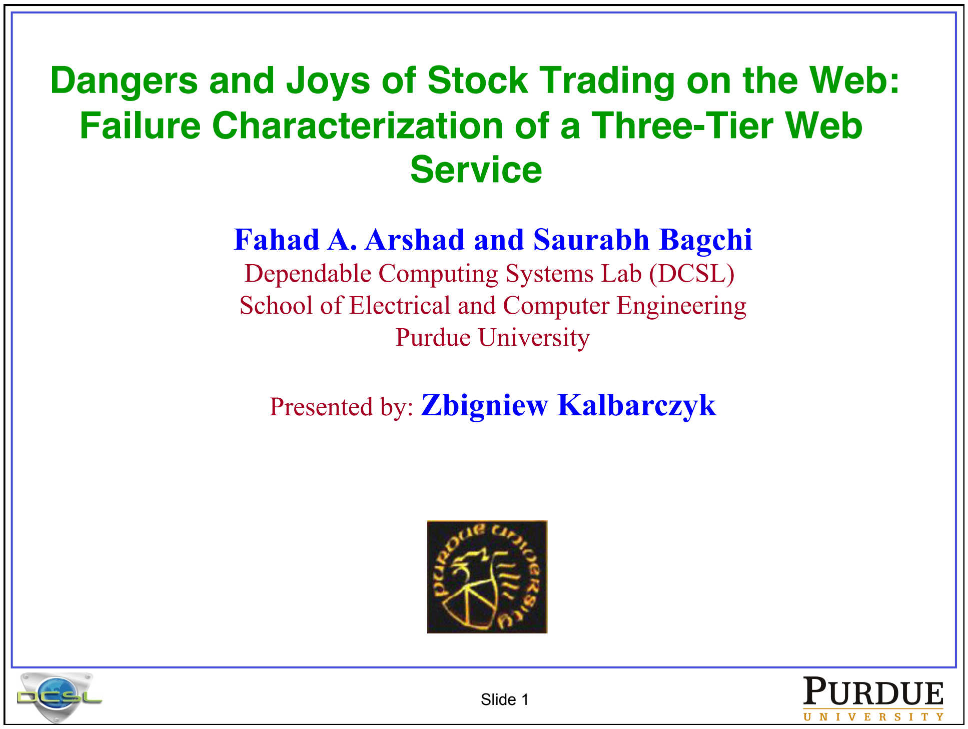
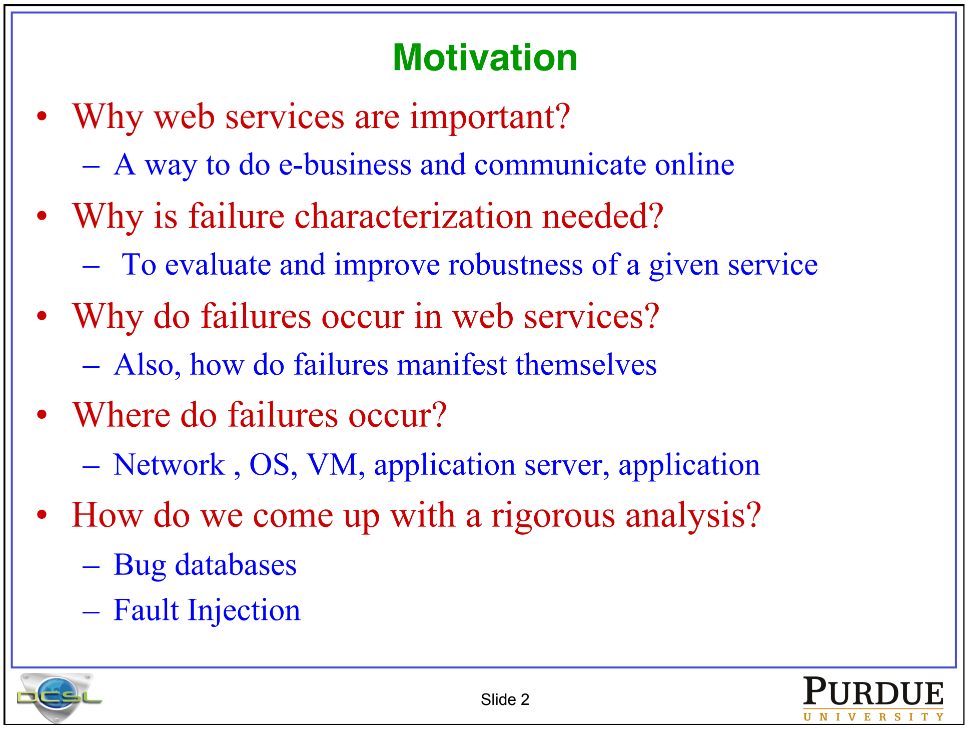


