Mobile Testing-as-a-Service (Mobile TaaS or MTaaS)

Jerry Gao, Ph.D. Professor
Computer Engineering Department
San Jose State University

W. T. Tsai, Ph.D. Professor
Sch Compt Infor & Dec Sys Engr
Arizona State University
Table of Contents

• Introduction
• What is Mobile TaaS?
• Why is Mobile TaaS Important?
• Mobile Testing Process
• Major Players and Tools
• Mobile Testing Approaches, Infrastructures, and Comparisons
• Mobile TaaS Approaches, Infrastructures, and Comparisons
• Expected Benefits and Features
• Test Models and Test Coverage Criteria
• User View of TaaS
• Cloud Computing View of TaaS
• Issues and Needs for Future Research
According to the latest study from Juniper Research, the market for cloud-based mobile applications will grow 88% from 2009 to 2014. The market was just over $400 million this past year, says Juniper, but by 2014 it will reach $9.5 billion. Driving this growth will be the adoption of the new web standard HTML5, increased mobile broadband coverage and the need for always-on collaborative services for the enterprise.
Introduction – Market Needs on Mobile

Figure 1: Of People With Smartphones, 33% Download Applications At Least Monthly

“Which of the following activities do you do on your primary cell phone or handheld wireless device at least monthly?”

- Receive SMS/text alerts: 70%
- Send/receive MMS (picture messages): 54%
- Check news/sports/weather: 53%
- Look up directions or maps: 50%
- Send/receive instant messages (e.g., MSN): 41%
- Send or receive work email: 38%
- Download applications: 33%
- Check financial accounts: 29%
- Research products for purchase: 23%
- Check flight, bus, or train status: 12%
- Enter a contest or vote via SMS/text messages: 9%
- Receive coupons/promotions: 8%
- Purchase products: 8%
- Access operator portal (e.g., AT&T MediaNet, Rogers Zone): 5%
- None of these: 7%

Base: 4,674 smartphone users

Source: North American Technographics® Benchmark Survey, Q2 2010 (US, Canada)

Figure 2: Global Smartphone And Tablet App Shipments In US$
Knowledge Background for Mobile TaaS Research

- Software Testing and Test Automation
- Mobile Computing
- Cloud Computing & SaaS Engineering
What Is Mobile TaaS?

No existing definition for Mobile TaaS.

Our first definition: (Mobile TaaS in a cloud infrastructure):

“Mobile Testing as a Service (known as Mobile TaaS) provides on-demand testing services for mobile applications and SaaS to support software validation and quality engineering processes by leveraging a cloud-based scalable mobile testing environment to assure pre-defined given service-level-agreements (SLAs).

Mobile TaaS on clouds offers a new business model for diverse mobile software validation services using the pay-as-you-test model to achieve cost-sharing and cost-reduction in mobile computing resources, networks, cloud computing and storage infrastructures.”
Mobile Testing Scope

- Multi-Tenancy Testing on Mobile Clouds
- On-Demand Mobile Test Run & Control
- Mobile Test Simulation and Virtualization
- Function and Behavior Testing
- Compatibility, Interoperation ability and Usability Testing
- Internationalization Testing on Mobile
- QoS Testing (Performance, load, Scalability..)
- Mobility and Connectivity Testing
Mobile TaaS Requirements and Needs

- Test Project-Oriented Multi-Tenancy & Customization Support
- On-Demand Scalable Mobile Test Infrastructures and Environments
- Control and Configuration Of Mobile Test Environments
- Easy Interoperation ability, Mobility & Connectivity
- Testing at Anywhere & Anytime (365/7/24) for Mobile App/Web App.
- Contracting, Utilization Billing and Reporting
- Large-Scale On-demand Mobile Test Services
- Test Tracking, Monitoring, and Coverage Analysis
Why Is MTaaS Important?

High costs on mobile test infrastructures due to:
- Diverse mobile devices and HW appliances & APIs
- Diverse mobile operation environments (platforms, connectivity, and configurations)
- Diverse wireless connectivity and configurations

Frequent changes and upgrades on:
- Mobile devices, appliances, and their APIs
- Mobile applications, features, and service plans
- Mobile platforms, technologies, input solutions

Complex mobile user interfaces
- Diverse mobile browsers and technologies
- Input approaches, display screens
- Multi-language mobile contents

Large-scale on:
- On-demand mobile test service requests
- Mobile test simulation and traffic loads
- Virtualization of mobile devices and environments

Multi-tenancy of mobile app/web applications
- Functions, behaviors, and QoS REQs
- Data, security features, user interfaces
Why Is MTaaS Important?

