

DEVELOPING A SOFTWARE ARCHITECTURE

**Architecture starts when you carefully put
two bricks together. There it begins.**

- Mies van der Rohe

INTRODUCTION

As with traditional architecture, software architecture begins when the first lines of a computer program are written. Whether the architecture is described by formal documentation or a barely imagined “castle in the sky”, the author of those first few lines of code and all the lines which follow them is being guided by a software architecture.

The analogy continues to apply as the software development process continues to unfold. Just as a well architected building achieves its potential in form and function as it nears completion, so a well architected software system achieves its potential in form and function as it nears completion. In contrast, there always seems to be something ‘missing’ when one encounters poorly architected buildings and poorly architected software systems.

The extent to which the completed building meets or even exceeds the customers goals and expectations is not an accident. Rather, it is the direct result of the quality of the building’s architecture. Similarly, the extent to which a completed software system meets the customers goals and expectations is also not an accident. It is the direct result of the quality of the software system’s architecture.

Exemplary software architectures, like exemplary building architectures, do not just “happen”. They are the result of the efforts of the system’s software architect and are reflected in the architectural diagrams and documents which flow from these efforts.

CUSTOM VS PRODUCT DEVELOPMENT PROJECTS

Substantive development projects can be classified into one of two categories:

- Custom development projects which are launched to meet the needs of a particular customer.
- Product development projects which are launched to meet the needs of the “marketplace”¹.

From an architectural perspective, there are three fundamental differences between a custom and a product development project:

- In a custom development situation, the needs of the customer can, at least in theory, be characterized by asking the customer the right questions. In contrast, there is no “customer” who can be asked any questions at all in a product development situation (there may be “representative customers” available to answer questions although, unfortunately, the extent to which the “representative customers” are truly “representative” is difficult if not impossible to ascertain and, as a consequence, even if the right questions are asked, it becomes difficult if not impossible to judge the value of the answers).
- In a custom development situation, the needs of the customer are (relatively) static whereas, in a product development situation, the needs of the “marketplace” are constantly changing².
- The “demand” for a custom development project’s deliverables³ is (relatively) unlikely to vanish or even diminish substantially during the life of the project. In contrast, a product development project could be rendered entirely pointless at any moment by the introduction of a roughly equivalent or even superior competing product.

The net effects of these differences are:

- the risk of failure due to an incorrect understanding of the project’s goals are unavoidably higher for a product development project than they are for a custom development project.

Fortunately, the practice of software architecture is primarily if not entirely focused on reducing this exact risk - i.e. the risk of failure due to an incorrect understanding of the project’s goals. Although this risk can never be reduced to zero and will always be of greater concern in a product development context vs a custom development context, the net reduction in risk is very real and, typically, quite substantial.

- the needs analysis component of a product development project must deal with “market experts” and “representative customers” who are fundamentally unable to answer many of the questions that they will be asked (including many that they believe that they are in a position to answer) whereas a custom development effort has at least some form of access to the actual customer.

1. One could argue that exploratory or research-oriented development projects represent a third category of software development project. From an architectural perspective, an exploratory or research-oriented development projects is almost if not entirely indistinguishable from a custom development project in which the “customer” is the lead researcher or possibly even the developer or developers themselves.

2. This is more of a “shades of gray” argument as the needs of a custom development project’s customer are rarely if ever truly static.

3. Dare I say “product”?

While broadly applicable to both types of software development projects, the remainder of this document focuses on how to develop a software architecture in a product development context.

The development of a software product's architecture is a collaborative effort which involves bringing together the goals, dreams and ideas of the products' 'sponsors' with the skills and insight of the software architect.

The result is a document or more likely a set of documents which describe:

- the needs and opportunities which the product is intended to address in the near and long term.
- the major (i.e. architectural) components and interfaces of the product with an emphasis on each component's roles and responsibilities, and on the information which flows across each of the interfaces.
- the key features which each major product release will introduce to the marketplace over a period of at least a few years.
- the technologies which the product development effort will leverage.
- the sequence in which the "first customer ship" (FCS) release of the product will be created.

