Lecture 1.1 By Alessio Ferrari : A Review

Muh. Riansyah
Faculty of Computer Science - University of Indonesia
Kampus UI, Depok 16424
muh.riansyah@ui.ac.id

***Abstract*—Using scientifical method.**

***Index terms*—** Empirical

1. Introduction

Welcome to this class on empirical metal doctor engineering I am my left upper body I am from CNR National Research Council, its a research it was shown in Italy and if you want to drop me an email, please use this email over to the what do you know. Florence that you wanna send to my information about this course so this course is dealing with software engineering. So we are not just dealing with programming, but with all ecosystem of aspects that are related to programming. So also testing but also designing software but also managing software engineers . we are speaking about empirical methods.

Empirical methods are scientifically based methods to analyze the activities related to software engineering. But first of all since I don't know which is your specific background in in software engineering, I will give you some some indication of what what we mean by software engineering. So what is our scope of interest.Then I will give you a brief overview of what you will learn during this course.

2. What is Software Engineering(SE) ?

1) So first of all what is software engineering ? We can read this slide, and then we see its the systematic design and development of software products and the management of the software process. So the keyword here is design development and management. So it's not just coding but also creating the architecture of the code designing the code planning for the software engineering process and managing the software engineering process. Another key word that is not read here is systematic. So systematic means that we have a set of strategy that we put into place to control the whole software process.
2) Let's read the second point that tells us about the goals of the SEineering. We have software engineering as one of its primary objective (is) the production of programs that meet specifications are demonstrably accurate, produced on time and within budget.

What does it mean? It means that the first goal of the software engineering discipline because it's a discipline like math let's say or physics. It's to create programs so running code that is in agreement with what the customer wants the meet specifications.meets the description of what is expected from this software are demonstrably accurate, so we can show that it meets the specification in a way that can be it can be demonstrated,proven, tested. Okay it's produced on time and within budget. These two aspects, time and budget. We will of course not touch so much in this course. But I put them here because we have to consider that in any activity related to management. You always have to do with aspects related to time and aspect related to budget like in any activity of engineering. it's always a trade-off between effort that you can spend (and),time and money that you have and you can invest, and objectives you want to achieve.okay, so it's always an engineering problem. it's always an optimization problem.

3. Scope of SE

Take this into account, while you think about software engineering what is more in detail our scope what will we deal with during this course. we will deal with a set of actors and a process basically what I was speaking what I mentioned before what the software engineering process basically what you see here in this light is an extremely simplified version of the software engineering process that aims to build software that is in agreement with the specification the everything starts in the top left corner with the a meeting between an analyst called requirements analyst or business analyst or analyst in general which is a person or a cell over a group of people that interact somehow with the customer or the users and that will detail later on the distinction between customers and users and try to understand what they want for example if I am working for a company for a software company that needs to develop tools for the management of hospitals which are particularly relevant in this moment. I have to interact with the management of the hospital, but of course most of the people that I will need to interact are also the doctors. That may be possible user of my system. I will need to interact also with the information technology manager of the hospital. So all these ecosystem or stakeholder,the sales of people are the ones that I have to start a dialogue to understand what are the requirements for my system.