# Dangers and Joys of Stock Trading on the Web: Failure Characterization of a Three-Tier Web Service

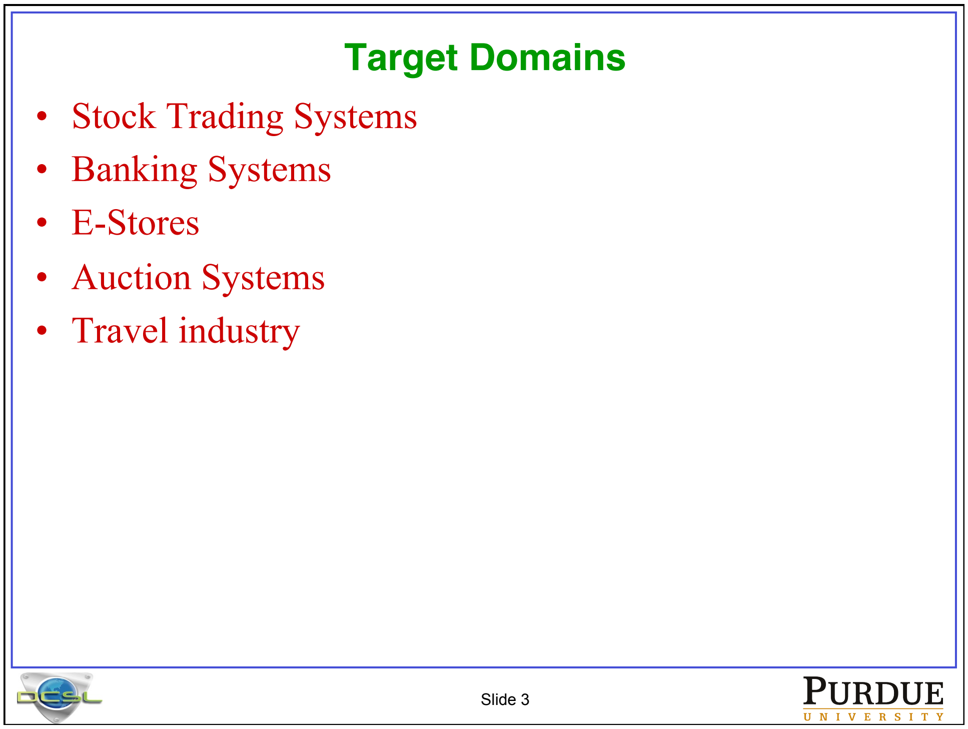
**Fahad A. Arshad and Saurabh Bagchi**
Dependable Computing Systems Lab (DCSL)
School of Electrical and Computer Engineering
Purdue University

