Pictures at QRS 2015

Day 1 (August 3)
General Pattern: Using Metaheuristics

- Model
- Objective Function
- Search Space
- Search Technique
- Problem
Insider Threat
Motivations and Challenges

- Mission-critical information = High-value target
- Threatens US and other Government organizations and large corporations
- Probability is low, but impact is severe
- Types of threat posed by malicious insiders
  - Denial of service
  - Data leakage and compromise of confidentiality
  - Compromise of integrity
- High complexity of problem
  - Increase in sharing of information, knowledge
  - Increased availability of corporate knowledge online
  - "Low and Slow" nature of malicious insiders
Our Approach:
Library Load Time Fault Injection

Injection Time:
- When does the injection take place?
  - To simulate failures by loading code in production
  - To test the system by forcing failures
- At library load time:
  - Our approach, linking against a custom error state generating library before the third-party library is loaded
Restricted Boltzmann's Machine (RBM)

RBM is the basic component of DBN. It is a two-layer network. The first is called the visible layer and the second is called the hidden layer.
Empirical software quality prediction - based on constructing and applying statistical/ML methods to historical datasets (Segev et al., 2006).

Large data repository for research, i.e. PROMISE and ISSE data collection and fault/none-response (Myrte et al., 2021).

Data missingness is a non-ignorable issue - significantly affects ability of models in knowledge discovery (Little an Rubin, 2002).
To obtain coverage, iteratively improve coverage based on the errors missed by fault tolerance mechanisms.
- Analyze the errors that are missed by the FT mechanisms.

Insert fault tolerance mechanisms in the application's source code.

Inject faults into the application that are protected with fault-tolerance mechanisms.
Software rejuvenation: A mechanism to counteract the influence of software aging.

- runtime phase:
  - Software rejuvenation
  - mitigate the effects and costs of ARBs
The Analysis Technique

- Implications
  - Only recursive data structures are considered (DAGs and general graphs are beyond our concern)
  - We only consider programs that preserve shapes of data structures
The Analysis Technique

Example: one execution of the loop body from [node] to [node]

```c
node* reverse(node* head)
{
    // [pre]
    node* q = NULL;
    while (node t = NULL) {
        // [node]
        t = p->next;
        p->next = q;
        q = t;
    }
    // [post]
    return q;
}
```
Background and Motivation

- Software Reliability and models
  - Software failure is a dynamic process and accelerated in software test.
  - Most software reliability growth models (SRGMs) utilize the fault detection record to describe the stochastic behavior of software failure in fault detection process (FDP).
  - The most popular parametric models are the non-homogeneous Poisson process (NHPP) models [1].
Introduction

- Previously works: Automated testing approach for finding vulnerabilities without or at least with little user interactions.
- Depiction of attacks as patterns [2, 3].
- Attack: A sequence of actions.
- Attack is nothing else than finding an interaction sequence that finally exploits a vulnerability acting as a planning.
Background

- Model Based Testing (MBT)
  - Nguyen et al., and Utting et al., 2012.
- Fault Modeling and Analysis
  - Tribble et al., 2004. Introduce FTA.
- Integration of Safety Analysis Techniques and Behavior
  - Ariss et al., 2011.
  - Kim et al., 2010
  - Gario et al. 2014
  - Test generation method for failures, but not Analysis only (Not Testing)
- Mitigation Modeling
  - Avizienis et al., 2004 (A Taxonomy of error handling & fault tolerant systems)
  - Lerner et al., 2010 (Identify several patterns)
- Regression Testing (RT)
  - Chen et al., 2002 (use UML activity diagrams and statecharts)
  - Orso et al., 2007 (use a statechart diagram)
  - Korel et al., 2002 (uses dependence analysis of a given program to reduce the size of a regression test suite (RTS)).
  - Chen et al., 2007 differ from Korel et al., in terms of the modified transition and how they define dependence...
Tribble et al., 2004. Introduce FTA
Integration of Safety Analysis Techni
Ariss et al., 2011,

Test generation meth

Identification Modeling
Ariss et al., 2004 (A Taxonomy of err
Over et al., 2010) Identify several patter
Regression Testing (RT)
Over et al., 2002 (use UML activity
Over et al., 2007 (use a statechart d
Over et al., 2002 (uses dependency
size of a regression test s
Over et al., 2007 differ from Korel t
transition and how they)
BACKGROUND (2/2)

- **Non-homogeneous Poisson Process Model**
  Modeling of the cumulative number of software faults

- **Parametric model**
  The form of mean value function of NHPP is known in advance

- **Nonparametric model**
  Estimate the mean value function with data directly

---

Propose a constrained nonparametric maximum likelihood estimator (CNPMLE) of mean value function of NHPP
BACKGROUND (2/2)

- **Non-homogeneous Poisson Process Model**
  - Modeling of the cumulative number of software faults
  - **Parametric model**
    - The form of mean value function of NHPP is known in advance
  - **Nonparametric model**
    - Estimate the mean value function with data directly

\[ \text{Cumulative number of faults} \]
\[ \text{Testing Time} \]

- Boswell (*Annals of Mathematical Statistics (1966)*)
- Propose a constrained nonparametric maximum likelihood estimator (CNPML) of mean value function of NHPP
RELATED WORK (NHGP MODEL)

(a) \textit{Stochastica} (1981)

Non-homogeneous gamma process (NHGP) characterized as a modulated gamma renewal trend function

(b) PRDC (2008)

Several NHGP-based SRMs and compare NHGP models having eight trend functions in NHPP models
Introduction

As the functional requirements of software keep growing, the large scale to solve various practical problems, and the growing need for researchers to quantify the behavior of software.

