Title: 웹 소프트웨어의 신뢰성 19

- ✓ **Instructor: Gregg Rothermel**
- ✓ **Institution: KAIST**
- ✓ **Dictated: 강단비, 김주현, 김지현, 주다은**

[00:00]

Hello, today I will present my research conducted in this course, which is 'Challenges in Applying Fault-Localization Techniqus on Web Applications'.

As you know, in debuggin process, we need to…

Sorry.

In debugging process we need to find causes of program failures after detecting program failure. But the finding failure causes is very time consuming because we need to analyze, execute the source code for finding faults.

So to detect this problem, many fault localization techniques targetting C, or Java have been proposed.

Those techniques are called slicing-based, coverage-based, and statistical model based, depends on approaches they use for localizing faults.

however, they just concentrate on traditional languages such as C or Java,

They do not concern web languages.

As web applications are becoming popular, fault localization techniques for web, need to be researched.

As you know, many web, services are delivered by a web, and the developer, nowadays, developer try to develop applications that work on the web.

So, to assist program, web programmers, we need to develop and research fault localization techniques on web.

However, there are very few works on web fault localization.

So, in this paper, I surveyed existing fault localization techniques on web and traditional languages.

And I analyzed possible challenges and issues when applying fault localization techniques on web applications.

By doing this, I hope there will be some…

I hoped, I found some issues that late the fault localization techniques, and I hoped make some ideas for the locating false in the web.

So, this talk consists of motivation that I delivered, and fault localization techniques, what I surveyed.

And, challenges in applying fault localization techniques on web applications.

Finally, I will deliver the conclusion.

In this section, I'll just present few techniques, most representative techniques for fault localization.

But I surveyed several techniques in the paper.

So, why do we need fault localization?

As you know, when we face the, when we detect failure, usually we face failed executions for finding program faults that cause program failure.

But if the program is consisted of very large lines of source code, for example, 1 million lines of source code, we cannot inspect all of the lines in the source code for finding faults.

So, to reduce lines of source code, that need to be inspected for finding faults, fault localization techniques have been proposed.

By using this technique, programmers can find fewer search space, for finding program faults.
In this sense, there are techniques called slicing-based, and coverage-based fault localization technique.

Okay. Slicing-based fault localization is the technique that slices lines of code that can affect detected program failure.

For example, if we face abnormal value, when executing the program, then the slicing-based approach trying to slice portions of source code that can affect abnormal value of the variable.

So, there are 2 kinds of techniques, static slicing and dynamic slicing.

Static slicing analyze target source code statically without executing the program.

That is, static slicing tries to show possible portions of source code that can affect abnormal value of the variable, by analyzing source code statically.

[05:00]

And dynamic slicing analyze target source code dynamically.

That is, they use information gathered from the real execution.

So they reduce search space for finding program faults.

This is the first work in the static slicing and this is know as the first work in the dynamic slicing.

So, in this example, I will briefly explain slicing-based approach and coverage-based approach.

Sorry. I will briefly explain static slicing and dynamic slicing.

So, in this example, the program returns, prints value depends on values of variable 'a', and value of variable 'b'.

And the value of variable 'a' is greater than 'b', then the retVal equals to the 1, and prints 1.

When 'a' is less than or equal to 'b', than retVal is -1, and the program prints -1.

So, when we execute the program by using this case, the line number 2,3,4,5, and 8 will be executed.

And statics… Let's assume that we detect abnormal value of variable at line number 8.

That is we detect the program failure at line number 8.

So, static slicing approach will show slices that can affect value of variable as line number 8.

In this case, line number 4,5, and 7, can affect value of variable retVal at line number 8.

So, this lines will be presented to the programmer that can contain program faults.

In case of dynamic slicing, dynamic slicing just concern about executed lines of source code.

That is they use information gathered from real execution, so they do not contain line number 7, which is not executed in the, in this execution.

So the dynamic slicing approach reduces searches phase for fining program faults.

So, at this point, the question is that.

We know which lines can affect abnormal value of the variable.

However, if the programmer tries to find program faults by inspecting those lines, how can we inspect which line should be inspected first?

Usually, programmers may inspect line number 8 and 5, 4 and 2, in a backward way, but it can be very inefficient for finding faults, if the fault is placed in the starting point of the execution.

For example, if the fault is place in line number 2, we need to inspect all most all of the lines of source code that are executed.

So, in this sense, we need to prioritize a statement or source code that can affect program faults for finding program fault effectly.

So the coverage-based source localization technique have been propesd in this sense.

This technique tries to give high priority to suspicious code that can contain program fault.

That is, if a statemet have, recieves higher priority than other statements, the statement is inspected first by the programmer.

So, there are many techniques have been proposed for prioritizing statements that can contain program faults.

There are more than 100 papers of this work, this area, but I will explain basics of this technique.

Which is presented at ICSE2002 by Jons et al.

[10:00]

So, this is similar example as described in the previous page, but in this program there is fault place in line number 5.

In line number 8, we can see the assertion of the this program.

The program should return, when value 'a' is greater than 'b', and it should return '-1' when 'a' is less than or equal to 'b'.

So, when executing line number 5, the program failure will be detected because the retVal is equal to 10, which is incorrect value.