Presented by: **Zbigniew Kalbarczyk**



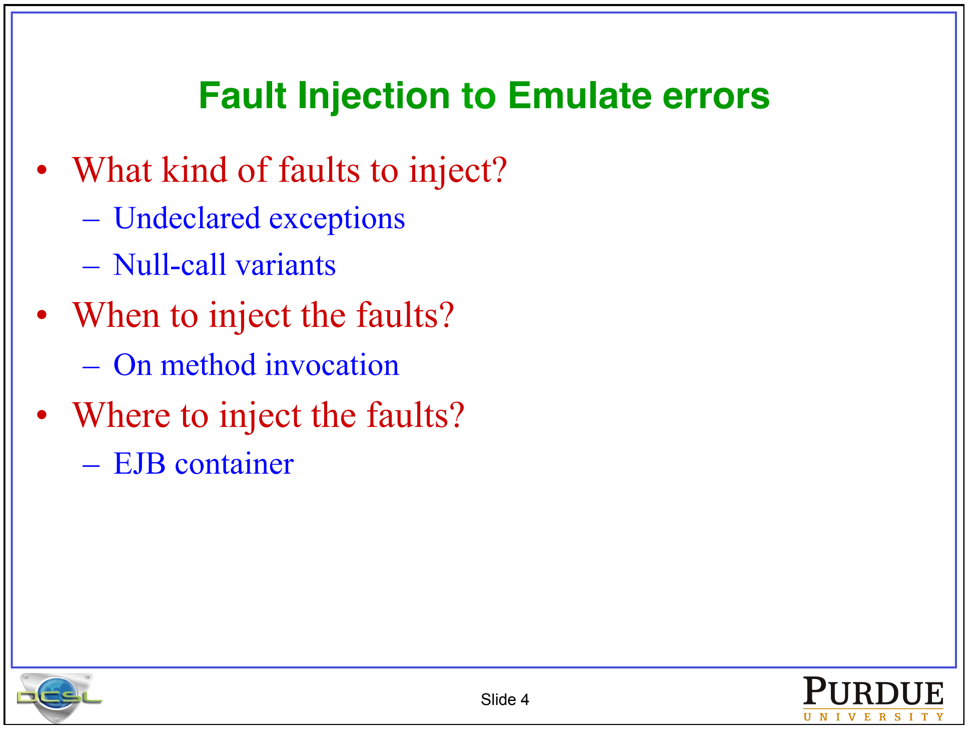
## Motivation

- Why web services are important?
  - A way to do e-business and communicate online
- Why is failure characterization needed?
  - To evaluate and improve robustness of a given service
- Why do failures occur in web services?
  - Also, how do failures manifest themselves
- Where do failures occur?
  - Network , OS, VM, application server, application
- How do we come up with a rigorous analysis?
  - Bug databases
  - Fault Injection



## Target Domains

- Stock Trading Systems
- Banking Systems
- E-Stores
- Auction Systems
- Travel industry



## Fault Injection to Emulate errors

- What kind of faults to inject?
  - Undeclared exceptions
  - Null-call variants
- When to inject the faults?
  - On method invocation
- Where to inject the faults?
  - EJB container

## Fault Injection : Null-call and Unchecked Exceptions

- Null-call
  - Null-Return
  - Null-Object-Return
  - No-Op
- Unchecked Exceptions
  - Arithmetic Exception
  - IndexOutOfBounds Exception
  - ClassCast Exception

## Fault Emulation : example code

```
foo(){
...
RObject x = bar();
...
}
RObject bar(){
...
return RObject;
}
```

**Original Code**

```
foo(){
...
//RObject x = bar();
Class RObjectClass = RObject.getClass();
RObject x = RObjectClass.cast(null);
...
}
```

**Null-Object-Return Code**

```
foo(){
...
//RObject x = bar();
RObject x = null;
...
}
```

**Null-Return Code**

```
foo(){
...
//RObject x = bar();
RObject y = new RObject();
RObject x = y;
...
}
```

**No-Op Code**

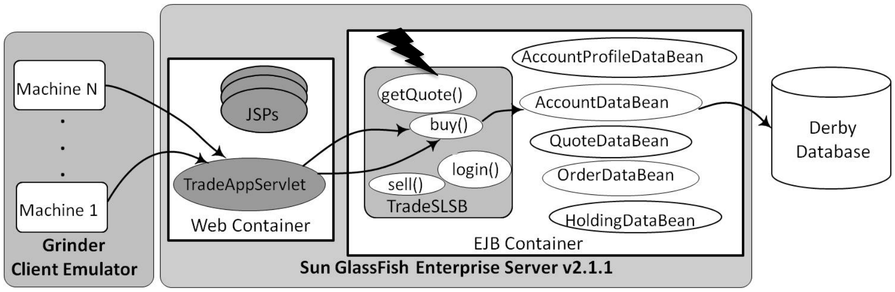
```
foo(){
...
RObject x = bar();
...
}
```

```
RObject bar(){
throw new java.lang.RuntimeException();
...
return RObject;
        }
```

**Unchecked Exception Code**

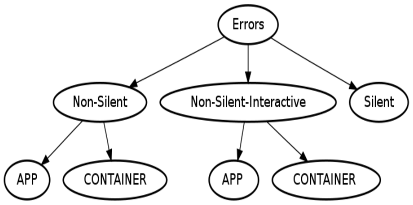
## Application: Three-tier Web Service (DayTrader)

- Front-end presentation in web container
- Middle-tier business logic in EJB container
- Back-end data-source in Derby database
- Faults injected in business logic in EJB container



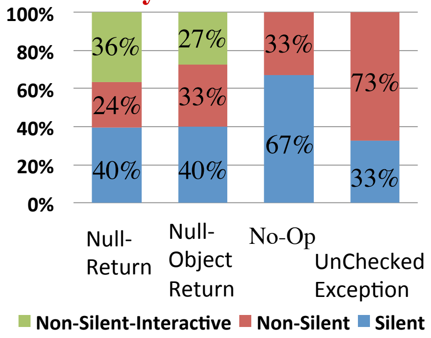
## Failure Manifestation and Classification

- Non-silent
  - Explicit error messages from infrastructure to user
  - E.g.: HTTP 5xx, blank page
- Non-silent-interactive
  - Partially correct response,
    only noticeble interactively
  - 3 results returned for 5 *getQuote* requests
- Silent
  - Unnoticeable to user or admin
  - E.g.: I buy 100 stocks of IBM and tomorrow I do not find them in my portfolio
  - Most worrisome class

