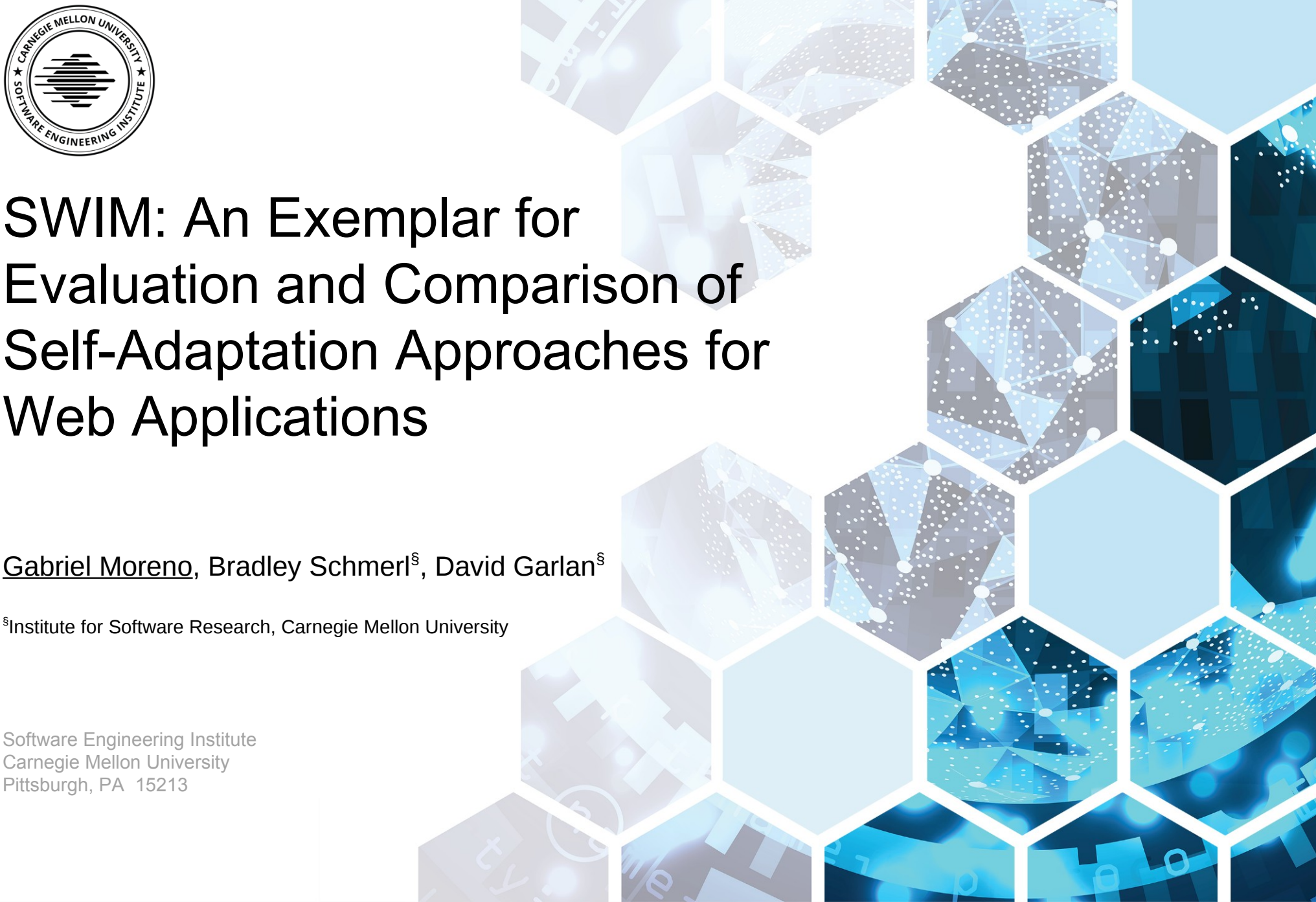


SWIM: An Exemplar for Evaluation and Comparison of Self-Adaptation Approaches for Web Applications

Gabriel Moreno, Bradley Schmerl[§], David Garlan[§]

