웹 소프트웨어 신뢰성

- ✓ **Instructor: Gregg Rothermel**
- ✓ **Institution:** 한국과학기술원
- ✓ **Dictated:** 김은비, 박원정, 한소영, 김보미

[00:00]

Hello, my name is 정형권, and today I'll present this paper.

So, web service … developer and company now days use web service for their applications.

So, they thanks to the web services, they don't have to develop the implement all of their application.

Just outsource the web service such as Ebay, Amazon and there are many web services.

So, service keep changing. Because to meet the various uses need the web service keep changing and the … to make application.

So, developer and company compose multiple web services.

The web service composition, so there is a compatibility issue, so if some of change in the composition can change that may affect the service compositions.

So, his approach is using audit testing.

It is a [01:48] regression testing so regression testing ensure that the existing functionalities are preserved at the applications changes.

So, in the regression testing, there are 3 major problems for 3 test cases selection and test case minimization and test case prioritization.

The test case selection is selecting test cases related to the changes.

And the test case minimization is to reduce the redundant test cases and the test case prioritization is to order test case to fine fault fast.

[Student Speaking]

So, your question is different between selection and prioritization?

So, I think selection is fine test cases related to the program changes.

[03:02]

And prioritization is ordering test cases to fine fault fast, is it clear?

[Student Speaking]

If … base on your opinion, it could be prioritization technique also be a selection technique.

But generally, a prioritization use the to … fine the most important test case which fine fault fast.

Generally, there are two techniques coverage based additional coverage based techniques.

So, I'll explain it in the later slide.

[Student Speaking]

I think service composer … developer yes … conduct audit testing in order to make sure that the existing functionality work whereas the service change.

[06:00]

So, is it clear?

[Student Speaking]

Service provider changes their service.

So, that is the problem because service consumer doesn't know the service change if the provider tell them their service change, that is kind of problem when we test the regression testing.

But we can know that many services provide the service change document for the service consumer.

So, service consumer can test their application based on all there are many approaches about to deal with the problem like search provide the notify the change information.

[Student Speaking]

Yeah, because if unless the provider notify the service change than the consumer will not know.

[Student Speaking]

So, there is the challenge when prioritization case in service testing.

So, service consumer cannot see the entrance structure of the service so, there is a low accessibility and observability.

[09:00]

And the invoking service can … is money.

If the number of invoking is above the limitation the service, provider define than the money will be charged to the users that is the kind of problem and because consumers cannot see the entrance structure.

So, they don't have enough coverage information of individual service and there is a issue the specific service changes peculiar properties of service compositions.

So, those problems will be handle later slide.

So, service composition. There are characteristic.

The firstly, it has aspect of globalization and personalization.

So, globalization is to aggregate data and resources from multiple service providers.

Multiple services which can be diploid the different serve and personalization is the service composition … to serve many various kind of users and it considers the users specific profiles.

And service composition has multiple input and output channels.

So, input channel is connection to the data and resources of service providers.

And output channel is connection to users.

And usually in the service composition, the problem with one channel doesn't effect to the other channels or little effect on the other channels.

So, this is the example of service composition.

So, there are 3 interfaces and back and component.

[11:56]

So, if there is change on the car service than user who do not use the car service will not affect the service change.

And there are two handlers according to the country.

So, that … only specific user can affect the changes so I'd like to talk about the prioritization more detail.

So, usually we use this because we cannot test all test cases because we don't have enough resources and time and cost.

So, the goal of prioritization is to fine the most important test cases.

Goal of it is the most important test cases are executed first.

And there are two particular techniques so one is coverage-based prioritization and the other one is additional coverage-based prioritization.

[Student Speaking]

So, I think the most important test cases is … likely to fine fault.

The test cases we make test case to fine fault so the good test case is to find fault that is the most important test cases they mentioned I think.

[Student Speaking]

There are coverage based … you mean that …

[Student Speaking]

[15:00]

That is his approach, yes

[Professor Speaking]

First question about selection versus prioritization through the first question you had.

We got an existing set of tests.

Your existing set of tests regression existing tests, technically speaking the selection problem is to partition this in the two parts.

Test you need to run, and test you don't, okay?

And you are going to run all of them, no notional order okay?

So, it's just going to say, hey run this, you don't need those, that's selection okay?

Prioritization says, here is your set of tests. Remember set! Doesn't to flying order.

Take those and put them into order and run them in that order, now they can obviously be intersection.