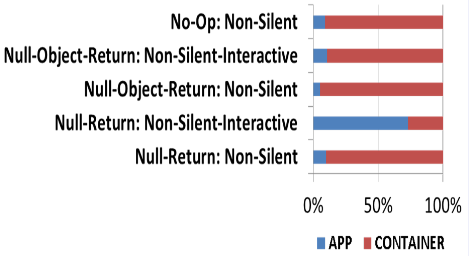


# Failure Distribution

## By Error Class



## By where failures are logged

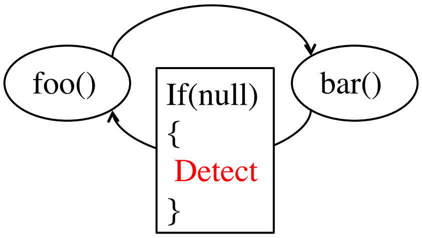


- No-Op causes significant silent errors
- Unchecked Exceptions causes significant cause non-silent errors

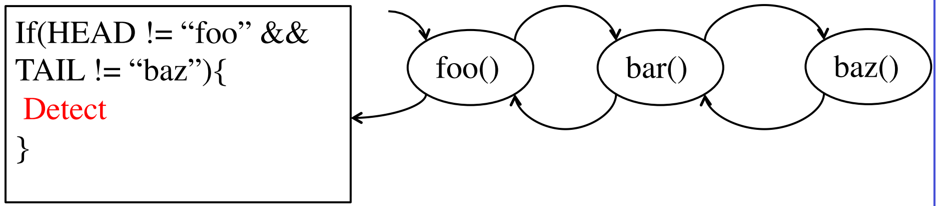
- Majority of non-silent failures are logged by CONTAINER

# How to detect these failures ?

- Detection Checks
  - Application Generic
    - Null-call check
  - Application Specific
    - Call-Length check
    - Head-Tail check



Normal Call-length = 3



## ACCURACY: Application Generic Null-call Check

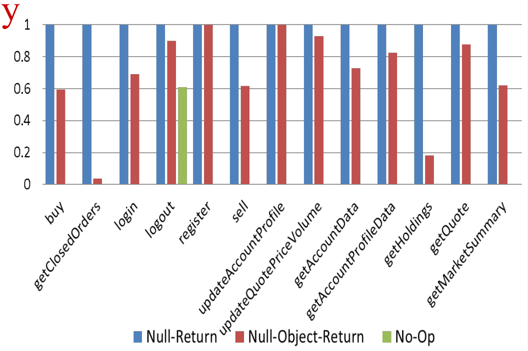
- Null-Return is caught in all 3 failure classes

(100% accuracy)

- No-Op is not detected by this check except for Logout

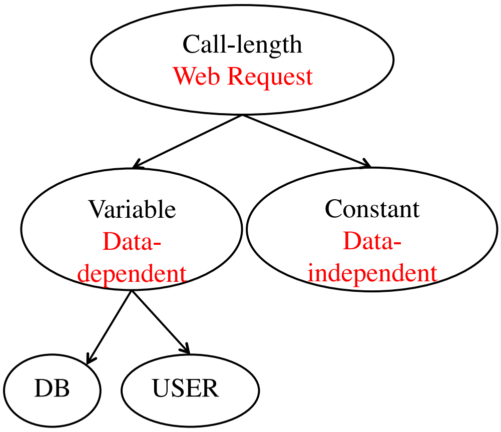
Accuracy of Null-Call Check

| Type of Injection | Failure Class | Accuracy |
|---|---|---|
| Null-Return | Silent | 100% |
| | Non-Silent | 100% |
| | Non-Silent-Interactive | 100% |
| Null-Object-Return | Silent | 30% |
| | Non-Silent | 22% |
| | Non-Silent-Interactive | 35% |
| No-Op | Silent | 1% |
| | Non-Silent | 0% |
| | Non-Silent-Interactive | 0% |



## Application Specific: Call-Length Check

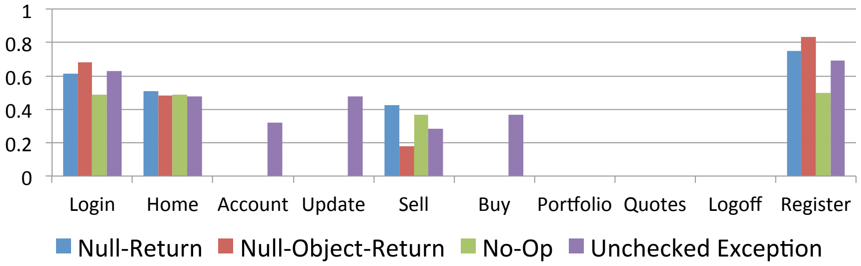
- Data-independent
  - Login=5
- Data-dependent
  - Variable call-length, due to different number of stocks owned or searched by a user
  - DB: Portfolio ≥ 2
  - USER: Quotes ≥ 1



