aujava executive brief

2019.11.21

Overview

- aujava (or au4j in short) is a Java/J2EE application understanding tool, in its current form a prototype. This short briefing focuses on the following topics.
- Value proposal tries to answer the following questions:
  - Why do we need a seperate tool to understand our application? Cannot we just understand it w/o any tool support?
  - Why and for whom is it practical to understand the application?
  - What is aujava and what it is not? Who are its intended users? What does it do, what features does it have?
- Business proposal tries to answer the following questions:
  - What are the business cases when you might need aujava?
  - What is the current status of aujava and how can one aquire it?

Value proposal: why do we need it?

- Do you have a large Java/J2EE code base which lacks live (always-up-to-date) documentation? If yes, then you might need an application understanding tool:
- **if you have a large code base**, say 1 million lines of code, which cannot be remembered by a developer, or even by a team of developers – since 1 million line of code means 18 000 printed pages, War and Peace x 14.
- **and if you don't have a live documentation** of your large code base, which is often the case – since many times, the source code is the only source of truth about how your business (application) works.
- then you might need a tool that can help you understand your code base: what it does and how it works. **That is basically what aujava does: helps you to understand an otherwise undocumented code base that is too large to remember. Question: why is that practical?**

Value proposal: why is it practical?

- aujava (in its current form) have 2 business cases where you can benefit from its usage:
  - **Application modernization**: When you are about to decide or have already decided to replace your legacy Java/J2EE application with a modern one.
  - **Application maintanance**: When during Java/JEE application maintanance you have to concert the collaboration of different IT professionals.
- **In sum aujava can help you whether you are still maintaining your legacy Java/JEE application, or you already decided to replace it. Question how?**

Value proposal: application modernization

- If you want to port or are thinking about porting your legacy Java/JEE application to a newer platform – say to cloud native environment, or Spring Framework, or something else…
- then whatever the target platform is, you have to manage your modernization project: you first have to identify the business functions embodied within the legacy code in order to make a rough estimation of the costs and to define the scope of the job to be done.
- **That is one thing that aujava can do: it can extract the business logic from a Java/J2EE application code base so then you can estimate the development costs of modernization and put it into development.**

Value proposal: application modernization (contd)

- aujava provides a brand-new approach to extract the business logic from the application code. Code analysis traditionally goes top-down, starting from the entry point(s), and traversing the directed graph of the control flow. Things quickly become unclear, get out of hand, because complexity grows exponentially on the way.
- aujava takes the opposite direction. It detects the particular points in the code, where a piece of persistent data is created (called *output endpoint*'s). From there, the traverse goes backward, gathering the code fragments that are involved in creation of the persistent data. **The result is the clean business logic, "distilled" from any other aspects of the architecture, e.g. logging or management**.

Value proposal: application maintanance

- If you are struggling with the maintenance of a large Java/JEE application, where there are many different IT experts who should work and collaborate with each other, ie. developers, analysts, testers, operators and security experts…
- Then you might need a tool that can speak to the different languages, understand the different viewpoints of the professionals…
- **That is the second thing that aujava can do, it can speak the language of developers, testers, operators and security experts, which in sum can save time and money during maintenance.**

Value proposal: application maintanance (contd)

- **4 testers/developers**: aujava can identify and present the impact of a code change. It can then decrease the regression bugs discovered after deployment and/or your testers don’t have to do a full regression test before deploying a change.
- **4 support/developers**: aujava can identify the code (JPQL) that corresponds to a problematic query (SQL) found in logs. So that your operation team don’t have to wait days while the developer tracks this down manually...
- **4 security experts**: aujava can identify the high value targets in the application interface, those linked to high value data. So that security professionals will be able to make targeted analysis and risk calculation, besides general vulnerability analysis.

Value proposal: what aujava is and is not?

- **aujava is a Java/JEE application understanding tool intented to be used by IT professionals**: developers, analysts, operators, testers and security experts
- **aujava is not a Java/JEE code audit tool and especially not a reporting tool** - there are very nice, free and popular tools for this use case and aujava does not compete with them
- **aujava helps to solve some of the management problems** of Java/JEE application modernization/maintanance, **but it is not intended to be used by managers**

Business proposal: why and when might you need it?

- **Application modernization**: Java/JEE, once a flagship Java technology for enterprise application development, recently became legacy. Big vendors (like Oracle, IBM) decided to drop it in favour of cloud native platforms. Hence Java/JEE applications written in the last 20 years might soon start to move from this legacy platform to new ones.
- **Application maintanance**: There could be legacy Java/JEE applications which cannot be quickly ported to a new, modern platform. Investment protection could be another reason to maintain a legacy application. Whatever the reason is if either the maintanance costs start to increase, or the activity rate starts to drop, then a tool might be useful which can hold back the costs and/or increase productivity.

Business proposal: status of the product and how to aquire it

- **aujava builds upon our existing technology of application understanding for RPG, and COBOL. It extends our solution to the Java world.**
- **aujava is currently a prototype, not yet a product.**
  - The prototype development will finish at the end of November.
  - The first real life test will supposed to start this month.
- **Since aujava is not yet a product, it is sold through projects**. The concept is the following:
  1. Project: We adopt/customize aujava to the customer needs. Being a prototype this could mean software development/updates as well.
  2. Usage: Then, after the adoption/customization the customer can use it for its own needs.