TAMPERE UNIVERSITY OF TECHNOLOGY

Guidance to selecting technologies for project

Matti Vuori

18.2.2015

Contents

Introduction

- This is a short intro to the question of how to select technologies to you project.
- Those include programming languages, libraries and so on.

First note

- The idea on the course is not to reinvent the wheel but produce as solid implementations for the customer as possible
- So using libraries and (free) components is a good idea
- Spend to coding effort to doing the unique things!

General strategies 1/4

- Don't automatically use the latest hyped thing, it might not be the best for you.
- Use only something that several of you already know and can use – time in project has better uses that learning a new language, and what if the one who is proficient get sick?
- Think of what could be good for the customer.

General strategies 2/4

- If you wish to create a community, find out what most people use, so it is easier for others to step in.
- To reduce hassle, keep the number of components small – this is related to managing the scope of product: there is only so little that can be done during the project.
- Think of licenses and costs – there are open source components for almost anything.

General strategies 3/4

- Have a consensus for choices – don't let anyone dominate.
- Evaluate alternatives (more about this later).
- Be prepared for problems though: your implementation may be an unique use case that a component has not been used for. In that case it is good to have open source components that you can fix yourself.

General strategies 4/4

- Anyway: don't trust anything, test everything that you do properly whether you made a component yourself or used an existing one.
- Include component issues in the risk analysis and be ready to make changes if there are problems.

Evaluate alternatives 1/4

- Technical issues: Suitability to your need, maturity, availability of tools and libraries and support: are there good forums where you can find help.
- Product quality: Will the technology product reliability, security, usability & other things that the users may wish? Security is critical. Check for security problems and also how fast they (if any) have been corrected in the past. Be paranoid!

Evaluate alternatives 2/4

- See if you will find advice and problem reports in the net. Check the bug databases of alternatives.
- See if the community behind a component is alive
  - Don't use a dead or dying project or something that others are moving away from.
- Think of platform support. You may now think of for example one mobile platform, but what if you want to support something else. Consider what the potential users are using now and later.
- Development performance: How productive working with the technology is; does everyone like (preferably love) it?

Evaluate alternatives 3/4

- Testability.
  - Can you test the component with common tools?
  - Are there testing tools for a platform or language?
  - Automated testing tools are often needed for continuous integration tests and tests during deployment.
- In choosing a language, think of how easily you can use with it things done in other languages.

Evaluate alternatives 4/4

- As you want to be agile, think of how simple and easy it is to change the implementation when your ideas change. This helps not only during this project, but later on, if the development continues.
- Think of changeability of the whole component if the component turns out not so good (also make your architecture so that changing is easy).

Practical testing

- Do a little implementation in all relevant alternatives and see how it goes. Ideally, some of you have already used the technology for something non-trivial.
- Test the critical features that you are dependent on.
- Assign the practical evaluation to someone as a clear task with resources.

Open source components 1/2

- Check the licenses and other IPR issues.
  - That you can utilise the component as you wish and that a restricted license does to attach to your code. There is a separate slide set about those.
- Start listing the components and licenses early on to prevent problems.
- Consider licenses for testing components too, if you wish to distribute them.

Open source components 2/2

- About open source licenses:
- http://opensource.org/licenses/category
- Comparison of licenses:
- http://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses
- This is a good read (In Finnish, though):
- "Matti Saastamoinen: Avoimen lähdekoodin lisenssit kaupallisessa liiketoiminnassa" (Open source licenses in commercial business)
- https://tampub.uta.fi/bitstream/handle/10024/93510/gradu01157.pdf?sequence=1