The Current Team

- TASSL Rutgers University
  - Autonomic Computing Research Group
    - Viraj Bhat
    - Manish Agarwal
    - Hua Liu (Maria)
    - Zhen Li (Jenny)
    - Manish Mahajan
    - Vincent Matossian
    - Venkatesh Putty
    - Cristina Schmidt
    - Guangsen Zhang
  - Autonomic Applications Research Group
    - Sumir Chandra
    - Xiaolin Li
    - Taher Saif
    - Li Zhang
    - Hailan Zhu

- CS Collaborators
  - MPOC, University of Arizona
    - Saleh Hayat
  - Biomedical Informatics, The Ohio State University
    - Taha R. Sun, Joel Saltz
  - CS, University of Maryland
    - Alan Sussman, Christian Hansen
  - Applications Collaborators
    - CSM, University of Texas at Austin
      - Margaret Parsons, Mary Wheeler
    - IG, University of Texas at Austin
      - Myrals Sam, Paul Stoffa
    - ASCI/CACR, Caltech
      - Michael Aivazis, Julian Cummings, Dan Meiron
    - CRL, Sandia National Laboratory, Livermore
      - Jaideep Ray, Johan Steensland

Computational Modeling of Physical Phenomenon

- Realistic, physically accurate computational modeling
  - Large computation requirements
    - e.g. simulation of the core-collapse of supernovae in 3D with reasonable resolution (560') would require ~ 10-20 teraflops for 1.5 months (i.e. ~100 Million CPUs) and about 200 terabytes of storage
    - e.g. turbulent flow simulations using active flow control in aerospace biomedical engineering requires 5600 x 1000 x 500 = 2.5 x 10^9 points and approximately 10^7 time steps, i.e. with 1GFlop processors requires a runtime of ~7 x 10^6 CPU hours, or about one month on 10,000 CPUs! (with perfect speedup). Also with 700B/pt the memory requirement is ~1.75TB of run time memory and ~800TB of storage.

- Complex couplings
  - multi-physics, multi-model, multi-resolution, ...

- Complex interactions
  - application – application, application – resource, application – data, application – user, ...

- Software/systems engineering/programmability
  - volume and complexity of code, community of developers, ...
    - scores of models, hundreds of components, millions of lines of code, ...
The Grid

- The Computational Grid
  - Potential for aggregating resources
  - Computational requirements
  - Potential for seamless interactions
  - New applications formulations

- Developing application to utilize and exploit the Grid remains a significant challenge
  - The problem: a level of complexity, heterogeneity, and dynamism for which our programming environments and infrastructure are becoming unmanageable, brittle and insecure

- System size, heterogeneity, dynamics, reliability, availability, usability
  - Currently typical proof-of-concept demos by "hero programmers"

- Requires fundamental changes in how applications are formulated, composed and managed
  - Break current paradigms based on passive components and static compositions

- Autonomic, adaptive, interactive application on the Grid offer the potential solutions
  - Autonomic: context aware, self-configuring, self-adapting, self-optimizing, self-healing, ...
  - Adaptive: resolution, algorithms, execution, scheduling, ...

AutoMate: Enabling Autonomic Applications

- Objective:
  - Investigate key technologies to enable the development of autonomic Grid applications that are context aware and capable of self-configuring, self-composing, self-optimizing and self-adapting.

- Overview:
  - Definition of Autonomic Components:
    - Definition of programming abstractions and supporting infrastructure that will enable the definition of autonomic components
    - Autonomic components provide enhanced profiles or contracts that encapsulate their functional, operational, and control aspects
  - Dynamic Composition of Autonomic Applications:
    - Mechanisms and supporting infrastructure to enable autonomic applications to be dynamically and opportunistically composed from autonomic components
    - Compositions will be based on policies and constraints that are defined, deployed and executed at runtime, and will be aware of available Grid resources (systems, services, storage, data) and their current states, requirements, and capabilities
  - Autonomic Middleware Services:
    - Design, development, and deployment of key services on top of the Grid middleware infrastructure to support autonomic applications
    - A key requirement for autonomic behavior and dynamic composition is the ability of the components, applications and resources (systems, services, storage, data) to interact as peers...

AutoMate: Architecture
AutoMate: Architecture

- **AutoMate System Layer:**
  - builds on the Grid middleware and OGSA and extends core Grid services to support autonomic behavior.
  - provides specialized services such as peer-to-peer semantic messaging, events and notification.
- **AutoMate Component Layer:**
  - addresses the definition, execution and runtime management of autonomic components.
  - provides supporting services such as discovery, factory, lifecycle, context, etc.
- **AutoMate Application Layer:**
  - builds on the component and system layers to support the autonomic composition and dynamic (opportunistic) interactions between components.
- **AutoMate Engines:**
  - context-awareness engine composed of context agents and services and provides context information at different levels to trigger autonomic behavior.
  - deductive engine composed of rule agents which are part of the applications, components, services and resources, and provides the collective decision making capability to enable autonomic behavior.
  - trust and access control engine composed of access control agents and provides dynamic context-aware access control to all interactions in the system.
- **AutoMate Portals:**
  - provide users with secure, pervasive (and collaborative) access to the different entities.
  - using these portals users can access resource, monitor, interact with, and steer components, compose and deploy applications, configure and deploy rules, etc.

AutoMate: Components

- **Accord:** Autonomic application framework
  - H. Liu, M. Agarwal
- **Rudder:** Decentralized deductive engine
  - Z. Li
- **Squid:** P2P discovery service
  - C. Schmidt
- **SESAME:** Dynamic access control engine
  - G. Zhang
- **Pawn:** P2P messaging substrate
  - V. Matossian

Autonomic Computing Workshop/Tutorial

- **Autonomic Computing Workshop**
  - www.caip.rutgers.edu/ams2003
- **Autonomic Computing Tutorial**
  - automate.rutgers.edu