So, let's see test case 1.

When we execute the test case 1, where the value of variable 'a' equals to 3, and value of variable 'b' equals to 2.

Then the test case 1 will execute line number 2, 3, 4, 5 and 8.

And the program failure will be detected because the retVal equals to 10, which violates assertion in line number 8.

By this way we can get coverage of each test case, and each execution result.

In this case, test case 1 cause program failure, so it is marked as fail.

By this way we can… Let's assume we have 4 test cases that execute like this and we have passed fail status.

By using this information, Tarantula technique calculate suspiciousness of each line.

In this case, line number 2, the suspiciousness of line number 2 will be calculated as 0.5, because the number of failed test cases that execute the line number 2 is 2, and the number of total test cases in given test set is 2.

And the number of passed test cases that execute line number 2, is 2.

And the number of total passed test cases in given test set, is 2.

So, by this way we can calculate suspiciousness of line number 2.

By this way, we can calculate the suspiciousness of each line in the program like this way.

In this program, the suspiciousness of line number 2 is 0.5, and line number 3, 0.5, and line number 4, 0.5, and so on.

So by using this suspiciousness score, the coverage-based fault localization prioritizes statements that can contain program faults.

In this case, line number 5 will have highest priority than other statements.

Then the programmer can refer to this rank, and the programmer will inspect statement that has highest priority first.

So, by this way, coverage-based fault localization tries to remove search's phase by finding program fault.

So, when we apply these kinds of techniques on the web applications, there will be several challenges that can affect ?[13:54] of fault localization.

For example, web applications have different architecture from the traditional applications which are features of web language or network structure that is used for web application.

So, because of different features, there can be many challenges.

I categorized those challenges in 2 categories.

First one is that, detecting web application's failure is not easy.

As you know, to apply fault localization, we need to detect program failure.

But detecting program failure in web application is very difficult because of feature of web languages.

And in addition, web languages have different syntexes and different grammar checking method, so the web language can have features that can reduce accuracy of fault localization.

[15:00]

As you know, for testing web applications, the testing techniques try to explore web page's UI states, but...

While exploringn UI states, the technique should determine the current UI state is correct or not.

In this sense, HTML validator can detect malformed HTML, but they do not concern about UI states in point of users' view.

So, to detect program failure, we need to define correct UI states precisely.

But it can be very difficult, because a web page is consisteed of several external services and web page are dynamically changing.

So, specifying correct UI states for all external serbvices, and all dynamic possible passes that can be changed is very challenging.

In addition to this, even if we have correct specifications for UI states, maitaining UI states while program is runnig, can require huge resources.

For example, in JavaScript, DOM tree representing UI state is maintained while program is running.

But exploring DOM tree while program is running, can be, can consume a lot of time, because a DOM tree consisits of several nodes that can construct UI state of current web page.

Okay.

Also, web languages have different features from the traditional languages such as C or Java.

For example, executing program faults, do not cause program failure many times.

For example, in JavaScript, function call without parameters is not reported as program failure.

If we don't pass the parometer for the function whole, bu the JavaScript do not report that as errors.

So because of this feature, the accuracy of coverage-based fault localization and feasibility of backward dynamic slicing approach can be severely attacked.

For example, because of this feature, a failure will be detected after executing several lines of code after executing the fault.

So, we need to inspect several lines of code for finding program faults, when we apply dynamic backward slicing approach.

And also, there will be so many coincidentally correct test cases.

Coincidentally Correct Test case is that test cases that execute program faults but do not cause program failure.

As we saw in the previous example, the suspiciousness of fault line is calculated based on the number of test and failed test cases that execute the fault.

So when there are many test cases that execute the fault, the suspiciousness of fault will be decreased.

So the accuracy of coverage-based fault localization will be decreased by this feature.

So this example shows that explicitly.

In this function, the function updates image in the web page periodically.

In line number 12, time out is set, for the display function.

The display function is called every 10,000 miliseconds.

[20:00]

And the display function updates image by refering dumb object and by changing property in the image object.

But in this function, the call for this function do not contain parameter.

But in JavaScript, it is not reported as errors.

It is, that is, executing line number 2, do not cause program failure.

We can detect program failure that comes from this line, at line number 8.

When the display function is set by this function, the line number 2 will clear current time wall and line number 3 will set new time wall for the display function.

And line number 5 refers document object for extracting image object.

And line number 6 extract image object for next image object thaht will be updated.

And line number 8 refers to next iamge object for updating current image.

But in this case, the null-value difference exception will be occurred because next image object do not have real obejct for the image.

Because we do not pass the parameter for display function.

So, when we apply dynamic slicing in this example, we need to explore almost all of the lines that are executed.

In this case line number 7, 6, 5, 4, 3, 2 and 12 will be inspected when we apply dynamic backward slicing approach.

Okaky, so…

Because of this feature, the coverage-based fault localization, the accuracy of coverage-based fault localization will be severely decreased.

Let's assume there is an additional line, at line number 2, which is, if image_id is breater than 100.

In this case, when we call, when we set the time wall at time number 14, the, as you know, the parameter is not set.

So the, in the JavaScript, image_id is assigned as random value in this function.