I could select the sub set and then put those in order.

I could take my ordering and say select the first 50% but in their pure forms they are two different things and this paper is looking at prioritization.

[Student Speaking]

Yes, minimization is [17:17] testing their … we got in the test with … we talk about coverage testing right?

And the test coverage different code items, if all you are interested in coverage, you may have redundant tests.

Tests don't give you any new coverage and so minimization tries to reduce this to the subset coverage the same things minimally.

It's [17:44] problem. So you only [17:46]

One way to do selection is to minimization relative two some set of items you want to cover.

That's a minimization selection technique.

[18:00]

The second thing has to do with prioritization there are two things going on the prioritization.

You got the set of tests okay?

You have an objective function that's your objective thing you're trying to maximize or improve them up and you have a heuristic.

So, the common object to function the one to talk about here is detecting faults faster, faster okay?

If I have 10 tests and the test one to test faults one, two, three.

Test two to test fault one. Test three to test faults two and faults three.

[19:00]

I can detect them faster by just running test 1. That gets them all. Ok?

Whereas if I do test T2 one first and then test T3, and then.. excuse me, test 2, 3 and 1, that's a slower detection.

So, if my goal is to detect fault faster, the way is not in our context in the original prioritization, it was based… we had the notion we wanted to detect fault faster.

I've got a thousand tests. They take me 2 weeks to run. I'd like to detect the fault in first 2 days if I could, cause I can start debugging them all.

The problem is, I don't know which test detects which fault. If I did, if I knew, I'll put all the fault detecting ones early. But I don't know. I don't know until I run them.

So I try and find some heuristic that I can hope predict the test that will get the faults.

So, one heuristic is code coverage.

I got a bunch of tests I want to know which one detects fault fastest.

I'm going to say the test that cover the most code are the most likely to detect fault first.

So if I got one that covers 80 statements and one that covers 60 and one that covers 30, powered on that and that and that. It's just a heuristic. They may not be true.

But that's what's called the total technique, I mean, that's the coverage based technique.

Additional coverage based says, never mind the number, it's the number of unique things you cover.

So this covers, initially, this may cover the most.

It looks like this covers the next most, but if all 60 of these are recovered in here, already, and this covers 20 new things, maybe it's best to do this next.

So that's like pick the one that gets me the next most coverage.

There, the intuition is that covering code that hasn't been covered is likely to give reveal fault faster.

[21:05]

But you get to separate these two things out.

Usually you have a goal, detecting the fault faster.

But you got no way to get that directly, cause you don't know which test to detect which fault.

So you find that heuristic, that you hope will maximize this.

And that's exactly what's going to happen here, they'd like to get fault faster, they're going to get a heuristic and that's what we're going to hear about. Does that make sense?

[Student Speaking] I'm sorry, but how can we know which coverage are additional, I mean, if we compare 2 test cases, 2 coverage, that I covered by two test cases, then we can compare additional coverage, but in additional testing, how can we know which one is?

Maybe I left this out. These techniques used to coverage of prior version of the program.

Maybe you mean this, I will show this, you have P, and T was associated with that version, you created a new version. You haven't run the tests on this yet.

I mean, certainly, if you wanted to put the test in order, here, I'll propose this. You want to run the test, in a good order, run them all on this see which ones detect the fault best, and then that's the all you need to run them on it.

Well, that's silly because we're trying to predict what to do this without running the tests.

So, in coverage based, we're usually using coverage information from P, the assumption is that that would be a predictor of the coverage they'll get on P'.

So that's another source of approximation. That's in that. Make sense?

[Student Speaking] Yes. So we don't need to have test cases to program P'? We don't execute test cases of P'?

Right. So the whole assumption is, you're trying to do this all analysis before you do the execution in order to maximize something on that execution.

So what information is available? You have prior coverage information, you have code change information, you may have document change information, you use all that to make an order in that you hope will achieve your objective.

And then in the papers, you have experimenting with these all the different heuristics trying to find one that does do better.

In experiments we can compare to the optimal ordering, cause I can take in an experiment, I can take, run all the test, I'll finally, well, the best ordering, would it be the one that put them like this. So you can do these comparisons. There. That.

[Student Speaking]

[24:10]

So, this is the example of the coverage based prioritization and additional coverage based prioritization.

So that is the sample program, so there are three test cases, so coverage based prioritization is how many code lines each test case cover, so, in that case, test case 3 has the most coverage, so test case 3 will be selected first.