For example as imagine in this case that I need to develop a new system to allow all the patients to record the status of fever for example or of cough so that I can know all the symptoms not just of the people that are there in the hospital but also of the people that say at all in this in this period so to set up everything to set up everything I will have to interact also with the family with the general practice doctors so the family doctor to understand what is their typical process for visiting patients and and for diagnosis and for notifying for example to the Central Hospital their disease and I have to understand whether a process is already in place or something needs to be added so this just to give you a glance of the complexity of this starting point and the needs of interaction with several several subjects and several stakeholders okay so once I have interacted with all these people I can step to the second phase which is the so called requirements phase in which I write down my requirements writing down the requirements means really writing down a document this document is a sort of contract and for for a long time it will be a living contract within the software process basically in this document to write down statements such as when the family doctor interact with the patient and know that he has a fever he has to write down the name of the patient and send press this button that sends the information to the central database of the hospital so requirements are documents in which everything it's it's written down and they act as a form of contact with between the customer and the analyst okay so basically the representative of the hospital will write a signature in the ending the requirements document not in the requirements document but in another form of agreements let's say that say okay what is written there it's really what I want okay then you have different forms of requirements like you have the one that the customer can understand and there are the ones that are more detailed and more oriented to the system designer and the developers because this document once it's written it is useful for the system designer who is we pass to the second step so you have your requirements document and the system designer needs to interpret what are the actual needs of the Crysta colder and needs to build the project of the software so the system designer is like the architect the software architect the one that I have designed here the house to represent the fact that somehow I'm creating the projects for something are not really developing at this time. it's like designing for example with UML. if you are familiar with that or sysml. if you are dealing with more complex hybrid systems and this project apart is passed on to the developer. the developers of course here I put just one subject, but imagine that all the time here you have a team of people and al so I didn't include management people we will see we will see later on but basically after you have designed your software the developers have to each wha each developer has to build a part of the software if you see if it is very large and they have to be of course integrated later on then you have your software you produce your software and afterwards this software needs to be tested so you need to verify that what the software does actually matches with the requirements but then you have also to verify so that you don't have bugs and you don't have both that it runs correctly and is in agreement with the specification so this piece will not speak about specification I speak about as a synonym in this case of requirements that are not they are not really seen on him in practice but for what we are concerned here basically when I speak about requirements I speak about specification for the soft okay so the tester is checking that the software verse works appropriately and but then you have to test with the customer so you have to check that the customer is really satisfied and that the analyst really understood what the customer wanted this is one of the most difficult thing to achieve because of course in this line in this path they have made several incidents may have occurred several misconceptions several there are several different subjects involved and things may have may have gone in the wrong direction so the customer might not be satisfied so the customer interact again with with the system analyst and then requirements are updated at this point there is a system designer again or the system maintainer that takes the role of mantener could be the same person could be another person that updates the project and then the project passed back to the developer and this cycle continues and it continues also when you have some new request from the customer because probably like for one year we work with you with our all with our system that controls the status of of the people at all but then we understand that some additional symptoms it is relevant.

For example I don't want to monitor the coronavirus anymore. I want to monitor also for this problem that is emerging that is affecting the skin of people. okay so I need to modify the software so that general practice doctors can also enter this type of symptoms or the people themselves can enter these this type of symptoms okay so in this case the customer goes again to the software company requirements our updater the designer is redesigning the software and the developer develops so this is like in in theory. how things should work of course in practice things work in a much different manner everything is more chaotic and is naturally more chaotic because first of all you have to have the customer there it is the customer don't always have time. The customer is made of several stakeholders these stakeholders they have to do their daily work they're not always reachable and people in your company may have assumptions so and they may develop something that is not really what what was written in the requirements for example and the several other aspects that may happen for example your developer is new is a newcomer and is not expert enough for this type of system doesn't have a background for example in medical system device so it there are a lot of aspects that you have to take into account in this process ok ? ?

so this is to give a brief brief idea what software engineering is dealing with. its dealing with all these people and all these artifacts,requirements, software testing is not represented here but you have the tester and the design the project ok ? the architecture. so in this scope and you question up to any questions so far ? no(kata muridnya). so this is the basically our scope.

4. Typical SE problem

So in this scope, what are the typical software engineering problems? So by software engineering problem, I mean problems that are asked to a software engineering researcher or to someone in the company who has a management role. Somehow all these different subjects of these different developers and testers and architects etc, ok ?

1) How can I find bugs in my code? So i see that my code for for in my company, my software company, I see that there is a lot of problems with this software, that crashes and I need to find I need to find a way to identify more bugs.
2) How can improve my requirements? I see that always. These requirements are whenever I come back to the customer, they are never really right. There is no way to to get these requirements documents in a proper way or I can see that every time I hand on this requirement to the developers they just don't look at them and they they get rid of the document and start and start coding because they didn't say they don't understand the requirements so how can I do with that. So this is another type of problem another type of problem in that same process.
3) How can improve software development speed? So I can see that everything worked nicely and smoothly but we are very slow in delivering so one of the main requirements of the customer is I need this in this short time and I need this early because this outbreak is spreading. So I need your solution for the covid-19 in short time ! So how can I improve the development speed also in this particular context should I a trade or some or some or some parts of the process.
4) How can I reduce the resources dedicated to testing? For example or other problem, how can I reduce the resources dedicated to testing? I see that like in my team I have two developers and three people in testing and these are needed to achieve a court that doesn't have bought but I'm feeling that I'm investing too much in testing and I could invest more of the money in in using people for developing using people is not it's not a good word. But let's say in having people developing code instead of having them doing the testing activity.