Crashes by OS Version Normalized (12/1 - 12/15)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Language(s)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Java, C, C++</td>
<td>Open Source OS (based on Linux) developer.android.com</td>
</tr>
<tr>
<td>bada</td>
<td>C, C++</td>
<td>Samsung’s mobile platform running on Linux or RealTime OS developer.bada.com</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>Java</td>
<td>J2ME compatible, extensions enable tighter integration n2.blackberry.com/eng/developers</td>
</tr>
<tr>
<td>iOS</td>
<td>Objective-C, C</td>
<td>Requires Apple Developer Account developer.apple.com/iphone</td>
</tr>
<tr>
<td>MeeGo</td>
<td>Qt, Web Apps, C++, others</td>
<td>Intel and Nokia guided open source OS (based on Linux) meego.com/developers</td>
</tr>
<tr>
<td>Symbian</td>
<td>C, C++, Java, Qt, WebApps, others</td>
<td>Open source OS built from the ground up for mobile devices <a href="http://www.forum.nokia.com/symbian">www.forum.nokia.com/symbian</a></td>
</tr>
<tr>
<td>webOS</td>
<td>HTML, CSS, JavaScript, C</td>
<td>Supports widget style programming, (based on Linux) developer.palm.com</td>
</tr>
<tr>
<td>Windows Mobile</td>
<td>C#, C</td>
<td>.NET CF or Windows Mobile API, most devices ship with J2ME compatible JVM developer.windowsmobile.com</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>C#, VB.NET</td>
<td>Silverlight, XNA frameworks create.msdn.com</td>
</tr>
</tbody>
</table>
Mobile Web vs. Native Apps: Differences in Testing

Native Apps
- Testing begins with app installation and launch
- Testing on some mobile devices requires access to a device ID
- Functionality and Usability need to be tested on multiple devices. Consider:
  - Operating System and version
  - Screen size
  - Custom themes
  - Interruptions
- Many native apps have access to additional APIs. Those connections need to be tested.

Mobile Web
- No installation required
- How does the site render in different browsers
  - iOS runs Safari
  - Android runs a stock browsers, but also supports Chrome, Dolphin, Opera Mini and third party browsers
  - Windows Phone runs IE
  - Blackberry runs a native browser, Opera Mini or Bolt
- Requires an internet connection,
  - Connectivity varies by location
- Load time is extremely important to mobile web users (source): 60% of users expect sites to load within three seconds, 74% of users will only wait five seconds
## Major Players and Tools in Mobile Testing

<table>
<thead>
<tr>
<th></th>
<th>MIT</th>
<th>MonkeyTal</th>
<th>seeTestMob</th>
<th>in</th>
<th>Neoload</th>
<th>SandStorm</th>
<th>MobileClou</th>
<th>Sikuli</th>
<th>Calabash</th>
<th>eggPlant</th>
<th>MonkeyRu</th>
<th>Robotiu</th>
<th>Dollop</th>
<th>Selenium</th>
<th>Android</th>
<th>WebDriver</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing Focus</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional Testing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Regression Testing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Performance Testing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Load Testing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Platform Support</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Windows</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mac</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Tests on Mobile O.S</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>iOS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Windows</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Testing Mobile Web Apps</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Testing Native Apps</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Source code based testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tests done using</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emulator</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Real Device</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Script languages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Jython</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ruby</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Perl</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Record &amp; Replay</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>License / Subscription</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Open Source</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Mobile Testing Environments and Infrastructures

