INTERRUPT INSIDE

An in-depth magazine about IoT and embedded technology from Data Respons 1, 2019

Bringing the internet to the internet of Things:

- the current state of 6LowPan connecting Bluetooth IoT devices to the internet

Software:
Code quality assurance with PMD

Autonomous driving:
Data logging and autonomous vehicles

Introduction to whitepaper:
The optimal toolbox for open source development

data respons

WHAT'S **INSIDE**

EDITOR

Times are data driven. Gartner has identified 'autonomous things' as the number one tech trend for 2019. In fact, they predict that 10% of all vehicles will have autonomous driving capabilities by 2021, compared to 1% in 2018. They also claim that by 2022, 40% of new application development projects will have AI co-developers on the team. For these opportunities to succeed, all parts of the development cycle need knowledge and experience from sensor level to the final application, as well as connectivity, security and data analytics.

Specialists across our companies have in-depth experience in AI, IoT and digitalisation projects across all industries and share their knowledge in this magazine. Our core focus is to contribute to our customers' business-critical product development – enabling them to stay ahead in the innovation race. At Data Respons, sharing our expertise and knowledge with our colleagues, customers and the industry is part of our core values. We see it as a part of being 'data responsible'.

This issue of Interrupt Inside addresses the more fundamental issues of connectivity related to connecting Bluetooth devices to the internet in the article Bringing the Internet to the Internet of Things. Also in this issue, Crister Nilson from our subsidiary Sylog in Sweden takes us through the various steps to vehicle autonomy, from 'no autonomy' to 'full self-driving'. In addition, our experts at Techpeople introduce their white paper guide on open source development, which can be downloaded from our website.

I hope you enjoy reading it,

KENNETH RAGNVALDSEN

CODE QUALITY ASSURANCE WITH PMD

An extensible static code analyser for Java and other languages

BY: Andreas Dangel
Software Engineer
MicroDoc GmbH

Developing new software is great. But, writing software that is maintainable and sustainable is not so easy. Luckily, there are tools available that can help you achieving better code quality. One of these tools is PMD.

```java
public class Example {
    void run() {
        try {
            // do something
        } catch (Exception e) {
            e.printStackTrace(); // Not good - this exception should be logged via a logger.
        }
    }
}
```

Figure 1: Example for AvoidPrintStackTrace

WHAT IS PMD?

PMD is a static source code analyser. It scans your source code and searches for patterns, that indicate problematic or flawed code. Sometimes it is just a too complex solution which might increase maintenance costs in the long term or it might be a indication of a real bug. In that sense, PMD can be seen as another pair of eyes, that reviews your code.

For example, PMD can be used to find usages of the printStackTrace() method, which is often generated by the IDEs when surrounding a statement with a try-catch-block. Just printing the stacktrace might result in swallowing the original exception, since the output might end up somewhere. Ususally such output should be logged with the appropriate logging framework. PMD provides the rule AvoidPrintStackTrace, which detects such cases. See figure 1.

The abbreviation "PMD" is not exactly defined, it is actually a backronym. But "Programming Mistake Detector" or "Project Mess Detector" are the most logical meanings. However, the tool is usually known and referred to simply as "PMD", sometimes with the tagline "Don't shoot the messenger". See Figure 2 for the official logo.

Figure 2: The PMD logo

The patterns, that PMD is searching for, are defined by rules. PMD is shipped with more than 250 built-in rules, that can be used immediately.

When the rules detect a problematic piece of code, a rule violation is reported. Furthermore, own rules can be developed in order to adapt PMD to specific project requirements. With so many possible rules, it is clear, that one cannot simply enable all rules. Some rules even contradict each other. And some rules just have different coding conventions in mind, that might not be suitable for the concrete project at hand.

In the field of code analysers and so called linters, there are other products available. For Java projects, often checkstyle is used in order to enforce a common (project- or company-wide) code style. Having a common code style helps a lot if multiple developers working together on the same project, since each part of the project is then be read and skimmed as easy as any other part - regardless of the author. Checkstyle concentrates on the source code directly including whitespace checks like correct indentation and also documentation via JavaDoc comments.

PMD doesn't support whitespace checks, but it has basic support for comments, like enforcing the existence of JavaDoc comments for classes or fields. Other tools like FindBugs and its successor SpotBugs are analysing the compiled bytecode of Java projects instead of the source code. They have therefore access to the compiler optimised code and might see slightly different code. Moreover, SpotBugs can rely on the structure of a classfile and does not need to deal with syntax errors. SpotBugs can only be used after the project has been compiled, while Checkstyle could be run before.

PMD can be seen in between these two tools: While the starting point for PMD is also the source code, PMD takes advantage of the compiled classes. This feature in PMD is called "type resolution" and it helps PMD to understand the analysed source code better in order to avoid false alarms. E.g., if PMD knows the return type of a method call, rules can be written that only apply to a specific type. Otherwise, the rule would need to "guess" and assume the type by looking at the type name only and do a simple string comparison. If the project has an own class with the same name, then we might mix up the classes. A concrete example can be seen in unit tests: PMD provides several rules for JUnit. But if the project uses a different test framework with the same class names (but obviously different packages), then these rules would find issues, which are maybe irrelevant for the other test framework.
There are other big players for code quality tools on the market like SonarQube that support a more integrated solution to also monitor quality improvements or regressions over time.

When PMD is integrated into the build pipeline, it can act as a quality gate. For example, if rule violations are detected, the build can be failed or the commit can be rejected. This can be used to enforce a specific quality goal. The build pipeline could also be configured to only make sure, that no new rule violations are introduced, so that the code quality doesn't degrade and hopefully improves over time.

There is one other component in PMD, that is often overseen: CPD - the Copy-Paste-Detector. This is a separate component, that searches for code duplications in order to follow the DRY principle (Don't Repeat Yourself).

OVERVIEW / HOW DOES IT WORK?

PMD analyses the source code by first parsing it. The parsing process consists of the two steps:

- lexing, which produces a stream of tokens
- and parsing, which produces an abstract syntax tree (AST).

This tree is the equivalent representation of the source code and has the root node "Compilation Unit". In Java, you can define multiple types in one source file (as long as there is only one public) and

>>

```java
public class Example {
    public void method1() { }

    public void anotherMethod() { }

    public static class NestedClass { }
}

class OtherClass {
}
```

Figure 3: Source code of the AST example

classes can be nested. Classes itself can have methods, which in turn have zero or more statements. Figures 3 and 4 show a simple java class and the corresponding AST.

When the source code could be parsed to an AST, then the syntax is correct. Nowadays, it is recommended to use PMD after the project has been compiled in order to take advantage from type resolution. This means that PMD can concentrate on valid syntax, e.g. if the parsing fails, the analysis of this source file is simply skipped. Technically an own grammar for JavaCC is used to implement the parser for the Java language. Therefore, failing to parse a specific source file might actually indicate a bug in PMD's own Java grammar and does not necessarily mean, that the source code is not valid.

After that, the AST is enriched by a couple of visitors: First, the qualified names for the types, that are defined in the source code, are determined. This is later helpful when referencing this class (and its nested classes and lambdas) itself. Second, the symbol facade visits the AST. It searches for the fields, methods and local variables and looks up their usages within the scope of this source file. The information collected in this step is made available to the rules, e.g. they can easily figure out, if a (private) field or method is used or not. The found variables are organised in different scopes, that are nested. The third visitor is the "Data Flow" facade. It's goal is to follow variable definitions, assignments and reassignments and their accesses throughout the program flow. It allows to detect anomalies such as assigning a new value to a variable after it has been accessed. It's currently limited to a single method. The last visitor is the "Type Resolution" facade. It traverses the AST and resolves the concrete Java types of variable declaration, method parameters, and classes whenever a referenced type is used. It uses the compile-time classpath (also known as the auxiliary classpath) of the project that is being analysed.

Now, after the AST has been created and filled with additional information, the rules are executed. While all rules for one file are executed one after another, the analysis of multiple files (and ASTs) is executed multi-threaded. Each rule has the possibility of reporting rule violations, which are collected in reports. The violation contains the information about the rule, the location (like line and column in the source file) and a message. In the end, the reports are transformed into the desired output format, such as XML or HTML.

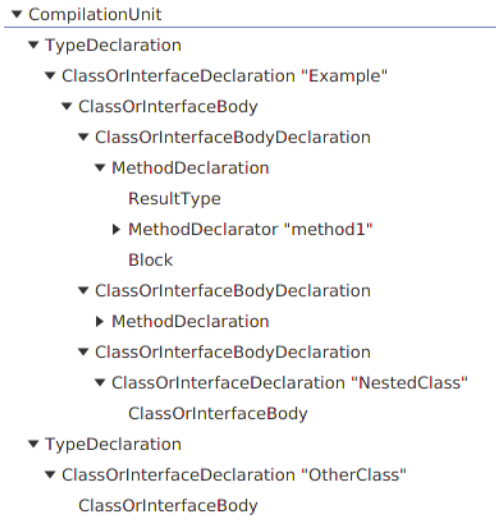


Figure 4: AST example

When utilising PMD for a project, there are a few different approaches possible. For greenfield projects, it's a no-brainer: PMD is active with a basic set of rules from the very beginning. So, every code, that is added, will be checked by PMD. For projects with an existing code base, the situation is most likely different. It can be overwhelming, if a whole bunch of rules are activated at once. You might be drowning in violations and it is not clear, which one to fix first. For this situation, an incremental approach is recommended: Prioritising and enabling one rule at a time.

Alternatively, all the selected rules can be enabled at once and the current number of violations are monitored. The goal is then, to reduce the violations with every commit and not introduce new violations. This however requires support from the build environment and is not possible with PMD alone. But it can

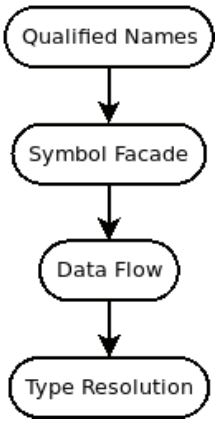


Figure 5: Visitors

be implemented using a quality gate in SonarQube.

PMD should be integrated into the development process as early as possible. The earlier PMD is used, the less issues need to be fixed later on. Therefore there are also IDE plugins that execute PMD while developing code. For Eclipse, there are today 3 different plugin implementations:

- The official pmd-eclipse plugin
- eclipse-pmd
- and the qa-eclipse-plugin.

For other IDEs and editors, there are plugins, too. For the full list, see the Tools

/ Integrations documentation page. Especially if your project is using Apache Maven as the build tool and you are using Eclipse, you should have a look at m2e-code-quality plugins, which transform the configuration from your Maven project files and make them available for the PMD, Checkstyle and Findbugs plugins in Eclipse. This means, you can configure your code quality tooling within your build tool and it is automatically working in Eclipse.

To compile, build and package software projects, usually build tools are used, such as Apache Maven, Gradle or Ant. For Ant, PMD provides an own task, that can be used. For the other build tools, plugins are existing, that can execute PMD. And most importantly: these plugins can fail the build, acting as a simple gate keeper. The Maven PMD Plugin can create a report for the project site and also contains a check goal, to fail the build, if PMD rules are violated. It also supports CPD, the copy paste detector.

All the previous tools are good, if you are building the project locally. But if a whole team is working on the project together, there is usually a central continuous integration server. Basically, such CI servers could just execute the build tool with its configuration for PMD, but they often provide a little bit more support for code quality tools like PMD: Since they regularly build the project and can keep a history, they allow to compare the reports generated by PMD from build to build. This allows you to see the development of the code quality over time like new introduced violations or violations that are resolved. For Jenkins, there is a PMD Plugin available, which produces a simple graph of violations.