And then next is test case 1, and test case 2.

But in the additional coverage based prioritization, the test case 3 will be selected first, but the next test case 1, the coverage of test case 1 is already covered by test case 3, but test case 2, there is one additional statement, which are not been covered yet, so the next test case will be the test case 2. That is the additional coverage based prioritization.

So, the coverage based might not good indicator for the prioritization, so he showed this example, so that is the service composition, and there are 5 test cases, and if service changes in the composition, and only the test case 4 can find its fault, related to the service change, then when we use the coverage based prioritization, the test case 4 just, smallest covered.. the smallest part, so it will selected last so that is the problem they say.

So that is the motivation of using his approach.

So some code portions which are ready to the service invoking, so many service provider provides their service change document on their website.

[27:05]

So the service change document could be a key to identify the code portion related to the service invoking part in the service composition.

So they prioritize test cases based on terms in the code, the terms appear in the service change description. So, to do that, he used the information retrieval technique.

[Student Speaking] I'm sorry but, if we achieve 80% coverage, I think they will be many words than 70% statement coverage. So I mean, frequency of terms occur in

an execution when collecting coverage achieved by the test case.

That might be related, but I think if test cases are… test case has many terms that appear in the service change document, then the test cases will be likely to be affected by the service change.

[Student Speaking] So there are resist corrasion based on changed description, we can find test cases that are related to changing description.

Yeah. Finding test cases related to the service change documents.

[Student Speaking]

So your question means if we use source code rather than service change document, then we will get better performance, yeah, that is true.

[29:58]

But in the web service area, we cannot see the source code over web service, because as a service consumer, we use the services from service provider.

And service provider provides just the interface of service. So the service consumer can just invoke operation in the service, so it's kind of black box testing.

[Student Speaking] Previously, before this technique, how did service consumers prioritize, in your explanation, coverage is the best prioritization but is not applicable for this area. Then what was the previous technique for this prioritization?

Can you say the question again? I didn't understand.

[Student Speaking] From your explanation, my understanding is that coverage based prioritization is not applicable for this area, web service, then what was previous technical prioritization?

In the service composition area?

There are... when you see the related to work part, then there are some previous approaches, and one approach is coverage information, but the service provider provides service interface document with code WSDL, and the WSDL and WSD schema and so on.

So they.. the one approach it use those interface document to find the most which use the coverage based technique.

[Student Speaking]

[33:08]

So information retrieval is to rank test documents, so for example, Google search, so when we search at Google, we can find the most relevant document with respect to query.

So we can see the trace, trace information of each test case as the document, and we can see the change description document as a query.

So there are many execution information of each test cases, and there is a query as a service change document, and find most relevant test cases by using IR technique. So that's the approach.

So they may identify a document by analyzing the coverage report its test cases and which include the code lines and they split the identifier document into the composing terms and kind of processing like filtering and stemming.

And this is the service change, the example of service change description.

So they also manually keep some keyword out of documents, and so that is the query they use.

So they also point out that this could be a big point of their approach, because they should pick some keywords from service change document manually.

And so for the information retrieval tool, they use Apache Lucene, and Google Desktop.

So those are which the two tools use vector space model, it represent document as a vector and relevant semantic indexing, so it's like reduce dimension, get rid of noise information.

[36:06]

So, case study, firstly evaluation matrix, they focus on the fault detection rate, so they made a fault matrix to measure the rate of fault detection.

So here is the fault matrix table.

So those are the fault and these are the techniques, prioritization techniques, and this index indicates the index of first test case that reveals the fault.

And they made a sample program which is called eBay finder, so it compose multiple web services, like eBay finding APIs, eBay shopping APIs, Google map APIs.

And the code lines are 20,000, and they made 160 test cases.

And to develop this application, the application is java code, so Google Web Toolkit

transforms the Java code to Java script.

And the test suite is based on the functional adequacy criterion like black box testing. And they are categorized faults.

So faults, there are two type of faults.

So, firstly, generic fault is related to the… is affect all input and output channels.

It is independent on the use of profile.

And the other fault type is profile-specific fault and is affect one particular input and output channel.

It is dependent on the user profile.

(Student Questioning)

[39:01]

The execution.

So, we can see the service composition as multiple input and output channels in the paper.

So the generic fault affect the code coverage which is not independent on user profile such as… user profile contains… includes like place and time like that.