Emulation-Based Testing

Simulation-Based Mobile Testing

Device-Based Mobile Testing

Remote Device-Based Mobile Testing
## Conventional Mobile Testing Approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Function Testing</td>
<td>Limitations on device-specific functions</td>
<td>Limitations on device-specific functions</td>
<td>Support</td>
<td>Support</td>
</tr>
<tr>
<td>End-to-End System Behavior Testing</td>
<td>w/o real mobile device behaviors</td>
<td>w/o real mobile device behaviors</td>
<td>Support</td>
<td>Support</td>
</tr>
<tr>
<td>Mobile Compatibility &amp; Interoperability Testing</td>
<td>Very limited</td>
<td>Very limited</td>
<td>Require diverse mobile devices and platforms</td>
<td>Require diverse mobile devices and platforms</td>
</tr>
<tr>
<td>Mobile GUI and Usability Testing</td>
<td>Limited w/o devices</td>
<td>Limited w/o devices</td>
<td>Good for real device experience</td>
<td>Good for real device experience</td>
</tr>
<tr>
<td>Load Testing</td>
<td>Require server-oriented load testing solutions</td>
<td>Easy to generate simulation-based system loads</td>
<td>Costly to generate real mobile traffic loads</td>
<td>Costly to generate real mobile traffic loads</td>
</tr>
<tr>
<td>Wireless Connectivity Testing</td>
<td>Limited coverage</td>
<td>Limited coverage</td>
<td>Good coverage but require local wireless network connection control</td>
<td>Good coverage, but require remote wireless network connection control</td>
</tr>
<tr>
<td>Operation Costs</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Download &amp; deployment</td>
<td>Not support for Mobile Native App</td>
<td>Not support for Mobile Native App</td>
<td>Support</td>
<td>Support</td>
</tr>
</tbody>
</table>
# Conventional Mobile Testing Approaches

<table>
<thead>
<tr>
<th>Perspectives/Different Approaches</th>
<th>Emulation-Based Testing</th>
<th>Simulation-Based Testing</th>
<th>Mobile Device Based Testing</th>
<th>Remote Device Based Mobile Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability Testing</td>
<td>Limited Support</td>
<td>Limited Support</td>
<td>Not Support</td>
<td>Not Support</td>
</tr>
<tr>
<td>Performance Testing</td>
<td>Check emulation-based mobile system performance</td>
<td>Check simulation-based mobile system performance</td>
<td>Check mobile device user and system performance</td>
<td>Check remote mobile device and system performance</td>
</tr>
<tr>
<td>Multi-Tenancy Testing</td>
<td>Limited coverage on multi-tenanted GUI features for specific devices</td>
<td>Limited coverage on multi-tenanted GUI features for specific devices</td>
<td>Supported with very high costs</td>
<td>Supported with very high costs</td>
</tr>
<tr>
<td>Mobility and Location-based Testing</td>
<td>Limited testing based on configured locations</td>
<td>Limited testing based on simulated locations</td>
<td>Limited to signed device testing</td>
<td>Limited to signed device testing</td>
</tr>
<tr>
<td>Mobile Security Testing</td>
<td>Limited coverage on real mobile OS/platforms and mobile clients</td>
<td>Limited coverage on mobile OS/platforms and mobile clients</td>
<td>Support testing on very limited scale</td>
<td>Support testing on very limited scale</td>
</tr>
<tr>
<td>Test Automation</td>
<td>Easy for GUI testing</td>
<td>Easy for GUI testing and load testing</td>
<td>Required integrated mobile test beds with wireless connectivity</td>
<td>Required integrated mobile test beds with wireless connectivity</td>
</tr>
</tbody>
</table>
Different Mobile TaaS Infrastructure

- **Emulation-Based Mobile Testing on Clouds**
  - Mobile App Instances
  - Wireless Internet
  - Emulation-Based Mobile clients

- **Simulation-Based Mobile Testing on Clouds**
  - Mobile App Instances
  - Wireless Internet
  - Simulation-Based Mobile clients

- **Device-Based Mobile Testing on Clouds**
  - Mobile App Instances
  - Wireless Internet
  - Mobile Device in a private cloud
Mobile TaaS Infrastructure for Native Mobile App
Mobile TaaS Infrastructures for Mobile Web App.
Expected Features & Benefits of Mobile TaaS