Nowadays, such CI servers are available as a service, too. Especially for open source projects they are often free to use. PMD itself uses e.g. Travis CI. GitHub as a code hosting platform provides integrations with various 3rd party services, that can be enabled. Two such services already use PMD to offer their service: Code Climate and Codacy. These services can also be integrated for verifying pull requests to get early feedback. Since these service also create a history, you can see the results over time.

PMD provides many different built-in rules. Since PMD 6, these rules are organised into 8 categories: Best Practices, Code Style, Design, Documentation, Error Prone, Multithreading, Performance, and Security. The recommended approach is, to create an own ruleset, which references the rules that should be used for the specific project. This ruleset should be part of the project, so that it can be easily shared between developers and build tools. For Maven projects, often an extra module with the name “build-tools” is created, which can be used as a dependency. This is described in the Multimodule Configuration for the maven-pmd-plugin.

You might also find yourself in a situation, that you need a very specific rule, which is not available in PMD itself. Since it is very specific to your project, it might not be even useful outside of your project. Therefore you can define own rules, and the code for these custom rules naturally goes into the “build-tools” module as well.

The ruleset can also contain project wide file exclusion patterns, e.g. if you don’t want to analyse generated code.

While referencing the existing rules in your ruleset, you can configure them exactly to your needs. Many rules can be easily customised via properties. The rules also define the message, that appears in the report, if a violation is detected. This message can also be overridden and customised. A typical customisation is the priority. You can give each rule a specific priority and during the build, you can decide to fail the build because of an important rule violation but ignore other rules. You can also add own rules. See Figure 6 for an example of a custom ruleset.

FEATURES

It’s now time to look at a few selected

```
<?xml version="1.0"?>

<ruleset name="Custom RuleSet"
    xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 http://pmd.sourceforge.net/ruleset_2_0_0.xsd">

    <description>
Custom RuleSet
    </description>

    <!-- Just use the rule DuplicateImports as is -->
    <rule ref="category/java/codestyle.xml/DuplicateImports"/>

    <!-- Use the rule EmptyCatchBlock, but raise the priority -->
    <rule ref="category/java/errorprone.xml/EmptyCatchBlock">
        <priority>1</priority> <!-- this is high priority -->
    </rule>

    <!-- Use the rule CyclomaticComplexity and configure the thresholds -->
    <rule ref="category/java/design.xml/CyclomaticComplexity">
        <properties>
            <property name="classReportLevel" value="40"/>
            <property name="methodReportLevel" value="5"/>
        </properties>
    </rule>

     <rule name="AvoidPublicField"
           language="java"
           message="Fields, that are not constants, should be private"
           class="net.sourceforge.pmd.lang.rule.XPathRule">
         <description>
     Data encapsulation would be broken, if fields are accessible from outside. Declare the field as private and use getters and setters to allow access if needed.
         </description>
         <priority>2</priority>
         <properties>
             <property name="xpath">
                 <value>
 <![CDATA[
 //FieldDeclaration[@Public='true'][@Static='false'][@Final='false']
 ]]>
                 </value>
             </property>
         </properties>
     </rule>
 </ruleset>
```

Figure 6: Example of a custom ruleset

features, that PMD provides. The first feature is the support for XPath based rules. Since the AST is a tree structure, it can be dealt with like a XML document. The document can then be queried using XPath expressions, to find nodes within the AST, that match certain criteria. This provides an alternative API to develop rules, if you don't want to implement a rule using the visitor pattern to traverse the AST. This is a very convenient way to create ad-hoc rules. There is even a graphical rule designer to make it easier to develop XPath rules. The designer shows the parsed AST and executes a given XPath query. You can see the matched nodes directly. In the end, the developed XPath expression can be exported as a custom PMD rule in XML format, that you can add to your own ruleset. Since the rule designer displays the AST, it is also a valueable tool for developing rules in Java using the visitor pattern. See Figure 7 for a screenshot of the designer. This way of providing access to the AST and reuse XPath to write custom rules is a unique feature of PMD, that does not exist in other static code analysers.

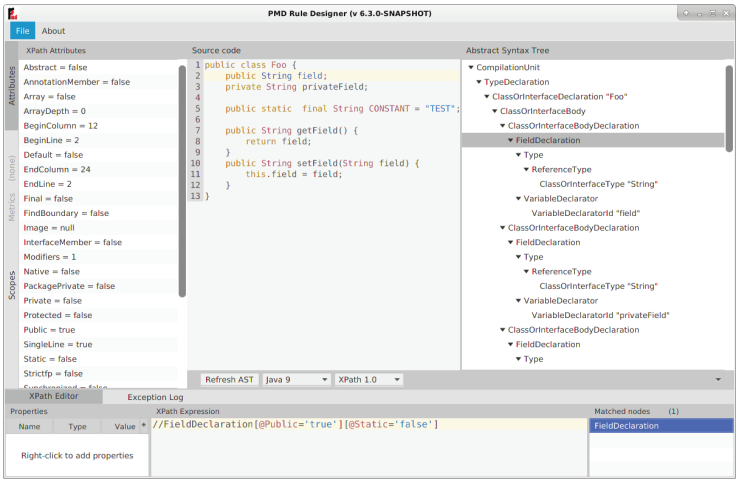


Figure 7: PMD Designer

Another feature of PMD is the so called type resolution. As explained above, type resolution happens as an extra step after parsing the source code. The goal is, that the AST is enriched with concrete type information whenever possible. Consider the following source code:

```
import org.slf4j.Logger;

public class Example {
    private static final Logger LOG = Logger.getLogger(Example.class);

    public void someMethod(String arg) {
        LOG.debug("This is a debug logging message: " + arg);
    }
}
```

Via type resolution, the field declaration for `LOG` is assigned the type `Logger`, which (through the import) is identified as `org.slf4j.Logger`. If the library "slf4j-api" is on the auxiliary classpath, then PMD can attach a concrete instance of `Class<org.slf4j.Logger>` to that node in the AST and the rule can access it. The rule can now first verify, that this field really is a logger, instead of simply relying on naming conventions of the field name or the simple class name. This helps greatly to reduce false positives for rule violation detection. In the example code snippet, PMD is correct to suggest to use the slf4j placeholder syntax (`"... message: {}", arg`), but PMD would be wrong, if the logger would be of a different type. Since the rule has access to the concrete class instance, it can even use reflection to gather more information as needed. This type resolution does not only work for 3rd party libraries, but in the same way it works within the same project, that is being analysed by PMD. That's why it is necessary, that the project is compiled first before PMD is executed. This means that references to other classes within the same project are resolved exactly the same way and the concrete class instances are made available.

There are a couple of rules, that make use of type resolution. And more rules will make use in the future, since type resolution is enabled by default for new Java rules. For example, the rule "LooseCoupling" finds usages of concrete collection implementations which should be replaced by the collection interface (e.g. use `List<>` instead of `ArrayList<>`). The fairly new rule "MissingOverride" actually uses type resolution and reflection to figure out, which methods are overriding methods from the super class and are missing a `@Override` annotation.

Type resolution has been available for a long time now in PMD. However, it is still under development. There are currently limitations for determining the types of method parameters, especially when overloading is in use and generics come into play.

The next feature is quite new: Metrics. It was added in 2017 during a Google Summer of Code project and provides a clean access to metrics of the analysed source code.

The metrics are e.g. access to foreign data (ATFD) or weighted method count (WMC). There are more metrics available already and the whole framework is usable by other languages, too. The metrics can be accessed by Java rules as well as by XPath rules. In the easiest case, these metrics can be used to detect overly complex or big classes, such as in the rule "CyclomaticComplexity". Multiple metrics can be combined to implement various code smell detectors such as "GodClass".

The next step in this area is to support multi file analysis. Currently, PMD looks only at one file, but for metrics it would be interesting to relate certain numbers of one class against, e.g., the total number of classes within the project. There are also benefits for the symbol table, if it has a view of the whole project. This will then allow to do full type resolution. Each rule has then access to all information which makes the rules more robust to false positives and also allows to find otherwise ignored special cases. Implementing this involves sharing data between the different file analysers - possibly involving an additional processing stage. The challenge is of course, to provide this functionality and not affecting the performance of the analysis negatively.

BEYOND JAVA

PMD started as a static code analyser just for the Java programming language only. This was the status for PMD version up to and including 4.3 (except for a little support for JSP). With PMD 5, a big refactoring took place, in order to support multiple languages. And with the initial release of PMD 5, three new languages were included: JSP, JavaScript (aka. ecmascript) and XML. Later on, support for PLSQL and the templating language Apache Velocity has been added while keeping the Java support up to date. The last big addition was support for Salesforce.com Apex.

Now, PMD supports in total 10 different languages including rules. Most rules are for Java, of course. Adding a new language takes quite some effort, but it is described in the step-by-step guide "Adding a new language". It involves in-

tegrating the language specific parser, mapping the language AST to the generic PMD interface types and last, but not least, writing new rules. Most of the PMD framework can be reused, so you'll immediately benefit from the possibility, to write XPath based rules for your language. The Copy-Paste-Detector (CPD) on the other hand supports many more languages. This is, because you only need to support a language specific tokeniser, which is much simpler than a full language grammar with productions. PMD provides even a "AnyLanguage" for CPD, which basically tokenises the source code at whitespaces. Language specific support is needed to improve the results of CPD, e.g. correctly identifying keywords and statement separators. With more effort, there is also the possibility to ignore identifier names during copy-paste-detection. This allows then to find duplicated code, which only differs in variable names, but is otherwise structurally the same. This feature however is only available for Java at the moment.

THE PROJECT

The following is a summary of the history of PMD that Tom Copeland wrote in the book "PMD Applied. An Easy-To-Use Guide for Developers". It covers the years 2002 till 2005.

The project PMD was started in Summer 2002. The original founders are David Dixon-Peugh, David Craine and Tom Copeland. The goal was to replace a commercial code checker, which these three guys were using in a government project in the US. They decided to write their own code checker and got approval to open source it. Now PMD was living on SourceForge. In November 2002, PMD version 1.0 was released with already 39 rules and a copy/paste detector. In March 2003, thanks to Dan Sheppard, XPath rules were introduced with PMD 1.04. Since PMD 1.3 (October 2003), the BSD license is used, which helped a lot to adopt it. Since then it has been integrated into many products.

The copy/paste detector has been rewritten a couple of times and improved in performance. With every release of PMD, new rules or report formats have been added and existing rules fixed. With PMD 2.0 (October 2004) the data flow analysis component has been added. With PMD 3.0 (March 2005) support for Java 1.5 was added.

Java 1.6 was added with PMD 3.9 (December 2006), Java 1.7 with PMD 4.3 (November 2011), Java 8 with PMD 5.1.0 (February 2014), Java 9 with PMD 6.0.0 (December 2017), Java 10 with PMD 6.4.0 (May 2018), Java 11 with PMD 6.6.0 (July 2018), and Java 12 with PMD 6.13.0 (March 2019).

A big step happened between PMD 4 and 5: A major refactoring took place in order to properly support rules for multiple languages. This introduced many breaking API changes and was released in 2012. Also with PMD 5, Apache Maven is being used as the primary build tool instead of Ant. Support for PLSQL was added in February 2014 with PMD 5.1.0. With PMD 5.2.0 (October 2014) the code was completely modularised into a core module and several language modules. This made it easier to add new languages. With PMD 5.5.0 (June 2016) Salesforce.com Apex has been added. With PMD 6.0.0 another small, but important refactoring took place. It has unfortunately a bigger impact on end users: All the rules have been categorised, so that they are easier to find. They have been moved into different rulesets. However, we are keeping the old rulesets for backwards compatibility, so that the existing custom rulesets still continue to work.

