Multi-agent System Modeling to Explore the Adoption of Security Analysis Tools in Software Development
Overall Research Goals

- Software bugs and defects can increase the cost of software projects significantly, and the use of security analysis tools may find bugs and defects cheaper or faster than manual inspections.

- Adoption of security analysis tools is not very common in software developments.

- The overall goal of the project is to develop a simulation model to improve understanding of the adoption of security analysis tools in software developments.
Broad Impact

- A framework that incorporates
  - Rational decision making of developers and their manager in a project
  - Realistic representation of heterogeneity of tasks and organizational structure
- Improve understanding of the factors influencing the adoption of security tools in software development
Specific Research Goal

- Develop a decision model to capture the dynamic decision making of developers, working on a set of tasks that fulfill the goals of a software project.

- Develop a sanction model for managers to explore the influence of sanctions on adoption of security analysis tools.

- Simulate developers of heterogeneous skills in alternative organizational structures under sanction mechanisms.
Model Description

The multiagent system model $O = \langle A, M, T \rangle$ contains three components:

- $T$ is a set of tasks in a project
- $A$ is a set of developer agents who perform tasks
- $M$ is a manager who assigns tasks and sanctions

Diagram:

- Manager
- Developer
- Product
- Sanctions
- Tasks
Tasks and Artifacts

- A project consists of a set of tasks $T = \{t_1, t_2, t_3, t_4, t_5, \ldots, t_n\}$
  - Each task $t$ has a state \{not coded, coded, tested\}
- Deliverable of a project is a product
- Product consists of a set of artifacts
  - Each artifact has two attributes \{functionality, security\}
Developer Agents

- Project P has d developers \{dev_1, dev_2, \ldots, dev_d\}
- Each developer has two skills \{code-skill, security-skill\}
- Manager assigns n tasks to each developer

- Developers have a perspective of the assigned tasks and their skills
  \[ S_{dev1} = \{
  \{t_1, \text{not coded}\}, \{t_2, \text{not coded}\}, \ldots, \{t_n, \text{not coded}\},
  \{dev_1, \text{code-skill}_1, \text{security-skill}_1\}\} \]
Developer Agents: Actions

- Developers have five actions {code, learn-code, security, learn-security, other}
  - Action **code** changes task state from not coded to coded
  - Action **security** changes task state
    - From coded to tested if no bugs are found
    - From coded to not coded if bugs are found
  - Action **learn-code** increases developer’s **code-skill**
    - Decreases time to code
    - Increases artifact’s **functionality**
  - Action **learn-security** increases developer’s **code-security**
    - Decreases time to (test) **security**
    - Increases artifact’s **security**
Developer Agents: Utility Function

- On taking any of action a, developers reach state $S'_{\text{dev1}}$
- Developers make decisions to maximize expected utility:

$$\text{arg max} \sum_s P(S'|a)U(S)$$

Utility of state based on action a
Probability of change in state if action a is taken
Developer Agents: Decision Network
Manager and Project State

- Project P has **one** manager
- Project state consists of task state and developer state

```
{t₁, not coded, dev₁}, {t₂, not coded, dev₂}, ......................
{dev₁, code-skill₁, security-skill₁},
{dev₂, code-skill₂, security-skill₂},
{dev₃, code-skill₃, security-skill₃},
......
```
Manager: Sanctions

- Manager sanctions based on the product’s state
- Product’s state
  
  \[ S_P = \{ \{\text{art}_1, \text{functionality}_1, \text{security}_1\}, \{\text{art}_2, \text{functionality}_2, \text{security}_2\}, \ldots, \{\text{art}_t, \text{functionality}_t, \text{security}_t\} \} \]

- Sanctions can be individual or group sanctions
Proposed Data Collection

- Developers decision model
  - Survey, time sheets that may explain the distribution of project time of a developer in coding, running security tools, learning and other works
  - The change in the skill level of developers for coding and running security tools at the beginning and end of a project

- Manager’s sanction model
  - Interview of managers to understand the sanction mechanism
Success Criteria

- Developing a simulation framework for the interactions of the developers and the manager
- Capturing realism in the framework
- Applying the framework for different organizational structures and skills of developers
Anticipated Difficulties, Limitations, and Criticism

- The developers and the manager make rational decisions in every time tick
- The developers’ utility from working on a project, is a function of their improvements in skills or completion of the project
- The manager doesn’t have any explicit objective function. The manager applies sanctions for timeliness of product delivery, functionality and security of the product