1. A description of the high-level domain / environment
2. A key problem that happens in this high-level domain
3. Current solutions to that key problem
4. A problem with the current solutions
5. A glimpse at how you solve the problem in (4)
6. An overview of your evaluation / experiment results

// add cost relation of RE in early

The software industry has changed and grown remarkably over the last decades. Software system are developed over million lines of code, number of modules and documents. The primary goal of the software system is to satisfying the users by developing the software that can meet up their needs and expectations. This goal is achievable by applying different methodologies and engineering techniques. One of the main factor is understanding the needs of users i.e. formally Requirement engineering (RE). RE is the key phase in the development of the software. RE is the first complete phase in the proposed literature e.g. waterfall model. As time passed software development models gets a different vision now the RE is the part of the w life cycle from the beginning until the end of the project e.g. agile model. Requirement engineering can be defined as

[1]Nuseibeh and Easterbrook, 2000

by the is Initially requirement engineering was the first complete phase of the software process model.. It provides the essential information and idea for the product without building it.

.. copied from the article//Except for well-defined problem domains and strict contractual procedures, most software-development projects address requirements specification and design issues simultaneously—and justifiably so. Achieving a separation of requirements and design steps is often difficult because their artificial ordering compels developers to focus on either aspect at any given time. In reality, candidate architectures can constrain designers from meeting particular requirements, and the choice of requirements can influence the architecture that designers select or develop. Based on our experience in industrial software-development projects, my colleagues and I use an adaptation of the spiral life-cycle model. We informally call this model Twin Peaks to emphasize the equal status we give to requirements and architectures

With Twin Peaks, developers can identify requirements and match architectures with commercially available products, rapidly and incrementally. The developer benefits by quickly narrowing the selections or making key architectural decisions to accommodate existing COTS solutions//

//We find that app metrics such as price, rating, and download rank are significantly different between the three completeness levels. We show that correlation analysis can find trends in the data that prevail across the partitions, offering one possible approach to App Store Analysis in the presence of sampling bias.//

// In the last part of the talk, we will discuss the lessons learnt during the analysis of the last 25 years of research in requirements elicitation and how they can be applied to guide the next steps of the research in requirements elicitation. In [12], lack of tools and technical support and inertia have been recognized as the three main causes of the distance between research and industry. This seems to suggest that the consistent interest in developing tools based on current technologies to support the elicitation process is the right direction to follow. However, the time and effort required to develop a usable tool and the needed support might not be feasible in practice for a research team. Moreover, since elicitation is a discipline which requires a lot of human skills, best practices could be taught with simple trainings, which might be sufficient in the general case and be also the successful approach to interrupt the inertia within companies.

A different direction already emerging in other areas of software engineering is to use new technologies to augment the information that the analyst can access to during the elicitation process. Possible directions rely on the use of biofeedback, video and audio analysis

//

t. Wano and Iio [241] analysed the textual content of 856 reviews from 500 apps in the Japanese App Store, and found that the review styles differed between apps in different categories.

//Many studies have produced tools which can aid in the summarisation and requirements extraction from reviews, but these tools have not been widely adopted as of yet by developers

Study shows that user analytics tools will help developers to deal with the large numbers of user feedback (e.g., user reviews) by filtering, classifying, and summarizing them, to decide what requirements and features they should add, change, or eliminate

As a AF3 case study:

<https://af3.fortiss.org/main-features/requirements/>

<https://visuresolutions.com/>

1. AF3 is an open source software by fortiss. For this developer get feedback or requests for the support of the software from the user as a bug, change request, feature, and support. Then manually a person check that list and assign it to the relevant person. For automatically assignment of the assignee, ML can be used. We can use the existing data set as an input for the ML.

We can also see the other trends in the dataset and conclude the tractability information of the current and old implemented requirements. The impact of the new features and change feature on the software.