- Crossing devices
- Crossing Platforms
- Crossing Diverse Wireless Networks
- Mobile Test Environment Costs
- Mobile Device Costs
- Mobile Testing Operation Costs
- Manual Test Operations
- Test Service Process Tim
- Pay-As-You-Test
- Test Anywhere
- Large-Scale Test/Simulation
- Mobile Test Efficiency
- Mobile Test Automation
- 7/24/236
## Different Mobile TaaS Approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing Service Model</td>
<td>Emulation-based Mobile TaaS Service Model</td>
<td>Simulation-based Mobile TaaS Service Model</td>
<td>Private Mobile TaaS Service Model</td>
<td>Remote Mobile TaaS Service Model</td>
</tr>
<tr>
<td>Business and Billing Model</td>
<td>Pay-as-you-use on device emulation could and TaaS server cloud</td>
<td>Pay-as-you-use on device simulation could and TaaS server cloud</td>
<td>Pay-as-you-use on private device could and TaaS server cloud</td>
<td>Pay-as-you-use on remote device could and TaaS server cloud</td>
</tr>
<tr>
<td>Testing Environment</td>
<td>Emulation-Based Mobile TaaS Infrastructure on Clouds</td>
<td>Simulation-Based Mobile TaaS Infrastructure on Clouds</td>
<td>Private Mobile TaaS Infrastructure on Clouds</td>
<td>Remote Mobile TaaS Infrastructure on Clouds</td>
</tr>
<tr>
<td>Costs</td>
<td>No costs on mobile devices</td>
<td>No costs on mobile devices</td>
<td>High-costs for mobile devices</td>
<td>Pay-as-you-use for third-party mobile devices</td>
</tr>
<tr>
<td>Mobile End-To-End Transaction Testing</td>
<td>Complete Support</td>
<td>Complete Support</td>
<td>Complete Support</td>
<td>Complete Support</td>
</tr>
<tr>
<td>Mobile Function and Behavior Testing</td>
<td>No coverage on device-specific GUI function &amp; behaviors</td>
<td>No coverage on device-specific GUI function and behavior</td>
<td>Complete Support</td>
<td>Complete Support</td>
</tr>
</tbody>
</table>
## Different Mobile TaaS Approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalable Function Testing</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable GUI Testing</td>
<td>Not Supported for device-specific features</td>
<td>Not Supported for device-specific features</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable Behavior Testing</td>
<td>Lack test coverage on device-specific behaviors</td>
<td>Lack test coverage on device-specific behaviors</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable Transaction Testing</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable Compatibility &amp; Interoperability Testing</td>
<td>limited support on mobile</td>
<td>Very limited support on mobile</td>
<td>Good support</td>
<td>Good Support</td>
</tr>
<tr>
<td>Scalable QoS Testing</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable Mobility and Location-based Testing</td>
<td>Limited Support</td>
<td>Limited Support</td>
<td>Supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Scalable Testing Simulation and Virtualization</td>
<td>Emulation-based</td>
<td>Simulation-based</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>
Proposed Testing Infrastructure for Mobile App. and SaaS Systems

Major Objectives:
• Reduce mobile testing costs and complexity
• Easy to set up an integrated mobile test environment for projects
• Enable to support large-scale testing different types of mobile testing
Proposed Mobile Test Stack for Mobile App. and MSaaS

**Mobile TaaS Cloud**
- Mobile TaaS Server
  - MTaaS Contract and Billing
  - MTaaS User Management
  - Mobile TaaS Manager
  - Mobile TaaS Big Repository
- Mobile TaaS Agent
  - Mobile Test Control Agent
  - Mobile Test-Ware Agent
  - Mobile Test Migration Agent
  - Mobile Test Comm. Agent

**Mobile Test Instance**
- Mobile Test Engine
- Deployed Mobile APP.
- Mobile Test Platform
- Mobile Device Test Bed
- Mobile Test Network

**Mobile Test Bed Cloud**
- Mobile Test Platform Server (s)
### Proposed Mobile Test Stack for Mobile App. and SaaS

<table>
<thead>
<tr>
<th>Mobile Test Stack</th>
<th>Mobile Test Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined Mobile Test Intelligence</td>
<td>Mobile Test Engine</td>
</tr>
<tr>
<td>Defined Mobile App. Deployment.</td>
<td>Deployed Mobile APP.</td>
</tr>
<tr>
<td>Defined Mobile Test Platform</td>
<td>Mobile Test Platform</td>
</tr>
<tr>
<td>Defined Mobile Device Test Bed</td>
<td>Mobile Device Test Bed</td>
</tr>
<tr>
<td>Defined Mobile Test Network</td>
<td>Mobile Test Network</td>
</tr>
</tbody>
</table>