So in the our typical software engineering problem, to give you an idea of what questions can I ask in that context and these are the typical software solutions.

5. Typical SE solution : when we are not empirical

1) Solution for How can I find bugs in my code?" is "This new testing environment will allow you to find all the bugs" If you don't consider empirical software engineering, so if you don't take into account what is the main scientific focus of of software engineering. So without a scientific glances, the typical answer for "how can I find bugs in my code ?" is this new testing environment will allow you to find out the box. So there is. The solution is always fine searching for some tool. let's solve my problem, okay ?
2) Solution for : How can improve my requirements? is "We can use a controlled language" Another example, How can i improve my requirements? We can use a control natural language so instead of writing the requirements in English or in Italians or in Italian or in French or German we used ah we use a constraint language a simplified language that is controlled and that is not ambiguous. So I am sure that everyone will understand them clearly.
3) Solution for : How can improve software development speed? is "Let us use this new prototypical programming language" For software development speed let's use this new prototypical programming language. So let's let's introduce a new programming language that for example is easier for everyone and let's let the people deploy the code faster. For example instead of using Java let's all use Python because so that instead of writing *system.out.println* I can write I can simply write *print* and this will speed up the development.
4) Solution for : How can I reduce the resources dedicated to testing? is "Lets use model checking" Let's use motor checking this is a new technique. But it's a technique for replacing testing basically all at least this is the the general hypothesis that you can replace testing of the code with exploding the system space and searching for four particular states. In this in the graph that represents all the software evolution so it's a solution tool based often try to solve my problem.

6. Typical SE Failures

But then in this typical software engineering solution received like not, not so let's say happy answers or like we didn't as a software engineering. We didn't succeed in this direction in building tools that were solving the world problem. Actually the the answer where this one yellow (see in the slide).

1) Failure for "introduce a new testing environment" is "It is very complex! I need to retrain all my team!" Like for example if the if I introduce a new testing environment, (then) i need to consider that it can be quite complex to introduce a new tool in the team, so I have to train and I to dedicate a lot of time instead of searching for bugs. To train the team in using this tool because it's very complicated and I don't have time to compare different tool and select the best one. So I have to (and) I need to I need to consider always the time and cost aspect of the software process.
2) Failure for "How can I improve my requirements? " is """

user control natural language ? yes ! but then I tried the language and does not allow me to express what I want ! So on the one hand this control is not ambiguous but on the other hand I cannot write everything that the customer wants.

3) Failure for "software development speed" is "" For software development speed ? well, yes! This programming, new protegee with a programming language is very simple very effective but it doesn't work on real cases like it's okay in tiny programs it may work but it doesn't work for real cases.
4) Failure for "" is "" And then for for testing let's use model checking! But then I try the model checker and I see okay but here I need a very good expert very an expert in model checking and it takes a lot of time to to learn to learn this new tool and the language that I have to use is too strict so I have several problems that before I wanted to solve with a general solution and general purpose solution mostly based on a new tool or a new language and that required most of the time D to disrupt the existing software process in a company to change it radically.

7. SE illusion

So this was not the direction to go. The illusion was that given a problem we would invent a solution and then make money of course we thought that our tani solution would scale up to all real world cases and we thought that making quite successful simple example was sufficient to ensure that our idea for solving a software engineering problem was working so the thing the main problem was that we could pass without doing who pass from problem to solution And from theory to real practice, real world without,basically no in the world,ok ?? So we thought we could change the world as it's written here without actually first studying the world and understanding better the problem space. what is the problem in the software engineering development process ? it before we didn't ask this we mostly asked ! we mostly focused on a possible solution that would make us happy in a tiny in a tiny working environment.

