What is Software Architecture (2):Quality Attributes

Malcolm Davis made the following comment on my attempt to define software architecture:

not a complete definition

Arnon's (and most of the industry's) definition of software architecture is counter to the point of architecture and engineering in the world of construction. Architecture approaches the problem from a user/business problem space. Engineers solve the technical space, and not necessarily the user/business space. The following is from Alan Cooper: "Architects synthesize people, purpose, and technology. If you just take people and technology, you have art and entertainment. If you just take technology and purpose, it's engineering. And people and purpose without technology is psychology. Architects have to synthesize all this, to create a vision of a solution. People must get something practical achieved."

Arnon's vision, as with most, sees architecture as an abstract/higher level of engineering and design. The overreaching theme is technology and purpose. In reality it is the inclusion of business and people. And business and people are left out of Arnon's final definition for software architecture.

I cannot count the number of interviews I have had with so-called "Architects". One of the first questions I always have is about the business model, and how the application is integrated into the business model. Many don't have a clue about the business model or the application's ROI. For instances, how is the quality ROI determined?

Architecture does not end upfront, but continues throughout the construction lifecycle.

The reason for an architect is that requirements analysis and design do NOT fully solve the problem.

I don't agree that I left out people. I did talk about the "system quality attributes" but I probably should have explained better what quality attributes are.

Quality attributes are non-functional requirements that are derived from stakeholder's needs.

The most common quality attributes used as input for architectural design are those seen from the end-user's perspective:

- Performance
- Availability
- Usability
- Modifiability
- Security

However, business stakeholders will probably be more interested in things like:

- Time to market
- Cost vs. Benefit
- Interoperability (e.g., with legacy systems)
- Projected life time
- ROI
- etc.

Developers concerns are again probably a little different; for example:

- Maintainability
- Portability
- Reusability
- Testability

There are many stakeholders to the project and we have to balance all these sometimes conflicting attributes. ISO/IEC9126-1:2001, a standard for the evaluation the quality of software, provides a hefty list of quality attributes to draw from few examples include:

- Functionality. A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
    - Interoperability
    - Compliance
    - Security
- Reliability. A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
    - Recoverability
    - Fault Tolerance
- Maintainability. A set of attributes that bear on the effort needed to make specified modifications.
    - Stability
    - Changeability
    - Testability
- Portability. A set of attributes that bear on the ability of software to be transferred from one environment to another.
    - Installability
    - Replaceability
    - Adaptability

If we go back to Malcolm's comment then yes, the architecture is an "as an abstract/higher level of engineering and design"; that supports/allows several quality attributes (assuming the architect did a good job) to be met. That is because the architecture is the "blueprint". The architect's role on the other hand is in fact to work with the different stakeholders and balance their relative needs including business aspects (which, as mentioned above, I find helpful to express as system's quality attributes).