## ACCURACY: Application Specific Call-Length Check

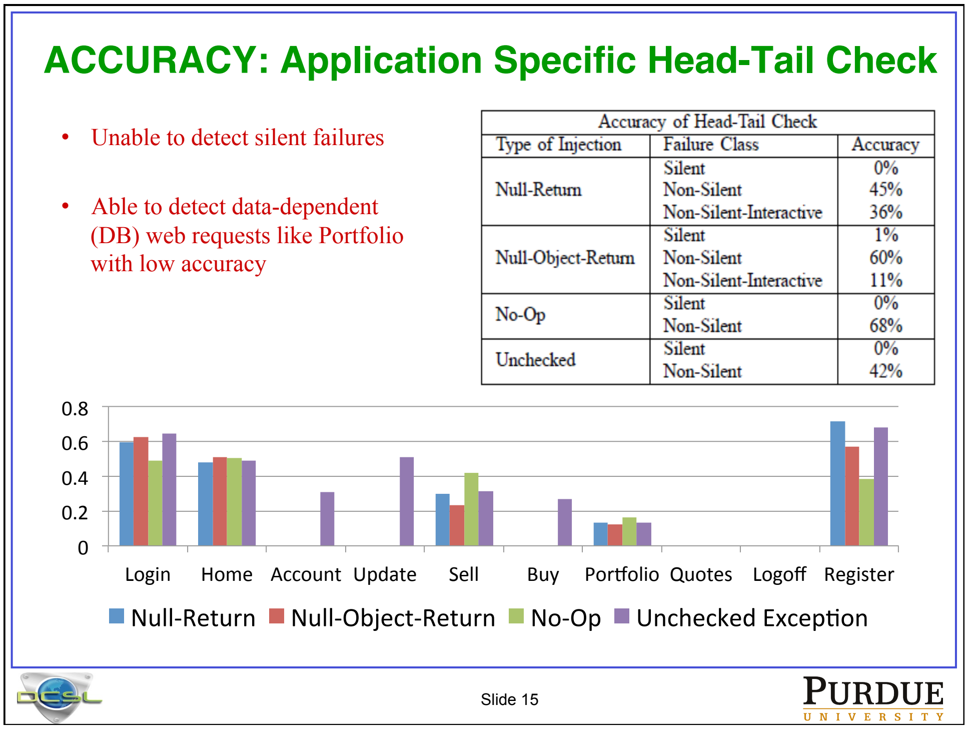
- Low detection accuracy for silent failures
- 60% of non-silent failures from No-Op detected
- Data-dependent web requests not detected

| Accuracy of Call-Length Check | | |
|---|---|---|
| Type of Injection | Failure Class | Accuracy |
| Null-Return | Silent | 0% |
| | Non-Silent | 48% |
| | Non-Silent-Interactive | 30% |
| Null-Object-Return | Silent | 1% |
| | Non-Silent | 57% |
| | Non-Silent-Interactive | 11% |
| No-Op | Silent | 0.2% |
| | Non-Silent | 60% |
| Unchecked | Silent | 0% |
| | Non-Silent | 41% |



## Application Specific: Head-Tail Check

- Match first and last EJB request names
  - **`getClosedOrders`** → `getHoldings` → `getQuote` → `getQuote` → **`getQuote`**
- Implemented using `ThreadLocal` API
- Able to detect some data-dependent requests i.e. “Portfolio”
  - Detects Portfolio only when target of injection is either **`getClosedOrders`** or **`getHoldings`** and request is cut-short.
  - Expected tail ejb-request is `getQuote` from learning which will fail to satisfy the check if the web request is cut short.

