Session-2 Transcribed

L:Author 1
E: Author 2
U: Author 3

0:00 - 1:35

L: Öncelikle herkese geldiği için çok teşekkür ederiz. Bugün aynı şekilde bu oturumun kaydedilmesine izin verdiğiniz için teşekkürler. Bugün genel olarak proje sırasında ortaya çıkan yazılım eserlerinin (software artifacts) görsel bir şekilde analizinin ve bu bağlamda alınacak kararlarda destek sağlayacak Software Artifact Analyzer (SAA) adlı aracı size tanıtmak ve en önemlisi sizin hem araç hakkında hem de genel olarak düşüncelerinizi öğrenmek için toplandık.

L: First of all, we would like to thank everyone for coming. We also appreciate your permission to record this session today. Today, we are gathered to introduce you to a tool called Software Artifact Analyzer (SAA), which visually analyzes software artifacts that emerge during the project and supports decision-making in this context. Most importantly, we are here to learn your thoughts and feedback about both the tool and the general topic.

1:35 - 2:16

L: İsterseniz biz bilgisayarlarımızdan görünen wifi'ye bağlanabilirsiniz. Bu sırada ben de kısaca bugün bu çalışmada yapacaklarımızı özetlemek istiyorum. İlk olarak bu haftanın başında sizinle paylaşmış olduğum ön ankette verdiğiniz cevaplar üzerinden gideceğiz. Onlar hakkında konuşacağız. Daha sonrasında ben genel olarak bu projede kullandığımız konseptler hakkında kısa bir anlatım yapacağım. Ardından size SAA aracımızın küçük bir demosunu yapacağım ve üzerinden bir anlatım gerçekleştireceğim.

L: If you would like, you can connect to the visible WiFi from your computers. Meanwhile, I would like to briefly summarize what we will be doing in this session today. First, we will go over the responses you provided in the pre-survey that I shared with you at the beginning of this week. We will discuss them. After that, I will give a brief presentation on the concepts we use in this project. Following that, I will give a short demo of our SAA tool and provide an explanation.

2:16 - 2:51

L: Daha sonrasında dediğimiz gibi sizden bu SAA aracını denemenizi isteyeceğiz ve bu bazı yapmanız için size 2 tane küçük görev vereceğiz. En sonunda da aslında hem bu çalışma boyunca ki deneyiminizi hem de genel düşünce ve görüşlerinizi paylaşmanız açısından sizi bazı sorular yönlendireceğiz ve bu sorular bağlamında da tartışmanızı isteyeceğiz. Bu şekilde şimdi

o zaman isterseniz ön anketle devam edelim. Bu arada ana amacımız sizin düşünce ve görüşlerinizi öğrenmek. O yüzden istediğiniz bir noktada istediğiniz gibi katkı verebilirsiniz.

L: Afterwards, as we mentioned, we will ask you to try out the SAA tool, and we will give you two small tasks to complete. At the end, we will ask you some questions to share your overall experience during this study as well as your general thoughts and opinions. We will then ask you to discuss these questions. Now, if you are ready, let's continue with the pre-survey. Meanwhile, our main goal is to learn your thoughts and opinions. Therefore, you can contribute at any point and in any way you like.

2:43 - 3:41

L: Şimdi ilk tur olacağı için ilk başta kendinizi tanıtıp başlarsanız diğer katılımcıların da sizi tanıması ve herkesin birbirini daha iyi tanıması açısından.

L: Since this is the first round, please start by introducing yourself so that other participants can get to know you and everyone can get to know each other better.

L: Pre-Survey_1-2. Soru: İlk 2 sorumuzda aslında hangi version control sistemi ve bug tracking sistemi kullandığınızı sormuştuk. Zaten katılımcıyla da çoğu Git tabanlı bir araç kullanıyor. Bunun dışında bu session'da olan herkes Jira kullanıyor diye anladım. Bu arada şu anda 10 katılımcıya ait veri var. Çünkü bir de geçen hafta bir session yapmıştık. Bu sonuçlar hakkında yorum yapmak isteyen olur mu?

L: Pre-Survey_1-2. Question: In the first two questions, we asked which version control system and bug tracking system you use. Most participants already use a Git-based tool. Besides that, I understood that everyone in this session uses Jira. By the way, we currently have data from 10 participants. Because we also had a session last week. Would anyone like to comment on these results?

K1: Standart yani, Github, Jira.

Participant1: It's standard, Github, Jira.

K2: Community zaten destekliyor.

Participant2: The community already supports it.

K1: Daha önemlisi bu iki ürünü kullandığınızda diğer ürünleri de çok kolay kullanıyorsunuz. Mesela herhangi bir analitik platformunu kullanmak isterseniz hepsinin bu ikisine de desteği var. Yani başka bir araca ihtiyacınız olmaz.

K1: More importantly, when you use these two products, you can easily use other tools as well. For example, if you want to use any analytical platform, they all support these two. So, you don't need another tool.

K2: Bir de API'leri var çok güzel. Issueları indiriyorsunuz excelden düzenleyebiliyorsunuz. Bir de CI/CD çok iyi, özellikle Gitlab'ın.

Participant2: And they have great APIs. You can download issues and organize them in Excel. Also, the CI/CD is very good, especially Gitlab's.

L: Pre-Survey_3. Soru: Tamamdır, teşekkürler. Daha sonrasında da aslında size herhangi bir projenizle görsel tabanda bir Software analiz aracı kullanıp kullanmadığınızı sormuştum. Bu oturumda da sanırım 3 kişi kullandıklarını söylemiş. Onlar hangi araçları kullandı ne amaçla kullandı açıklamak ister mi? Bu arada ilk önce kendimizi tanıtırsanız, seviniriz konuşmaya başlarken.

L: Pre-Survey_3. Question: Okay, thank you. After that, I actually asked if you have used a visual-based software analysis tool in any of your projects. I believe three people in this session said they have. Would they like to explain which tools they used and for what purposes? By the way, please introduce yourselves before you start speaking.

K3: Ben hayır dedim.

Participant3: I said no.

K4: Aslında şöyle bilinen bir araç olmadığı için ben hayır dedim fakat. Bu arada ben Havelsan'da çalışıyorum adım (Katilimci adı). Biz Azure DevOps kullanıyorduk o projede. Havelsan'da hem Azure DevOps var hem Jira var. Projelere göre değişiklik gösterebiliyor. Azure DevOps kullanırken orada Havelsan'da iç geliştirmesiyle yapmış olduğu bu bugla relate olabilir mi veya bu test case değiştirirsen belki şu requirementlara da bir bakman lazım gibi analiz bir şüpheye düşmüş durum var onları da bir doğrulaman gerekiyor gibi yazılmıştı. Kendi yazdığı tool Havelsan'ın böyle bilindik veya varsa sektörde ben de bilmiyorum ama hem özel bir visual eden bir tool kullanmadık.

Participant4: Actually, I said no because there wasn't a well-known tool. By the way, I work at Havelsan, my name is (Participant's name). We were using Azure DevOps in that project. At Havelsan, we have both Azure DevOps and Jira. It can vary by project. When using Azure DevOps, there was a situation where it was written that we had to verify whether the bug was related or if changing the test case would require looking at other requirements as well. Havelsan's self-developed tool isn't well-known or at least I'm not aware of any in the industry, so we didn't use a specialized visual tool.

L: Zaten sorumuz genel olarak kullanıp kullanmadığınızdı o yüzden aslında intuitional bir cevap bekliyorduk.

L: Our question was general about whether you used it or not, so we were expecting an intuitional answer.

K2: Visualization pek değil de SonarQube kullanıyorduk biz de. Kod analizi için. Onun dışında bu Jiralarla yapılan kodda geliştirmeleri takip etmek çok kolay değil. Çünkü kodun ilerisine ne denk gelecek bilemeyebilirsiniz. Visual bir şeyi yok. Kod analizi için.

Participant2: Not much visualization, but we were using SonarQube for code analysis. Apart from that, tracking code developments with Jira is not very easy because you can't know what will come up in the code. There's no visual aspect. Just for code analysis.

K5: Ben (Katılımcı adı). Ben bunu hayır demiştim. O yüzden ekleyeceğim bir şey yok.

Participant5: I'm (Participant's name). I said no to this. So, I have nothing to add.

L: Tamam. L: Okay.

K1: Ismim (Katılımcı adı), ben Mobile Action'dayım. Biz aslında şöyle. Farklı araçları farklı amaçlar için kullandık. Veya yani SonarQube kullandık biz de. SonarQube yani statik kod analizi için kullanıyoruz. Aynı şekilde şöyle biz daha çok engineering productivity ölçmek için mühendislerin çalışma metriklerini ölçebilmek amacıyla yaklaşık 7-8 araçtan demo aldık. Bir tanesi ile çalışmaya başladık, sonra onu bıraktık, ikincisine geçtik derken böyle bir tecrübemiz var. Neyi kullanıyoruz derseniz Swarmy adlı bir tool kullanıyoruz. Swarmy ne yapıyor? Jira ile ilgili data inceleyip bize çeşitli dashboardlarda grafikler sunuyor. İşte doğrama metrikleri denen kendi CI/CD toolarının metriklerini de çıkartıyor. Genel bir organizasyon sağlıyor.