So it is not related the… that kind of fault is not related to the specific user profile.

(Student Speaking)

Sorry, can you…?

(Student Speaking)

It's related to coverage.

And the coverage… the code portion is not related to the user specific profile.

So it affects general code portion.

There's a paragraph of that on the page 640 on the right.

So, they started saying generic means affecting all IR channels but then later on they say generic involves locations executed independently of user profile stating.

There's profile-specific… the locations… only are given to certain users.

You know, only certain users' experience.

[41:58]

They discuss it was actually two, but they did't use those terms there.

As I see it, a generic one is one if it might affect anyone who use this regardless of how they use this.

And a profile-specific might affect some users but not others cause it depends on how they use it.

Does that seem placed?

Maybe, I can show you the example.

so, in this service composition, example, so user profile means like users country or the some user only use car service, some user don't use the other service, hotel service, dry service, so user specific fault means only the fault is related to the user profile like country.

So the eBay service change the country code or like… some user specific code in the later version then it might affect the service composition.

So that is the user specific fault.

And generic fault is… regardless of channels, they might affect the service composition.

So first study is about generic faults.

So they perform the coverage-based and additional-coverage-based and IR-based test prioritization.

So coverage and coverage-based and additional-coverage-based produce only one ranking regardless of changes but if we use IR-based technique, then it produce multiple… different ranking according to the changes.

[45:11]

So there are… they injected five faults, generic faults in the service composition, application and…

So if we use IR-based technique, then it produces different ranking information according to the changes.

And they define two types of identifier documents.

One is just consider the class and method signatures covered by a test case.

That is the example.

And CMR-I also consider the request contents like the bid, count like that.

And they define two types of query.

Firstly long query is just the service change description itself.

And short query means the awesome manually the… extract some related key word from the service change document.

So they use the five change descriptions

And based on these changes, they injected five generic faults into the program.

So this is the result.

So when they use short query, fault 3 and 4 and 5, all techniques perform well but in terms of fault 1 and 2, the IR-based prioritization has a better performance than the coverage and additional coverage prioritization.

And in terms of fault 2, when we consider just class and method signature, then the IR 2 cannot find any relevant test case about fault 2.

[48:08]

But when we consider the request message which contains invoking service, it can find the relevant test case.

And the second table is about using long query.

As you see, the performance was not good, so the author recommend not to use long query.

And study 2 is about profile-specific faults.

So they use changes in the eBay Finding Service.

They injected 8 faults which I handled by 8 specific eBay stores, so they use 42 test cases.

So this is the result.

So, as you see the performance using the IR-based prioritization, outperform the one of the coverage-based prioritization.

So those are the related work about web service composition and testing… eBaytion testing.

So this is a conclusion.

So this paper is talking about web service composition and web service evolution.

So, as web service change, we need test the web service composition and his approach is about test case prioritization.

So they perform the coverage… they compare the coverage and additional-coverage and IR-based coverage technique.

And in the result, all three technique perform well on generic faults but in the profile specific fault, IR-based technique outperform the other techniques.

So that is the end of my presentation.

[51:01]

(Student Questioning)

Can you explain why IR-based technique outperforms other techniques on profile default.

So, your question is… I think related to this example.

So when service change and only test case fault can find the fault related to the service change.

But let's say the test case fault had small coverage.

In that case, if we use the coverage-based technique, the test case fault will not be selected first, so that is the answer I think.

So any other questions or discussion?

In context of regressive testing, the one most important thing is that preserving functionality before version change, right?

Then, in this work just show… in this case studies just show they find differences only… they find fault only.

That doesn't imply the existing other functionality are still preserved.

So how do you think this test cases, this metric in this paper?

In regressive testing, preserving functionality is most important factor.

[54:08]

But in case study in the paper doesn't provide that kind of information.

They just found fault or not.

so if there is fault, then obviously the functionality is not preserving but if there is no fault then that doesn't say anything about preserving functionality or not.

So… but in the paper doesn't say about preserving functionality at all.

So, how do you think about that doesn't make sense in context of regressive testing?

We cannot find all faults in the program so… but when we use his approach, we can find test cases which are likely to find faults related to the service changes fast.

So… did your question is…

It's actually interesting question because it actually goes back to a higher level onto all of regressing testing, right?

So you started by saying preserving functionality is the goal.

First of I say is one goal, but also you enhance systems right.

Sometimes you're correcting bugs… sometimes you're enhancing the system.