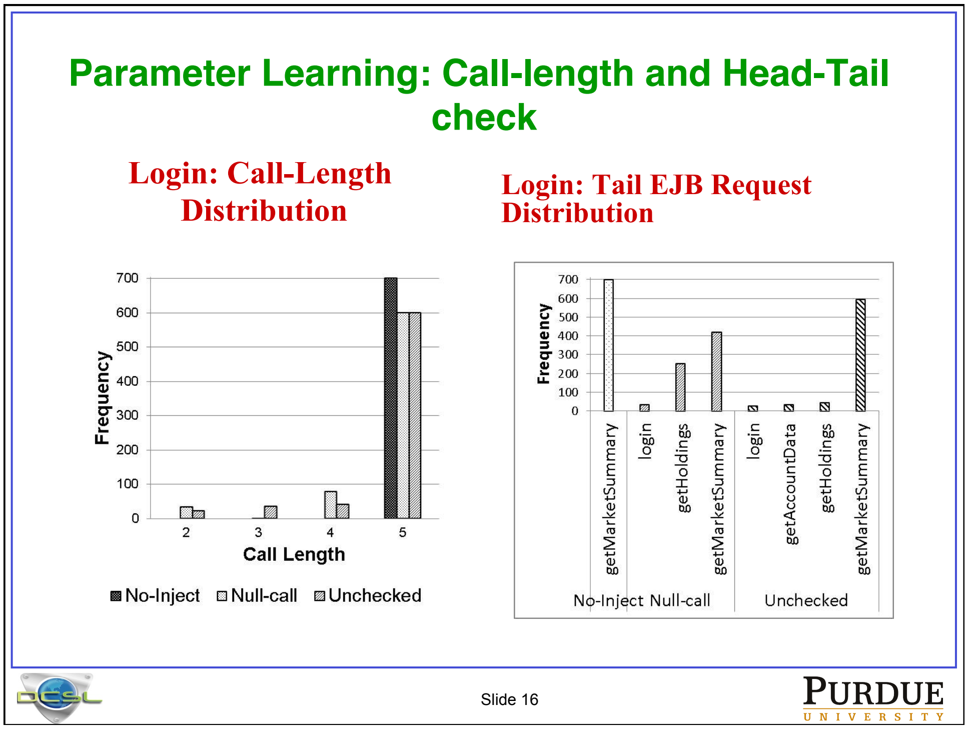


# ACCURACY: Application Specific Head-Tail Check

- Unable to detect silent failures
- Able to detect data-dependent (DB) web requests like Portfolio with low accuracy

Accuracy of Head-Tail Check

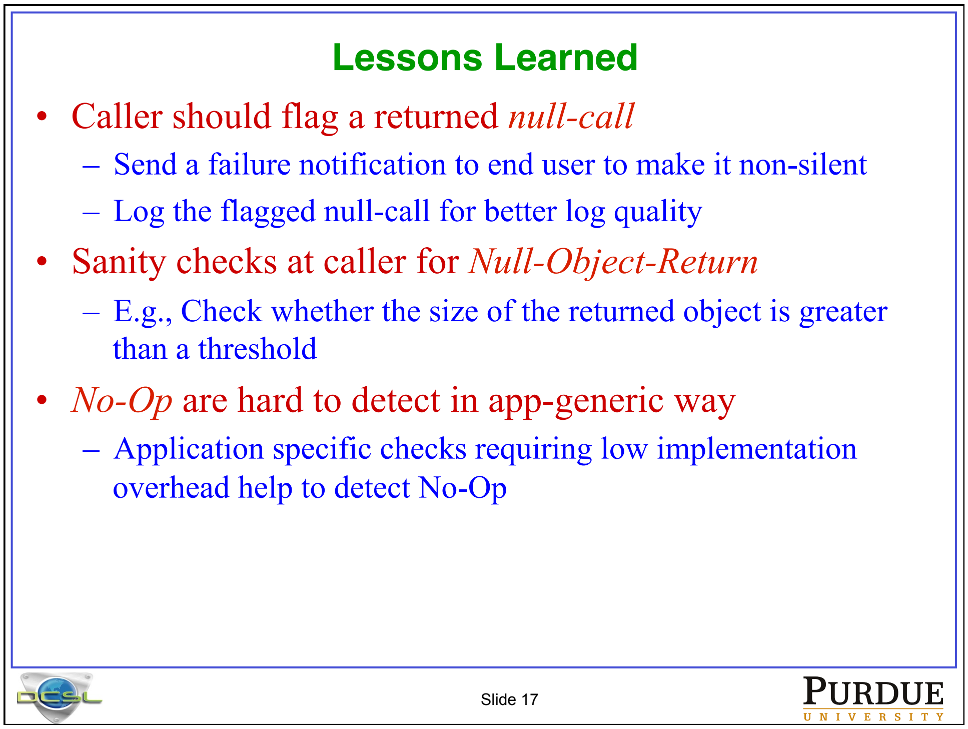
| Type of Injection | Failure Class | Accuracy |
|---|---|---|
| Null-Return | Silent<br>Non-Silent<br>Non-Silent-Interactive | 0%<br>45%<br>36% |
| Null-Object-Return | Silent<br>Non-Silent<br>Non-Silent-Interactive | 1%<br>60%<br>11% |
| No-Op | Silent<br>Non-Silent | 0%<br>68% |
| Unchecked | Silent<br>Non-Silent | 0%<br>42% |