Participant1: My name is (Participant's name), I work at Mobile Action. Actually, we used different tools for different purposes. We also used SonarQube, for static code analysis. Similarly, we tried around 7-8 tools for measuring engineering productivity and working metrics of engineers. We started with one, then moved to another. This is our experience. If you ask what we use now, we use a tool called Swarmy. What does Swarmy do? It analyzes data related to Jira and presents various graphics on dashboards. It also generates metrics from its own CI/CD tools, providing a general organization.

L: Ne gibi sonuçlar için bakıyorsunuz?

L: What kind of results are you looking for?

K1: En doğrama metriklerinde en yaygın olan zaman yönetimi. Daha önce kullandığımız araç biraz daha iyiydi. Orada biraz daha mesela kod review sürecinin doğru yönetip yönetilmediğini de takip edebiliyorduk. O araç daha iyiydi. Ne yazık ki biz aldık sonra 3 ay sonra kapatmaya karar verdiler. O dahil ama işte. Yani. En azından günlük aktivitelerin nereden kaldı, nereye gittiğim işte sizin asıl sorularınızda yer alan bazı grafikleri görüyoruz. Hani gerekirse tool üzerinde de gösterebilirim. Hani şu anda daha iyisini gördükten sonra biraz eksik kaldı.

Participant1: The most common metric isPsrt. The tool we used before was better. We could also track whether the code review process was managed correctly. Unfortunately, we started using it, and three months later, they decided to shut it down. But, we can at least see some

graphs related to your main questions, like where daily activities left off and where they went. If needed, I can show it on the tool. After seeing a better one, it feels lacking now.

K3: (Katılımcı Adı), ben. Carbon Health'ten geliyorum. Biz kullanmıyoruz ama SonarQube kullanıyoruz da onu çok visual olarak düşünmemiştim. Yani daha böyle kod yönlendiriyor. Bir de şey var. Github'da ki bütün mentionları logluyoruz biz. Orda böyle çeşitli grafiklerle kim ne yapmış görebiliyoruz.

Participant3: (Participant's name), from Carbon Health. We don't use it, but we use SonarQube, though I didn't think of it as very visual. It guides the code more. Also, we log all mentions on Github. We can see who did what through various graphs there.

L: Pre-Survey_4. Soru: Daha sonrasında aslında yine benzer bir soru sormuştuk. Görselleştirmenin proje sürecindeki insightları yani anlamamıza katkısını sormuştuk. Genel olarak herkes zaten pozitif olarak değerlendirmiş. Bu soru üzerinden yorum yapmak isteyen olur mu? Bu da mesela 3 diye cevap veren katılımcılardan biri cevap vermek ister mi?

L: Pre-Survey_4. Question: After that, we asked a similar question about the contribution of visualization to understanding insights in the project process.

Katilimci 2: Visualization dediğiniz zaman o çok geniş bir şey aslında. Neyin görüntülenmesi yani? Çok geniş bir şey. Sonuçta Kubernetes kodlarının görsel olarak görüntülenmesi de bir visualization'dır. İnsanların Jira'daki hareketlerinin ne kadar çözdüklerinin görüntülenmesi de bir görüntülemedir. Ne açıdan?

Participant 2: When you talk about visualization, it's actually a very broad thing. What is being visualized? It's a broad concept. For example, visualizing Kubernetes code is also a form of visualization. Visualizing how people are resolving issues in Jira is another form. From what perspective?

L: Şimdi mesela hani text-based analiz olabilir veya herhangi bir görsel elementi kullanarak olabilir. L: For instance, it could be text-based analysis or using any visual element.

Katilimci 2: Yani görsel daha iyi tabi.

Participant 2: So, visual is better, of course.

Katilimci 1: Tabi o kesin canım, hani kör yürümektense görselle desteklenmesi tabii ki daha iyi. Yani metrikler tabii ki her yerde var, lograin içinde de var, orada da var, burada da var ama bir bakışta görebilmek en azından bir sıkıntı olduğunda, "A burada bir sıkıntı var" diyebilmek çok önemli.

Participant 1: Absolutely, it's better to have visual support than walking blind. Metrics are everywhere, in logs and other places, but being able to see them at a glance, to say "Ah, there's a problem here," is very important.

Katilimci 3: Ben buna 3 demiş olabilirim. Şimdi görselleştirme önemli ama belli bir seviyede olsun. Hep de görsel görsel, text-based bilgi de önemli yani. Yani onun ikisinin böyle birleşik versiyonu bence daha makul bir deneyim sağlayabilir.

Participant 3: I might have rated this a 3. Visualization is important, but to a certain extent. It's not all about visuals; text-based information is also important. A combined version of both might provide a more reasonable experience.

L: Teşekkürler, başka bir katkıda bulunmak isteyen olur mu acaba?

L: Thank you. Would anyone else like to contribute?

Katilimci 4: Görsellikte sanki paterni bir bakışta yakalayabilmek adına fayda sağlayabilir diye düşünmüştüm ben burada. İşte örnek veriyorum. Hani hep aynı kişiye açılan buglar, aracımızdan hareketle söylüyorum ya da işte aynı source kodları sürekli işaret eden bir bug durumunu texte görmekten ziyade görselliği çok hızlı bir şekilde. Ha demek ki bu source code, bu dosyada bir sıkıntı var ya da bu şeyde bir problem var. Daha böyle bir bakışta o paterni yakalamak için daha faydalı. Ama tabii ki detaylara girmesi lazım. Yani detay için belki bir şey söyleyemedim ama bir ilk bakışta doğru patern yakalatma konusunda fayda olacağını düşünüyorum.

Participant 4: I thought visualizing could be useful to quickly catch patterns at a glance. For example, bugs always assigned to the same person or a bug constantly pointing to the same source code. It's much faster to see this visually rather than in text. So, it indicates there might be an issue with this source code or this file. It helps to catch the pattern at a glance. But of course, it needs to go into details. I might not be able to comment on the details, but I think it helps to catch the correct pattern at first glance.

Katilimci 1: Hani bizim denemesini yaptığımız toollardan biri Linear'di zannedersem. Rework metric diye bir şey çıkartmışlar. Yani toplam kod base atıyorum. 1.000.000 satırda işte 10.000 satırlık kod kısmı sürekli olarak yeniden PR açılıyor, yeniden yapılıyor. Bu rework metric olarak geçiyor. Yani burada bir problem var ki sen buraya sürekli dönüyorsun diye ölçüyor. Yani yeni build etmek mesela koda satır eklemek şeklinde olur ama aynı koda sürekli editlemek, orada bir problem var ki sen buraya dönüyorsun şeklinde ölçüyor. O da yüzde olarak da veriyordu, çok güzeldi.

Participant 1: One of the tools we tried, I think it was Linear. They came up with a rework metric. Out of a total code base of 1,000,000 lines, if 10,000 lines of code are constantly being re-opened for PR and reworked, it's measured as a rework metric. It indicates there's a problem in that area. Instead of adding new lines to the code, constantly editing the same code suggests there's an issue. It also provided percentages, which was very nice.

L: Pre-Survey_5. Soru: Diğer sorumuzda da aslında proje süresince en fazla etkileşime geçtiğiniz software artifact'ları sormuştuk. Burada da gördüğüm kadarıyla en fazla kod dosyaları, issue'lar, commit'ler ve onu takip eden de pull request'ler seçilmiş. Bu sonuçlar üzerine konuşmak isteyen olur mu, sizce neden bu artifact'lar daha önde çıkmış?

L: Pre-Survey_5. Question: In our other question, we asked which software artifacts you interacted with the most during the project. From what I see, code files, issues, commits, and followed by pull requests were chosen the most. Would anyone like to discuss these results? Why do you think these artifacts stand out more?

Katilimci 2: Yani sonuç olarak source code her zaman değişen bir şey. Yani onun için örneğin diyagramı ilk başta çiziyorsunuz, o orada kalıyor. Yüzde doksanda o ilk haliyle kalıyor, sonra gidip güncelleyen yoktur.

Participant 2: In the end, source code is always changing. For example, you draw the diagram initially, and it stays there. 90% of the time, it remains in its initial state, and no one updates it later.

Katilimci 3: Ben az çıkan hakkında konuşabilirim, şu mode UML. Bunlar genelde böyle daha oturmuş kurumsal mı denir? Belki öyle firmalarda oluyorlar. Ama start-up gibi ortamlarda çünkü firmalarda daha böyle kâr etmek gerekiyor. Mesela ben hiç kullanmadım yani database'e göre saymıyorsak son 3-4 yılda bir defa bile kullanmadım.

Participant 3: I can talk about the less common ones, like UML. These are generally found in more established, corporate settings, I guess? They might exist in such companies. But in start-up environments, because companies need to be more profitable. For example, I haven't used it at all, unless we're counting databases, in the last 3-4 years, not even once.