#### Mobile Test Instance

- Mobile Test Engine
- Deployed Mobile APP.
- Mobile Test Platform
- Mobile Device Test Bed
- Mobile Test Network

#### Mobile Test Bed Cloud

- Mobile Test Server(s)
Test Models and Test Criteria

- Mobile User Interface Test Model & Criteria
- End-To-End Transaction Test Model & Criteria
- Mobile App QoS Test Model & Criteria
- Compatibility Test Model & Criteria
- Mobile Security Test Model & Criteria
- Mobile Behavior Test Model & Criteria
- Interoperability Test Model & Criteria
- Mobile Usability Test Model & Criteria
- Mobile Function Test Model & Criteria
- Mobile Envir. Test Model & Criteria
- Connectivity Test Model & Criteria
- Multi-tenancy Test Model & Criteria
- Mobility Test Model & Criteria
Modeling for Mobile Test Environments

Mobile device
- Web phone
- Tablet
- Smartphone

Select-1

Function
- Fi
- Dj
- Ek

Mobile Device

Network Connectivity
- WAN (2G/3G/4G..)
- Wireless internet
- Wireless PAN (such as NFC/BlueTooth)

Appliance APIs
- API-A
- API-B

Platform/OS
- S40
- Android
- iOS
- Window RT
- BlackBerry

Select-1

Version #1
- Version #k

Smartphone A

Cellular Network (such as GSM/PCS)
Scalability & Performance Modeling in Clouds

Dynamic Scalability and Performance Models for Testing

- Dynamic Models for Scalability Costs and QoS & SLAs
- Adaptive Dynamic Resource-Oriented Scalability Models

- i.e. System dynamics

Modeling dynamic behaviors of MSaaS/application performance in scalable cloud environment.

Modeling scalability-based economic costs, QoS Performance and Scalability based SLA agreements for MSaaS

Modeling dynamic behaviors of allocating computing resources at the cloud level (such as cloud infrastructures)

Above Clouds
Cloud Level
SaaS Level

Resource Measurement
Performance Measurement
Cost Measurement
Scalability Measurement
Multi-Tenancy of SaaS and Mobile SaaS

Single Tenant

- Mobile GUI Structure
- GUI Events
- MSaaS App Workflows
- MSaaS Business Logics
- DB Schema & Service

Operation Flows
Forms or Frames
Service Functions
MSaaS Business Logics
DB Tables

Isolating

Multi-Tenants

- Mobile GUI Structure
- GUI Events
- App Workflows
- MSaaS Business Logics
- DB Schema & Service

Operation Flows
Forms or Frames
Service Functions
MSaaS Business Logics
DB Tables

Sharing
Multi-Tenancy Test Models

Mobile GUI forms

Mobile Style & Profile

Service Workflow

Decision Table

State diagram

API flow diagram

Mobile GUI Flow Model

Mobile Client Structure Model

Tenant-A

GWF

GWF

GDT

GSD

GDM

GAPI

WF

Switch node

Loop node

SQL DB

NoSQL DB

Service Workflow

Decision Table

State diagram

API flow diagram

Mobile GUI forms

Mobile Style & Profile

Mobile GUI Flow Model

Mobile Client Structure Model

Tenant-A

GWF

GWF

GDT

GSD

GDM

GAPI

WF

Switch node

Loop node

SQL DB

NoSQL DB

Service Workflow

Decision Table

State diagram

API flow diagram

Mobile GUI forms

Mobile Style & Profile

Mobile GUI Flow Model

Mobile Client Structure Model

Tenant-A

GWF

GWF

GDT

GSD

GDM

GAPI

WF

Switch node

Loop node

SQL DB

NoSQL DB
Multi-Tenancy Testing for MSaaS
Cloud Computing as New Computing

• SaaS (software-as-a-service), PaaS (platform-as-a-service), and IaaS (infrastructure-as-a-service) as three principal components of cloud

• SaaS ≠ software, PaaS ≠ platform and IaaS ≠ infrastructure.

• In fact, SaaS ⊃ software, PaaS ⊃ platform, IaaS ⊃ infrastructure.