Over the last years, the project gradually moved more and more infrastructure from SourceForge towards GitHub. The complete subversion repository has been converted to git. It contains the full history back to the year 2002. While at the beginning every sub-project was in the same repository, we have now several separate repositories, e.g. for the eclipse plugin or other extensions.

The move to GitHub was a step forward in terms of presence and attracting new contributors. The GitHub web interface is more user friendly, easier to use and feels faster than SourceForge. GitHub especially encourages contributions through the concept of pull requests. GitHub is now the primary location for the source code and the issue tracker. On SourceForge, we still have the mailing list running and a webspace and the archive of old releases. There are other services PMD uses, e.g. travis-ci as a build server. It builds every push and deploys the snapshot via the OSS Repository Hosting service by Sonatype. For releases, this build server is even able to deploy the final artifacts directly to Maven Central.

Also, every pull request is built automatically. Other services are e.g. coveralls for test coverage and BinTray for hosting the eclipse plugin update site.

In 2017, PMD participated the first time in Google Summer of Code. This is a student stipend program offered by Google. Students all around the world have the opportunity to work during semester break on various open source projects. Open source organisations provide projects and mentors and the students apply for a project with a proposal. In 2017 two students worked on type resolution and metrics. In 2018 PMD is participating again.
As of today, the project has 3 active maintainers, about 100 different contributors, 500 merged pull requests. According to cloc it contains about 100k Java lines of code, surprisingly 88k XML LOC (which probably are the test cases) and many other types.

THE FUTURE

What's left to do for PMD? Aside from keeping the support for Java and other languages up to date and fixing bugs, adding new rules, adjusting additional rules, there are a few topics, that sound promising. In order to lower the barrier of using PMD, specialised rulesets might be useful.

There could be a "Getting Started" ruleset, that has just enough generic rules, that are useful for any project. This might be the default ruleset and could be a template for creating an own customised tailored ruleset for the project. There could also be use-case based rulesets, the group the rules not by category but by another topic, e.g. Unit testing, Logging, Migration of library usages, Android specific patterns.

Another interesting feature is autofixes. Since PMD has the knowledge, where a violation exactly is in the source code, it is for some rules trivial to provide a fix. The goal is, that PMD provides directly the fixed source code, that can be confirmed in a IDE plugin and applied automatically. Then, besides type resolution, which is still not completely finished, there is also the data flow analysis (DFA) part. PMD has a good start for the DFA, but it's still very limited. A related feature is control flow analysis. With that available, rules could be written which can detect unused code.

Or rules, that verify that a specific guarding method must be called before another method. Having the call stack available, would make this possible to verify. This requires, similar to the mentioned multi file analysis, an overview of the complete project that is being analysed.

And last, but not least, a possible future feature could be cross language support. Since PMD already supports multiple languages, this would put multi-language support onto the next level: Some languages allow to embed other languages, e.g. JavaScript inside HTML, or PHP+HTML+JavaScript. Or there is Salesforce.com VisualForce with Lightning.

When and if these features are implemented is unknown. The project is driven by volunteers and contributors and all this depends on the available time. New contributors are always welcome to work together and make PMD

WANT TO FIND OUT MORE?

Go visit https://pmd.github.io

ATLASSIAN SUITE:

tools for every team and more agility in projects

A report on how EPOS CAT GmbH uses the collaboration tools of the Australian software manufacturer Atlassian in the automotive industry and what experience it has gained in the course of the years.

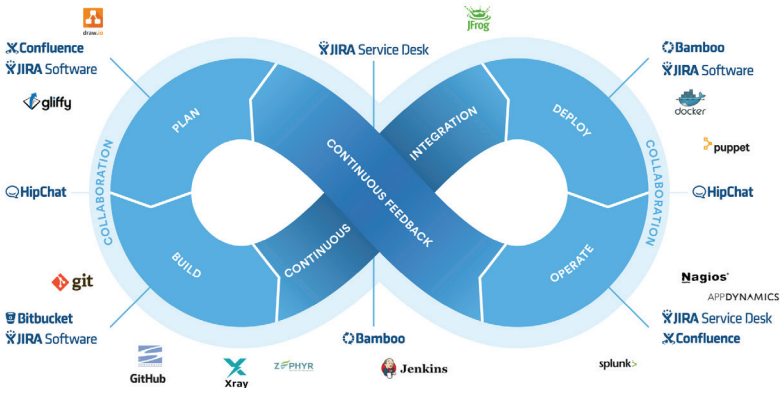
BY: Alexander Sowatsch
Solutions Architect,
EPOS CAT GmbH

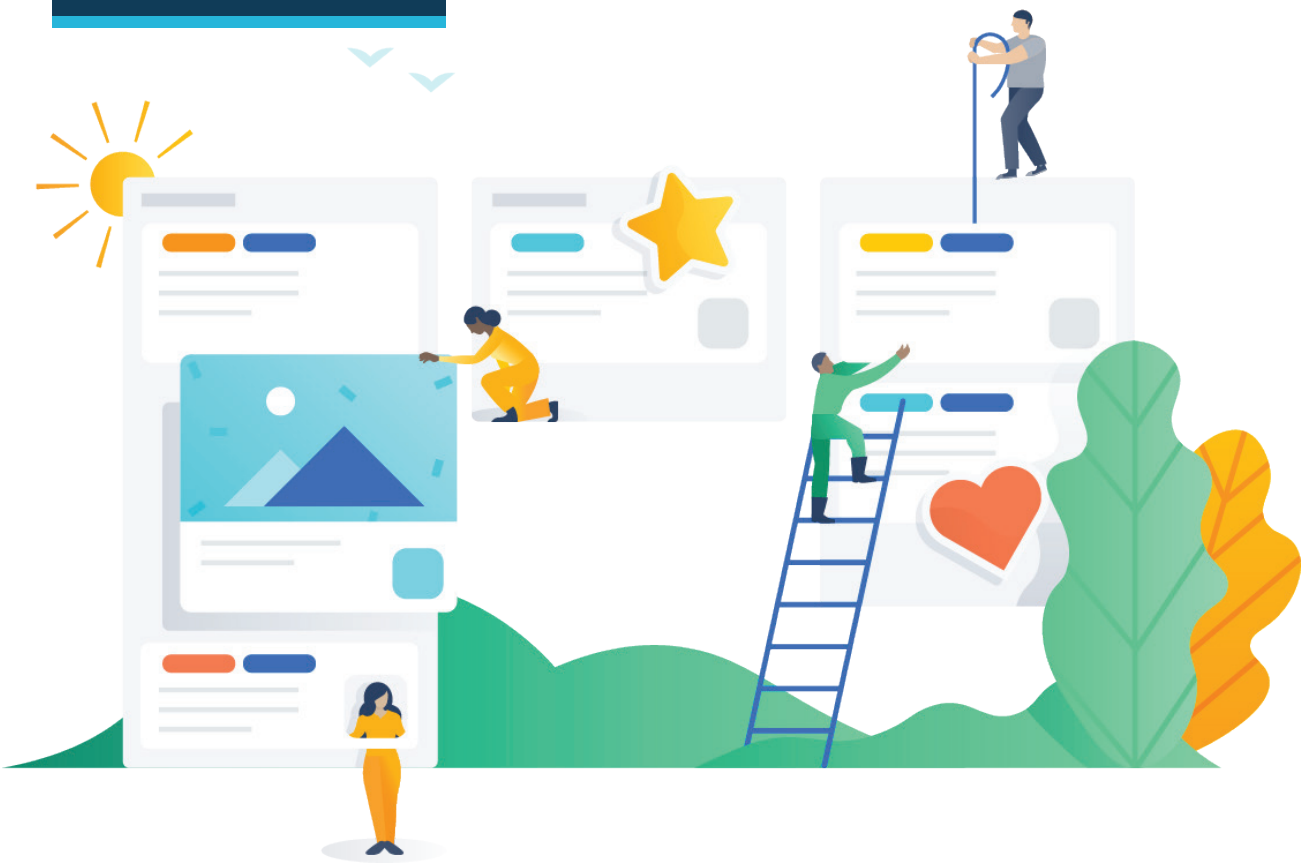
Compared with 20 years ago, we are now seeing entirely different business models emerge, developed by quick start-ups which turn new ideas into marketable products with astonishing speed. Organisations benefitting most from this success are those that are versatile, flexible or agile enough to be capable of uncompromising customer orientation. A start-up is usually established by the decision makers from scratch, allowing them extensive creative freedom. But what about the many organisations whose decision-makers see the need for agility, but whose structures and culture have developed far from the market?

With a collaboration team of about ten people, EPOS CAT GmbH looks after both such customer groups, and over the last few years has implemented a wide variety of projects in which extensive experience has been gained.

UNDERSTANDING USERS AND REQUIREMENTS

For the most part, software developers do not need to be told about Atlassian tools and their benefits - they generally use them as a matter of course and without major difficulties. However, for business teams in the fields of design, IT service management (ITSM) or human resources the implementation requires more explanation and is more challenging. And even here there is a strong culture gap. While young employees often demand up-to-date work tools - and are thus often the drivers behind the introduction of Jira or Confluence - older employees are frequently afraid of, or have at least strong reservations about these modern tools. Functions we know and are familiar with from social media, which enable the mentioning of colleagues in a text or the sharing of articles without sending an e-mail, require explanation. Needless to say, project teams which are forced to use collaborative tools are doomed to fail. The lack of interest or acceptance, and in the worst case a boycott, all prevent progress. This applies more than ever to the introduction of new working procedures. We have thus found that the success of a project depends entirely on taking in the entire team, with all its different roles and diverse (cultural) prerequisites, and on arousing enthusiasm in individual cases for the implementation. After all, the product owner's requirements for an application differ from those of the colleague on the support desk, or the tester. The Atlassian suite has the right tool, specific extensions as well as plug-ins for all parties.





KNOWLEDGE MANAGEMENT WITH CONFLUENCE

In our projects, Confluence has proven itself as an entry point. Depending on the requirements of the project, additional tools from the suite complement the enterprise wiki. Easy to use, it can replace Word and, thanks to its central location, complement or in some cases replace the classic e-mail. The collaboration team at EPOS uses this wiki for working together on documents, creating and maintaining a manual, setting up FAQs or drafting offers and exporting them in one click as a PDF; these are just a few examples of the diverse application possibilities.

PROJECT ORGANISATION WITH JIRA

According to Atlassian, Jira is the most widely used project management add-on for Confluence users. While it is well-suited to agile processes in software development, Jira is more of an organisational tool to most business teams. The tool lends itself to restrictive processes that require traceability and

> One of our team members has the sole task of training users and adjusting configurations. In order to be able to keep this technically sophisticated system up to date, an internal billing model has been established by our customer. The research and development departments therefore use their Jira in their corporation as a paid service.

It may sound simple at first, but there are still hidden risks. Therefore, the EPOS team has created an overview with several rules for successful project implementation. For example, new accounts require new passwords or an authorisation structure that ensures that only your own team members have access to specific areas. Once again it is important right from the start to reduce inhibitions and minimise hurdles. Why not ask new colleagues to introduce themselves to the team in the wiki instead of writing a portrait for the intranet? Or encourage new colleagues to organise their entire training on Confluence in order to become familiar with the tool and its features.

transparency or that need to be evaluated; budget approvals or decision-making processes are examples of possible applications.