Katilimci 1: Şöyle olan değil olmaya bakmak lazım.

Participant 1: It's about looking at what should be rather than what is.

Katilimci 2: Ama şöyle, o UML şeyde gerekiyor. Hani büyük projelerde UML çizmeniz lazım ki mikro servisleri, onları, buraları çıkarabilirsiniz, o yoksa çok sorunsuz çıkmıyor.

Participant 2: But, UML is needed in big projects. You need to draw UML diagrams to define microservices and other parts. Without it, it doesn't come out very smoothly.

Katilimci 1: Yani bizde yirmiden fazla mikro servis var. Hiç UML çizmedik.

Participant 1: We have more than twenty microservices. We've never drawn UML diagrams.

Katilimci 1: Ama şey, hiç bilmediğiniz bir domaini analiz ederken kimin ne yaptığını görmek açısından gerekiyor.

Participant 1: But, when analyzing a domain you know nothing about, it's necessary to see what everyone is doing.

Katilimci 3: Şöyle, illa bir kural setine ihtiyaç olmuyor. Kendin bir flow diagram kendin çizebiliyorsun.

Participant 3: You don't necessarily need a set of rules. You can draw a flow diagram yourself.

Katilimci 1: Çizilse ve yazılmıyor.

Participant 1: It might be drawn but not written.

Katilimci 3: Yani UML uyumak gibi bir derdimiz yok. Participant 3: We don't have the issue of adhering to UML.

Katilimci 1: Bir toplantıda karar alınıyor ve geçiyor. Yani o döküman bir daha da kaybediliyor gidiyor yani.

Participant 1: A decision is made in a meeting and that's it. That document is lost and gone forever.

Katilimci 2: Hazır diyorsun, onayını alıyorsun, kurduğundan sonra gideriz. Daha baktığımızda zaten yani 10 katılımcının. Bizim mesela 100 kere sıkın Instagram çizdim dediğiniz gibi bir incelendikten sonra bir daha atlattanmıyor, yani şey de öyle oldu zaten. Bir süre sonra hele projede son teslim zamanında dönüp onu kim güncelleyecek? Zaten müşteri de dönüp dönüp acayip şeyler söylemeye başlayıp başta dediğiniz tam tersi demeye başlıyor.

Katilimci 2: You say it's ready, get approval, and move on. Looking back, we have 10 participants. For example, I've drawn Instagram 100 times, and after a review, it's never touched again. The same happened in the project. Near the deadline, who will update it? The customer starts saying strange things, contradicting what was said initially.

Katilimci 5: Bir kez çiziliyor, sonra da ellenmiyor yani.

Participant 5: It's drawn once and never touched again.

L: Pre-Survey_6. Soru: Daha sonra, hiç projelerinizde expert developer bulunması için harici bir araç kullanıp kullanmadığınızı sormuştuk. 2 katılımcı kullandıklarını söylemiş. Nasıl bir araç kullanıyorlar? Ondan bahsedebilir mi?

L: Pre-Survey_6. Question: We then asked if you have ever used an external tool to find expert developers in your projects. Two participants said they have. Can they talk about what kind of tool they use?

Katilimci 2: Yani genelde bunlar böyle çıkmıyor artık. Yani bizden expert olanı yazdığı koddan görüyorsunuz. Evet çözdüğü hatta bir de dönülüyorsun, iş veriyorsun, çözüyorsun bu evet bu iyi ve böyle biraz daha iş veririm diyorsun öyle gidiyor.

Participant 2: These things usually don't come up. You see who is an expert from the code they write. You give them a task, they solve it, and you think, "Yes, this is good," and you give them more tasks.

Katilimci 3: Bir takım oturmuşsa, bir yerden sonra herkes birbirini tanıyor ya. Oradan biraz öyle. Yani araç yok ama herkesin kafasında, bizde de aynı şekilde.

Participant 3: If a team is established, everyone knows each other after a while. It's like that. We don't have a tool, but everyone has an idea in their heads, and it's the same with us.

Katilimci 4: Bizde de aynı şekilde. Yani işte şu servisle ilgili hata varsa onu şu kişi bilir daha iyi bilir aynen aynen ben öyle bir şey var. Kafada bir mapping ama bir araçla ortaya çıkartmıyoruz.

Participant 4: It's the same with us. If there's an issue with a service, everyone knows who understands it better. There's a mental mapping, but we don't use a tool to identify it.

Katilimci 1: Git blame de sonuç veriyor. Tool olarak en fazla onu söyleyebilirim.

Participant 1: Git blame works for that. It's the most I can say as a tool.

Katilimci 5: Fakat Git blame bazen yanıltıcı sonuç da verebiliyor. Bir issue yanlış kişiye assign oluyor.

Participant 5: But Git blame can sometimes give misleading results. An issue can be assigned to the wrong person.

L: Pre-Survey_7. Soru: Son sorumuzda da aslında process smelleri ile ne kadar tanıdık olduğunuzu, ne kadar bildiğinizi sormuştuk. Katılımcıların çoğu zaten bu konu hakkında bilgisi varmış aynı zamanda. Çalışma ortamınızda daha önce hiç process anomalilerini tespit eden bir uygulama kullanmış mıydınız diye sormuştuk. Fakat sanırım kullanan olmamış. Böyle bir araç kullanan var mı yani?

L: Pre-Survey_7. Question: In our last question, we asked how familiar you are with process smells and how much you know about them. Most participants were already knowledgeable about this topic. We also asked if you have ever used an application to detect process anomalies in your work environment. I guess no one has used it. Is there anyone who has used such a tool?

Katilimci 2: Yani sonrasında bizde yok ama şöyle bir şey var, genelde bir olay falan olduğu zaman sonrasında toplantı yapıyoruz. Orada konuşarak process smelleri ortaya çıkarıyoruz.

Participant 2: We don't have such a tool, but after an incident, we usually hold a meeting. We discuss and identify process smells there.

Katilimci 3: Düzeltmek adına bazen şey oluyor, code review yaparken de bazen ortaya çıkıyor. SonarQube de çıkarmıyor bazen ama dışarıdan baktığınızda zaman patlayacak kısmı kendin ortaya çıkarıyorsun.

Participant 3: Sometimes it comes up during code reviews for correction. SonarQube doesn't always detect it, but you can see the part that's going to fail when you look from the outside.

Katilimci 4: Bir de process smell ve code smell farklı şeyler sonuçta. SonarQube code smell'leri ortaya çıkarmak üzerine. Genelde şirketin ya da takımların işleyişi ile ilgili sorunlar olabiliyor.

Participant 4: Also, process smells and code smells are different things. SonarQube is for identifying code smells. Usually, issues are related to the company's or team's workflows.

L: Aynen. Bad practices diyebiliriz.

L: Exactly. We can call them bad practices.

Katilimci 2: Ya genelde şey olmuş oluyor, exceptionlar handle edilmemiş oluyor. %90'ında geliyor, patlıyor.

Participant 2: Usually, exceptions are not handled. In 90% of cases, they come up and cause problems.

L: Onlar biraz daha code smell'e giriyor. Process smell biraz daha süreç içinde yapılan uyumsuzluklar. Yani süreç içerisinde, diyelim ki yanlış hareket mesela atıyorum işte kendi kendine bir pull requesti review etmiş kendi kodunu gibi.

L: Those fall more under code smells. Process smells are more about inconsistencies within the process. For example, someone reviewing their own pull request is a process smell.

Katilimci 1: O tarz şeyleri genelde blokluyoruz zaten.

Participant 1: We usually block things like that.

Katilimci 4: Bazı metriklerimiz var. 1.000 satırlık kodun ortalama review süresinin dokümanları varsa, mesela o da 1.000 satır ama 2 buçuk saatte onaylanmış. Neden? Ya da mesela 1.000 satır 2 dakikada onay vermiş. Bunların neresine baktığınızda 1.000 satırı 2 dakikada okumaya yetmez. Ama bunu örnekleme yoluyla yapıyoruz. Bize dokümanlar çıkıyor. Gidip o PR'ı gözden geçirmiş kişilere bak bu PR çok çabuk kapatılmamış mı diye soruyoruz. O da diyor ki o konfigürasyon değişikliğiydi zaten biliyorduk işte vesaire gibi bir yorumda bulunuyor. O şekilde ama bu bir süreç aslında. Bir araç bazında yapmıyoruz, bir süreç olarak yakalanıyorlar. O metrikler sprint boyunca toplanıyor ve aykırı olanlar inceleniyor.

Participant 4: We have some metrics. If a 1,000-line code review document is approved in 2.5 hours, why? Or if a 1,000-line code is approved in 2 minutes, you can't read 1,000 lines in 2 minutes. We do this by sampling. We get documents and ask the reviewers if the PR was

closed too quickly. They might say it was a configuration change, so they knew about it. It's a process we follow, not tool-based. These metrics are collected during the sprint, and anomalies are investigated.

