solution – drawbacks:

- We need to change many places.
- Hard to maintain.
- Code duplication.
- How do we know that we corrected all the places?
- How do we know that the new code is correct?
Aspects – the banking system:

The Aspect way

All the changes are defined externally.
SSDL: Aspects

Aspects – advantages:

- Only one new piece of code for each concern.
- We can reuse the code.
- Automatic update of the original system.
- The original system remain un–touched.
- Easy to maintain.
Aspects – questions:

- Do aspects influence the system correctness?
- What about the relationship between aspects.
  - Can one aspect stop working because of a second aspect?
- Does the order of the aspects matter?
SSDL: Aspects

Aspects – SSDL research

SSDL hosted and led research and tool development on verification for aspects in the EU Network of Excellence AOSD–Europe, with the CAPE (Common Aspect Proof Environment)—11 academic and industrial participants (incl. IBM and Siemens)
Goal of the VeriTech Project

- To improve our ability to use computers for proving that the system we design is correct
- Verify: Hardware, Communication Protocols and Software
- Integrates existing tools, permits translating from one tool to the other
VeriTech: translating models until now
VeriTech: plug-in 2-way translations
- Help with equipment and software
- Grants for student projects
- Support for research of joint interest
- Teaching fundamental and advanced topics
- Joint projects