It is also obvious that tools are only put to everyday use if they are simple to operate and their functions are mastered. Accordingly, training and coaching as well as the subsequent support of the user are important. The training courses are mostly short standardised introductions to convey the features by demonstrating best practices and use cases. In addition to these basic training courses, there is also individually tailored coaching on Scrum, Kanban and plug-ins for software development and test management.

PRACTICAL EXPERIENCE

That was the human side of things. Now let us turn to our experiences with the methodological and technical requirements and hurdles. Last year our collaboration team had the task of equipping a young company specialised in autonomous driving with the complete Atlassian suite. The team consists of approximately 200 people and wants to use the tools to develop software for autonomous driving from layers 1-5. From the beginning, the applications were organised uniformly, with no configurations nor customising. Jira was set up with exactly three different project types with remarkably simple authorisation structures, so that each team can look at

everything, apart from areas with explicit data protection restrictions. It is kept quite simple and is thus very sustainable, as it can be maintained cyclically.
The use of Atlassian applications has also proven itself in companies and corporations that have grown over the decades or which have a hierarchical organisation, as the next project example shows. For almost ten years, EPOS has been operating a Jira in automotive R & D. It was introduced in version 3.14 and is currently maintained by our experts in version 7.3. By March 2018, it had around 600,000 tickets with 1.7 million comments and around 4,000 active users per month. This Jira is controlled centrally and, as a corporate service, is designed so that project-specific configurations can take place. The ratio of projects by business teams and software development is estimated at 80% to 20% and therefore there are numerous project workflows which map out request evaluation processes or approval processes for the commissioning of external service providers. One of our team members has the sole task of training users and adjusting configurations. In order to be able to keep this technically sophisticated system up to date, an internal billing model has been established by our customer. The research and development departments therefore use their Jira in their corporation as a paid service. This makes it possible to finance lifecycle, support or documentation and to offer a technically and methodically clean system. Financing through an internal billing model has proven itself many times and is therefore often standard in the enterprise environment. For strategically placed business services, it is particularly important that products are up-to-date, which is why we are principally concerned with keeping customising low. It pays off to take into account the follow-up costs for licenses right from the start of the project, and to stay as close to the product as possible and to keep up with product updates cyclically.

The dangers of customising include translations into your own language. In our experience this often leads to confusion and can also lead to amusing or misleading errors. We therefore urgently advise to use the tools in the English original. The wording can be found in the introductions, making it easier to seek help from Support or further workshops. The use of specialist terminology ensures that all team members worldwide can communicate in a consistent and comprehensible way.

After ten years of Atlassian projects, our experience in short is quite simple: take the whole project team through training and continuous support, motivate them in the individual roles for using the new tools to assist them in their daily work, and keep the applications technically and methodically simple and lean.

The EPOS team would be pleased to hear about other experiences and to answer your questions. We will be at the next Atlassian Camp in autumn in Vienna. If Ingolstadt or Spain are not on your travel plan, local user groups are a good alternative. In German speaking countries, especially in Germany, there are Atlassian user groups in many cities online at https://aug.atlassian.com/. In Scandinavia there are also local groups and partners with great expertise in Oslo, Copenhagen or Stockholm.

This is EPOS CAT

A leading consulting and service company for automotive IT and computer aided testing (CAT). EPOS CAT designs, develops and operates tailor-made software solutions to support and optimize customer's business processes mainly targeting the automotive industry.

Modern vehicles contain increasingly complex IT system driving demand for software development, test and technical support to comply with strict industry safety regulations. Measurement and test systems represent a significant cost factor in vehicle development and quality assurance. Their proprietary "computer aided testing" (CAT) software solution supports customers in managing the ever shorter product development cycles in an efficient and secure way. The company's engineers are located close to customers to secure efficient development, rapid respond support and evolve their industry know-how.

www.epos-cat.de

BRINGING THE
TO THE INTERNET

All is not what you think when IoT devices get connected: One of the more curious things about IoT devices is that most of them are not actually a part of what we today loosely describe as the “Internet”. The “Internet” is usually defined as a global network consisting of TCP (Transmission Control Protocol) combined with protocols like IPv4 and IPv6, and various other supporting protocols. Despite the name “Internet Of Things”, IoT devices rarely have support for any of these protocols, making a direct link to (or from) them impossible. Based on the idea that the Internet Protocol could and should be applied even to the smallest devices, 6LowPan is now entering the arena. This article offers a look into the inner workings of 6LowPan, together with assessing the current state of adoption of the standard. And although many difficulties lie ahead, one thing is for sure: 6LowPan over Bluetooth is not standing still.

INTERNET OF THINGS

- the current state of 6LowPan connecting Bluetooth IoT devices to the internet

BY: Peter Dons Tychsen, TechPeople

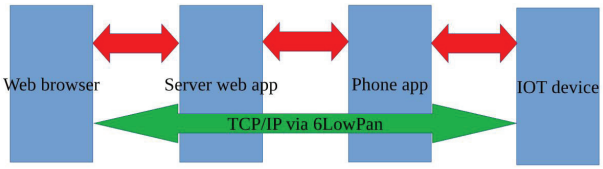
GATEWAYS PROVIDE CONNECTIVITY

Instead of a direct link, a steady stream of proprietary gateways of many shapes and forms are used to connect to IoT devices or to collect data from them. This holds true for low-power radio systems like Bluetooth LE, ZigBee, Z-wave and similar technologies. This is in contrast to other technologies like WiFi (IEEE 802.11) which are often accompanied by a TCP/IP stack, which can make them a part of a local or global network.

The problem with the current approach is that you need proprietary software on an intermediate device like a phone, server or router to be able to talk to your IoT device. This makes sense if your device is one device of many in a larger farm of devices, where only the combined data from all the devices is of interest. However, if your device is a low-power, self-contained device, you might want to simply talk to the device directly. That way you can avoid all the other data pit-stops, which makes your product more complex and requires constant maintaining. Keeping an IOS, Android or Linux server applications working over time is far from free, and can lead to unnecessary complexity. Security can also become more tricky, and peer-to-peer security can be harder to achieve, as all the pit-stops might need some kind of awareness of the data being transmitted. Even if this is not the case, it would be optimal, if existing and proven security protocols like “TSL” and “SSL” (which are used for e.g. secure web browsing) could be used directly to connect to the device over IPv6. Bluetooth LE is a fairly young technology, while the fundamentals that make up for the security on e.g. the Internet are old, mature and well proven.

BLUETOOTH IOT, SIMPLIFIED

So what would a typical example look like in the real world? Let's take an example, where a user wants to query a specific Bluetooth IoT device for some data via a browser. To do this currently it would involve a web-server to process the requests and a proprietary application on e.g. a mobile phone which has a Bluetooth link to the device. But what if the IoT device was the web server? This would eliminate the need for an external web server all together, and would reduce the mobile phone to a simple TCP/IP router with no knowledge of the device's application or the data. You could also keep the web server, and only eliminate the phone application, if the server needed to process the data.



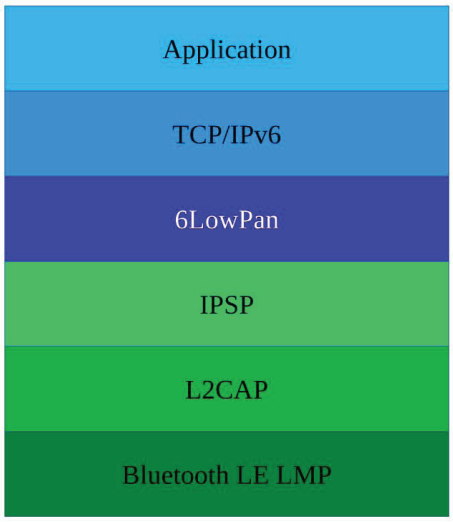
This typology is something we already take for granted, when it comes to other technologies. People would e.g. find it very odd, if they needed a special app on their WiFi router for their TV to work. Normally you would assume that any WiFi device that is connected directly to your home network is on the "internet". This is not empirically true, but merely a result of standards and protocols having evolved this way. To help Bluetooth and other protocols evolve to this step, 6LowPan enters the arena.

6LowPan stands for "Low-Power Wireless Personal Area Networks", and is like other internet standards defined by the "Internet Engineering Task Force", via the "RFC 6282" document, and further defines "RFC 7668", which details how specifically TCP/IPv6 is mapped on top of Bluetooth Low Energy with regards to addressing and such.

Further, to incorporate it into to world of Bluetooth, bluetooth.org defines the "IPSP" standard, which maps the 6LowPan protocol into L2CAP on a fixed port number. With all this in place, the IoT device can now talk directly to a TCP/IPv6 network, such as the "Internet" as we know it.

WHAT PROTOCOLS MAKE THE MAGIC WORK?

Combining all the mentioned standards and protocols makes up a complete protocol stack from application to physical transport. The protocol stack ends up looking like this:



To avoid having to introduce new addresses to the system, the Bluetooth address (which is a 6 byte MAC address issued by IANA) is mapped directly into a reserved range of IPv6 addresses, which will therefor not collide with any existing IPv6 addresses. This is a clever trick, as this means that you can deduct the address of your remote IPv6 device, simply by knowing its Bluetooth address.

In the end the whole point of this is the application. This means that existing TCP/IPv6 applications and protocols can now be placed directly on top of the protocol stack. This could give life to older TCP based protocols like "FTP" and "Telnet", which are fairly bandwidth effective, but could also be used with newer protocols like "HTTP".

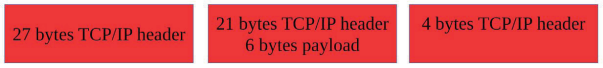
Looking at the entire protocol stack, one might deduct that this is all a bit too much for a small microcontroller to handle, as this requires processing of additional packet check sums and protocol handling. However, in practice, an efficient and minimal implementation of this can be done in a couple of kilobytes of code on a microcontroller like the ARM Cortex M0, which is an entry-level controller at the time of writing. It goes without saying that for high data throughput, larger controllers will be needed. Bluetooth 5 introduces new, faster packet types that can hold more data and transmit faster.

MAKING LARGE PACKETS FIT INSIDE SMALL CONTAINERS

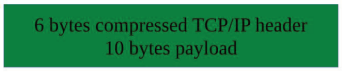
One of the largest challenges of using TCP/IPv6 on devices with limited resources is the fact that IPv6 was initially designed for large systems with large payload sizes. An original IPv6 and TCP header is usually around 48 bytes. As the hardcoded payload area for a Bluetooth LE can be as low as 27 bytes, this is a challenge. Furthermore, you would not like to see all your battery power and bandwidth go up in smoke because of excessive headers. To remedy this, 6LowPan utilises something called "Header Compression" to vastly reduce the size of the problematic headers. The compression of 6LowPan datagrams is specified in standards such as "RFC 6282".

One of the most significant compression mechanisms is the "LOWPAN IPHC" which defines how IPv6 headers can be reduced to a few number of bytes by using some carefully defined logic that causes much of the information to become implicit. Other schemes define how TCP/UDP and other internet protocols can be compressed in a similar fashion.

If you have a bandwidth limited link, this will make a world of difference. For a 10 byte payload sent inside a 27 byte MTU (Max Transmission Unit) physical link, it would look like this without any compression:



Via header compression this can be reduced to:



Great Scott! This means 3 times less power, and 3 times less bandwidth is used. That is a significant reduction and makes the technology much more appealing.

The things that are compressed are usually addresses and port numbers. As an example, because the Bluetooth address is mapped directly into the IPv6 address, the remote address can be deducted to zero, if the receiver of the IPv6 packet is the same entity as the device receiving the Bluetooth LE packet. In a similar fashion, other fields can be eluded by using implicit knowledge to deduct the values.