U: Siz süreyi önemsiyorsunuz orada.

U: You are emphasizing the time there.

Katilimci 4: Süre var, kod sayısı var. Repo'ya göre şu kişilerin bakması gerekiyor dediğimiz metrikler var. Bunlar code review süreci ile ilgili olanlar. Bug tracking süreci ile ilgili kurumsal süreçlerimiz var. Belki o da buraya gidiyor mudur? Çok emin değilim. İşte bug açılır açılmaz birine assign olmuyor. O zaten önce ilgili modülün teknik yöneticisine gidiyor, o bir analiz ettikten sonra bazılarını yönlendiriyor, bazılarını boarda yolluyor, vesaire vesaire gibi. O zaten kurumsal bir süreç.

Participant 4: There are metrics like time, code lines, and certain people who need to review according to the repo. These are related to the code review process. We also have corporate processes related to the bug tracking process. Maybe that applies here too? I'm not sure. A bug is not assigned immediately to someone. It first goes to the technical manager of the relevant module, who analyzes it and then directs some to others, sends some to the board, etc. It's already a corporate process.

Katilimci 2: Bu bizde demek ki herkes istediğini yapmıyor.

Participant 2: This means not everyone does whatever they want with us.

L: Tamamdır, son sorumuzdu. O zaman sunum ile devam edelim.

L: Alright, that was our last question. Let's continue with the presentation then.

19:20 - 30:06 (Presentation Session About the Concepts)

30:06 - 52:56 (SAA DEMO) 30:06 - 52:56 (SAA DEMO)

Katilimci 1: Bana biraz şey gibi geldi, bu process anomalileri daha çok open source bir data üzerine geliştirildiği için bize çok map etmiyor. Bizde mesela bug'ın unutulması gibi bir case yok yani. Ya yapmıyoruzdur, ya da yaparız. Bu daha çok community içinde oluşabilecek bir durum gibi.

Participant 1: It seems to me that these process anomalies are developed more on open-source data, so they don't map well to us. For example, we don't have a case where a bug is forgotten. Either we don't do it or we do it. This seems like a situation that could occur more within a community.

Katilimci 2: Bir de normal bir projede bu kadar fazla artifact olmayabilir. Core artifact'larda çok fazla değişiklik olur, herkes oraya değiştirilir. O orada patlayabilir.

Participant 2: Also, in a normal project, there might not be so many artifacts. There are a lot of changes in the core artifacts, and everyone changes them there. That can explode there.

U: Yani artifact sayısı, PR sayısı çok mu azdır diyorsunuz?

U: So you are saying the number of artifacts, PRs, is very low?

Katilimci 2: Jar sayısı. Benim artifact olarak düşündüğüm jar.

Participant 2: The number of jars. What I consider an artifact is a jar.

U: Biz burada ama genel olarak kod dosyalarını, PR hepsini artifact olarak kabul ediyoruz.

U: But here, we generally consider code files and PRs as artifacts.

Katilimci 2: Biz jar'ları push ediyoruz. Normal dönemde file bazında gitmiyorsunuz. Mesela çok fazla file olur. Paketliyorsunuz ve onları gönderiyorsunuz.

Participant 2: We push jars. Normally, you don't go file by file. For example, there are many files. You package them and send them.

U: Burada biz bazı assumpsiyonlar yapıyoruz. Development team'in, size'ının düzineler mertebesinde olduğunu farz ediyoruz. Open bir space çalışıyorsunuz, sesini yükseltsen bile duyarlar.

U: Here, we make some assumptions. We assume that the size of the development team is in the dozens. You work in an open space, and even if you raise your voice, they will hear you.

Katilimci 1: Mesela open community daha önemli. Turnover'lar da oluyor.

Participant 1: For example, the open community is more important. There are turnovers as well.

Katilimci 4: Götürsem yarın bizim şirkette birinci gün aktif kullanıma başlanır. Yani bizim ürün için söylüyorum. Biz bir ürün yazdığımız için bir de 70 kişilik bir ekip olduğumuz için şeyde güzel oluyor. Çünkü domain'i ayrılmış takımlar var. Bir domain diğer domain'e bir entegrasyon yazacağı zaman gerçekten kendi takımını iyi biliyor ama diğer takımda o entegrasyonun ucuna kime sorması gerektiğiyle ilgili eksik bilgisi olabiliyor. Tabi ki hepsi bunun iletişime çözülebilecek şeyler ama biraz böyle bir remote çalışma falan da çok gün. Olduğu için evinde otururken şimdi yandakine kalkıp sormak var. Oradan metrim mosta yazacak herkese de sen de bakıyorsun sen de bakıyorsun. Ben götürsem bunu şey yaparım. Parali satın aldirirm.

Participant 4: If I take this to our company, it would be actively used from the first day. For our product, I'm saying. We are writing a product, and since we have a team of 70 people, it works well. Because there are teams separated by domain. When one domain writes an integration for

another, they know their own team well but may lack information on who to ask in the other team. Of course, all of this can be solved through communication, but with so much remote work nowadays, you have to get up and ask the person next to you. Instead, you write it in the metrics and everyone sees it. I would make them buy this tool.

U: Bir de zaten şeyde geçecek ama sunum geri kalanında büyük resim şu. Yani bu bir platform aslında bir framework. Bu framework'ü herkesin ihtiyaçları doğrultusunda biz geliştirebiliriz yani. Küçük şirketlerin işine yarayacak şeyler ya da işte farklı ihtiyaçları olan ki mesela siz süre dediniz ya yani süre ve kod sayısı şey büyük değil dediniz ya mesela o bizim hiç düşünmediğimiz bir şeydi, o da entegre edilebilir.

U: And it will pass, but the big picture for the rest of the presentation is this. This is a platform, actually a framework. We can develop this framework according to everyone's needs. Things that small companies will benefit from or different needs, like you said about time and code lines, which we hadn't thought about, can be integrated.

Katilimci 4: Kurumsallarda belki katılır arkadaşlar. Kurumsallarda şöyle bir durum oluyor, gerçekten birinin performansını ölçmek çok zor bir kurumsal şirketteyseniz startuplarda vesaire ya da girişimcinin sermayesini kendi verdiği yerlerde o kişiyi zaten çok yakın takip ediyor. Ya da zaten kişi sayısı az oluyor, hızlıca da görünüyor. Böyle bir sosyal ahlaklık mı deniyordu? bir yasa vardı. Hani 50 kişilik ekipte yatan 5 tane adam var, onu bulmanızı ve ispatlamanız çok zor ve iş kanunu gereğince de fiyatlarımızda bir yaptırımınız yok. Böyle bir tool mu olsa sen yazdığın kaynak koda sürekli bug geliyor. Analiz aracı da böyle böyle söylüyor. Bununla ilgili seninle bir one to one görüşme yapalım deyip 3-5 one to one sonra da artık seninle yolumuzu ayırmak zorundayız. Çünkü bu araca göre de çok da iyi bir performans yok diyebiliyor oluyoruz. Biz o metriklere de o yüzden aslında çok önemsiyoruz. Hani 1000 satırlık oda 2 dakikada bakmış olamazsın baktıysanda bir açıklaması olmalı ya da bakmış olmak için baktığın anlamına gelir gibi bir şeyler çıkarmaya çalışıyoruz. O yüzden kurum yani bizim gibi kurumsallar belki ki daha iyi olabilir.

Participant 4: Maybe others will agree about corporates. In corporates, it's really hard to measure someone's performance. In startups or places where the entrepreneur provides the capital, they already closely monitor the person. Or there are fewer people, so it's quickly visible. There was a law called social accountability or something like that. In a team of 50, there are 5 people who don't work, and it's very hard to find and prove that, and there's no penalty according to labor laws. If we had a tool that says, "You keep getting bugs in your code, the analysis tool says so," we could have a one-to-one meeting and after 3-5 meetings, say, "We have to part ways because this tool shows your performance isn't good." We actually care a lot about these metrics for that reason. If you look at 1000 lines of code in 2 minutes, there must be an explanation, or it means you just looked at it superficially. So, for companies like ours, corporates might benefit more.

Katilimci 5: Ben şu kullanımda katılıyorum yani. Takımlar yani developer açısından işte 6 kişi, 7 kişi, 8 kişi var. Takım içi ben bilirim, kim ne yazmış. Bazen diğer takımın ellediği noktaya

gitmemiz gerekiyor. Ben bilmiyorum diğer takımda kim ne yazmış falan bu toola bakıp kim neyi biliyormuş anlayabilirim. O açıdan bayağı avantajlı geldi bana.

Participant 5: I agree with this usage. In teams, from a developer's perspective, there are 6, 7, 8 people. Within the team, I know who wrote what. Sometimes we need to go to a point touched by another team. I don't know who wrote what in the other team, but by looking at this tool, I can understand who knows what. That seems quite advantageous to me.