8. The Hard Truth

The reality was that of course we were wrong because software development is complex this constant dependent. As I said before there are several stakeholders source objects that have some some relevance throughout the software process different professionals, different needs and different domains. Imagine that 20 when you think about software you are thinking about something that is both in your mobile phone and also in an aircraft and also in your washing machine.

8.1. Different Domain Apps

If you don't have a very old washing machine you may have software also in that and also in your car so all these domains they all have typical technical aspects. That are all different so a mobile app to track your diet diet is very different from a software that controls a shuttle ok ? so different processes are needed for them and different knowledge is needed so the real solution that works for every real-world case it doesn't exist ok ?

8.2. Different Company

And consider that you have different contexts also in terms of companies because you may have a small company of just two people you may have a very large company that multinational company or a company like Google that that is so large that of course you don't know much of what's going on in another branch of the company or your colleagues and the actual process that is carried out may be different even within the company so even the within branches of the company why not.

8.3. Building from legacy vs scratch

Another very important point that you should consider also from your from your work maybe is that you rarely start the development from scratch so we always teach we always teach students to start a program write down the code to solve a specific to solve a specific problem but the reality is that whenever you go to a company software already exists there and you have to refactor a softer extend the software there is always the legacy system to refactor also when for example in the example of the of the system to monitor the coronavirus. The the fact is that very often the company that will be called to do. That software most likely already has some interaction with with them with the doctors if the company already has some software some basic software for for the hospitals because otherwise. Also it is not possible to build everything from scratch in short time so you have always to consider that there is a past. This past is made of contacts, but also made of existing software. And even in the rare cases in which the software to be developed is new all the developers have specific backgrounds and skills that have an impact on the development so if you are someone who's been working always on on web application and you start a new you get offered a job in a railway company - you have to develop software for the for the railway company your background will influence the way you program and the role that you will have, and of course the speed that you will have in developing in the specific context so these are complexity dimension that are very personal very subjective but have an impact on the old software development process and the software engineers that is monitoring so how is a process should be considering this disperse another aspects

8.4. Requirement Change since the emergin new Need

Also you really know how the project will go as the context is surely going to change too throughout the project.

Again it is good to consider the coronavirus example the context. Now is we need a system to monitor people at home should we all stay at home. No probably this thing will change in a few weeks so probably new ideas will come to change the software new needs we come to change the software. After this period, so hmm and I don't know what this need will be I gave before the example of the skin problem, skin symptom to be added if another disease come but it could be also that I want to monitor for example the contacts of the people who already had the virus and the people that didn't have the virus and in this case other problems rise for example privacy problem so consider that.

8.5. No Simple and General Solution

All these factors make impossible to invent a simple solution that works for everyone both for testing for and that is good for the testing of all system. Writing requirements of all system or solving the identification of bugs for all system . it is very difficult to find this this type of solution so each particular context each company or each domain has specific solution. And the thing was that rarely in practice software engineering solution came from research.

9. Understand the world/reality with empirical studies

So we understood also, that before changing the world (with) introducing new tools and that may be needed to change the reality, we should first learn about the world. We should first learn what is the reality. So observe before inventing. And this is why empirical software engineering research which is the topic of this course as born. Empirical software engineering research is,let's read the slide, the use of a scientific method to investigate software engineering problems and seeing a scientific method. Because there is no "the" scientific method. There's "only" scientific method and you will see that there are different techniques to face different problems. All of them are flawed and you will learn this different scientific method. The only idea of scientific method intended as a general way of thinking and the general paradigm is that we start from the observation.

1) So we observe the world
2) We formulate hypotheses
3) so we try to think make conjectures try to think what will happen if I do something.