• From user point of view: XaaS means online, rentable, composable, searchable, scalable, virtualized, seemingly unlimited computing, storage and networking
XaaS Internal Point of View

• **Decentralized operations**: Parallel and distributed computing, P2P operations, and MapReduce operations
• **Metadata-based computing**: Separating metadata from data, and use metadata to control almost all execution
• **Data-centric computing**: Metadata and data will be retrieved often, significant computation requires data retrieval from the associated databases, new database model such as eventual consistency, writers over readers
• **Automated redundancy and recovery**: Automated triplicate writes, redundant components, data, processors, storage, and communication
• **Automated scalability**: Multi-level scalability architecture with redundant load balancers, stateless service design, automated migration, automated data partitioning, automated workload detection
SaaS Example: Salesforce.com [3]

- Distributed operations
- Metadata-driven computing
- Data-centric computing: runtime application generator dynamically builds applications in response to specific user requests
- Automated redundancy and recovery management
- Automated scalability

![Force.com Architecture Diagram]
Example Scalability Architecture (Salesforce.com)

Threshold User Capacity = Add a POD

Network Services

Storage Services

Backup Services

Monitoring Services
Redundancy and Recovery Example (Salesforce.com)

- **PaaS level**: A PaaS system is often a black box and provides chunk-level redundancy and recovery.

- **SaaS-oriented PaaS**: This divides the PaaS resources to different tenants, such as partitioning storage and network resources (Cisco, VmWare, NetApp for tenants).

- **SaaS-on-PaaS**: Design SaaS R&R mechanisms on top of a PaaS. R&R can be at the scheduler level, ontology, data, metadata levels, and other levels.
TaaS from User Point of View

TaaS means testing software will be online, composable, Web-based, on demand, searchable, scalable, virtualized and secure testing environment, unlimited computing, storage and networking

- **Test case generation**: Users or crowd can develop, debug, and evaluate test cases online using automated test case generation tools online or TaaS. One set of users can develop test cases, a different set can evaluate test cases, all participants can be ranked.

- **Test script development**: Users or crowd can develop, debug and evaluate test scripts using TaaS tools in a collaborative manner, or use SaaS approach where software is constructed by reusing components.

- **Test script compilation and deployment**: can be done automatically by TaaS or as user directed.

- **Test script execution**: Parallel, distributed, autonomous, or MapReduce.

- **Test result evaluation**: Database operations such as automated saving, intelligent retrieval, concurrent, parallel, distributed and MapReduce evaluation execution.
Concurrent Testing and Analysis in clouds

Eliminate configurations from testing considerations

TA Analysis

Combinatorial Testing

Interaction (X, F, P, N, U)
Configuration (X, F, P, N, U)

Automated detection of X or F configurations using existing test results

(1) Identify pass/fault configurations
(2) Identify fault location
TaaS from Internal Point of View

- **Decentralized operations**: Testing tasks may be
  - Executed in a parallel, distributed, autonomously, on-demand, or triggered by policies;
  - Testing tasks may be migrated, performed in a redundant manner, or embedded within cloud operations.

- **Metadata-based computing**: Use metadata to
  - Specify and control testing tasks such as time, frequency, multi-tasking, redundant testing, parallel testing
  - Specify test cases, scripts, and environment
  - Specify test results such as indexing, location, organization, profiles
Built-in Continuous Testing

(a) SOA model
(b) SaaS model
(c) Continuous Testing in SaaS
Test Cases Generation from Metadata

• Test cases can be generated by examining metadata:
  – For example, length of customer must be 64 bits. One can generate a collection of customers of 64 bits, another collection with 128 bits or any other bits.
  • Random number from $0 \sim 2^{64} - 1$
  • Another set is negative numbers
  • Greater than $2^{64} - 1$
  – The boundary value test cases can be generated from {-2, -1, 0, +1, +2} around boundary of the constraints, specifies by the metadata.
  – The need for International standard on test metadata for TaaS
**TaaS from Internal Point of View (continued)**

- **Data-centric testing**: Use data in the SaaS, PaaS, or TaaS database for testing. This may include:
  - Automated test results saving in test databases like in-memory databases where data are classified as hot, warm, or cold
  - Storing monitoring data intelligently with automated index generation
  - Analysis of test data to optimize test strategies
  - Multi-tenancy test script composition. i.e., one test code for all testing needs, and use different components to customize different testing needs
  - Automated logging like LogStash or Splunk manner
TaaS from Internal Point of View (continued)