U: Bir de biraz vurgulamaya çalıştık. Tabi böyle paperların şeylerini falan koyarak ama şimdi. Tipik olarak işte git blame baktığınız zaman sadece işte number of line oluyor vesaire ama burada biraz daha şeyli computational bir şeyle işte graf algoritmalarıyla onu çıkar. Böyle bir de recency de şeye katıyor. Yani 3 sene önce bir sürü commit yapan bir adam standart zaten işte bir sürü satır getirmetot yazmıştır orada duruyordur ama asıl önemli metotları değiştiren daha resim şeyleri ekleyen kişi farklı olabilir. ve yeni yöntemler ortaya çıktıkça bu.

U: We tried to emphasize a bit. Of course, by including things from papers and such. Typically, when you look at git blame, it's just the number of lines, etc., but here, it's a bit more computational, using graph algorithms. It also includes recency. Someone who made a lot of commits three years ago might have written many methods that are still there, but the person who added the important methods recently might be different. And as new methods emerge, this

Katilimci 2: Mesela şey zaman, hani şunu zorluyor, bir issue vardır, çok kolay çizersin. Bir issue vardır, bir ay boyunca bütün modüllere bir şey eklemeniz gerekiyor. Küçük küçük bir sürü bir şey eklersiniz. 1 ayda da hiçbir şey yapmıyor gözükürsünüz, bir işi yapıyor gözükürsünüz ama zor bir şeydir aslında.

Participant 2: For example, time pressures, like having an issue that's easy to fix. Another issue might require adding something to all modules over a month. You add many small things, and it looks like you haven't done anything in a month, but it's actually a difficult task.

52:56 - 01:12:10 (SAA Framework Presentstion)

1:23:00 - 2:15:00 (User Trial Session)

Katilimci 2: Commit'lerin kritik derecesi de özelliklere eklenebilir. Ivır zıvır bir şey mi acaba, critical blocker bir issue mu çözmüş yoksa normal bir sorun mu çözmüş.

Participant 2: The criticality of commits can also be added to the features. Is it something trivial, or has it solved a critical blocker issue, or just a normal problem?

Katilimci 1: Bir de şey eklenebilir. Belirli bir kod dosyası setini kendi seçip onlar bazında bakmak. Birden fazla dosyaya bakmak gerekir. Bazı dosyalar birlikte değişen dosyalar oluyor. Onlar hakkında toplam bilgiyi bulmak açısından.

Participant 1: Another feature could be selecting a specific set of code files and analyzing them together. Sometimes multiple files need to be examined together. Finding collective information about files that change together would be useful.

Katilimci 1: Şeyi tavsiye edebilirim. Issue bazlı değil de file gruplarını, yani aynı PR içinde birlikte değiştirilen file'ları bir cluster edip. Aynı PR'da beş file değişir, diğer PR'da dört file değişir ama onların bir cluster olduğunu anlıyorsunuz. Onlara odaklanmak daha mantıklı olur. Çünkü file biraz yanıltıcı olabilir.

Participant 1: I can suggest focusing on file groups instead of individual issues, clustering files that change together in the same PR. Five files might change in one PR, and four in another, but understanding they form a cluster is more practical. Focusing on these clusters makes more sense because a single file can be misleading.

Katilimci 2: Ya genelde şey oluyor çünkü dediğim gibi onları class class değil de gruplayıp gidiyorlar. Bir issue'ya bağlı oluyorlar. Ben commit'lere issue numarasını bütün file'lara yazdırıyorum. Yani dört tane file değişiyorsa aynı commit'te, issue numarası hepsinde bulunuyor. Böylece ilgili dosyalar oradan gruplanıp, "Ha bu adam bunu bitirmiş, ondan sonra da bu verilebilir" şeklinde ilerleyebiliriz. Yani issue bağlarken tek dosya değil de dosya grubuna.

Participant 2: Usually, as I said, they don't go class by class but are grouped together. They are tied to an issue. I make sure the issue number is written in all files for commits. So, if four files are changed in the same commit, the issue number is in all of them. This way, related files can be grouped, and we can proceed, "This person completed this, then it can be assigned," not just for a single file but for a group of files.

E: Biz de zaten n tane dosya ile ilgili olduğunu biliyoruz. Trace ediyoruz zaten. Yani tek bir file bağlamaya çalışmıyoruz.

E: We already know how many files are involved. We trace them. So, we don't try to link just one file.

Katilimci 2: Yani burada tek file var ya.

Participant 2: Here there's only one file, right?

E: O örnekte öyledir.

E: That's how it is in that example.

Katilimci 2: Tamam zamanında bak şurada da fail olarak tek bir tane dosya seçiliyor ve tek dosya açtım şurada.

Participant 2: Okay, but look, here a single file is selected as a failure and I opened a single file here.

E: Hıhı. E: Yes.

Katilimci 2: Tek de olsa gidiyor ya şurada, orada birden fazla. Ama bunun üzerinde yani o?

Participant 2: Even if it's a single file, it goes there, but multiple files are involved. So?

E: Beş tane dosya grubu vereyim. Bu, bu beş dosyayı en iyi bilen kişiyi bulmak istiyorsunuz mesela, anladım, anladım.

E: Let me give you five file groups. You want to find the person who knows these five files best, right, I understand, I understand.

Katilimci 2: Tamam tamam.

Participant 2: Okay, okay.

Katilimci 1: Bazı dosyalar herkesin gidip adını yazıp çıktığı dosyalar olabiliyor ama bazı domain'ler içinde mesela atıyorum, file upload için bir class'ı değiştirmeniz gerekiyor. Ama onu değiştirirken de onunla ilgili servislere birer satır yazmanız gerekiyor. Ama bunların hepsini birleştirince aslında uzmanı kim olduğunu bulabiliyorsunuz.

Participant 1: Some files are those where everyone writes their name and leaves, but for some domains, for example, you need to change a class for file upload. While doing that, you also need to write a line for related services. Combining all these, you can actually find out who the expert is.

Katilimci 2: Ya ya da şey oluyor, zaten proje şeylerinde servislerde tek bir dosyada yazılı olmuyorlar. Bir giriş noktaları oluyor, bir işlem dosyaları oluyor. Grup grup oluyorlar.

Participant 2: Or it happens that services in projects are not written in a single file. They have entry points, processing files, etc. They are grouped.

Katilimci 1: Layer layer düşündüğünüz için hani bir servis layer'ı oluyor, bir endpoint'i oluyor, bir şey oluyor.

Participant 1: Thinking in layers, there's a service layer, an endpoint, something like that.

E: Onu UI'da söylemesi zor yani şey mi ya.

E: It's hard to describe that in the UI, isn't it?

Katilimci 2: Mesela hani önce bunlara ilişkilidir deyip buna göre öner diyebiliriz. Yani dışarıda kişi seçsin, şu şu şu dosyalarla en çok ilgilenen kişileri. Hani o konuda bilgili olanları. İşte sonuçta.

Participant 2: For example, we could first say these are related and suggest accordingly. Let the person outside choose, the ones most involved with these files. Those knowledgeable in that area.

Katilimci 1: Veya ek bir analiz de düşünebilirsiniz. Bu file'la birlikte olan file'lar kimlerdir? Traceability kullanarak, işte bu file'la birlikte olan file'lar kimler. Ek bir analiz çıkarabilirsiniz.

Participant 1: Or you could consider an additional analysis. Who are the files associated with this file? Using traceability, you can find out who worked on files associated with this one. You can come up with additional analysis.

E: Bu arada bizim başka bir ürünümüz var. Biraz bunu yapıyor. Mesela işte biraz önce dediniz ya, mesela A.java'yı değiştirirken B'yi de hep beraber değiştiriyorum. Mesela eskiden 10 kere commit yapmışım, onunla da A'yı değiştirmişim, B'yi değiştirmişim, A'yı değiştirmeye çalıştığımda uyarıyor, "Normalde B'yi de değiştiriyordum bununla beraber" gibi. Mesela öyle bir şey yapıyor, buraya da uygulanabilir.

E: By the way, we have another product that does something similar. Like you said earlier, for example, while changing A.java, I always change B as well. If I've made 10 commits in the past, changing A with B, it alerts me, "Normally you would change B along with this." For example, it does something like that and can be applied here too.

Katilimci 1: Bazı şeyler de mesela rutin faaliyetler mi? Bizler yeni bir konfigürasyon ekleyeceğimiz zaman 50 tane dosyayı o konfigürasyonla yazmamız gerekiyor. Sen unutursan patlıyor, mesela onu link edip gösterse çok işe yarar. Nasıl sistem patlıyor mesela onu da söylese çok faydalı olur.

Participant 1: Some things are routine activities, for example. When we add a new configuration, we need to write it in 50 files. If you forget, it fails. Linking and showing that would be very useful. Also, explaining how the system fails would be very helpful.

Katilimci 4: Aynen aynen, konfigürasyon tarafı için çok geçerli. Bu da aynı şekilde.

Participant 4: Exactly, exactly, very valid for the configuration side. This too, similarly.