I select certain methodologies who verify that who my Hypothesis is actually true and validate my hypothesis with respect to reality this is the very old scientific method so observe create hypotheses select the methodologies for verification and validate their Portage's with respect to reality this is a four large form of experimentation somehow the idea is that in software engineer if I can understand how thing working practice I can always find way to improve them but if I just have a vague idea what is the problem and I try to find a tool for solving its this is not the right direction so knowledge and understanding is not this is a particular thing that I want to you to pay attention to so while while you may think that things finished they're like I start from observation making hypothesis and then I check the potency sold

maybe the end of the path actually knowing the reality through this scientific process is oriented to solving real world problems. Because at the at the beginning there was always the set of questions from the from the developers from the people from the software engineering community asking to solve real-world problem. So we have to know something to solve some problem. And in our context, also, evaluating whether my solution to the problem observed, I've solved the problem is also in the scope of Empirical software engineering. so in Empirical software engineering is both studying the reality but also transforming the reality and checking that my transformation of the reality, actually solved the existing problem ok ? I will give you an example of cycle in the next slide.

10. Typical EMSE Cycle

10.1. Two Space of EMSE Cycle

So as I said do we have as I said we have two spaces. I didn't say we have two spaces. but somehow I implied that we are considering two spaces

1) Reality Space: One is the reality space so the space of the of the software process, the picture that I showed you before with the customer with the developer etc that is the reality space.
2) Theory space : And then the second is the theory space. The space of abstraction. The space in which I as a software engineering researcher as I try to understand what is the problem and how to solve it.

10.1.1. Five Steps of EMSE Cycle. So the typical software engineering cycle consider consider these five steps.

1) Observe Reality First steps is always observe reality ok ? For example I have, I observed in a company that I have a lot of parts in software and I know that people in my company are working hours in hours. but still, I have lots of bugs and this is a problem for me, okay? why is the problem? because it costs a lot! I can find several motivation for which. This is a problem, but I want to solve it! Because lots of bugs is not good, ok ? then I can start observing that say about this bug.
2) Formulate Problem Theory
3) Evaluate Theory Against Reality
4) Formulate Solution Theory
5) Evaluate Solution Against Theory

I can try to formulate a theory, so I create some hypothesis. I try to abstract from the reality and reason on why these things happen well I can say backs may be produced by too many hours of work because they say the people works a lot of hours so I can think that maybe all this working time does not have a good effect in the in the quality of the code bugs may also be associated to complex code for example this is can be another hypothesis of why things are there is a lot of bugs so my coders are writing very long functions for example very complex function nested and they introduce back without considering that these are problem theories so basically are ways to explain the problem that where do they come from they come from some inference some observation but also from my background that's research ok ? from my also cultural background because maybe i stimmt Elysees if you work too much maybe you do it wrong ok ? because you cannot focus enough so the formulation or problem.

11. Theory vs Reality

Theory is very dependent on the on the person and that's why you always have to evaluate the theory against reality ok ? and you have always to evaluate the theory against the reality.