Automated test redundancy management and recovery:

- Testing tasks can be partitioned and sent to different processors for parallel and redundant processing
- Test results can be merged following a mathematical principle to increase the confidence of test execution
- Test and test results can be recovered in case of failures in a processor or in a cluster due to automated redundancy management
- Recovery follows the metadata-based approach
TaaS from Internal Point of View (continued)

**Automated test scalability**: Need to deal with scalability of 1) applications, 2) cloud infrastructure; 3) TaaS.

- Testing scalability of applications and cloud infrastructure: Need to follow cloud scalability mechanisms (such as 2-level scalability architecture, stateless service design, and automated migration), testing needs to set up an environment including test input, relevant SaaS and PaaS services, execute test, and evaluate the performance.

- TaaS scalability: as the SUT scales up at runtime (with more resources and more data generated), TaaS needs to scale up proportionally to track and monitor the execution. TaaS scalability mechanism may include:
  - Multi-level TaaS load balancers used to dispatch (stateless) testing tasks to different processors for execution
  - Automated testing tasks migration to optimize test execution
Combinatorial Testing TaaS

• Combinatorial Testing (CT) can detect failures triggered by interactions of parameters with a covering array test suite generated. Many outstanding algorithms are available, and they focus on test coverage.

• But cloud computing need to identify faults, not just failures. Also need a algebra to allow to Inference of the test results of a subset to the test results of its superset.
  • Allowing asynchronous merging of test results.
  • Allow test problem space to be decomposed.
  • Allow concurrent and asynchronous testing
  • Allow storage of intermediate test results
Tenant Application Testing with Fault Detection

- 10 components and 10 values in each component. Thus, the total number of configurations is 10 billion.
- Faults scale (only for two-way faults): the number of 2-way interactions is 4500.
- Simulated different number of faults: 5, 50, 100, 200, 300, 500, that represent 0.1%, 1.1%, 2.2%, 3.3%, 6.67%, and 11.1% of 4,500 2-way interactions respectively.

![Graph showing final faults/faults](chart.png)
Five Status Indicators

- **Infeasible (X):** For certain interactions, it may happen that no feasible test is permitted to contain this interaction. For example, it may be infeasible to select two GUI components in one configuration such that one says the wall is GREEN but the other says RED.

- **Faulty (F):** If the interaction has been found to be faulty.

- **Operational (P):** Among the rest, if an interaction has appeared in a test whose execution gave an operational result, the interaction cannot be faulty.

- **Irrelevant (N):** For some feasible interactions, it may be the case that certain interactions are not expected to arise, so while it is possible to run a test containing the interaction, there is no requirement to do so.

- **Unknown (U):** If neither of these occurs then the status of the interaction is required but not currently known.
## Multiplication Rule

<table>
<thead>
<tr>
<th>$\otimes$</th>
<th>X</th>
<th>F</th>
<th>P</th>
<th>N</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>F</td>
<td>X</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>P</td>
<td>X</td>
<td>F</td>
<td>U</td>
<td>N</td>
<td>U</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>F</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>U</td>
<td>X</td>
<td>F</td>
<td>U</td>
<td>N</td>
<td>U</td>
</tr>
</tbody>
</table>

- The binary operation has the following properties of commutatively and associativity. Superset of X or F are still or F.

\[
V(\mathcal{T}_1) \otimes V(\mathcal{T}_2) = V(\mathcal{T}_2) \otimes V(\mathcal{T}_1), \\
V(\mathcal{T}_1) \otimes (V(\mathcal{T}_2) \otimes V(\mathcal{T}_3)) = (V(\mathcal{T}_1) \otimes V(\mathcal{T}_2)) \otimes V(\mathcal{T}_3).
\]
Concurrent Testing Framework
Merging Concurrent Testing Results

The testing results of a interaction \( T \) in different servers should satisfy the following constraints.