WITH GREAT PROTOCOLS COMES GREAT POWER

One thing to always worry about is power. Contrary to the name, Bluetooth Low Energy is not very low energy if you are sending a lot of data frequently. So you really want to keep your IoT device in a sleeping state as much as possible. One way to do this is simply to apply reasonable values for Bluetooth LE related parameters such as the "connection interval", which limits the number of slots used for detecting incoming traffic.

But when IPv6 is introduced into the mix, a new problem arises. Apart from the information you want to send, the gateway itself could be sending vast amounts of broadcast information, which is not necessarily useful for your device. As an example, many networks utilise discovery protocols to detect which devices and what services are available on the network, such as the IPv6 "neighborhood discovery".

To battle one of these problems the "RFC 5741" introduces optimisations to what "neighborhood search" requests are sent to the 6LowPan enabled device. Another, more simple approach, is to simply make sure that the network devices on your network are only setup to route the most necessary packets to the device. This can typically be controlled via firewall and routing rules. However, as the gateway might be an off-the-shelf phone, that might not be an option.

Power consumption will always be in focus, and 6LowPan will not change that. However, we might see new challenges in existing protocols along the transition towards IPv6 (if we are going that way), as many of them were simply not designed with such power critical devices in mind. However, standard extensions like "RFC 5741" seems to ratify this rather nicely. And it will likely not be the last.

PLAYING AROUND WITH 6LOWPAN IN THE PENGUIN'S SANDBOX

When 6LowPan was developed, an open implementation was needed to test and verify it with. As Linux is widely accepted as the de facto standard for embedded open source development, implementing 6LowPan as a module for Linux was obvious. So since kernel 3.17 (from 2014) this has been possible to some extent. As the standard evolved, so did the kernel module, so a newer version of the kernel is recommended. Additionally, the kernel module will need to be enabled, and the documentation on how to do this is available several places online. Searching for the keywords "Linux" "Bluetooth" and "6LowPan" should get you started.

For Embedded or Windows usage, a proprietary stack is needed, as Windows does not currently support 6LowPan out of the box. This can be obtained from various sources, including the author's own "DTBT – DonTech BlueTooth", which has support for 6LowPan on Windows, MacOS and embedded targets. Some embedded targets, like Texas Instruments 2630, natively support 6LowPan and IPv6.

To help the developers along, the sniffers and packet tracers used for Bluetooth and networks have adopted support for these new protocols as well. Sniffers from Wireshark, Frontline and Ellisys have gained support for parts of the 6LowPan stack. Below is an example trace of the classical "IP ping" being sent from a Linux device to a Bluetooth connected Windows implementation with DTBT:

WHO IS ON THE 6LOWPAN TRAIN, AND WHERE IS IT GOING?

There are many backers of the 6LowPan standard, including the biggest producers of Bluetooth controllers such as "Texas Instruments" and "Nordic Semiconductor". Other companies are also pushing 6LowPan for other low-power network types. The thing they have in common is that they are chipset manufacturers and not end product manufacturers. The biggest challenge with the 6LowPan is not the standard, the producers of chipsets or the specific implementations. The big hurdle is adoption.

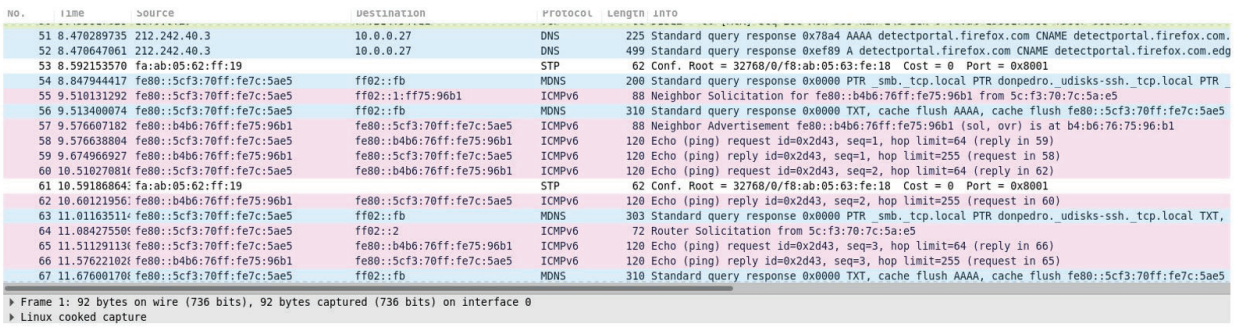
IoT is currently a very hot subject, but various companies want to make their own IoT ecosystem, where they can sell more of their own products. This is a classical problem. In the early days of the Internet, many companies like Microsoft, AOL and CompuServe tried to push their own version of the "Internet", so they would have complete control over devices and services sold on it. Today, people would hardly accept a solution where e.g. your TV only worked with a specific Internet connection.

Companies like Apple and Google still seem to be interested in this philosophy. On one hand they present things as an open-ish platform, so other manufacturers can participate, but on the other hand they do not wish to employ open standards to accomplish their goals. They still want control, and this bait-and-switch tactic can be something that keeps consumers and companies from fully embracing the technology. However, if history is to repeat itself, technologies like 6LowPan could change all that.

So for now, there is no 6LowPan on Android or IOS. Actually, Linux is the only platform supporting it out-of-the-box. As Android is based on Linux, they could of course suddenly announce its arrival, if they felt it did not get in the way of any of their own visions of deploying devices like "Google Home".

One thing is for sure, though. 6LowPan over Bluetooth is not standing still. The more recently released "Bluetooth Mesh Profile" for Bluetooth LE has already gotten a new sister RFC called "draft-ietf-6lo-blemesh-00". As the name indicates, it is still in draft. But the goal is to extend 6LowPan to cover mesh networks using Bluetooth. So companies that are looking into Bluetooth mesh might also want to take a look at topping if off with IPv6 connectivity.

Bluetooth™, Linux™, Windows™, IOS™ and MacOs™ are trademarks of their respective owners.

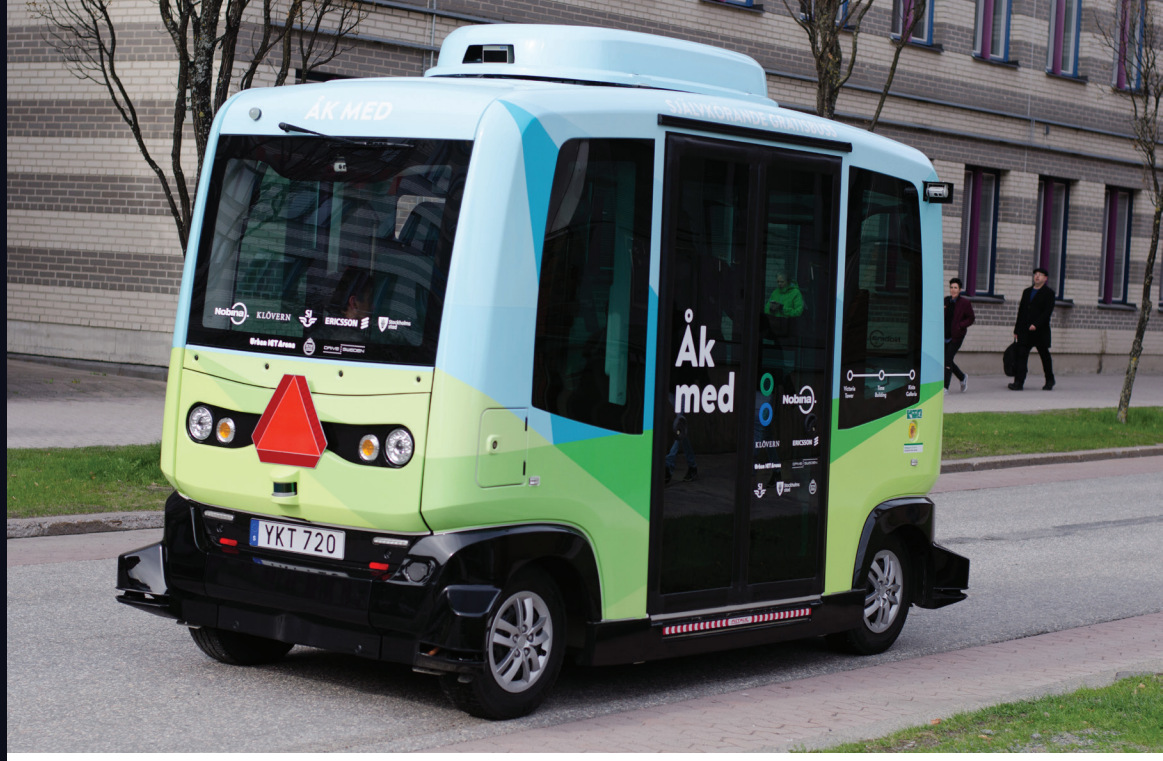


No.	Time	Source	Destination	Protocol	Length	Info
51	8.470289735	212.242.40.3	10.0.0.27	DNS	225	Standard query response 0x78a4 AAAA detectportal.firefox.com CNAME detectportal.firefox.com.
52	8.470647061	212.242.40.3	10.0.0.27	DNS	499	Standard query response 0xef89 A detectportal.firefox.com CNAME detectportal.firefox.com.edg
53	8.592153570	fa:ab:05:62:ff:19		STP	62	Conf. Root = 32768/0/f8:ab:05:63:fe:18 Cost = 0 Port = 0x8001
54	8.847944417	fe80::5cf3:70ff:fe7c:5ae5	ff02::fb	MDNS	200	Standard query response 0x0000 PTR _smb._tcp.local PTR donpedro._udisks-ssh._tcp.local PTR
55	9.510131292	fe80::5cf3:70ff:fe7c:5ae5	ff02::1:ff75:96b1	ICMPv6	88	Neighbor Solicitation for fe80::b4b6:76ff:fe75:96b1 from 5c:f3:70:7c:5a:e5
56	9.513400074	fe80::5cf3:70ff:fe7c:5ae5	ff02::fb	MDNS	310	Standard query response 0x0000 TXT, cache flush AAAA, cache flush fe80::5cf3:70ff:fe7c:5ae5
57	9.576607182	fe80::b4b6:76ff:fe75:96b1	fe80::5cf3:70ff:fe7c:5ae5	ICMPv6	88	Neighbor Advertisement fe80::b4b6:76ff:fe75:96b1 (sol, ovr) is at b4:b6:76:75:96:b1
58	9.576638804	fe80::5cf3:70ff:fe7c:5ae5	fe80::b4b6:76ff:fe75:96b1	ICMPv6	120	Echo (ping) request id=0x2d43, seq=1, hop limit=64 (reply in 59)
59	9.674966927	fe80::b4b6:76ff:fe75:96b1	fe80::5cf3:70ff:fe7c:5ae5	ICMPv6	120	Echo (ping) reply id=0x2d43, seq=1, hop limit=255 (request in 58)
60	10.510270816	fe80::5cf3:70ff:fe7c:5ae5	fe80::b4b6:76ff:fe75:96b1	ICMPv6	120	Echo (ping) request id=0x2d43, seq=2, hop limit=64 (reply in 62)
61	10.591868643	fa:ab:05:62:ff:19		STP	62	Conf. Root = 32768/0/f8:ab:05:63:fe:18 Cost = 0 Port = 0x8001
62	10.601219563	fe80::b4b6:76ff:fe75:96b1	fe80::5cf3:70ff:fe7c:5ae5	ICMPv6	120	Echo (ping) reply id=0x2d43, seq=2, hop limit=255 (request in 60)
63	11.011635114	fe80::5cf3:70ff:fe7c:5ae5	ff02::fb	MDNS	303	Standard query response 0x0000 PTR _smb._tcp.local PTR donpedro._udisks-ssh._tcp.local TXT,
64	11.084275509	fe80::5cf3:70ff:fe7c:5ae5	ff02::2	ICMPv6	72	Router Solicitation from 5c:f3:70:7c:5a:e5
65	11.511291136	fe80::5cf3:70ff:fe7c:5ae5	fe80::b4b6:76ff:fe75:96b1	ICMPv6	120	Echo (ping) request id=0x2d43, seq=3, hop limit=64 (reply in 66)
66	11.576221026	fe80::b4b6:76ff:fe75:96b1	fe80::5cf3:70ff:fe7c:5ae5	ICMPv6	120	Echo (ping) reply id=0x2d43, seq=3, hop limit=255 (request in 65)
67	11.676001706	fe80::5cf3:70ff:fe7c:5ae5	ff02::fb	MDNS	310	Standard query response 0x0000 TXT, cache flush AAAA, cache flush fe80::5cf3:70ff:fe7c:5ae5

▸ Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▸ Linux cooked capture

DATA LOGGING & AUTONOMOUS VEHICLES

For the vehicle industry, the expression ”Self-driving cars” has become a slogan which engages people who have not previously had the slightest interest in traditional automotive technologies. Everyone (almost) can relate to sitting in a car that drives you from one place to another without your involvement, some people imagine it with alarm others with gratifying enthusiasm.