# Parameter Learning: Call-length and Head-Tail check

**Login: Call-Length Distribution**

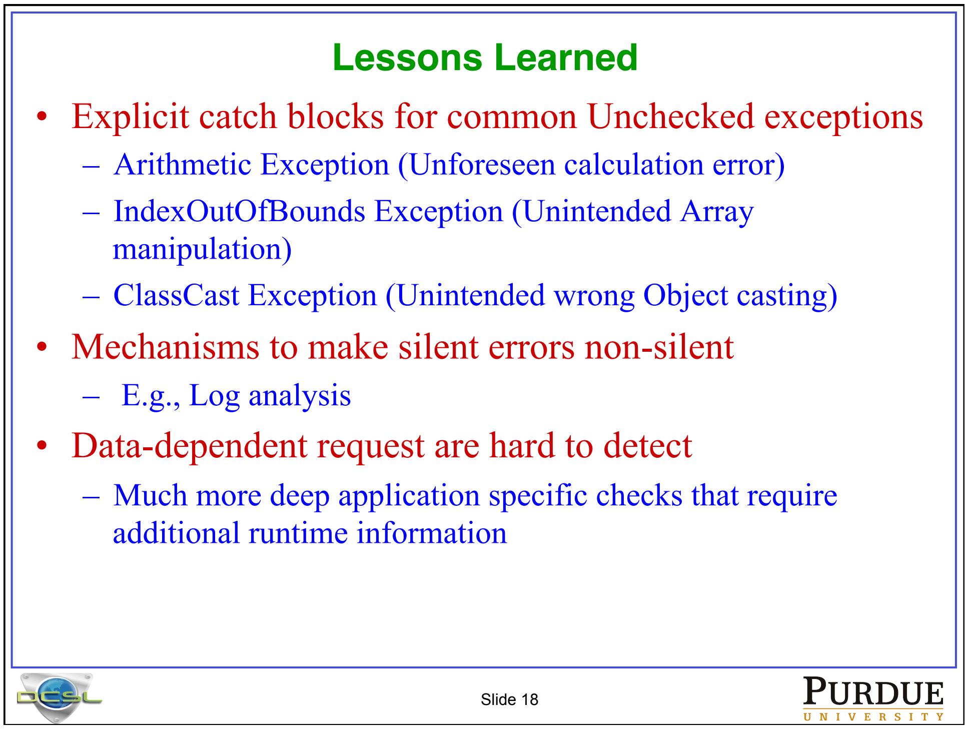
**Login: Tail EJB Request Distribution**



## Lessons Learned

- Caller should flag a returned *null-call*
  - Send a failure notification to end user to make it non-silent
  - Log the flagged null-call for better log quality
- Sanity checks at caller for *Null-Object-Return*
  - E.g., Check whether the size of the returned object is greater than a threshold
- *No-Op* are hard to detect in app-generic way
  - Application specific checks requiring low implementation overhead help to detect No-Op

PURDUE
UNIVERSITY



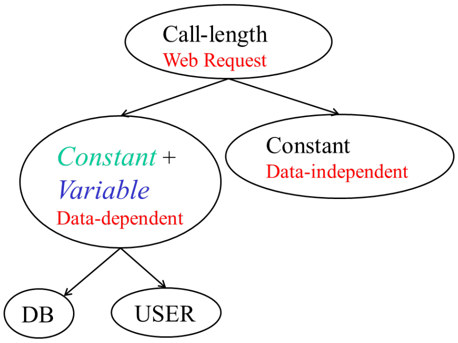
## Lessons Learned

- Explicit catch blocks for common Unchecked exceptions
  - Arithmetic Exception (Unforeseen calculation error)
  - IndexOutOfBounds Exception (Unintended Array manipulation)
  - ClassCast Exception (Unintended wrong Object casting)
- Mechanisms to make silent errors non-silent
  - E.g., Log analysis
- Data-dependent request are hard to detect
  - Much more deep application specific checks that require additional runtime information

PURDUE
UNIVERSITY

## Future Directions: Detect Data-dependent Requests

- Each data-dependent request has a *constant* and a *variable* part in normal case
  - **Portfolio**:getClosedOrders()->getHoldings->getQuote()*
- Extract data-dependent information at runtime
- Match expected runtime call-length to observed call-length



## Future Directions

- Failures due to concurrency in java based web services
- Failures due to using different design and architectural patterns in a given three-tier web service
- Identifying design patterns that lead to robust web services

