Implementing automatic test of architecture rules

When developing software a number of architecture rules and requirements must be fulfilled for all code. These architecture requirements are documented in guidelines to the developers. The guidelines are located in a web based library accessible to all developers. All developers must be educated to understand the guidelines and must keep them in mind when developing code.

The education of guidelines is time consuming for both architects presenting the guidelines and the developers learning them. It is also hard for the developers to keep the guidelines in their mind and mistakes are made.

To attack these difficulties, the guidelines can be complemented by adding checkers and automatic tests in our tool-chain. When a developer creates code that creates a breach of a guideline, the checker finds this and gives fast feedback. The feedback can consist of a short error message and a link to the justification. The developer can then correct the code and learn about the architecture requirement.

The target of this master thesis is to implement some of our guidelines, with architecture requirements, as checker or automatic tests into our tool-chain.

If you are interested, please, contact makoto.kawamura@sonymobile.com for further information. All contacts must be made in English. The location for the master thesis is in the Sony building in Lund and is supervised by Robert Lagerstedt.

Background: To keep quality and maintainability of software, a number of non-functional requirements must be fulfilled. These non-functional requirements include architecture requirements. The requirements are documented in guidelines and design documents.

Examples of guidelines are:
- API design guideline
- Platform coding style
- Naming convention
- Customization guideline
- Patch guideline
- Copyright headers
- Software update rules

In Sony Mobile, a central organization called “Global System Management” creates architecture guidelines that are enforced in all development organizations of Sony Mobile. These guidelines are located in a central library and are constantly communicated to all architects and developers.

Problem statement: The communication of the guidelines is very time consuming for the architects at “Global System Management” and other architects and developers. This communication needs to be reoccurring since the guidelines are extended and changed from time to time. It is also easy for the developers to make mistakes and create breaches of the guidelines since it is hard to keep them in mind all the time.

When the breach is found, often by a quality or cost report, a long time has passed since the developer created the code. This increases the time to correct the code since the developer must task switch back to the code area in the development system as well as the mind.
Proposed solution: When developing code much of the work is done in a tool-chain. The tool-chain contains tools like text editors, compilers, linkers, static analysis tools, automatic test frameworks, etc. Many of the guidelines can be implemented as checkers or automatic tests in the tool-chain. This prevents breaches and it also educates the developers in a learning feedback loop.

Scope of Master thesis work: Implement a number of checkers or automatic tests in the Sony Mobile tool-chain to communicate the architecture requirements and verify the compliance. The work should include:

- **Analyze** the architecture requirements and categorize them in a way that identifies the possibility of creating automatic checkers and tests.
- Identify a number of architecture requirements that can be implemented and **design** a solution of a checker based on the text in the guideline. (Sometimes the text in the guideline only gives the intention of the requirement and this needs to be transformed into a checker algorithm.) The solution can include what tool should be extended, what data from the code is needed etc.
- **Implement** the checkers into the tool-chain as an informative checker. (Meaning that it will not break the code but instead give a warning to the developer about the breach. This is to make sure that the build is not broken if there is legacy code that have breaches.)

Extent: 2 x 30 hp Master Theses