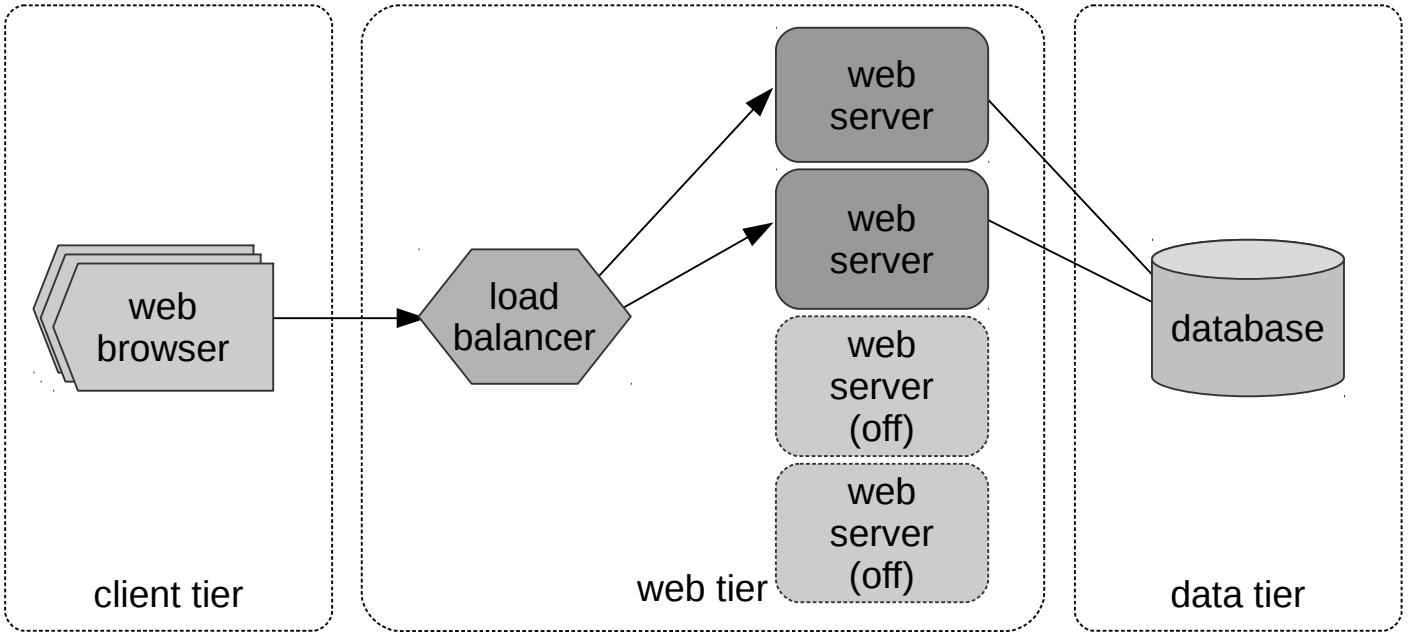
[§]Institute for Software Research, Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0694

3-tier web applications (Znn.com, RUBiS)



Adaptation goal:

- keep response time below threshold
- serve as much optional/high quality content as possible
- minimize cost (servers)

Adaptation tactics:

- add/remove servers
- increase/decrease proportion of optional content (a.k.a. dimmer)
- switch between low/high quality content

Using actual web servers for research

Although web applications like Znn.com and RUBiS were developed for research, they are not simulated.

Using actual web servers has drawbacks:

- These systems are not easy or cheap to deploy
  - Several machines needed
  - Even if cloud deployment is scripted, VM cost is incurred
- Run-time conditions cannot be replicated exactly for comparing self-adaptation approaches
  - Uncontrollable factors: background processes, network delays, etc.
- Running experiments is time-consuming
  - Real web access traces are hours long
  - For some experiment designs, running the experiments set takes days

SWIM

Only one machine needed: a single process simulates

- Clients sending traffic to web application
- Load balancer
- Multiple web servers

Replicates run-time conditions

- Traffic replayed from prerecorded trace files
- All random number generators are seeded

Speeds up experimentation

- Discrete-event simulation fast-forwards time to next relevant event

What SWIM simulates

- Round-robin load balancing
- The time it takes to process a request
  - Service time drawn from a random distribution according to response type
    - One distribution for mandatory content, one for mandatory+optional content
- Processor sharing
  - Up to a configurable number of concurrent requests per server
  - Other requests queued
- Cold cache effect
  - Processing requests in newly added servers takes longer
  - Effect gradually disappears

Adaptation manager as simulation module

SWIM implements most the MAPE-K loop

Adaptation manager needs to implement Analysis/Planning

Constraint:

- Adaptation manager must be implemented in C++

Benefit:

- The simulation controls the passage of time and fast-forwards time to next relevant event
- Example: 18-hours of traffic with 29 million requests going to a 60-server cluster can be simulated in 5 minutes on a laptop computer

Adaptation manager as external program

SWIM provides a simple TCP-based interface

- Probing commands get monitoring information
- Effector commands to execute changes

Limitation:

- Simulation must be run in wall-clock time

Benefit:

- The adaptation manager can be implemented in any language that support TCP socket communication
  - Existing self-adaptation frameworks Rainbow [Garlan 2004] and CobRA [Angelopoulos 2016] have been used with SWIM in this way

Demonstration

Get SWIM

Docker image:

https://hub.docker.com/r/gabrielmoreno/swim/

Source code:

https://github.com/cps-sei/swim

Additional Slides

What SWIM does not simulate

- The network and its delays
- The actual functionality of the web application
- Distinction between CPU processing and I/O for serving a request