E: O şeyde de var aslında. Çünkü kola alan müşterilerin %90'ı cips de alıyor, yanında falan gibi. Yani o dosyayı değiştirdiğinde front-end'de bir şey değişiyorsun, arkada da hep onu değiştirmen gerekiyor. Buraya da benzerini yapmaya çalışabiliriz. Yani biraz daha farklı olur, güzel bir feature olur bu.

E: It's similar in that too. Like how 90% of customers who buy cola also buy chips. So when you change a file, you need to change something in the front-end and back-end as well. We could try something similar here. It would be a nice feature, a bit different.

1:26:00 - 1:44:00(Yemek Molasi)

Katilimci 2: Hani şey çok iyi olur, zor issue çözen commit'lerin yanına böyle o bilgiyi de verirse veya işaretlense.

Participant 2: It would be great if it provided that information next to commits that solve difficult issues or if they were marked.

2:15:00 - 2:42:00 (Post Question Session)

L: İlk sorumuz şu şekilde: aslında size sunumda yaptığımız bu SAA'nin graph modelini ne kadar yeterli buldunuz? Projeyi yansıtmak açısından önümüzdeki sayfalardan inceleyebilirsiniz modeli. Bu bazda alt sorumuz, siz olsaydınız hangi tür software artifact'lerin, ilişkilerin veya özelliklerin eklenmesi gerektiğini düşünürdünüz? Sizce modele neler eklenebilir ekstra olarak bunu sormuştur.

L: Our first question is: how adequate did you find the graph model of the SAA presented to you during the presentation? You can review the model in the upcoming pages to reflect on the project. Based on this, our sub-question is, if you were in charge, what type of software artifacts, relationships, or features would you think should be added? What additional elements do you think could be included in the model?

Katilimci 3: Bence relation'lar gayet kapsayıcı ve yeterli ama. Issue'ların önemi olsun, file ve commit'lerin boyutu olsun, belki bazı katsayılar eklenebilir. Yani onun dışında zaten baya çeşitli gözüküyor.

Participant 3: I think the relations are quite comprehensive and sufficient. However, the importance of issues, the size of files and commits, maybe some coefficients can be added. Otherwise, it already looks quite diverse.

Katilimci 2: Tabii tabii.

Participant 2: Of course, of course.

Katilimci 3: Ben düşünüyorum günlük hayatta neler kullanıyorum, bunları kullanıyorum.

Participant 3: I'm thinking about what I use in daily life, and I use these.

Katilimci 1: PR badge size deniyor, toplam line of code sayısı.

Participant 1: It's called PR badge size, the total number of lines of code.

L: O var aslında hocam.

L: We actually have that.

Katilimci 1: Ayrıca mesela review ilişkisi, toplam review edilen line sayısı. Aynı şekilde review gidip geldi, kaç kez turn oldu. Çünkü mesela review ediliyor, sonra tekrar değişiyor, tekrar review ediliyor. Turn sayısı.

Participant 1: Also, for example, the review relationship, the total number of reviewed lines. Similarly, the review goes back and forth, how many times it turned. Because, for example, it's reviewed, then changed again, and reviewed again. Turn count.

Katilimci 1: Ya ama o turn sayısı da çok mantıklı olmayabilir çünkü. Bazılarının yazamıyorlar. Çünkü issue açıldığında da çok açık olmuyor. Benim issue'larım çok fazla gidip gidip geliyor. Benim müşteriden kaynaklı olarak. Çünkü küçücük az bir satır koyuyorlar yani. "Kervan yolda dizilir" diyeyim de hepsini şey yapıyorlar, tabloyu etkilenecek diyor ya bu tablonun kolonları yok mu diyorum, oraya açıklayın, size şunu da ekleyelim, bunu da ekleyelim. Bilmem ne, o 100 tane şey var camdan, yani o yüzden o turnover biraz şey. Yani issue ilk başta düzgün açıyorsan mantıklı ama genelde çok işlemiyor.

Participant 1: But the turn count might not be very logical because some can't write clearly. Because when an issue is opened, it's not very clear. My issues go back and forth a lot, because of the customer. They put in a tiny bit of text. "Kervan yolda dizilir" as they say, and they keep adding things, saying it will affect the table. I say, aren't there columns in the table, explain it there, and we'll add this, add that. There are 100 things through the window, so that turnover is a bit tricky. If you open the issue correctly at first, it makes sense, but generally, it doesn't work well.

Katilimci 1: Issue type ile de ilgili mesela. Yani improvement gidip gelebilir ama. Mesela bug ise onun gidip gelmemesi lazım.

Participant 1: It's also related to the issue type. For example, an improvement might go back and forth, but if it's a bug, it shouldn't go back and forth.

Katilimci 1: Bir de reopen oluyor issue'ların bazıları. Yani aynı hata çözülüp tekrar geliyor. Onlar ne kadar çok dönüyor ona bakmak lazım.

Participant 1: Some issues get reopened. That is, the same error is solved and comes back again. We need to look at how often they go back and forth.

E: O anomaliler arası var, onu söylüyorum.

E: There are those anomalies, that's what I'm saying.

Katilimci 3: Bir de şey, bir şey söyleyebilirim. Jira'da, bazen yanlışlıkla işi kapatıyorum. 10 saniye sonra geri açıyorum. Bu tarz durumları datadan elemek açısından bir analiz olabiliyor.

Participant 3: Also, I can say something. In Jira, sometimes I accidentally close a task and reopen it 10 seconds later. An analysis to eliminate such situations from the data can be helpful.

E: Biz onu aslında anomaly dahil etmemek için onu ediyoruz başında. Yani hani onu.

E: We actually exclude that at the beginning to avoid including it as an anomaly.

Katilimci 3: Tamam ama mesela hiç alakam olmayan bir şey, issue bakıyorken, okuyorken bir kısa yolu oluyor. Klavye diye bir şey.

Participant 3: Okay, but for example, sometimes something I'm not related to, while looking at an issue, reading it, there's a shortcut. Like on the keyboard.

Katilimci 1: Direkt zaten datayı yüklerken o durumlar elimine edilebilir. Çünkü bazen bazı caseler oluyor mesela. Sistem patladı, şey orada olabilir ama herkes sonuçta programlara dalmış yani. Onu buraya katma değil. Hani bunu sistemden çıkartmak gerekebilir yani.

Participant 1: Those situations can be eliminated while uploading the data. Because sometimes there are cases, for example, the system crashes, it might be there, but everyone is engrossed in the programs. So, it's not about including it here. It may need to be removed from the system.

Katilimci 4: Burada tabii bu artifact'ler üzerinden konuştuk ama kurumsallarda da vardır işte aslında. O PR'daki CI başarılı, işte ya da ne bileyim birden fazla CI olabiliyor. Bazıları bir tane başarılı bir build yeterli oluyor ama onun 3 tane build var aslında koşan. Hani oradaki şeyleri de değerlendirmeye alınabilir. Bütün build'ları başarılı olarak geçmiş olan o author'ları öner işte gibi ya da. O yani build mekanizmasıyla ya da oradaki başarımla ya da bu bir yere release edilebilmiş commit'lerin author'ları öner gibi güzel şeyler çıkabilir ya bizdeki örnekten yola çıkarsam bizde 3 ayda bir işte front-end'ler için söyleyeyim, mesela library olabiliyor mu? Çünkü biz bir ürünümüz bizden türelecek olan ürünler onu kullanacak ama defaultta lazım değil oluyorsa geçiriyoruz bir yere. Çünkü benim ürünüme lazım değil ama benim ürünümden türeyeceklerini engelleyeceğim. Diğer ikisi geçtiyse bu geçmiş oluyor ama aslında bunu da geçmiş. Olan commit'leri yapan kişi daha doğru düzgün, güzel kod yazmış demek olduğu yazdığında. Onu oradaki kişilerden önerse daha hoş olur, güzel iyi.

Participant 4: Here, we talked about these artifacts, but it's the same in corporates. The CI in that PR is successful, or maybe there are multiple CIs. Sometimes a single successful build is sufficient, but there are actually 3 builds running. Things there could also be taken into consideration. Suggesting authors whose builds passed successfully, or suggesting authors of commits that were successfully released somewhere, could bring nice outcomes. From our example, for front-ends every 3 months, we might have a library. Because our product will be used by derivative products from us, but if it's not needed by default, we move it somewhere. Because it's not needed for my product but I'll prevent derivatives. If the other two passed, this one has also passed. So, the person who made the commit has written proper, good code. Suggesting those people would be better.

Katilimci 3: SonarQube mesela require check değil. Olmasa da olur. Yani bazı bulunmaması ama orda yeşil alanları onun.

Participant 3: SonarQube, for example, isn't a required check. It doesn't have to be there. I mean, some things aren't found, but there are its green areas.

L: Bu dediğiniz analizlerin eklenmesi için sizce ekstra hangi artifact'ler eklenebilir? Yani kayda geçmesi için?

L: What additional artifacts do you think could be added to incorporate these analyses? To record them?

Katilimci 4: CI run'ları, mesela biz 3 build varsa 1 tanesi geçse de kabul ediyoruz ama bunun belirtilmesi açısından eklenebilir.