So when we call the display function, the, this line will be executed only when image_id is greater than 100.

So, executing the fault line do not cause program failure many times.

So there will be many passed test cases that execute the fault and line number 14.

So because of test cases that execute line number 14 but do not cause program failure, the suspiciousness of line number 14 will be decreased when we apply coverage-based fault localization.

So in this talk, in this paper I surveyed several fault localization techniques, for C or Java as well as web applications such as PHP or JavaScript.

So, as web applications are becoming popular, fault localization techniques targetting web languages need to be researched.

But, at this point there are very few works.

So to develop new techniques I surveyed and investigated existing fault localization techniques and analyzed possible issues when applying fault localization techniques on web applications.

You can look this in my paper. Thank you.

Hi, my name is Kyoung-Rok and today I'll introduce the works done in the field AJAX testing.

This is the outline of my presentation.

I'll first provide some background knowledge and then I'll introduce the challenges in AJAX testing.

[25:00]

Next, I'll show the approaches proposed so far, and then conclusion is findings and future work.

And this is the motivation of survey, my survey.

And Ajax is for now, essential for building a modern web application with user interface. Testing is necessary to build a reliable software, but testing an Ajax causes severl challenges, since Ajax is different from the traditional web appications.

So, I thought surveying the works done in Ajax testing, in the field for Ajax testing would be helpful for understanding the current stated of the field, recognizing the further steps.

So, first of all, what is Ajax?

Ajax is group of existing web development techniques, which is used to implrement a web application that provides responsive, desktop-like user interface.

And the characteristics of Ajax is followed.

First, enables the cline to asynchronously to communicate with server.

And next is… web page can be updated partially.

It doesn't require the full page update, and so it can update web page dynaically which means it can update it in real time.

And these are the components of Ajax.

First, HTML and CSS is used in combination to compose a web page.

And Dom has 2 roles.

It represent the HTML document as a tree like structure.

And also, it provides API, a set of API that can be used to manuplate DOM, I mean the HTML document.

This API is usually accessed by JavaScript.

And XML and JSON serve as a data interchange format, between server and client.

And XML Http Request is the most important component in the Ajax technologies.

It enables the synchronous communication between a server and client.

It also can be accessed using JavaScript.

And finally, JavaScript is the most commonly used scripting language in web development.

And it's role is to handle data communication to modify, dynamically modify DOM.

And a compare the characteristics of traditional web application to that of Ajex web application.

First a treditional web application is composed of multiple static pages.

Which means once the page is loaded then the page will stay static wouldn't be changed.

And the request is dierctly sent to a server, synchronously.

Which means a user should wait until the server responds to the request.

And pages, as I mentioned just before, there are multiple pages in traditional web application.

And those pages are interlinked with hyperlinkes.

And transition between pages is incurred by clicking the elements.

The HTML element which is dedicated to represent the hyperlink.

And the page should be, I mean the whole page should be updated even whit the samll change is the page.

So it is very ineffitient in terms of ?[29:24].

And here are the characteristics of the Ajex application.

Theoretically Ajex application can be composed of a single and dynamic page.

So it can have only one page then dynamically modify its state to represent different states of web application.

And a request to a server is done asynchronously, through XMLHttpRequest object.

And then so, that the working user ?[30:2].

[30:00]

And DOM state can be dynamically modified using DOM API and Javas' callbacks.

Javas' callbackes will decode when asynchronous responses received from the server.

Then finally a page can be partially updated using DOM API.

Which can explicit, save the ?[30:36].

So here are the challenges in Ajax testing.

First it's asynchronous communication.

As you guys know the asynchronous communication or process is the main cause of unexpected software bavavior.

And partial communication.

It didn't present in the traditional web application.

So many tools couldn't understand this kind of partial communication.

And next is dynamic DOM update.

This feature also didn't represent in traditional web application.

As many tools and techniques addopted that testing the web application with only multiple static pages.

It cannot detect the dynamic state changes within the single page.

And finally detecting dynamic state enrties.

Oh, sorry I forgot to explain this part.

This is the part of google map UI, user interface.

And even though those are not linked elements, commonly used in traditional web application.

This kind of elements can also encur the state changes.

And it's virtually every elements can incur such a change, it is hard to detect reliability such elements.

So this problem finding the entry points to under states is also problem for testing Ajex applications.

Here, and now I will introduce you a approaches introduced so far.

First one is state-based testing of Ajex application.

As I mentioned before, Ajex applications can modify the page in real time

For instance, in google map, those part of user interface is dynamically updated with the user interaction.

And traditional techniques and tools can captuer such dynamic state changes occur in a single page.

[?]This tequnique try to draw infor of fin? State Machine that reflects such a dynamic DOM modification.

The inferred model in composed of abstract DOM states.

And the transition is an Ajex callback or user event that incur DOM update.

And to extract a model, user districtly uses a set of execution traces generated by an user.

The left is the example of exception traces generated from the simple cars Application

There are two, actually three events, add, remove, and empty events.

And exception traces would store Information of such an events and as the DOM states.

Incurred by such an event cuase.

And right one is a sample efficient for communication application.

Only novle part is inferring such a model from, I mean that reflets the dynamic states.