Stockholm, Sweden, 4 may 2018: A Self-driving bus on the street in Stockholm. Rolling the street for a trial period of 6 months. (Image: iStock/JarlJ)

BY: Crister Nilson
Consultant Manager & Automotive B
usiness Area responsible
Sylog AB

THE ANSWER TO THE QUESTION:
What is self-driving car? Among the general public, there is no clear conception of what a self-driving car is. Within the vehicle industry there are 6 levels that define how self-driving a vehicle is.

Level 0, no automation:
The driver is in control of the vehicle's forward motion.

Level 1, driver assistance:
The driver is in complete control of the vehicle's forward motion, but can utilise certain assistive functions such as ABS and Cruise Control.

Level 2, partial automation:
At this level the driver can hand over control of the vehicle to the car's system in well selected scenarios, parking assistance for example. The driver is still responsible for taking over control in critical situations.

Level 3, conditional automation:
The driver can allow the vehicle's system to take over all safety critical elements, but the driver's attention is still necessary.

Level 4, high automation:
At this level the system can determine itself when it is safe to take over control of the vehicle and then do so. The system is not able to handle all dynamic situations that can arise, it then hands over control to the driver.

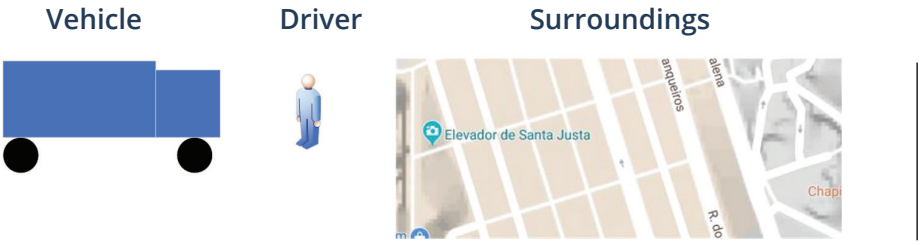
Level 5, full automation:
Requires no interaction with the driver in any situation.

ROADMAP
Today the majority of vehicle manufacturers have levels 1, 2 and 3 technology on the market. But the higher levels are not far away, Tesla, for example, has already launched self-driving vehicles at level 4. Most other vehicle manufacturers are aiming to offer level 4 vehicles between 2020 and 2025 and be able to offer vehicles at level 5 from 2025.

	Past		2018 - 2020		2020 - 2025	2025 - 2030
Level	0	1	2	3	4	5
	No automation	Driver assistance	Partial automation	Conditional automation	High automation	Full automation

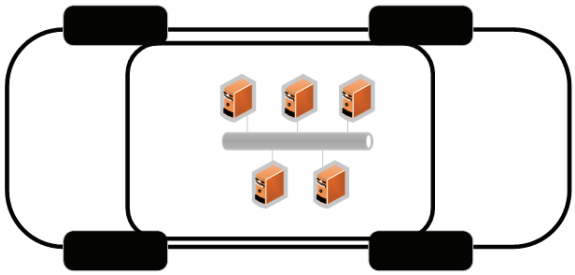
COMPONENTS

The sensor systems that are needed to achieve self-driving cars are usually divided into three main groups: camera, radar and lidar based systems. Both camera and radar systems are currently used on cars for levels 1 and 2. Subcomponents in these systems are also sufficiently advanced to be used for the higher levels. What is elegant about this is that it can be utilised to collect data to be analysed for the next level's autonomous functions.

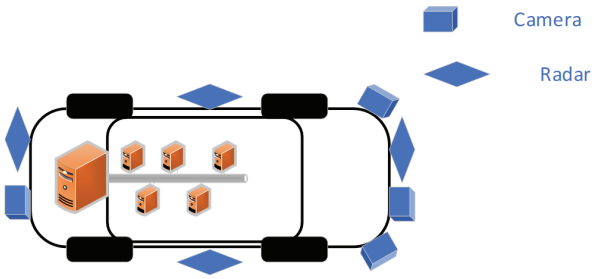


The vehicle's current components, which are usually connected in one or a number of CAN networks, will be supplemented by a completely new layer of components and networks which will replace the driver and the driver's choice of actions in different situations that can arise when driving.

The difference in number of sensors and processing power will increase markedly with each stage of automation. The traditional network with relatively simple computers is designed to manage the vehicle's transducers and sensors. All interaction with the surroundings requires a driver.



Replacing the driver in certain or all situations will require a sharp increase in new components and computing power. A camera and radar system generates a relatively large amount of data which has to be analysed in real time together with data from the vehicle's traditional systems.



To analyse camera and radar information, today's distributed computers must be supplemented with computers with advanced processors and greater memory capacity. The CAN bus does not have the capacity to handle the quantities of data, rather it is necessary to introduce elements such as an Ethernet. Similarly, logging of vehicle data must be developed.

DATA LOGGING

Traditionally the vehicle industry has logged CAN communication since it was introduced. The quantities of data have gradually increased from the first logging of J1935 at 1 Hz which requires relatively low memory capacity to logging of CCP/XCP with up to 1 kHz where the memory requirement has increased to at least Gbyte level.

The next step is the introduction of logging equipment, partly with traditional CAN connections,as well as with both Ethernet and analogue inputs for film and radar logging. Logging of image and radar data drastically increases the requirement for memory management. It is not just the actual size of the memory that has to be taken into account. The memory's capacity to store and upload in the shortest possible time is of major importance. To facilitate management of film and radar sequences, preprepared formats can be used for storage instead of logging the analogue flows. It will also facilitate time stamping of events that are logged where, for example, an alarm signal on the CAN bus can be analysed together with a film sequence. The major challenge will be to analyse all data that is logged and to use data from logging effectively in, for example, simulations in the lab for future autonomous functions.

RAISING THE LEVEL OF AUTOMATION

A prerequisite to raise the level of automation, with all the attendant complexity, is to have and be able to analyse data collected. The data consists of a multitude of different situations and types of data collected from the vehicle, the surroundings and driver interaction or AI.

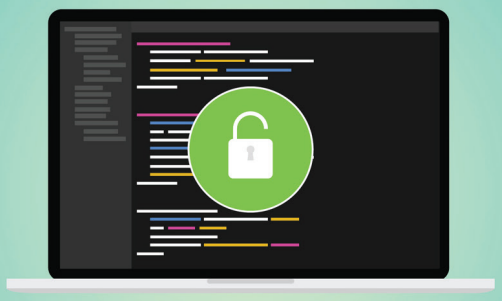
As several of the sensors that are needed for full autonomy are already fitted in vehicles with a lower level, it is a perfect situation to start logging data from all systems involved and from the driver's actions. Imagine a vehicle of level 3 type and that the radar system produces an emergency braking call but the driver simply releases the accelerator. The film that has been recorded shows that there was a large black bin bag on the roadway. Using data collected from all components will subsequently enable models to be developed so that the camera can determine what type of obstacle it is seeing.

Recorded data can also be used for feedback to functions under development in simulated environments, regression testing etc.

Environments within the Automotive field always place high requirements on environmental endurance. Combined with several interfaces, a high memory capacity and frequently a lack of space in the test object, this makes specifying and developing general log equipment complicated. All vehicle manufacturers have their own strategy to construct solutions to log automatic functions. At Dataresponse/Sylog, we have long experience of both data logging and the vehicle industry. Together with customers, we are looking at different solutions to help them raise automation to the next and subsequent levels.

THE OPTIMAL TOOLBOX
for Open Source Development

FREE WHITEPAPER TO DOWNLOAD



Are you looking for a fully integrated set of tools for state-of-the-art support of C++ development on Linux? Then look no further. Thomas Arnbjerg, software developer at TechPeople, the Danish subsidiary of Data Respons, has done the job for you, putting together the optimal Open Source development setup, and describing it in an extensive step-by-step guide, free for you to download from our website. Here is an introduction to the paper:

TechPeople A/S

Tools to manage software development involving various developers are often expensive. As an alternative, the Open Source world offers similar tools, but in the past they have been standalone or difficult to integrate. That has changed, according to Thomas:

- A large variety of tools and disciplines have been developed to assist in managing all these details, including integrated development environments which, among other things, provide assistance with formatting, troubleshooting and code completion. Also there is a test framework for unit tests, memory profiling, code coverage, etc., build/continuous integration servers, tools for statistical code analysis and versioning systems.

- In this guide I've tried to pull these tools together into a well-functioning, fully integrated and free set of tools which provide state-of-the-art support of C++ development on Linux. Many of these tools may also be used in other situations.

Thomas Arnbjergs guide addresses the well known challenges in coding:

- Ensuring that code written by different developers is uniform
- Ensuring that many developers may work on the same program at the same time
- Ensuring that each developer is able to QA his or her code before it's integrated into the system
- Ensuring that the overall system will function after changes are implemented

The guide details a process in which the individual developer creates a branch in the source code to develop a new feature without compatibility issues in relation to the work done by the other developers. The following tools are used to support the process – all installed under Linux:

- Jenkins as Continuous Integration Server with a wide variety of installed plugins
- Eclipse IDE for C/C++ development with an extensive range of installed plugins
- Git versioning systems

Developers will see how, at the start of a new feature, they get their own sandbox for the development. They receive tools support for development with code completion and formatting. On the build server, their codes are controlled using static code analysis and unit test. Code coverage is measured and quality criteria/goals can be set out. Once a feature is completed, 'develop' is merged into the feature branch and it is ensured that everything runs in Jenkins. After that, there is a merge back to 'develop' and the feature is finished. All things considered, the integration task becomes so much simpler and most of it happens in the feature branches.

Furthermore Thomas Arnbjerg drills below the surface and looks at the configuration and use in a Continuous Integration (CI) scenario in which Jenkins supports the development in feature branches.

According to Thomas, the only annoying aspect about Jenkins is that (like all other open source projects) Jenkins does not have a large marketing department which highlights its excellent qualities. Another thing is that all the functionalities available today as well as future improvements have required and will continue to require that a person who understands the problems and has the skills and the time to solve them (could be sponsored) works out an extension which will make life easier for everyone. These are the mechanisms which have created the super tool that can be installed today free of charge.