Participant 4: CI runs, for example, if we have 3 builds, we accept if 1 passes, but this could be added for clarification.

Katilimci 1: Benim dediğim file'ların oluşturduğu cluster'lar. Yani birlikte değişen file'lar.

Participant 1: The clusters formed by the files I'm talking about. The files that change together.

Katilimci 5: Comment etmek de expert bulma işlemine katılabilir.

Participant 5: Commenting can also be included in finding experts.

Katilimci 1: Bizim o eskiden kullandığımız araçta da o şekildeydi. Çok comment veren aslında kodu daha çok bilen anlamına geliyor.

Participant 1: In the tool we used before, it was like that. The one who comments a lot actually means they know the code better.

Katilici 2 : Bazen seyde oluyor musterinin commentleri bir aydır bekliyor diye bir şey yapalım.

L: Diğer bir sorumuzda da aslında şunu sormuştuk, sizce bu software artifact analyzer tool'una yeni bir analiz eklenecek olsa ne gibi analizler eklenebilir? Sormaya çalıştığımız şey aslında şu, dediğimiz gibi biz en temelinde bu software artifact traceability graph yapısını kullanıyoruz. Sizin aklınıza mesela bu graph yapısını kullanarak aslında aynı şekilde bu sisteme entegre edilebilecek bir analiz çeşidi geliyor mu? Veya hani sizin hani çalışma süresince keşke böyle bir araç olsaydı dediğiniz bir noktada olabilir. Ya aslında cevapladınız o yüzden yani yani bayağı var aslında. Daha detayli katki vermek istiyern olur mu acaba.

Katilimci 3: Şey güzel olur aslında böyle kendime yönelik geçmişte ne yapmışım şeklinde bakabilirim.

Participant 3: It would actually be nice to look at what I've done in the past for myself.

Katilimci 1: Orada developer bazında seçiliyor galiba benim şeylerim var. Hatta zamanda oynatabilirsin. Böyle artmış mı.

Participant 1: I think it's selected on a developer basis there. You can even play with the time to see if it has increased.

Katilimci 2: Bunun bir tane kullanım kılavuzunu yapacak mısın?

Participant 2: Will you make a user manual for this?

L: Var hocam.

L: Yes, we have it.

Katilimci 5: Şey eklenebilir, takımların gerçekte sizin düzenlediğiniz takım yapısına uygunluk olmadığını, kod dosyalar üzerinden çıkarılmasıyla ilgili çalışmalar vardı. Belki öyle bir şey olabilir. Developer o şekilde cluster etmek tarzında. Hani bazen bizim şirkette de 4 kişiyle başlıyor. Takım sekize çıkıyor, sonra bölüyorlar ama neye göre bölünecek yani veya başka bir konsepte göre yeni yapılacak işin mantığına göre.

Participant 5: Something could be added regarding whether teams actually fit the team structure you designed, based on code files. Maybe something like that. Clustering developers in that way. Like in our company, sometimes a team starts with 4 people, then grows to 8, and then they split. But how should they be split, or according to what concept should new work be structured?

Katilimci 4: Yani şey gibi bir analiz de eklenebilir mesela. Kritik developer'ların bulunması. Bu kritik, kritik onlar giderse ne olur gibi.

Participant 4: An analysis could be added to find critical developers. Like, if these critical developers leave, what happens?

E: Onunla ilgili benzer bir çalışmamız var mesela bus faktör falan gibi bir şey. Yani işte şu şu modülü sadece bu biliyor, bu modülü 3-4 kişi biliyor falan.

E: We have a similar study on that, like a bus factor. So, for example, only this person knows this module, while 3-4 people know this other module.

Katilimci 4: Bu bunu biz sözel olarak biliyoruz da bunu birine ikna ederken data gerektiğinde mesela sen onu ağzına söylüyorsun. Yani "Şu kişi derse yanarız" diye de yönetici herkes muhabbetidir. Aslında hepsinin yerine biri bulunur falan, sadece tabi 10'a çıkart bakarsanız mezarlıklar iyi.

Participant 4: We know this verbally, but when convincing someone else and needing data, you say it. Like, "If this person leaves, we're in trouble," and every manager talks like that. Actually, someone can replace everyone, but of course, if you look at 10, graveyards are good.

U: Eray o şapkayı taktığı zaman o konularıyla çalışıyor, ilgili çalışıyor. Bizim buradaki şapkamız ayrı bir şapka. Yani biz o çalışıp yapılan çalışmalara buraya entegre ediyoruz.

U: When Eray wears that hat, he works on those subjects, related to that. Our hat here is a different one. So, we integrate those studies done into this.

L: Diğer bir soruda aslında, bu gösterdiğimiz bu çizim kanvası projedeki software artifact'leri hakkında bilgi edinme veya genel olarak bu tarz işler için ne kadar katkı yaratıyor? Projedeki artifact'leri incelemek için bu çizim kanvasını kullanır mıydınız?

L: Another question is, how useful is this drawing canvas we showed for getting information about software artifacts in the project or for general tasks like this? Would you use this drawing canvas to examine the artifacts in the project?

U: Mesela Alperen kendisini buraya query edip sonra show all work diyebilir. Yani bir sürü olur.
U: For example, Alperen can query himself here and then say "show all work." There would be a lot.

Katilimci 3: Evet, beni çok kullanırdı ya. Dediğiniz gibi yaptığım işlere bakmak için.

Participant 3: Yes, it would be very useful for me. As you said, to look at the work I've done.

Katilimci 4: Ben de kullanırdım, işime yarardı. Ben BevOps'tayım mesela. Bir yerde bir problem çıktığı zaman mesela hızlıca kimseye sormadan gidip artık hep de vakit he bunu şurayı bozmuşlar, buradan da şunu bozmuşlar deyip gidip bulurum sahibini. "Buraları niye bozuyoruz?" derim çok.

Participant 4: I would use it too, it would be useful for me. I'm in BevOps, for example. When a problem arises somewhere, I can quickly go and find the person responsible without asking anyone. "Why are we breaking these?" I often say.

L: Tamamdır, şimdi isterseniz bu deneme bölümünde yaptığınız task'lar dolayısıyla verdiğiniz cevaplar üzerinden gidebiliriz.

L: Alright, now if you want, we can go through the answers you provided during the trial tasks section.

Katilimci 2: Valla ben o aracın önerdiğini yazdım. Tool önerdiği bana mantıklı geldi.

Participant 2: Well, I wrote what the tool suggested. The tool's suggestion made sense to me.

Katilimci 4: Recency denince ikinci olarak Hans çıkıyordu. Fakat katkı olarak bakınca Peter çıkıyordu. Onun da daha çok katkısı var ama daha eski.

Participant 4: When considering recency, Hans came out second. But when looking at contributions, Peter came out. He has more contributions, but they're older.

Katilimci 3: Hans ile Peter arasında çok az fark vardı. Ben Peter'ı seçtim. Graph açınca bir tür ilişki vardı. Yani tool'u kullandım.

Participant 3: There was very little difference between Hans and Peter. I chose Peter. When I opened the graph, there was a certain relationship. So, I used the tool.

Katilimci 1: Bu aslında şeyi gösteriyor yani, sorunun zor bir soru olduğunu. Yani bir bakışta çözülemeyeceğini gösteriyor. Bir tool support gerektirdiğini.

Participant 1: This actually shows that it was a difficult question. It indicates that it can't be solved at a glance and that tool support is required.

L: Tamam, devam edelim. Diğer bir sorumuzda da aslında hangisinin daha güvenilir sonuçlar verdiğini sormuştuk. Burada da sanırım genel olarak insanlar recency value ile birlikte SAA'yı tercih etmiş. Ama genelde SAA tercih edilmiş. Bu konu hakkında yorum yapmak isteyen olur mu, yoksa geçelim mi? Tamam geçelim.

L: Alright, let's continue. Another question we asked was which one gave more reliable results. Here, I think people generally preferred SAA with the recency value. But overall, SAA was preferred. Does anyone want to comment on this, or should we move on? Okay, let's move on.

E: Bu arada buradaki rakamlar bir önceki bir fokus grup 5 kişi daha vardı. Onun rakamlarına toplam görüyorsunuz.

E: By the way, the numbers here include a previous focus group with 5 more people. You're seeing the total numbers.

L: Diğer bir sorumuzda ise SAA hakkında inandırıcı bulmadıysanız nedenleri ne diye sormuştuk. Limited context önde gelmiş, bu konu hakkında yorum yapmak isteyen olur mu?

L: Another question we asked was, if you didn't find SAA credible, what were the reasons? Limited context came up as a leading reason. Does anyone want to comment on this?

Katilimci 4: Evet aslında şey, tool'da ki şeyleri ben customize edebiliyor olsam kimsenin bulamayacağı şeyleri bulabilirim. Ama hani oradaki set stabil geliyor. İlk etapta mesela burada şey diyebilirim. Ne bileyim önceliği yüksek olmayan şeyleri veya da lines of code'undan az olan alan gibi kendimce bir şeyler ekleyebiliyor olsam çoğu zaten. Eksiğini söylemek için söyledim.