And actually generating test cases among modle isn't that novle.

they actually utilize the traditional model of base-testing tedhniques

But one thing they have developed further is they have attemp to minimize the test size.

Because exhaustively generating best cases from a can lead to test case explosion.

[35:00]

So they incorporate the notion code symatrically interacting events.

Which is commonly used in GI Testing.

The main idea is that…

O.K. I'll first explain what the symatrically event is.

And say there is two Events E1 and E2

And those two events are symatrically interacting if swapping the order of them brings application to a different state.

For instance, there are two events at any empty and add…

If you swap the order, that brings I mean to state to different ones.

The goal of this kind of consideration is that it only consider events that is meaningful in terms of state changes.

Is it clear to you guys?

(student questioning)

Because if swapping the events does not change anything,

that means one event has no effect on application changes.

So in my understanding they try to extract only all orthogonal set of event sequences that can discover new state.

Is it clear?

(student questioning)

O.K anyway they succeeded in reducing the test suite size up to 87%.

They clean?

I don't know.

And there are drawbacks of this technique.

They utilise this execution traces ot infer model,

but , it means that they need enough size of execution traces.

Which doesn't seem to be always possible.

And also that events that didn't covered in exception trace won't be included in the model.

And next, even though they have utilised symmatrically interacting events,

but still the test suite size is huge if the test cases exceeds a cartain length.

for their codinf to their empirical study,

If they try to increase, I mean, they found out that test cases whose length is up to 4, maximum 4,

already generated defined cases, which is quite many.

but since longer event sequence has more power of, I mean more ?[38:28] power,

is quite limitation of this technique

And next is serch based testing.

and it's exactlly - tension of three techniques fit this testing.

And the goal of this technique is to reduce the size of, I mean lengthen the event sequences. That has more power, I mean, fault revealing power.

And at the same time, keep the test size reasonable.

Just cutting, I mean just removing some test cases will diminish the fault revealing power.

So they instead try to maximise the test case diversity.

What I mean by case diversity is the test case diversity executes software more diverse way.

Which is diverse ,I mean, discover more faults.

And they ustilise hill-climbing aldorithm to find the test siute result, maximum test set diversity

And this is how this algorithm work.

They starts with this initial set,

O.K I think this script is cracked.

[40:00]

They start with initial set of events and they find, what they call neighboring solutions, by ?[40:22].

O.K. this y axis is cost function, and x axis is solution to test its cost, I mean objective function.

So the solution is higher cost is better.

And the goal of this algorism is to find the solution of maximun cost.

So, we literally go up the hill in terms of cost.

Then and if you find the top point, then the algorism is stopped.

And the solution is the optiming solution.

So one one limitation of this technique is that it can satisfy with local maximum.

Which is not the highest point but actually …

They have addopted this algorism so they clime the hill by ? I mean adding, a single, one more test sequence, test case sequences.

And everytime they add one test squences they applied cost functions.

to test if next generated solution is better than the previous one.

so they climb the hill and if they found one sloution then maximise the cost, which is top test case diversity.

They masured case diversity by seeing the execution frequency and model coverage if and anyway.

If they find the solution that maximise the cost function.

Then they stop and they return solution as an optimise one.

They again suceeded in reducing the test size,

And in extream case the size is reduced upto 3%, which is quite high.

The generated test suites showed compairable within power that of exhustive test suite.

Which is quite good, because the test suits, result in test suit should be much smaller than previous ones.

And also the test suit showed much higher fault detection ratio since it had compairable performance with the smaller size.

And one drawback of this technique is that is requires complex compitition.

Next is invariant base testing of Ajax application.

The goal is simillar, to that of state base testing.

Due to try to infer the model from the dynamics states, Ajex states, application.

They, I mean this technique is based on crawer called CRAWLJAX,

which and automatically crawl dynamic DOM states of an Ajex application.

And after a midel is extracted, they tested each states, and the modle itself , using several kinds of invariance.

Also they apply - traditiona test case, I mean model based test - testing techniques.

And the CRAWJAX.

CRAWJAX aoutomatically detect clickaleds,

for instance those buttons in the google map,

and clickables can incur DOM state transition.

Then they actually keep them to discover new states of Ajax applications.

And they automatically infer the stateful graphs from it.

And they are identified the new DOM state, by compairing them using Levenshtine distance.

[45:00]

And the left part explains what the Lavenstine distance is.

And it's quite naïve way to compare the s- symilarity

So I think the this limitation, I mean this is quite naïve to compare complex DOM trees.

And these are Invariant types.

Generic DOM invariants, to check whether the HTML is well formed without synthetic error, something.

And state machine invariants, which act whether there is no dead paths, or backtracting works prdperly.

And application spacificinvariants is literally the application secified invariants that should be manually coded by testers.

And additionally, test cases are derived from a model, using traditional methods

So this, again, is not a novel way to test.

And they actually implemented to code ATUSA.

ATUSA is implemented baesed on CRAWLJAX, and provides API,

so that testers can make their own costom plugins to test their applications.

And from emprical studies, the technique is turned out to be scalable recuring minimal manual effort to run a test.

Since this is the goal of this technique is to automatically test Ajex application, this is quite good.