11.1. Long working hours and bugs

And I want to check whether there is a relation to the working time and i see that actually no relation. The real relation is that most of the bugs are introduced between 8 p.m. and 8 a.m. And if I see that if I check number of bugs in the code with respect to how long is the code and the code complexity, I see that there's no real relationship. So it seems that one hint from reality from the salvation of reality shows me that probably the complexity of code is not related to the number of bugs in my context and I see that actually what is related to the number of bugs is the time when this pre polar programming if people program by night it seems that they introduce back in during the day they don't introduce bug as at this point I have understood in practice what may be the source of my problem so I can pass to formulate a possible theory I want to solve this problem by basically preventing developer for working at night. So I see the bouncer in the night at a night. so let's let's put a system that does not allow them to work them to work in the night. so let us say at the office at night they cannot upload the software from home at night. so they cannot to work between 8 p.m. and 8 a.m. This is a constraint that I want to enforce and I solve the problem with a natural system a system that that blocks basically the system intended both as a software system that doesn't allow them to applaud for example and also a system made of fur rules set of rules that tell them hey guys do not program between 8:00 p.m. and there and they I am you are not allowed to you won't get paid for the time that you spend programming at that time then I have to check if the solution works because again here I am in the theory space I didn't check against reality so I have to need to evaluate the solution of a reality and I say hey if programmed on is these people don't program at night. Actually I see that I am able to reduce the number of bugs. So this is this is a very good very good outcome. But on the other hand I see that I cannot meet the delivery deadline. so this comes from another observation of reality so after I built my solution this is not enough. I have to check that I that no additional problems have been introduced because I have somewhat transformed the reality and I need to observe that my transformation somehow didn't introduce some some noble problem so that my solution it'll create more problem that the one that is soft at this at this point I have a new problem that I cannot meet the delivery deadlines below beside the other one so it seems that development speed may be lower during the day because if this is a possible theory for for my problem so for my new problem because because I have understood I have seen that I forbid the people to before I was meeting the delivery deadline and people were working day and night but during the night they were introducing a lot of bugs ok ? the roundest speed may be lower during the day so because now they cannot they cannot they cannot program at night so let's evaluate this theory against the allottee and i see that actually there is more correct code during the day but slower speed and in the night there are more bugs but faster speed so I get more code more code delivered but more buggy code so how can I solve this this problem I need to dedicate more testing resources for code developed at night so it is not possible to forbid the people to working at night because they seem to be more productive but the solution that I can find is that instead of instead of forbidding people to work at night to have correct code add more testing for coding development and night so I invest some money for the testing resources instead of reducing the effort of the people during the night I check the solution and I see that I have less bugs and I have at the same time also more softer so I can meet the delivery timeline so this is the typical cycle of empirical software engineering so start from reality formulate the problem theory check that my theory actually works against the reality then try so check that my theory is right basically then try to formulate a solution theory and finally try to evaluate the solution against against the reality ok ? so end and see observing the reality again that I didn't interfere with other aspects so the typical software engineering problems after introducing this type of cycle this new way of thinking they do remain the same still how can I find these bugs in the code how can I improve the requirements how can I reduce the resources dedicated to testing and up I improve software development speed and all the other type of problems related to the stakeholders to the to the task of the software process so problem is the same but the typical software engineering solution today is more based on understanding the reality so if someone asked how can I improve my requirements instead of proposing a new language that is instead of being English it's a it's a constrained way of writing in English is a restricted English is a simplified English I first asked let's see how you write your appointments now I asked the company the specific company to see what is the form of the requirements and I start from that reality and I want to see what are the main quality problems of the requirements so I may see that for example actually the requirement language is clear so I don't need I don't need a language for making them less ambiguous because it's very everything is very clear it's just that they are incomplete because you didn't meet probably enough with the customer so I am asking how many meetings do you know I might do with your customer well you may say one meeting just one minute at the beginning and then and then that's it okay let's do another thing instead of introducing any more a new technology let's try to schedule a meeting each week to revise the requirements with the customer so the idea here is that I don't even need to introduce a new software to solve my problem I just need to manage the thing in a different way so let's meet with the customer each week to revise the requirements so to interact with the customer more frequently because he will change his mind and also the reality will change so because dynamism is one of the most typical aspect of the software engineering you have to take into account that you always have to continuously interact with people so the way of approaching problem is different but you have to consider in this example I am always speaking about an interaction with a specific customer so with an I'm proposing a context specific solution that derives from reality and use a scientific method so the realities of course the reality that is observable is the reality of my client or a set of clients set of customers and typical scientifical.

12. Scientifical Method in SE

You should use the scientific method but which method well consider this image software engineering is a is a strange creature because as I was speaking to you I was speaking about mostly stakeholders I didn't mention softer soft why because software engineering as a discipline as a very strong technical face it because it is it involves developers testers so people who know about software development and software is is a very technical thing but on the other hand it has also human and social face it's a human and social profile why because you have different stakeholders coming from different domain but you have also a lot of people involved in the organization running the development of software software is not developed by one person in his own room software for a car or for any any any software that you might use even in your mobile requires the participation of a minimum of two or three people also for the tiniest thing so whenever you you have to consider something bigger you need a structure you need management and this management has to deal with ever-changing requirements and the difference,for example, with the other engineering domain.

13. Software Engineering as a Strange Creature

For example the requirements of house are the same for have been the same for for a long time you may have some specific requirements that that you need to address in a specific context because maybe you want more windows etc but in the end the higher level requirements is always I you need you need to have repair from the from the rain unit you need to be heated and you need to be have enough light you need to sleep there and you need to go to the toilets and you need to cook these are and you need to relax and you need space for all these activities these are the general requirements of a house but for software general requirements always depend may be radically different and you because as I told before you have a spacecraft soft that,for example, as a goal.