Participant 4: Yes, actually, if I could customize things in the tool, I could find things no one else can. But the set comes stable initially. For example, I could add things like low-priority issues or areas with fewer lines of code. I'm saying this to point out the shortcomings.

Katilimci 4: Sadece graph analiz yapıyorsunuz. Geri kalan değişik yaptığınız denge direk kuruyorsunuz ama belki o kodlardaki line of code değişikliği oranı veya işte hep aynı bölümleri değiştirdi. Ne kadar varyasyon sorular. İşte doksanla dokundu, 80 dokundu veya onunla dokundu ama 50 kere değiştirdi. Vesaire. Hani bu tür contextler belki biraz daha oradaki ekspertizi netleştirebilir. Doğru. Burada birden çok zeki biliyor.

Participant 4: You only do graph analysis. You directly balance the remaining changes, but maybe the rate of line of code changes or changing the same sections repeatedly. Questions like how much variation there is. For example, it touched 90, 80, or it touched it but changed it 50 times, etc. These kinds of contexts might clarify the expertise there a bit more. Correct. Here, many people know it.

L: Bir diğer sorumuzda ise GitHub ile bulma yönteminiz güvenilmez ise neden diye sormuştuk. Burada da limited content işaretlenmiş, bu konu hakkında yorum yapmak isteyen olur mu?

L: Another question we asked was, if you found the method of finding with GitHub unreliable, why? Limited context was marked here as well. Does anyone want to comment on this?

Katilimci 4: Ya tabii çıkarım yapmak için çok zor oluyor. Ondan ona gidiyim, ondan ona gidiyim.

Participant 4: Yeah, of course, it's very difficult to make inferences. Going from one to another.

Katilimci 5: Mesela ben gözümle yaparken Michael Mostarda'yı seçmiştim. Halbuki 15 yıl önceymiş. Çok dikkat etmemiştim ya o yüzden. Ben bulamadım mesela.

Participant 5: For example, when I was doing it manually, I chose Michael Mostarda. Turns out it was 15 years ago. I didn't pay much attention, so I couldn't find it.

Katilimci 1: Gözle aramak daha zor tabii ki, tool'da direkt sonuçları ve nedeni gösteriyor.

Participant 1: Searching manually is more difficult, of course. The tool directly shows the results and reasons.

L: Diğer bir sorumuzda da aslında genel olarak bu görselleştirmenin sonuçların anlamlandırılabilirliğine ne kadar katkı sağladığını sormuştuk. Cevaplara göre genel olarak pozitif katkı sağladığını düşünüyormuşuz. Bunu da mesela 3 diyen biri konuşabilir mi acaba?

L: Another question we asked was how much this visualization contributes to the interpretability of the results. According to the answers, we think it generally provides a positive contribution. Could someone who rated it as 3 speak on this?

L: Burada da genellikle olumlu cevap almışız. Bunun üzerine yorumda bulunmak isteyen olumlu yani görsellik mesela analizinizde nasıl size bir yardım sağladı? Yani sonuçların çıkarım inandırıcılığı sizin için arttı mı?

L: Here, we've generally received positive responses. Would anyone like to comment on how visualization helped you in your analysis? Did it increase the credibility of the results for you?

Katilimci 2: Göstermek iyi oldu tabi. Yani burada 5 var, bu 3 yapmış, bu 2 yapmış, ne yapmış bak. Görsel daha iyi tabi.

Participant 2: It was good to show it, of course. Here, there's a 5, this one made a 3, this one made a 2, see what they've done. Visual is definitely better.

Katilimci 3: Ben mesela Peter'ı kesinlikle ikinci sıraya koymazdım. Büyüklü bir süre sonra anlaşılıyor tabii ki.

Participant 3: For example, I definitely wouldn't have placed Peter in the second position. It becomes clear after a while.

Katilimci 1: Ben sadece visualization şeyi eklerdim. Hani graph gösterimi güzel hoş ama. Hani her şeyi anlatamıyorsun. Grafik biraz daha chart'lardır, işte raporlardır. Hani aynen dashboard koymak lazım.

Participant 1: I would just add something to the visualization. The graph display is nice, but you can't explain everything. Graphics are more like charts, reports. We need to add dashboards.

E: Bunun bir önceki fokus grupta da özellikle yani böyle dashboard'tan gitmek istiyor insanlar.

E: In the previous focus group, people also wanted to go with dashboards.

Katilimci 1: Zaman içinde böyle ne olmuş, aynen trend yönetimi.

Participant 1: What's happened over time, exactly, trend management.

L: Tamam, diğer bir sorumuzda da aslında eklediğimiz bu görsel elementlerin, mesela node badge'lerinin, anlayışa ne kadar katkı sağladığını sormuştuk. Bunda da genelde pozitif yanıt çıkmış. Bu konu hakkında konuşmak isteyen olur mu? Yani şu şekilde geliştirilebilir şeklinde de bir yorum alabiliriz.

L: Alright, another question we asked was how much the visual elements we added, such as node badges, contributed to understanding. This also generally received positive responses. Would anyone like to comment on this? We could also take suggestions on how it could be improved.

Katilimci 1: Az önce bahsettiğim gibi, buradaki 3'ü ben verdim. Graph yeterli gelmiyor çünkü bana.

Participant 1: As I mentioned earlier, I gave it a 3. The graph isn't sufficient for me.

L: Diğer bir sorumuzda aslında SAA ile expert bulmanın sizin normalde kullandığınız metoda kıyasla daha kolay olup olmadığını sormuştuk. Genellikle kullanıcılar daha kolay olduğunu düşünmüş.

L: In another question, we asked whether finding experts with SAA was easier compared to the method you normally use. Generally, users thought it was easier.

L: Son sorumuzda da aslında Jira ve SAA'yı karşılaştırdığımızda anomaly detection açısından hangisinde daha rahattı diye sormuştuk ama zaten konuştuk. Üstüne daha fazla konuşmak isteyen olursa. Çoğu kullanıcı SAA'yı daha useful bulmuş.

L: In our last question, we asked which was more comfortable for anomaly detection when comparing Jira and SAA, but we've already discussed this. If anyone wants to add more, most users found SAA more useful.

Katilimci 4: Şöyle bir yorumda bulunabilirim. Buradaki anomaliler bir varsayıma dayanıyor.

Participant 4: I can make a comment like this. The anomalies here are based on an assumption.

L: Genel olarak SAA hakkındaki deneyiminizi sormak istemiştik. Kafa karıştırıcı noktaları neydi sizce veya siz bu aracı kullanıyor olsaydınız en fazla hangi use case bazında kullanıyor olurdunuz? Hani inspection kullanırdınız yoksa yeni bir analiz mi entegre ederdiniz, yoksa hâlihazırda olan analizlerden birini mi kullanırdınız? Bu şekilde bir soru sormuştur.

L: We wanted to ask about your overall experience with SAA. What were the confusing points for you, or if you were using this tool, for which use case would you use it the most? Would you use it for inspection, integrate a new analysis, or use one of the existing analyses? We asked a question in this way.

Katilimci 2: Projede en çok çalışan kişiyi bulabiliriz. Bunu elden kaçırmayalım. Bu çocuk iyiymiş diye. Bize yol gösterir.

Participant 2: We could find the person who works the most on the project. Let's not miss this. We can say, "This person is good." It guides us.

Katilimci 4: Görsel olarak iletişim sağlamıyor birçok araç. Bir de navigate ederek görebiliyorum, ilişkisel olarak inceleyebiliyorum. Çok yönlü olması açısından da kullanışlı. Aslında tool'larımızda bir şeye göre bir şey söyleyebiliyoruz. Ama issue bir alanında şöyle bir veri varsa ve o developer şöyle bir iş yaptıysa gibi çok yönlü bir veriye sahip olmamızı sağlıyor.

Participant 4: Many tools don't provide visual communication. And I can navigate and see, I can examine relationally. It's useful because it's multifaceted. Actually, in our tools, we can say something according to something. But if there is such data in an issue area and the developer did such a job, it allows us to have multifaceted data.

Katilimci 3: Bir de bizde çok refactoring oluyor, onun çaresine bakabilse çok iyi olur.
Participant 3: We also do a lot of refactoring, if it could handle that, it would be great.

Katilimci 5: Graph'in expandable olması bence çok iyi olmuş.

Participant 5: The expandability of the graph is very good, I think.

L: Bir de bir tane survey daha göndereceğiz, onu da doldurursanız seviniriz. Onu sonra ile göndereceğim. Çok teşekkür ederiz. Çok katkıda bulundunuz. Çok sağ olun, çok güzel feedback'ler aldık, kullanacağız. Çok sağ olun.

L: We will also send another survey, we would appreciate it if you could fill it out. I will send it later. Thank you very much. You provided a lot of input. Thank you very much, we received great feedback, we will use it. Thank you very much.