And one drawback is as I mentioned before, using Levenstine edit distance might, so that it can produce some false positives on the negatives.

And next is regression testing of the Ajex applications.

It's also proposed by the authors who proposed the invariant test based testing.

So regression testing is to check wether a modified software introduces new bugs,

by reusing test suites for earlier versions of the software.

And comparison based approch is in regression testing,

saved outputs generated from previous test suite is used as an oracle.

I'm not sure whether this is regular term used in the ?[47:49] software testing, but they used the term.

But regression testing on a Ajex applications using this comparison based approch.

Might generate many false positives.

Since the DOM state is very dynamic and same input might generate same output.

Even though there isn't actually a fault.

And here are the challenges of regression testing on Ajex application.

One is nonditerministic behavior.

For instance, application that displays current timestamp, will always change timestamps whenever user opens web appplications.

But this is actually unexpected behavior.

But just nearly compairing the text-similar will   produce false-positives.

And next is..

Since different user interface state depending on backend state

Since they only identify the same, I mean…

Sincs the UI state depends on the backend state of web application

Just compairing the superficial part, I mean frontal part, cannot always identify symmatrically identical states.

So this is the example of this problem.

Say the test case is to add one new item to the application

So applying the test case multyple times will change the backend state.

[50:00]

Which is not understoodable by external too, which unequals the froten part.

So even though starting front-end state was the same is different with two item and one item

So it is a problem.

So this are the method they have taken.

They propose several types of oracle comparator.

That can strip off the valued changes nondeterminance changes for instance.

What they call daytime oracle comparator restruct strip off the timestmaps from application.

Then try to compare the rest, so that it won't produce the false-possitives.

And then the templet generation is that a web application often containn main structures such as tables or list.

And which that state changes and manifested, I mean, I think, I actually didn't agree with this authors -approch.

What they mean is that if an - state can be identified or resist one state by generating a templet or patterns.

So for instance, on a bulletin board, the new articles will be displayed as a form of table or list.

Which is an uniform formet, I mean patterns.

Even though there is different number of articles, three or four, by using the generated templet.

that can detect such a uniform structure, I mean repeating structure.

They can reduce, I mean ?[52:18] them into the same state.

But - is quite effective approch.

(student questioning)

One more time, what do you mean?

(student questioning)

O.K.

They could implement their new approch by integrating in CRAWLJAX.

And this approch turned out to be effective -able.

And according to their empirical syudy, conducted on google reader.

they succeeded in reducing false-positive up to fifty five percent.

And with their constant comcomparator, hand coded.

They actually succeed in, I mean reduced up to 95%.

which is quite high.

And one drawback is that, the goal of this technique is to identify symmatrically identical states.

even though they present different frontal state.

But they didn't concider the back-end states.

So I think this approch is quite limited.

So, this are my findings.

Most of the studies focused on modeling dynamic states of an Ajex application.

But, testing method, itself, wasn't that novel.

The most important, I mean the most focused part is to reliablly infer the dynamic model from Ajex applications.

And because of the server reasons, such as identifing set equivalance.

Reliably inferring a model was hard, seems to be hard.

So the futuer work should be develop techniques can reliably infer a model from a Ajex applications.

And I think they only concentrated on ? The frontal part of Ajex application.

I think they - should consider the interaction part of between the server and client to reliably test Ajex applications.

So I think this is the future work.

This is it.

Hi, my name is Jinhee Park.

Today I am going to discribe the serer study of Reliability Estimation Approches in the Web Environment.

[55:00]

I will classify the existing realibility estimation approches in the web environment.

And then I will evaluate them using scenario based evaluation.

This is outline of my talk.

First, I will introduce the basic information of softwaer reliability analysis.

And then I will compare them with the reliability estimation of SOA systems.

And in the overall approch.

I will discribe what I have studied for this project.

…for this project.

And then I will explain, I will compare the existing reliability estimation techniques and then using scenario-based evaluation I will evaluate the existing techniques.

And then I will, I will conclude by discussing remaining open problems.

The topic I want to present today is the software reliability.

So before we enter into the main part, we need to know the, what is the meaning of software reliability.

So the definition of software reliability is like this; the probability of software will not cause the failure of a system for a specified time under specified conditions.

In, traditionally, most software reliability researchers used this definitions for estimating software reliability.

They adopted a probability concept to explain the uncertaintys in the software reliability.

So to estimate this probability, many software reliability models were introduced.

Some researchers classified the existing models as four types like this.

First two models is widely, are widely used.

These models are called as software reliability growth model.

Using this model, we can measure or estimate software reliability, reliability.

This model uses the failure data during the operation, operational test.

So, I will briefly explain the, what they have, what I, they have done for estimating software, traditional software reliability analysis.

They used software reliability growth model during the operational testing.

They, at first project manager may establish a software reliability goal.

To meet that goal, they tested the software continuously while testing software, they record the failure data.

Failure data like this, this is cummulative failure number.

And then they fit the reliability growth model to the failure data.

It is, it can be one of the reliability growth model.

Using this model, they try to see the tendency of the failure occurrence.

Using this kind of models they can, they can estimate the remaining testing time for meeting the object, for meeting the object of software reliability.