So enhancing may even involve removing.

But, definitely, one goal of regression testing is to make sure things that used to work haven't stopped working, okay?

Now there may also be new stuff added, that now you want to work.

Think in a… in this context, people who are providing services probably they want to make sure the things that used to work still work preserving functionality.

Sometimes they may be adding new stuff too.

[56:59]

Testing generally can't prove that a program is correct.

It can only find faults.

So I guess I want to say testing can't prove that you preserve functionality.

(Student Speaking)

OK. That's true. I mean one thing they could do in theory is run all their tests, all of them.

Never mind part or this. Do what we call retest all, 'run all the tests'.

I mean at least then, you can assault that you preserve functionality relative to those tests, right?

I mean it's just limited test rate but... and in a sense of prioritizing, if you only run some of the tests, you've really only shown… If you found no faults, well then you've shown you preserve functionality relative to those tests.

And… let me put it another way.

If you've got 50 tests and you can afford to run 10 of them and you want to show that you preserve or you want confidence, evidence that you've preserve functionality, you'd like to pick the 10 which if you had not preserve functionality would be most likely to fail… would be most likely to show you that fact ,right?

And so like 10 that go through the change are more likely to show you if you haven't preserve functionality.

And by inverse if you suggest that you have, then 10 that don't go near the change.

So basically I think so what they're do… if they are looking for test faults, it's related to showing that you preserve functionality

[59:04]

But on the other hand, their measure, I see something different there, too.

I mean what confidence that we have that we preserve functionality.

When we've run 1/10 of the test rate, it could be a whole another measure, right?

If you prioritize and stop short, what's the chance you may miss things?

And therefore wrongly assault that you preserve functionality.

So, what am I saying? It's an interesting way to look things this notion did you preserve functionality.

I think it's… in some sense, it's related to, it's likely the contra-positive of are there no faults related to changes, right?

If there are no faults related to changes, then you preserve functionality, ok?

But you can never say there are no faults related to the changes, cause you can't run an infinite number of inputs.

You can't say there is no faults related to the changes given my… in as far as I can tell from my test rate.

That's better of you.

But now if you run few of your tests, how confident are you in that?

That's someone knows. That's all.

Be a whole different measure for a success of prioritization instead of saying… OK.

They are talking about, well, how fast detect faults.

And a whole different thing is how confident are you in reliability of your system.

That's the different thing that how fast detect faults.

And so, people have talked a little bit about prioritizing tests in a way that let your confidence in reliability grow faster which is a different thing that let your detect faults faster
.
So, they are definitely looking at faults, not this other possible measure.

And most people have looked the faults.

But the other measure is interesting, I think.

(Student Speaking)

What are measures to case the reliability?

Oh, measures mean time to failure.

(Student Speaking)

The reason I asked this question is that the compared technique, additional coverage based and coverage based technique is more likely find fault, more likely check program behavior rather than fault finding faster.

So, maybe there can be another techniques and said same thing.

[1:02:03]

So, my concern is that this comparison is not suitable for showing their technique is better than other.

In that sense, it's your definition of better, right? Better.

If there definition of better is find faults faster, that's one definition of better.

That's some engineers are interested in.

But if, but it's not the definition, that says make the system more dependable.

And… so you're right.

There could be... if they have… you could be saying if this is all constructive ability [?1:02:43] threats, you are saying your measure of the goodness of prioritization order doesn't reflect what engineers would really want.

And that maybe true to some cases.

What's interesting is that most of prioritization literature looks at this fault detection thing.

Because you're right. There's a whole another potential goal.

So they've chosen that but you could say, they've chosen fault detection but that isn't really of you of what we are interested in in terms of reliability of our systems.

(Student Speaking)

Is it fair to compare with coverage based technique and additional coverage based technique in context of fault finding capability?

Yes, because that's what… there are two answers.

The early stuff I did was saying let's compare coverage, let's assume coverage helps detect faults faster.

And that's the way we prioritize and so that was using coverage… that was fault finding was the goal and we are looking at coverage to find faults.

And they are saying if fault finding is the goal, we've got another approach that's better at that goal.

But I've, another way to answer that is I think if you were interested in showing that your program is more dependable, covering code might do better than attacking

possible fault locations.

So, but you know, as long as we are talking about their comparison, there are couple other things that bother me.

Ok, they compare to basic coverage based approaches.