- If \( V(T) = U \) in one cluster, then in other clusters the \( V(T) \) can be either \( F, P, N, \) or \( U \).
- If \( V(T) = N \) in one cluster, then in other clusters, the \( V(T) \) can be either \( F, P, N, \) or \( U \).
- If \( V(T) = P \) in one cluster, then the \( V(T) \) can be either \( P, N, \) or \( U \) in all clusters;
- If \( V(T) = F \) in one cluster, then in other clusters, the \( V(T) \) can be \( F, N, \) or \( U \).
- If \( V(T) = X \) in one cluster, then in other clusters, the \( V(T) \) can be \( X \) only.

If these constraints are satisfied, then the testing results can be merged. Otherwise, there must be an error in the testing results. Addition operations

<table>
<thead>
<tr>
<th>+</th>
<th>E</th>
<th>X</th>
<th>F</th>
<th>P</th>
<th>N</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>X</td>
<td>E</td>
<td>X</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>E</td>
<td>E</td>
<td>F</td>
<td>E</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>P</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>N</td>
<td>E</td>
<td>E</td>
<td>F</td>
<td>P</td>
<td>N</td>
<td>U</td>
</tr>
<tr>
<td>U</td>
<td>E</td>
<td>E</td>
<td>F</td>
<td>P</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>
Partition Configurations

- Algebraic rules allow one to partition configurations into different sets for different servers to test, and these sets do not need to be non-overlapping.
- Fifteen configurations, assuming only interaction (c, d, f) is faulty, and only interaction (c, d, e) is infeasible, and all other interactions pass the testing.

1) Example 1: If one assigns 1-10 configurations into Server\textsubscript{1}, 6-15 configurations into Server\textsubscript{2}, and 1-5, 11-15 configurations into Server\textsubscript{3}.

<table>
<thead>
<tr>
<th></th>
<th>Server\textsubscript{1}</th>
<th>Server\textsubscript{2}</th>
<th>Server\textsubscript{3}</th>
<th>Merged Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a,b,c,d)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,c,e)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,c,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,d,e)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,d,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,c,d,e)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(a,c,d,f)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>(a,c,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,d,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(b,c,d,e)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(b,c,d,f)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>(b,c,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(b,d,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(c,d,e,f)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

2) Example 2: If one assigns configurations 1, 3, 5, 7, 9, 11, 13, 15 into Server\textsubscript{1}, configurations 2, 4, 6, 8, 10, 12, 14 into Server\textsubscript{2}, and 4-11 configurations into Server\textsubscript{3}.

<table>
<thead>
<tr>
<th></th>
<th>Server\textsubscript{1}</th>
<th>Server\textsubscript{2}</th>
<th>Server\textsubscript{3}</th>
<th>Merged Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a,b,c,d)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,c,e)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,c,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,d,e)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,d,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,b,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,c,d,e)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(a,c,d,f)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>(a,c,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(a,d,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(b,c,d,e)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(b,c,d,f)</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>(b,c,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(b,d,e,f)</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>(c,d,e,f)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Issues in Mobile TaaS

(a) Diversity of mobile devices and platforms
   → high complexity of mobile testing environments
   → high cost on set-up cloud-based mobile testing infrastructure
   - Diverse mobile devices with different configurations/operation environments
   - Diverse mobile appliances, drivers, and APIs from different vendors
   - Diverse wireless connectivity
   - Private/public cloud infrastructure for both mobile client and mobile SaaS servers

(b) Mobile test automation in large-scale
   → require intelligent large-scale concurrent test automation
   → require automatic test control and handling methods for test failures
   - Lack of high-level test automation control technology (for example, test control-oriented script language and protocol for remote control and control)
   - Lack of well-defined scripting standards, techniques and hierarchical control solutions
Issues in Mobile TaaS

(c) Lack of standards
→ higher cost in test tool integration and set-up for mobile test automaton
  o Lack of standard interfaces, languages, and connectivity protocols, and APIs among test tools
  o Lack of well-defined QA processes and test criteria for mobile Apps and mobile cloud services
  o Lack of standard test interfaces, APIs, and connectivity protocols to support mobile app testing and mobile cloud service validation

(d) Lack of effective open mobile test stack/frameworks, and test tools
→ the difficulties and challenges in mobile test automation
  o Lack of an effective mobile test bed for diverse mobile apps
  o Lack of a scalable mobile test tool to support large-scale mobile traffic loads and user accesses
  o Diverse wireless connectivity

(e) Globalization and internationalization
→ higher cost and difficulty in testing for diverse language contents and Mobile User interfaces