So this can be one of the usefulness of using software reliability growth models.

In traditional models, they used the, they considered there's just code-based faults because software development environment are, softwares are developed locally, in local environment.

[60:00]

So they consider logical or the computational fault, only they considered this kind of faults.

But the paradigm has shifted to the web-based software.

In the web-based software such as soft, service-oriented architecture, a lot of service are distributed globally.

So there are a lot of service, Google, Amazon, Naver, can be those services.

They interacting each other over the networks.

So two very good service, like service composition, they, service users invoke a lot of services all around the world.

So, so the environment is changed, totally different from the traditional software development environment.

So, so in this environment, the main cause of failures are faults of external services.

This service are, this service run remotely so it could be out of control.

And second one is dynamic web environment.

So server can be down or server, network can be congested, network queues are all full due to, due to too many concurrent users.

All this, all these kind of situations, some failures can occur so to estimate the reliability of the SOA system, we need to consider this kind of factors, which can affect the software reliability.

And this can be one example.

Two different users estimated different reliability for one same web services because due to the unpredictable communication links.

This might be, might happen so we need to consider this and unlike traditional software, reliability changes according to the network condition even though there is no fault correction.

So we need to consider this.

Motivation of my work is this, even though there are a lot of, even though software reliability of the web-based software is getting important, the reliability research of web-based software are studied rarely.

So, so we need to research for this area and estimating the reliability of the web-based software is very difficult because of our dynamic execution web environment, web circumstances as I mentioned in the previous page.

So I, what I tried to do is survey existing approaches to estimate the reliability of SOA systems.

And I want to identify the strengths and weakness of each approaches based on scenario-based evaluation.

Through this work, I can provide insights into the determining future directions of reliability researches.

This is overview of what I have done for this study.

First, I choose, selected the key papers among existing papers and then I categorized those papers, those techniques according to some criteria such as failure data source and application pace.

And then I analyzed the approaches and compared them using a scenario-based evaluation.

I evaluated each approach.

[65:00]

Through this evaluation, I can see how well each approach will perform in dynamic, dynamic execution environment.

And then I will suggest some open problems.

The goal is the estimate of reliability of SOA systems.

To do that, most of approaches are divided into two parts.

For atomic service, they first estimate the reliability of the atomic service which is a basic web service unit without the structural information.

And then they try to estimate the reliability of the composite service which is a new web service consisting of atomic service.

The distinction of each approach is failure data source, so I categorized each approach using, according to the failure data source like log information-based approaches.

Or testing-based approaches or collaborative approaches.

And to estimate reliability of composite service, those, these three informations are required.

First, reliability of all atomic service being used.

And compositional structures to make new service like this.

And usage profile information is required.

Actually, compositional structure can be diverse such as, like a structure chart, or a scenario model, or stochastic petri net.

They use…

And in SOA domain, unlike traditional software reliability research, they adopt the new, or they adopt the new definition of reliability.

Such as, like the probability that the system successfully completes its task when it's invoked.

Because it is suitable for the service delivery systems, they adopt this kind of definition which is called "reliability on demand".

And then to estimate that this probability value, they adopt the input domain based model such as Nelson model.

It is very simple model for estimating the reliability.

For example, left can be compute as the 1 minus failure rate.

Failure rate is, can equals to the total numbers of cumulative failures divided by total numbers of service invocations.

This is very simple, most of the approaches adopt a simple model for estimating software reliability, atomic web service reliability.

From now, I will briefly explain the characteristics of each approaches.

First, log information-based approaches use some kind of log information.

Service, there is UDDI, just service provider register their services to the UDDI and then they service users can look up the service from the UDDI and then find their services.

This approach extended UDDI to log some precious informations such as invoking or execution or failure information.

Using this information from the UDD, from the extended UDDI calculate the reliability of each atomic web services.

They used just a simple Nelson model for calculated, calculating.

[70:00]

This is the one example for the log information based approaches.

And then second one is testing-based approaches.

They used the failure data from group testing.

This approach is, suggests WebStar infrastructures.

WebStar will perform the rigorous check-in test on all services.

This, this can be the web services in the same group, these are, the group testing will be conducted for these service candidates.

And then using this testing, some result will be recorded at a loss and then reliability values will be calculated by Nelson model.

This information can be used for when service user try to, when they try to know their reliability information of atomic web services.

It can be the, one example of testing-based approaches.

And third one is collaborative approaches use similar user information.

Each users, there are a lot of users who want to use the service, web services, each user contributes failure information to the centralized through the client-side middleware.

And then, and then, so service user A want to know the failure probability of service A.
They can references the similar users' information.

So they just find the, they just find the similar users using a failure probability and then if the similar users use, have used service A, service 1, user A can reference that information.

So it can be the one approach.

This is, this is the approaches for composite web services.

Most approaches use the Stochastic Workflow Reduction Algorithms.

This algorithm is introduced for reductions loops such as sequence and branch and loops and parallel.

This can be the composite, compositional structure of new web services, it can be reduced into because aggregation using this kind of formulas.

So using this reduction loops, service users can simplify the complex structures.

So they can calculate, compute the reliability of web, composite web services.

This is the comparison result of each estimation techniques.

The phase, application phase were, they were a little bit different.