Right from a start you can see theirs are likely to gone, I mean, the intuition for theirs is that, faults are more likely to occur, the more times you invoke the services that have changed, right?

OK, that's obvious enough.

[1:05:00]

And just how many times you cover code doesn't track with that.

It's how many times you cover the invocations to code that just changed.

Or there are regular prioritization techniques whose goal is to cover the code that has changed.

And that's exactly what they are doing.

So in substance their technique is just… cover my early prioritization papers, one technique is cover the code, another is cover the changed code.

And that's just what they are doing the only different thing is how they identify the changes through the IR techniques.

(Student Speaking)

Right, so I think in English we sometimes we have a seeing, if I were to say, well, there are really comparing to 'straw man'.

Have you ever heard of that?

'Straw man' is a man made of straw.

Straw, grass stuff, straw, right? Not a real man, ok?

If you want to, so, ok, I am a warrior.

You are comparing my ability to kill enemies to that of a straw man?

There's no real comparison.

So when we say it's straw man, it really mean it's a thing you are comparing to that you are always going to beat. Ok.

That's the way of saying it's not the right comparison.

And I agree with you on that.

I think the coverage, another way to look at is as you observed in related work here. There are some papers trying to attack this problem.

(Student Speaking)

So and we would call, there's another, we sometimes talk about the state of the art and the state of the practice.

State of the art actually is research, the boundary of research.

State of the practice is how far they are going to practice.

This related work section talks about state of the art.

And they don't compare state of the art.

They compare to what I think is a straw man.

That's my opinion.

Other comments?

I think the experiment conducted in this paper are artificially controlled by the authors .

For example, they made their application and tests even they see their fault by their own.

Yeah, so what do you think about this, I think the result does not show their technique is effective at all.

[1:08:06]

Could you come and pose this question additionally?

Oh definitely. But let's see.

Yeah, I think there is the external ability and also internal ability.

So they could made fault for their approach.

So their approach better.

That is internal ability.

And there is, they use only just one application that they made.

So it's how to adjust the other well, applications well, service composition.

So, yeah, I agree with your opinion.

So answer the question becomes then how could they have done better.

Well, in terms of the applications, far better to go out and select one.

Now, you could still say oh, they went they looked at 15 of them, they pick the one on which they do best.

Take a shit, ok?!

You'd rather, that's why we are so much happier with random selection in some way.

But that's a hard thing to do that.

We are having more than one would help.

As far as the faults, well, the very best situation is when there are real faults cause then no one can say anything but usually there aren't enough of those.

So what some other authors have done is, basically, what I often do, I've got a full-time programmer in my lab.

And sometimes I'll give him programs since they well, here's classification here's text anomy of faults

I want you to put 20 faults into this program that are similar to ones you've seen in practice.

And I don't tell him anything about why or what the technique is, ok?

You get people who are not the experimentals or who are not the authors to do this ceeding. [?1:10:18]

You get, now, they I mean not be real faults.

That's you are concern then, well, how well did the people do.

People different from the authors and even better multiple people will give you a

much will help your ability a bit

That's what they could have done.

The last thing about that is single fault at a time where most all the prioritization work really is considered with multiple faults and the rate of fault detection not just what was your first test detected.

[1:11:00]

What are you doing for your project?

Yes, for this class.

For this class…

My project is also regression testing on web service composition and my approach is prioritized based on, my approach is the test case prioritization and the criteria is, I am, my approach is using the dynamic slicing of its execution trace of test case which means dynamic slice is the portion of code in the program which is related to the outcome of test case.

So, I am going to use that slice as a document as this approach and using IR technique.

So I will fine most relevant slice for the query which is the service change document.

That is the subject of my project.

So from a discussion here you've already heard something this people could complaining about in your study depending on what you do.

Obviously there's limited time here so, maybe, you might, you probably can go out and get 10 web services with a hundred faults on that.

Don't worry, we understand.

Just be aware of what you are doing and be able to say.

Are there any question?

(Student Questioning)

That conference is ICWS.

It's quite a famous conference in web service.

(Student Questioning)

[1:14:00]

(I am wondering, I don't know what their sub… (Professor Answering)

I can pretty much tell you I don't think this…

(Student Speaking)

The main issue would be the evaluation, I think.

But now the web services conference can be much broader field.

They may not have a lot of people… prioritization…

You can look up the acceptance rates…

Whereas so if you are start seeing, start seeing is 13 percent above…

Because obviously if you want to focus on web services reach the people who do those…