The one of Goa reaching the moon or the the diet software in Europe that has as a goal reducing your reducing your your weight so you have a lot of space for for variety let's say and these of course involves humans involves people involves the human factors of the development so we need to take empirical method both from the hard sciences so from physics maths biology let's say but also from social sciences so from those Sciences like political science sociology and this type this type of science that involves the study of the people so on one hand we have to study hard things like for example the code the running time of the code for example but also we have to study the feeling of the people the under the way the people understand things and in this case it is more useful to use methods coming from the social sciences metal from the earth sciences are mostly quantitative so I normally deal with numbers ok ? metals from social sciences are mostly qualitative and you normally deal with words.

You will learn about these different methods both from hard science and from social science. for example here you have a list of methods there is no say no one way of of calling these methods these are just example you have for example experiments with human subjects when we human subjects it's simply an experiment similar to the ones that you may have for medicines ok ? for example if you have to test a medicine. you take let's say 20 people and you give them a certain set a medicine and to the other 20 people you give a placebo ok ? in software engineering the same thing you can do it with a tool for testing for example 210 people.

I give these typical tool for this new tool for testing and the other 10 people they use the old testing tool ok ? and I compare for example how many bugs defined within one hour my experiments okay to understand whether the soft which one of the software is better but I'm testing with human subjects okay I can do an experiment just with softer subjects so for example I have to compare different different instead of wanting to compare the speeds of the people in using a certain testing tool I have some automated tool that does not realize a just push button does not rely on people and I want to compare let's say 10 tools that do automated testing and in this case the experiment is with softer subject thank bisko's and comparing these tools I'm not involving humans because as a Bay as a input I'm using for example a bunch of code that I know for which I know that I have certain bugs. so I check just which one is the tool that is in itself faster and also how many bugs they are able to find in other cases I may need to do a survey a survey is a question of a secure and for example because I want to know which are the most typical bugs in your company maybe I have to deal with a multinational company and they have to understand where to focus my investment for training for example for training the developer to avoid certain specific type of bugs but I don't know which are the typical bugs. so I do a survey with the testing team to understand what are the problem. So a questionnaire, I distribute a questionnaire in some cases maybe is not a very well for whose problem. so I so we're focused like type of bugs style most most typical bugs but it's more it's more related to the relationship let's say that people have I see that people in the team are really in that team are really frustrated they they don't they don't deliver and they seem unhappy and I want to understand what's the reason why what are the triggers for frustration and in this case I perform another type of study it's da koat interview study and then I do some ethnography so I participate to the environment I try to understand and it's different from doing the third way is different from doing an experiment because in dealing with people and I'm dealing with the real world other cases may be judgment study case study face study or literature reviews who you will see this during the lecture I don't go in deep into into each definition of the type of studies but we will see we will see probably at the next lecture an overview of the different types of studies with the with the presentation of the a paper that was published last year at the main software engineering conference at a software sorry it was presented in a journal and then presented also this main software engineering conference.