Everything in this guide is open source-based without costs for licenses and all the projects go several years back and are actively maintained. Following up on development and upgrading on an ongoing basis is thus manageable.

Last but not least, it is considered good style to give something back to those who have made it possible to create a professional development setup that does not require investment of large amounts of money. So, while you benefit from these tools you might perhaps consider donating part of the savings to the open source projects or contribute with developer resources to make the experience even better.

The full guide can be dowlnloaded from our website: datarespons.com/opensource

This is TechPeople AS

TechPeople is a consultancy house within the Data Respons group. The company is based in Copenhagen, and specialises in embedded solutions and IT business systems.

TechPeople have specialists within hardware, software, mechanic development, project management and product testing. TechPeople's innovative customers range from large international companies to creative start ups.

techpeople.dk

TechPeople

RAPID DEVELOPM

and beyond with Oracle APEX

Over the years, the requirements of software and its development have changed a lot. On the one hand there is an increasing customer demand for web applications which can be accessed easily via a web browser and can handle huge amounts of data. On the other hand there is growing demand on the part of software developers for faster development including faster prototyping. The well-known database manufacturer Oracle has found a way to combine both demands.

BY: Markus Kaml
Project Manager and Instructor,
EPOS CAT GmbH

Whenever it is possible and reasonable, users prefer web applications to desktop applications because of their easy accessibility in web browsers on desktop or mobile devices without the installation of any additional software. Furthermore, the amount of data in companies has increased over the years. At all times database systems are the preferred ways to store and handle data efficiently. The database manufacturer Oracle [1, 2] is well-known for its relational database system "Oracle Database" which provides many efficient features to read and write large amounts of data. To cope with the growing demand of developing web applications very fast, Oracle has created the online development environment "Oracle APEX" [3] which comes as a no-cost plugin for "Oracle Database" and is already included in "Oracle XE". "APEX" stands for "APPLICATION EXPRESS" and that is precisely what it is.

COMBINED DEVELOPMENT ENVIRONMENT AND DATABASE

Oracle APEX is fully supported through and available for Oracle Database. After its installation Oracle APEX provides a powerful development environment which is accessible online via web browsers. Oracle APEX is independent of the operating system underneath. Just a web browser is required.

Due to the fact that Oracle APEX is installed on Oracle Database, the corresponding database can be accessed directly through the online development environment shown in figure 1.

Every part of an Oracle Database, e.g. tables, views, triggers, etc., can be accessed in this way by using the SQL Workshop. Thus no persistence layer is needed to exchange data between the developed application and the database.

However, this means similarly that Oracle Database is required to use Oracle APEX. In view of the operating costs of Oracle Database, which can be very high depending on the preferred license model, Oracle APEX is more suitable for companies that already operate Oracle Database.

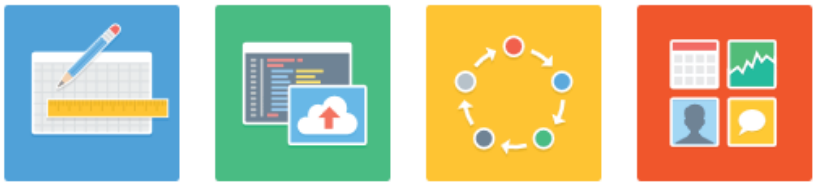
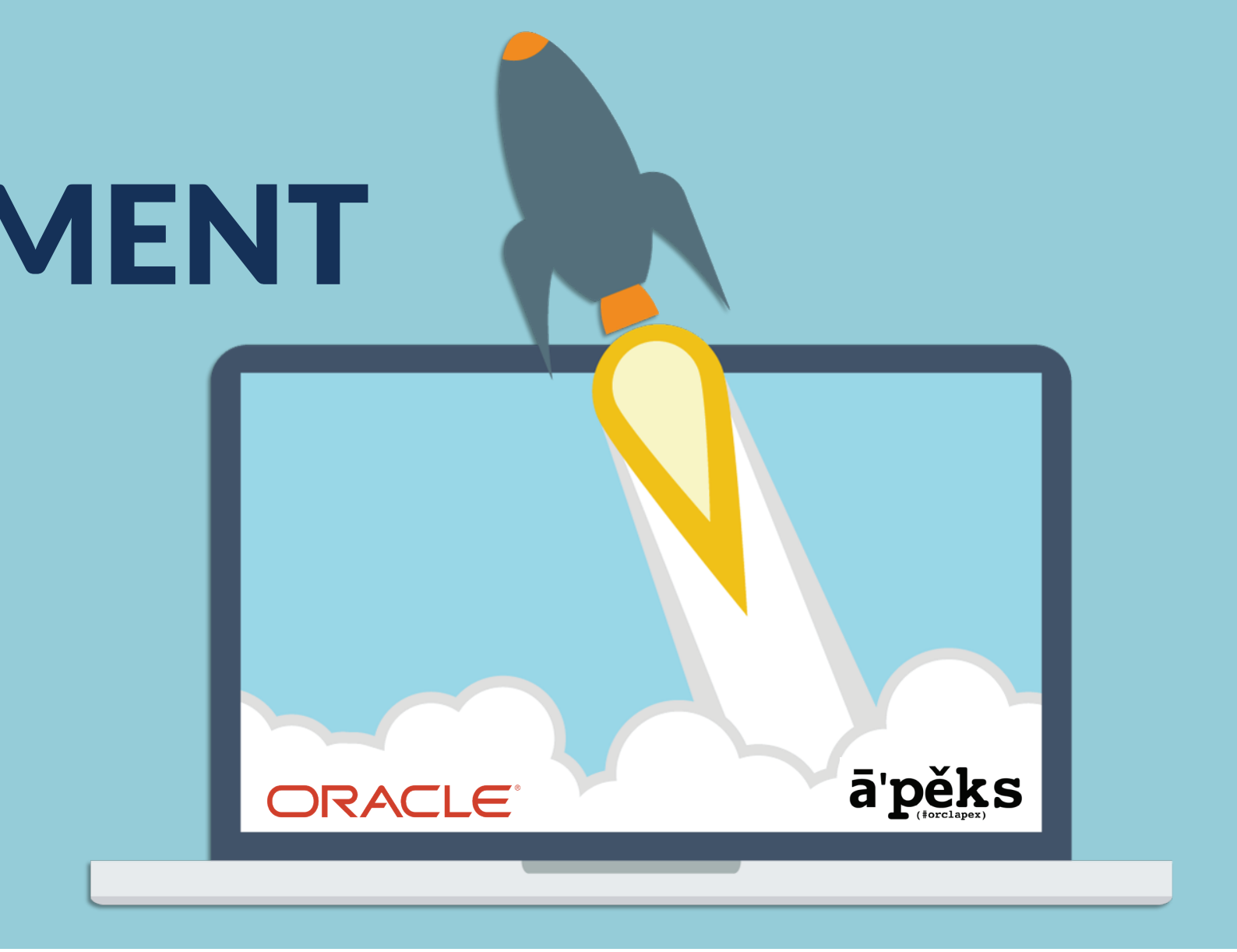


Figure 1: Oracle APEX development environment



PROGRAMMING DIRECTLY WITHIN THE DATABASE

Oracle Database is a powerful relational database system which can handle huge amounts of data and supports tables, views, sequences and triggers etc. In addition to those database items, Oracle Database provides packages and functions to enable developers to extend their databases by writing source code with PL/SQL. PL/SQL was invented by Oracle and extends the ordinary SQL functionality with features known from other programming languages. Thus "PL/SQL" stands for "Procedural Language/Structured Query Language" and makes it feasible to use, for example, variables, arrays, if-queries and loops directly in an Oracle Database. Even object orientation can be applied.

As mentioned before, no persistence layer is needed to gain direct access via the online development environment to the database underneath. In the same way PL/SQL does not require such a persistence layer to gain access to tables or views. In packages and functions PL/SQL can be mixed up with ordinary SQL. This enables a very easy, fast and lightweight way to write powerful PL/SQL scripts. It is feasible to use prepared statements as well.

NO ADDITIONAL PROGRAMMING LANGUAGE

Except for PL/SQL, which extends the ordinary SQL with powerful features known from programming languages, there is no additional programming or scripting language to learn. Oracle APEX is an online development environment which supports developers to create web applications. Thus all the established technologies in web development, e.g. HTML, CSS, JavaScript or jQuery, can be used to design web applications and make them do what should be done.

Furthermore, other technologies such as Java or Jasper Reports can be applied as well.

In addition to the off-the-shelf set of items like buttons, text fields or select lists, Oracle APEX can be extended with other powerful third-party items in the form of plugins. One of the most common plugin items is the "Select2 APEX plugin" [4] which is based on "Select2" [5] and improves the functionality as well as the user-friendliness of ordinary Oracle APEX select lists. Furthermore, it is also very simple to create own APEX plugins.

RAPID GUI PROTOTYPES

Besides the advantages mentioned, regarding software development itself, Oracle APEX supports developers in an earlier stage as well. As a default layout theme for the graphical user interface (consecutively GUI) Oracle APEX comes with the Universal Theme and the Theme Roller as an easy-to-use tool to adapt the Universal Theme individually. There is no need to spend time on the GUI at the very beginning. Thus the developer can directly start with implementing the business logic.

This is the reason why Oracle APEX is feasible to create rapid GUI-Prototypes without logic. Thus prospective customers can get an idea of how their future application will look.

One of the most efficient features of Oracle APEX is tabular reports of data which are used in so called Master-De-

tail-Pages. Master-Detail-Pages consist of a tabular master page and an item-based detail page. The master page provides an overview over the corresponding data in form of data sets while the detail page provides the possibility to create and edit a single data set.

To create such a tabular report just an ordinary SQL SELECT statement is sufficient. According to the selected columns Oracle APEX generates a tabular report including the same columns. The appearance of this report is based on the current theme used for the regarding application. Figure 2 shows an SQL SELECT statement as well as the corresponding tabular report.

```
SELECT
    QUANTITY,
    ARTICLE,
    BRAND,
    UNITPRICE,
    QUANTITY * UNITPRICE
        AS TOTALPRICE
FROM
    CART
ORDER BY
    ARTICLE;
```

Cart

Quantity	Article	Brand	Unit price	Total price
30	Galaxy S9	Samsung	€552,00	€16.560,00
30	Phone 8	Apple	€615,00	€18.50,00
30	P20	Huawei	€482,98	€14.89,40

1-3

Figure 2: SQL SELECT statement and corresponding tabular report

INTERNATIONALISATION

The world is getting more and more connected. Thus the demand for multilingual applications is growing rapidly as well. Oracle has taken this fact into consideration and has equipped Oracle APEX with powerful translation utilities to translate whole applications.

The language in which the APEX application is developed at the very beginning serves as the default language. All the labels, buttons, region titles, main and sub menu entries can be translated with no additional effort. There is no need to think about translating the application at the very beginning. Each APEX application can be translated rapidly at any time.

If an application is to be translated, Oracle APEX will store all translatable texts in a translation repository. This repository can be exported as an XLIFF file. "XLIFF" stands for "XML Localisation Interchange File Format" [7] and is an XML-based format to store and exchange translation data. Once an XLIFF file is created it can be easily extended with the translated texts and imported into the translation repository. As a final step the updated translation repository must be published before the translated application can be used.

However, the translation utilities of Oracle APEX just cumulate all translatable texts without any duplicate checks. This means, that e.g. every OK-Button of the application appears a number of times within the XLIFF file. Due to this, some texts have to be translated more than once. This confession must be made to keep the flexibility of translating an application at any time during the development process.