Software, as a typical artifact complex system, is treated as an abstract framework of which are represented in various ways. By decomposing modules or other entities into sub-modules, the inter-communicational entities as edges and attaching various attributes of interest such as structure of software data essentially represent a form of interconnected contents.

Since the complex network models of software are capable of describing not only the entities or system granularity or structure of software are available as well.
Background

- Defects are inevitable in software systems
Related Works

Utilize metrics of testing activities
(test coverage, the number of test cases, etc.)

- Cox Regression + SRGM
  → Shibata et al., ISSRE 2006

- Logistic Regression + SRGM
  → Okamura et al., ISSRE 2010

Utilize design metrics of software
(lines of code, complexity, etc.)

- Poisson Regression + SRGM
  → Okamura et al., HASE 2014
On the Viability of Using SRGMs for IT Help Desk Incident Predictions
BBC: Brute-Force Boolean Combination
Contents

- Introduction
  - Scientific Workflow Management Systems (SWFMS)
  - Security of SWFMS to the Cloud
- Kepler SWFMS
  - Workflow and components
- Securing Kepler
  - Problem Statement
  - Investigation: Kepler, Provenance
  - Security Analysis Package (SAP)
  - A Prototype of Secure Kepler
- Conclusion
Prioritization factors

Failure Impact

- user-driven measure of the severity of defects
- calculated based on historical user feedback reports collected from previous versions of system

\[ F_i(TC) = \frac{\sum_{i=1}^{n} F_i}{Max_F_i} \]

- Max-Fi: test case maximum failure severity
- n: failure indexes that a test case has detected
- Fi: failure impact of a test case
Combinatorial Testing

Parameter values:
- CPU: Intel, AMD
- OS: Windows, Linux, Mac
- Browser: IE, Firefox, Safari

Constraints:
- (IE ⇒ Win) \land (Safari ⇒ Mac) \land (Mac ⇒ \neg AMD)

<table>
<thead>
<tr>
<th>Test</th>
<th>CPU</th>
<th>OS</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intel</td>
<td>Windows</td>
<td>IE</td>
</tr>
<tr>
<td>2</td>
<td>AMD</td>
<td>Windows</td>
<td>Firefox</td>
</tr>
<tr>
<td>3</td>
<td>Intel</td>
<td>Windows</td>
<td>Safari</td>
</tr>
<tr>
<td>4</td>
<td>AMD</td>
<td>Linux</td>
<td>IE</td>
</tr>
<tr>
<td>5</td>
<td>Intel</td>
<td>Linux</td>
<td>Firefox</td>
</tr>
</tbody>
</table>

An SUT model:

Test suite
5. FoCuS
   • by IBM, 2010
   • Algorithm using BDD

6. CASCADE
   • by Chinese Academy of Sciences, 2012
   • using Pseudo-Boolean

7. Calot
   • AIST, 2015
   • Minimization of pair-wise tests
   • Using incremental SAT

8. CTE(Classification Tree Editor)
   • Tree-structured SUT model, 1993
   • By Mercedes, Berner and Mattner
Outline

- Dynamic Software Update (DSU)
- Event-based formal model of DSU
  - how to understand
  - how to formalize
- Optimistic DSU with runtime monitor
Inevitable Software Updates

Fix Bugs
Enhance Functionality

Change interface as requirements evolve
Conclusion and further work:

- For complex manipulation, probability finding is an interesting area.
- Future work:
  - Reducing size of the problem
  - Modifying reproduction
  - Modifying fitness
  - Using other MOE algorithms
  - Using symbolic regression algorithms
An Adjustable Risk Assessment for a Cloud System

Chi-An Chih, Yu-Lun Huang
Institute of Electrical & Control Engineering, National Chiao-Tung University
Introduction

- Cloud computing provides many advantages:
  - Reduced capital cost
  - Quick deployment
  - Less personnel training

- Security is currently still a big problem for wide-scale adoption:
  - Data security
  - Ease of control
  - Service availability
If \((x == y)\) then
Coverage criterion \(A = X\)

Coverage criterion \(A\) is not effective.
Contents

• Introduction
Optimizing Translations of Set Operators

- \( C \cap D \rightarrow \forall x (x \in C \land x \in D) \)
- \( X \setminus Y \rightarrow \forall x (x \in X \land x \notin Y) \)
- \( X - Y \rightarrow \forall x (x \in X \land x \notin Y) \)
- \( X \cup Y \rightarrow \forall x (x \in X \lor x \in Y) \)

If \( v \) is a machine variable, \( I \) is the identifier of \( a \) and \( t \) is the table holding sets of \( a \)'s type:

- \( v = t \)
  - For each \( v \) in \( t \)
    - Concept: square only if \( v \) is relevant, \( w \) in.
    - \( t = t \land v \leftarrow w \)
    - \( t = t \land v \leftarrow \) delete from \( t \) where \( e \) is valid and \( t \)
    - \( t = t \land v \leftarrow \) delete from \( t \) where \( e \) is valid and \( t \)
Software Trustworthiness

Hasselbring, Reussner: Software trustworthiness is based on all attributes of software as accuracy, reliability, safety, timeliness, availability, predictability and other attributes of software quality.

Outline

- Fault localization techniques
  - Binary similarity fault localization
  - Network fault localization
    - Traffic parameter
- Single fault v.s multiple fault localization
- SFL v.s NFL
Clock expression

Let $c, d$ be two clocks.

- $c \leq d$ means that clock $c$ precedes clock $d$, which means that every observation point of clock $c$ is earlier than or equal to the related observation point in $d$.
- $c < d$ means that clock $c$ strictly precedes clock $d$, which means that every observation point of clock $c$ is strictly earlier than the related observation in $d$.
- $c = d$ represents that clock $c$ and $d$ are synchronized with each other.
- $c \neq d$ represents that the clock $c$ is synchronized with clock $d$. Otherwise, at the same time.