14. Typical SE Solutions (Today)

15. Empirical Methods in Software Engineering

That is called the ABC of software engineering research that tries to let's say introduce which are the typical software engineering method that put them together according to a sound framework so we will see these different these different methodologies so let's come to a goal of this course and then we make a break so the goal of this course is to learn these methods that are commonly used in empirical software engineering research so you will learn some quantitative methods - like experimentation and some qualitative methods like interviews and ethnography and Aniyah grounded theory and some terminology will be given later on you will learn when it is appropriate to use a certain scientific method so when it is appropriate to do quantitative investigation and when it is more appropriate to look at qualitative data and when you do you may need to mix the methods or together to understand what is the reality and so you will learn also how to combine these different methods. Always keep in mind that all these methods so experiment surveys interviews judgment studies review of literature are all flawed why for example in experiment a real-world environment is always simplified as I have to focus only on a specific set of variables for example in the case of the people using using a certain tool for testing I am course I have to setup the same area have to use the same code and the code cannot be representative of all reality okay the code to debug people may have interacted with similar codes my people may have different experience so I have to check that there is always this there is a balance in the experience of the people but reality is different because in reality I might have people that that have all you know B's for example unbalanced unbalanced the level of experience so is the experimental environment is always a simplified version of reality that focuses on specific variables okay surveys the questioner people tend to say what they think and know what they do and it is it is quite hard to be sure that they have correctly understood the question so understanding if they actually understood what was asked to them and their questions are in the answer are reliable is not easy for interviews and ethnography there is always a researcher bias when if you think about going in a company study and deriving theories from observing you unavoidably put yourself in the middle so experiments may allow you to be objective but if you interview you are part of the you are part of the problem part of the solution you intervene in the reality case studies which are particular type of studies typical of software engineering in which you basically go to a company and try to understand the problem for in the company and possibly find a solution for the specific funny I always had to generalize and to other companies because everyone has his own context and it's always hard to separate the environment from the unit of analysis so if you want to analyze for example the bugs in the code you cannot separate them from the authors of the code from the history of the code so realities into woven and while experience try to create barriers between elements of reality in case studies it is always difficult because you have to deal with with the real field okay so the the issue is never stick to methodological purity so don't think that your results for any method would be the the real results so you have some confusion of your theory but you're never sure okay also and see there is no taxonomy for this method so you may find I will try to give you synonyms for things try to remember the names of experiments are these interviews case studies but in some cases you may have a mix between between names and some synonyms used so I understood that I thought that most of the participants were coming from Statistics from a background in statistics so I I tailored this Court's considering this aspect but I understand that you're coming also from different curricula so basically you will need to know the content of this course because it is a real-world application of statistical competencies and most of all is a way of approaching software and software engineering as scientists and you will learn methods that may be useful for you in research but can also be useful for you in in the reality in the job reality for example if you need to do some market research in companies to understand what is the feeling of people about a certain product that can be a software product or another product and doesn't matter you may need this case that you will learn in this in this course and it can help you also to improve your skills the your your issues your to improve your skills as problem solvers as problem solver that are always needed in any in any company both in software engineering and in other in other contexts so this outline of the course and I'm I will start with an overview of empirical methods so I will give you this overview today then I will pass to describing interviews and ethnography which is a qualitative method is quite lightweight to understand and it's interesting then I will speak about surveys so questionnaire then

16. Goal and Scope of this Course

1) To learn a set of methods commonly used in empirical software engineering research
2) To learn when to use a certain scientific method
3) To learn how to combine different methods

17. All Methods Are Flawed!

1. Experiments: Real-world environment is simplified, as I have to focus only on a specific set of variables (independent, dependent, controlled variables, we will see them later)
2. Surveys :People tell you what they think, not what they do, and it is hard to be sure that they have correctly understood the questions.
3. Interviews and Ethnography : Unavoidable researcher bias, as theories are derived from qualitative data
4. Case Studies (which could include (quasi-)experiments, surveys, interviews, etc.): Hard to generalise and hard to separate environment from unit of analysis, as case studies are real-world experiences (several confounding variables) in one or a few companies (limited scope):

18. Course Outline

19. References

I will spend well to do systematic literature reduce. This is important for any study that you do is very research oriented but it is it is something that you need you need to consider because as I told if you remember when I was speaking about the process the software process added the typical software engineering process I said ok ? now you create some theory okay so my policies that explain a certain situation and it is true that these I policies needs to be derived also not just from your background but from literature so understanding the literature understanding the scientific background on a certain problem is fundamental so you will need to learn for a systematic literature reviews because it's helpful to create possible hypotheses based on existing theory then you will learn qualitative not analysis methods and you will learn quantitative that analysis Metso experiments was experiments and hypothesis testing you will see probably I don't know if there will be time but I would like to teach you about mining software repositories and finally probably I will anticipate case studies before - of the repositories because it's more relevant and for for your context and action research so going to a company and try to a transform the process try to introduce a new way of doing things and how to verify it ok ? this is the reference book and you can you can find it in Amazon you can find it probably some addition and some you can find it also in in Internet if you if you search in form of ebook so this is the reference book okay and then I will start I always start this after after we do little break okay