Some techniques were for every SOA steps and failure data source were different.

So I'll explain the log information-based collaborative approaches and the important thing is the most of the approaches have adopted the Nelson model.

Simple Nelson model there for the atomic service and SWR algorithm for the composite services. I generated the hypothetical scenarios of web service reliability changes.

[75:00]

This scenario describe how well various reliability tech, reliability estimation technique describe dynamic web environment.

So, each scenario explains the factors which affect the web service reliability.

So some scenario is related to network condition changes and resource execution and service availability.

Using this kind of scenarios maybe each techniques will be evaluated.

For example, there is some dynamic situation between 6pm and 8pm.

A lot of service request occurs so maybe for this, during this, during this period maybe a lot of failures might occur so I just wonder each technique will consider, concern this dynamic situations.

So based on this scenarios each technique were evaluated by hand.

And actually some are very, some techniques were very good in some situations.

Some were mediocre and some were very poor.

And each approach, strength and weakness of each approach were suggested.

I want to discuss this topic from the survey study I have known there some reliability issue, atomic web services.

For atomic web services, estimation, left estimation most approaches use the Nelson model. Which is the input-based model.

The distinction of each technique were a failure, a way of failure data collection such as by group testing, or by log information, or by failure probability information from similar users.

Or they invoked the external web service hourly.

Some papers would estimate reliability from different viewpoint.

Some try to estimate the atomic web service from provider viewpoint.

And some are estimate them from client viewpoint.

So the result will be different according to the viewpoint of maybe client viewpoint, they consider more about the network condition.

But provider don't need to consider that.

And remaining open problems is there were no method or technique that, which covers all factors which affect the software, service reliability.

So hybrid method might be improvising, I thought like that.

And for composite web service, most of approaches applied architecture-based approaches.

As I mentioned, they use the Stochastic Workflow Reduction algorithm so I…

Just the, only difference was the representational way of a structure of services.

And the problem, remaining open problem was they only addressed simple compositional structures in the example.

And they, most approaches don't consider dynamic service binding.

This can be the…some web services adopt the dynamic binding for…

[80:00]

So dynamic service binding may affect the reliability of the service system but existing work don't consider that.

And dependency problem were not considered yet because in the composite service, each services might affect each other.

And assumption of usage profile information, they, usage profile information is the occurrence probability of each pass.

So this information were, this information was given by just the users by hand so this can be the big, big disadvantage of existing works.

And for the end-user programmer at the Mashup is the composition of data from various source.

So Mashup can be, if I assume the Mashup is very similar to SOA system, so SOA system, the reliability estimation techniques for SOA system can be applied to the Mashup.

So this is conclusion.

I identified current solution and limitations of work on the web-based reliability research.

Classified them based on some criteria and then I wil provide some insights for the future direction of reliability research.

As future work, as we know, SaaS is the emerging trend that has transformed the way of deliverying software through the Web.

So I will, I want to compare the existing reliability estimate techniques for SOA systems with the SaaS, SaaS work, work of SaaS.

This is all of my presentation.

If you have any questions?

Hello everyone, my name is Gyucheol.

Today I would like to talk about my class project.

The title of my work is Using human-generated invariants to improve testing quality in web software regression testing.

My project type is experiment so in this presentation, I will show some background, background knowledge and related works.

And I will show my experiment design and process and results.

And last, I will show some, I will show limitation of my experiment and discussions or feedback from participant.

Before we explaining main experiment, I'd like to talk about web application and testing.
The most modern web applications consists of modern web technology such as HTML, CSS, JavaScript and SML.

These technologies have dynamic characteristics.

So in a testing, software testing perspective, the web, testing web software, web application is is hard because of web applications have stateful, asynchronous, and event-based nature.
It means web application can update partial part of current screen, current webpage, and the most widely use of client side program language is Javascript.

So Javascript is loosely typed language so the type of variables can be dynamically modified during the execution time.

And there is a interaction between the servor and client-side.

[85:00]

And in the most execution time the client-side manipulation is especially in DOM.

So these data communication and DOM is occurred in backgrounds so users and testers cannot see the process of these data communicaiton and manipulation of DOM.

These are the reasons web application testing is hard.

To test web application the recent testing approaches are these.

So in a code side, there is Unit testing2 or JSUnit2.

It tests a Javascript.

Javascript code on a functional level.

In client side, in the browser the Javascript can modify the DOM, so DOM makes in the DOM-based testing.

There is a famous tool called Sellenium 3, this is most popular Ajax testing tool in the world.

It captures events fired by user interaction, and record them to reuse test cases.

Such tools like Sellenium3 can have access to the DOM and can assert expective new IP defined by the tester and replay the testers.

Replay the event.

But these tools have a problem because in this lot of manual effort on testers.

So automatic testing is needed.

To solve the manual effort in the testing researchers have proposed automated web application testing.

So they propose a tool that use crawl, to crawl web application states and save them to reuse test case to suite and to use the regression testing.

But it still need some manual effort for customizing an environment tools such as setting a depth level of setting of what should be clicked or what should not be clicked.

And these are customizing these professional time around 30 minutes required to test a single web application.

And there is some undetected default remained in this technique, this automated web application testing.