AUTHENTICATION AND AUTHORISATION

Although web applications can be accessed very easily by anyone using a web browser, it is not always intended that everyone can read or edit every data within the application. Different users should get different permissions to access data. Oracle has thus developed a comparably powerful and easy-to-use approach to secure web applications developed with Oracle APEX.

To allow or deny users to read or edit data, the application has to know the identity of the user which requests access. This process is called authentication. Oracle APEX provides the possibility to define and apply different authentication schemes. Particular authentication servers can also be used to adapt SSO, which stands for "Single-Sign-On". An authentication scheme needs to know where to find authentication information of the users and what to do with new users. This can be achieved with PL/SQL. Once a web application contains more than one authentication scheme, it is very easy to switch. Even for the APEX workspace users of the development environment there is a built-in authentication theme available off-the-shelf.

As soon as a user is authenticated successfully the next process, called authorisation, comes into play. According to the authentication schemes mentioned Oracle APEX provides authorisation schemes to manage what permissions a user has. Such an authorisation scheme needs to know where to find the permissions the users have. This can be achieved with PL/SQL as well. The following example works with a table "USER" containing the user identity, a table "ROLE" containing the existing application roles and a table "USER_ROLE" containing roles the users have. Figure 3 shows a PL/SQL function, located in the package "APP_SEC", which returns a value whether or not the committed user has administrator permissions or not. In addition the figure shows the corresponding content of the authorisation scheme which calls the PL/SQL function. Almost everything in Oracle APEX (e.g. pages, regions, items, buttons, validations, processes etc.) can be restricted with an authorisation scheme.

In addition to authentication and authorisation, Oracle has provided an additional functionality called Oracle VPD. VPD stands for "Virtual Private Database" and offers the possibility to implement multi-client capability into APEX web applications. With Oracle VPD and PL/SQL special columns of tables can be declared as conditions to separate data between different clients. An active Oracle VPD automatically adds an SQL WHERE clause to an SQL SELECT statement. This WHERE clause contains the

> "APEX has allowed us to migrate several disparate Excel and MS Access applications to a consistent, secure, web based environment. The speed and concurrency offered by APEX have been exceptionally valuable."
>
> - Eric Brandenburg, Senior Applications Architect, Brunswick Corporate IT [6]

> For close to 20 years, the Université du Québec à Trois-Rivières (UQTR) has used the Oracle PL/SQL technology to develop most of its internal and public systems on the Web platform (for example, the student portal). Moreover, we have integrated Oracle Application Express (APEX) to our development, and we are completely satisfied with it. Oracle Application Express is a quick, powerful, and mature development tool that allowed us to improve our productivity level."
>
> - Georges-Martin Caron - IT and Technology Project Manager – Coordinator of the Information Systems, Université du Québec à Trois-Rivières [6]

declared columns and thus delivers only data sets that match (row level security).

CONCLUSION

Oracle has created a powerful development environment in the form of a no-cost plugin for Oracle Databases called Oracle APEX. It can be used for both rapid development and rapid prototyping. Oracle APEX provides easy-to-use and at the same time powerful support for authentication, authorisation and internationalisation. In addition, with PL/SQL it is feasible to create efficient, multilingual, secure and future-proof web applications which are independent of their dimensions.

The German IT service provider EPOS CAT GmbH has been working with Oracle Databases and Oracle APEX for more than 10 years. Since 2005 more than 80 web applications for over 170,000 users have been developed. This shows the great success of Oracle APEX.

```
-- Checks if a user has a role.
FUNCTION hasRole(p_username VARCHAR2, p_roleid NUMBER)
    RETURN BOOLEAN IS
    l_ret NUMBER DEFAULT 0;
BEGIN
    SELECT
        COUNT(1)
    INTO
        l_ret
    FROM
        USER u
        INNER JOIN USER_ROLE ur ON ur.USER_ID = u.ID
        INNER JOIN ROLE r ON ur.ROLE_ID = r.ID
    WHERE
        TRIM(UPPER(u.USERNAME)) = TRIM(UPPER(p_username))
        AND r.ID = p_roleid;

    RETURN (l_ret > 0);
END hasRole;
```

```
-- Variable containing the unique id of the role.
g_roleid_administrator NUMBER := 1;

-- Checks if a user is Administrator.
FUNCTION isAdministrator(p_username VARCHAR2)
    RETURN BOOLEAN IS
BEGIN
    RETURN APP_SEC.hasRole(p_username,
        g_roleid_administrator);
END isAdministrator;

-- Content of the authorization scheme.
Authorization Scheme: isAdministrator
Scheme Type: PL/SQL Function Returning Boolean
PL/SQL Function Body:
return APP_SEC.isAdministrator(
    OWA_UTIL.GET_CURRENT_USERNAME() );
Identify error message displayed when scheme violated:
Administrator permissions are required to access this page!
```

Figure 3: PL/SQL function and calling authorisation scheme

REFRENCES

1. Oracle (07.03.2014). About Oracle. Called on 13.10.2018 from https://www.oracle.com/corporate/#info
2. Oracle: Oracle Fact Sheet - The Complete Cloud and Next-Generation Platform For Business. September 2017.
3. Oracle (03.10.2018). Oracle APEX. Called on 13.10.2018 from https://apex.oracle.com/en/
4. APEX-PLUGIN.COM (12.08.2013). Select2. Called on 13.10.2018 from http://www.apex-plugin.com/oracle-apex-plugins/item-plugin/select2_344.html
5. SELECT2 (2015). Select2. Called on 13.10.2018 from https://select2.org/
6. Oracle (03.11.2015). Oracle Application Express Customer Quotes. Called on 13.10.2018 from https://www.oracle.com/technetwork/developer-tools/apex/learnmore/apex-quotes-1863317.html
7. OASIS (01.02.2008). XLIFF Version 1.2. Called on 13.10.2018 from http://docs.oasis-open.org/xliff/v1.2/os/xliff-core.html
8. #orclapex (26.11.2015). apeks.png. Called on 13.10.2018 from https://blogs.oracle.com/academy-dach/oracle-apex-programming-competition-2016-fr-studenten-aus-nrw
9. Logok (18.11.2014).Oracle_logo-880x660.png. Called on 30.01.2019 from http://logok.org/oracle-logo/

INSIDE WRITERS

ANDREAS DANGEL

Software Engineer
MicroDoc GmbH

MARKUS KAML

Project Manager and instructor
EPOS CAT GmbH

ALEXANDER SOWATSCH

Solutions Architect,
EPOS CAT GmbH

CRISTER NILSSON

Software Engineer
MicroDoc GmbH

THOMAS ARNBJERG

Solutions Architect,
EPOS CAT GmbH

PETER DONS TYCHSEN

Bluetooth Expert,
TechPeople A/S

PUBLISHER:

Data Respons ASA,
Sandviksveien 26,
1363 Høvik

Tel: +47 66 11 20 00
info@datarespons.no

EDITOR-IN-CHIEF:

Kenneth Ragnvaldsen
CEO, Data Respons

EDITOR:

Elisabeth Andenæs,
Corp. Communications & Brand Manager,
Data Respons

Tel: +47 92 20 30 03
Email: ean@datarespons.no

TECHNICAL EDITORS:

Data Respons R&D Services,
Ivar A. Melhuus Sehm, Director
Email: ise@datarespons.no

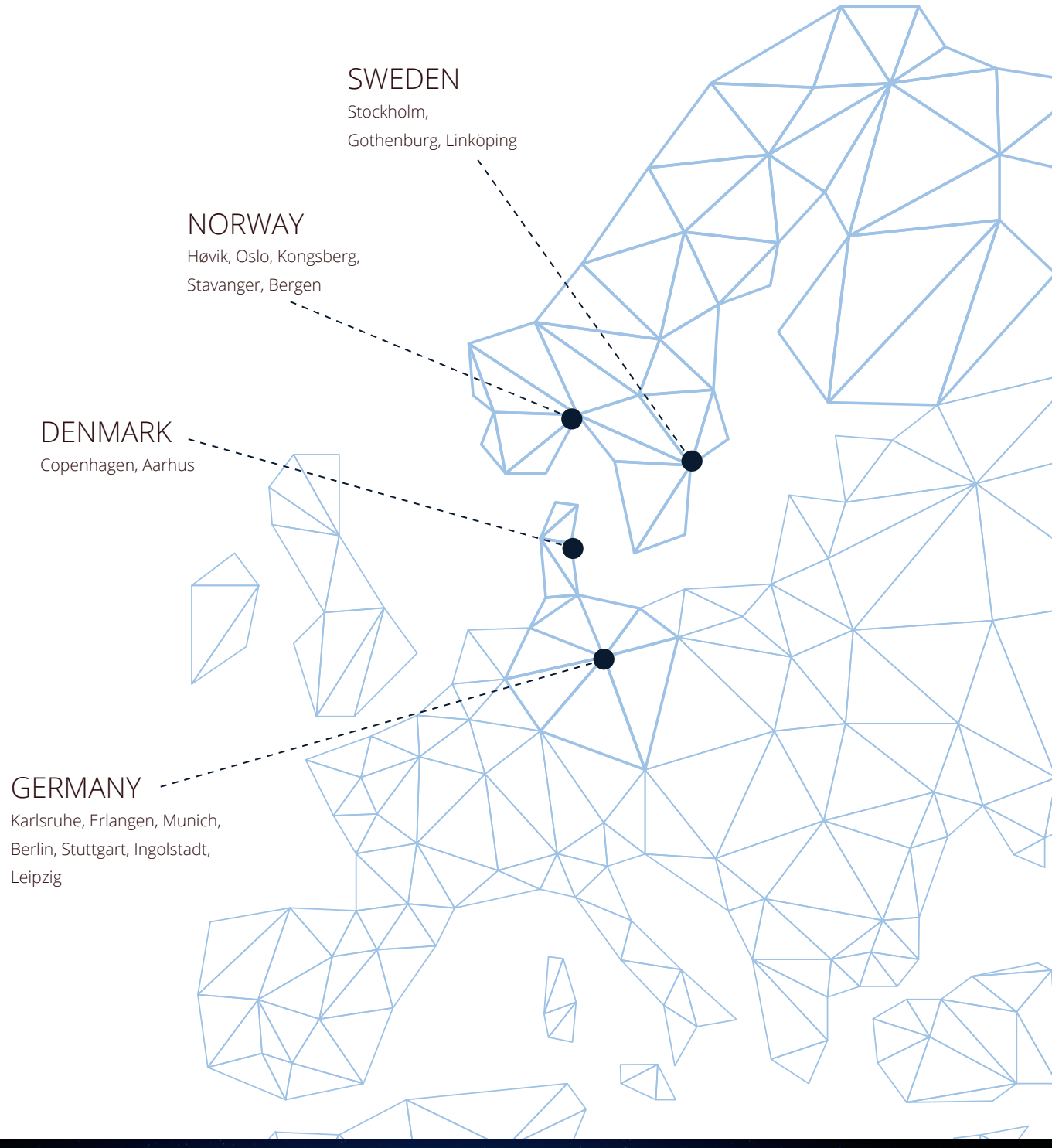
TechPeople A/S,
Kim Farhenholtz, Managing Director
Email: kfi@techpeople.dk

Sylog AB,
Johan Jacobsson, Managing Director
Email: johan.jacobsson@sylog.se

Microdoc GmbH,
Hans Kamutzki, Managing Director
Email: hans.kamutzki@microdoc.com

EPOS CAT GmbH,
Dr. Heidi Sauer, Managing Director
Email: heidi.sauer@epos-cat.de

data:respons



A SMARTER SOLUTION STARTS FROM INSIDE

A SMARTER SOLUTION
starts from inside