# Thank you

# Backup Slides

## Implementation: Call-Length Check

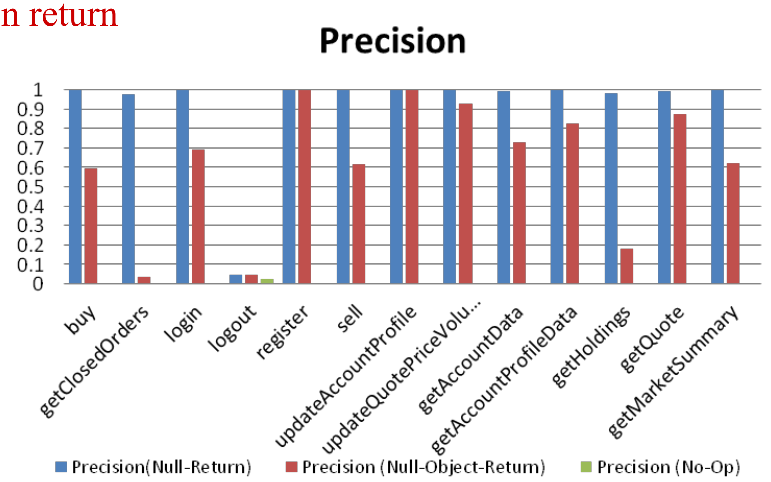
- Use `ThreadLocal` API
- Monitor no of EJB requests (call-length) invoked for a given web request
- Detect at the end of a given web request

```
[sell : getClosedOrders, sell, updateQuotePriceVolume]
[quotes : getClosedOrders, getQuote]
[update_profile : getClosedOrders, updateAccountProfile, getAccountData, getAccountProfileData]
[quotes : getClosedOrders, getQuote, getQuote, getQuote, getQuote]
[register : register, login, getAccountData, getHoldings, getMarketSummary]
[login : getClosedOrders, login, getAccountData, getHoldings, getMarketSummary]
[buy : getClosedOrders, buy, updateQuotePriceVolume]
[logout : logout]
[home : getClosedOrders, getAccountData, getHoldings, getMarketSummary]
[portfolio : getClosedOrders, getHoldings, getQuote, getQuote, getQuote]
[account : getClosedOrders, getAccountData, getAccountProfileData]
[portfolio : getClosedOrders, getHoldings, getQuote]
```

Sample Web and EJB requests

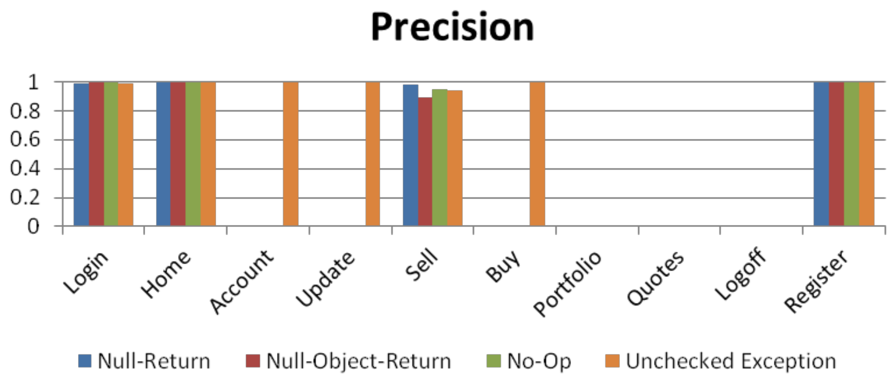
## Application Generic: Null-call Check (PRECISION)

- Logout results in low precision since it expects a **`null`** on return



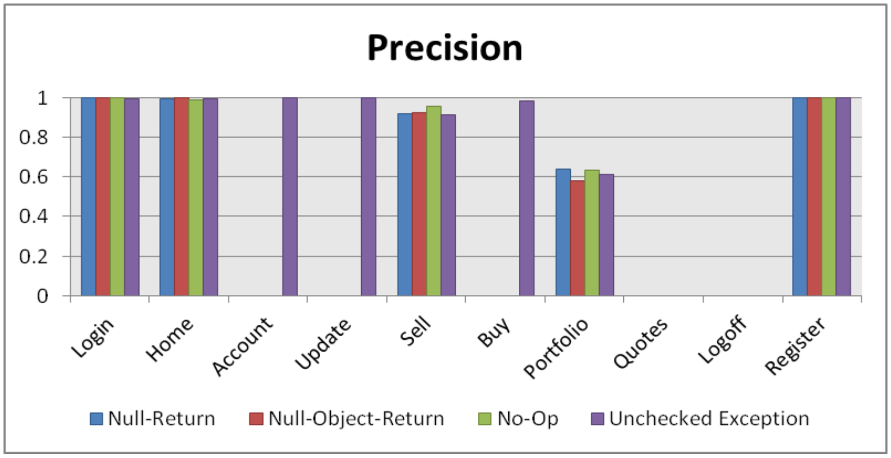
## PRECISION of Application Specific Call-Length Check

- Data-dependent web requests like portfolio and quotes are not detected.



## Application Specific Head-Tail Check PRECSION

- DB Data-dependent web requests like portfolio are detected with reasonable precision.


Precision
Login
Home
Account
Update
Sell
Buy
Portfolio
Quotes
Logoff
Register
Null-Return
Null-Object-Return
No-Op
Unchecked Exception