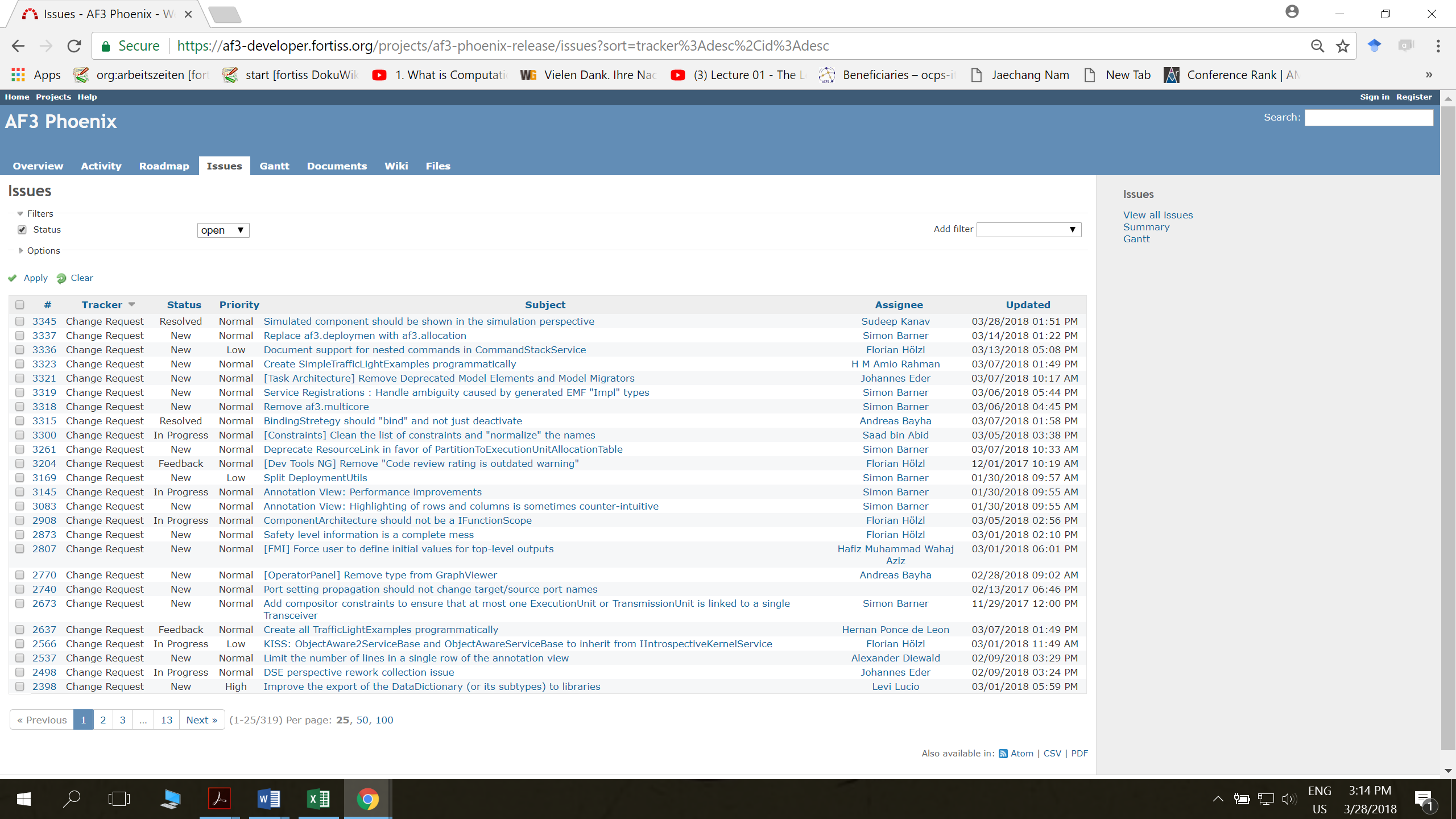
**Main categories**: bug, change request, feature, support.

**Features:** Number (#), status, priority, subject, assignee, updates

**Dataset:** total=2484; Bug= 958, Change Request =176, Support =433, Features= 899.

1. **Problem:** Extracting information relevant for design (and also testing) is sometimes hard

**Idea:** Extract relevant information for design (e.g. functions, inputs, outputs, operating modes, architecture elements) from the requirements, e.g. via ontology extraction.



1. The other problem that is noticeable from the literature is the cross project implementation for complementing the other dataset that has not enough dataset. If we have derived something in a domain D and we have conquered the results in the sub domain D’. At which extant D’ can be helpful for the prediction of other area of D. Was the covered area useful and helpful?
2. Reusability of Requirement
3. Finding relevant NFR for selected FR ( is it tracabilty of high leval and low levl req)
4. In Analyis of req identifying the related stakeholderwith ML? too vague
5. Getting requirments/ reviews from the same type of apps before building up the app in the same domain …. Review usability can lead us to requirement usability?
6. The requirement patterns imply a set of software requirements that handle each business problems. The requirement is expressed in norm [22] and stored in the requirement pattern repository that can be reused in a similar business domain. In order to identify the problem and requirement patterns, a domain analyzer first elicits the stakeholder model for identifying stakeholder and their roles, the process model for analyzing interactions between the system and stakeholders and the norms repository for storing common functional and non-functional requirements. Through the models, it builds a link between a process indicating a business goal and a norm handling the process. This type of the research only proposes a manual process for identifying requirement patterns from application specifc requirements specifcations without mentioning automatic or semiautomatic ways. Thus, the quality of domain modeling is critical to properly identify the patterns. Also, though the patterns are identifed, it only can be applied to the similar application domains. This is mainly caused by the lack of automatic ways to extract requirement patterns from a set of requirements specifcations
7. Is it possible to merge tweets and reviews and find the priority bugs or in other words can tweets be augmented by the app review or vice versa?

significant information from a SRS?

***RQ 1.1:*** What are Architecturally Significant

Functional Requirements (ASFRs) and what is

currently known about them?

***RQ 1.2:*** What categories of ASFRs have implicit

architectural impact for SAs?

***RQ 1.3:*** From a SA’s perspective, what mechanism is

used to unearth the unspecified architectural details in

a SRS?

***RQ 1.4:*** Having found answer to RQ 1.3, what kind of

information is discoverable using the mechanism

(identified in RQ 1.3)?

**RQ 2:** How to use the mechanism to equip BAs to ask more

architecturally relevant information? How could we design an

approach that integrates the mechanism in an organization’s

software delivery process?

**RQ 3:** Is the proposed way of working usable and useful in

practice?

***RQ 3.1:*** Can practicing SAs use the approach?

***RQ 3.2:*** Does the proposed approach take less effort

overall than the current way of working?

***RQ 3.3:*** Does it lead to architecture designs that are

comparable in quality to the current way of working?

***RQ 3.4:*** For which kinds of systems and development

1. projects can our approach be used?