So the problem is manual effort is still needed and there are still undetected defaults remained, so I thought there can be some improvement in these problems.

So in the manual effort perspective we can use the crowd sourcing of collective intelligence to collect data and use them to test.

It can help to detect more effort during testing.

However before we research that crowd sourcing or collective intelligence, I thought the preliminary research is needed so I think if I conduct experiment with human, can human help to improve testing quality such as fault detection?

[90:00]

So I conducted experiment using the state-of-art tool in a research area, JSART and small number of participants.

From this slide I will show 2 related works for my work.

In this slide I'd like to talk about JSR tool.

It means Javascript Assertion-based Regression testing.

In this paper their main propose and developed tool for automated web application testing.

These tools' test process consist of   4 main steps.

First, to intercept javascript code and instrument javascript code between client and servor.

And navigate the whole web application and produce 2 execution traces.

And second, using the trace data the tool generates dynamic invariant.

In this invariant generation step, the invariant is made from Daikon system.

And third is unstable invariant, they retest the web application with previous invariants and filtering the unstable invariant and produce the stable invariant.

And finally they transform the stable invariant into the stable assertions and inject back them to the web applications for the regression testing.

In this tool, however there are some undetected faults still remained.

That problem is explained in the following slide.

And this is other related work for my work.

This is about the crowds…related to the crowd sourcing.

To improve test coverage they use crowd sourcing to help generate test input and the automatic software testing.

They propose they developed puzzle-based automatic testing environment that converts the

Object oriented programming challenges such as object mutation and complex constraint solving problems into the small puzzles for humans to solve.

So in this site, you can see the puzzle today make a convert object mutation and complex constraint solving problems.

The main reason for they made this tool is the object mutation and complex constraint solving problems might not so be difficult for humans.

So like in this PAT webpage, website, the 2 automatically convert to the constraint solving problems to the this like puzzle, and users just type the values to the each variables and if the puzzle is solved this inputs are used to make a new test cases, like this.

[95:00]

These test cases used in a testing and these test cases can improve test coverage of target program.

So from this slide I'll show my experiment design, my study design.

The experiment idea of my work is using human-generated invariants to improve testing quality in web applications.

So my research question is how much improvement is achieved when we use and add human-generated invariants to improve test web applications.

So, before explaining about the process of my experiment, first I'll describe my object web application named same-game.

It is from JSART paper, I select this example because the result of using tool generative invariant testing result is from JSART can be a baseline of my experiment.

And actually I sent email to the authour but authour just send me this one example.

So actually I use this example, and this example has 293 lines of code(LOC) and it consists of 9 javascript functions.

It is very simple web game just like visual ?[1:37:36] or Anipang.

You can click some cells, and there is a same color neibors.

Neibors can be removed together.

So when I click this button, this red color box is removed, and click this blue box and this blue box is removed.

But this simple webgame is consist of just javascript functions so it can be used to javascript regression testing.

So I conduct study with 5 graduate students and 3 students from web science and web technology and 2 students from computer science.

So before start a main experiment I survery the participant background knowledge.
They thought their programming skill level is 1 to 3 or 5, and all participants didn't know about invariant.

And 3 of 5 participants didn't have experience with javascript programming.

And using the participant invariant, I compared the result testing using tool-generated invariants and testing using human-generated invariants and test using both tool and human invariants.

I had a pilot study with 2 graduate students and 10 minutes pilot study.

They provided feedback for me about…they wanted to see the whole flow of web application and they want to explicitly write about the global variables and the outputs.

[100:00]

And one of them told me that if I provide this examples of faults or mistakes in javascript programming can be help to write invariant.

So according to this feedbacks I conduct study like this process.

Survey background knowledge of participants and I taught a basic background knowledge of invariant, regression testing and JSART tool and explain about object to explain application structure and fault of web page application.

And provide writing guideline of invariants and last I…they use invariant using the paper which contain 2 functions will have fault.

And finally I survey feedbacks from the participants.

So I want to use JSART tool to set a baseline.

I…before pass the tool generate and human generate invariant, I processed the 1-3 steps to make stable invariants as you saw the relative work in JSART.

This is tool generated invariant is a baseline.

And I convert the participant generated invariant to the 2 readable file because JSART tool use a Daikon assertion5 to test web application.

So I manually I convege the invariant to the file.

And seed faults and test using human generated invariants and both.

These are the faults which are received object application.

This fault is cannot be detected using JSART tool, so I use these faults to test human generated invariant.

So from this slide I'll show result of my experiment.

These are the numbers of invariant made by participants.

I condcut study with 5 students and they made these number of invariants but the number of invariants per person is slightly different because I thought the according to the background survey knowledge of participant, the participant's programming skill level is different and experience of javascript program is different so maybe I think these results are provided.

So use this invariant, I test the faulty web application but I actually, unfortunately I did not find some improvement in my experiment.

So I summarize the feedbacks from participants and limitations of my experiments.

So some of participants told me that they cannot understand the details of invariants and target application.

So maybe I provide more easier examples to understand basic definition of invariant and javascript characteristics.

And I should provide more examples of invariants.

And in an object application 'same game' maybe I…some of them need whole application structure or whole application code.