What follows is a multi-phase approach to developing a software architecture. Some may argue that it is possible if not necessary to skip over some of the earlier phases in situations where a reasonable if colloquial understanding of the product's software architecture already exists. The obvious response to such a suggestion is that if such an understanding actually exists then the earlier phases should not take very long and could uncover aspects of the architecture which are, in fact, not sufficiently well understood.

PRODUCT POSITIONING PHASE

The first phase of software product architecture development is to position the product by identifying and documenting the market needs / opportunities which the product or product family might reasonably be expected to address. The planning horizon for the product positioning phase is context dependent although a horizon of about five years is usually appropriate as the horizon is then far enough out to stretch discussions beyond the needs of right now and near enough out to be subject to reasonably fact-oriented discussion (at least a modest effort should be expended on the task of looking out about twice as far out as the nominal planning horizon).

A major goal of this phase is to capture both the short term goals and intentions of the ‘sponsors’ while documenting enough of the long term vision to ensure that short term decisions do not rule out long term opportunities.

The output of the product positioning phase is a document or set of documents which the participants all agree captures the short term goals and long term vision alluded to in the previous paragraph⁴.

The product positioning phase ends with a careful review of the generated documents to ensure that they accurately capture the identified market needs / opportunities.

INITIAL ARCHITECTURAL DIAGRAM DEVELOPMENT PHASE

The next phase of software product architecture development is to develop high level (i.e. architectural) diagrams of how a set of software components might work together to satisfy the most ambitious market needs / opportunities identified in the product positioning phase. This phase is, almost of necessity, iterative in nature as early attempts at creating the architectural diagrams are likely to be “throw away” efforts.

As it becomes apparent that the current high level architectural view is likely to prove satisfactory, a reasonable amount of effort should be expended in informally discussing issues raised in the phases described below as such discussion is likely to either confirm the validity of or identify flaws in the current approach.

The initial architectural diagrams phase ends with a careful review of the architectural diagrams with an emphasis on whether or not the architecture can satisfy all of the identified market needs / opportunities.

FEATURE DEFINITION PHASE

The next phase of software product architecture development is feature definition. This phase documents the key features of a software product or product family which would be capable of addressing the market needs / opportunities identified in the product positioning phase. This phase should also document which features are required to satisfy each of the identified market needs / opportunities.

4. These documents must contain enough detail to capture the short term goals and long term vision while avoiding descending into details which are really beyond the scope of ‘product positioning’. Striking this balance can be a non-trivial exercise involving considerable discussion.



The architectural diagrams produced earlier should be annotated and possibly updated to indicate which components support which features (it will almost certainly prove useful in the long run to augment the architectural diagrams with written documents which identify the features supported by each component rather than create diagrams which contain too much detail to be comprehensible).

Care should be taken to ensure that the features generally drive the architecture rather than the other way around. A certain amount of architecturally driven feature definition is probably unavoidable but do not allow the architecture to rule out a genuinely useful/necessary feature⁵. Consideration must also be given as to whether or not other feature sets might also address the identified market needs / opportunities.

The feature definition phase ends with a careful review of the identified features with an emphasis on ensuring that the market needs / opportunities identified earlier are actually satisfied by the identified features and that the architecture as described by the architectural diagrams at least appears to be capable of supporting the features.

It may prove necessary to return to the earlier high level architectural diagram phase. It may also prove fruitful to essentially combine the high level architectural diagram phase with the feature definition phase although care must be taken to ensure that both sets of deliverables exist before the combined phase is considered to be complete.

DETAILED ARCHITECTURE DESCRIPTION DEVELOPMENT PHASE

The next phase is to create the product's detailed software architecture by creating a reasonably detailed description of each of the product's architectural components (major software components) and interfaces (how the components communicate with each other). The focus should be on the roles and responsibilities of each component and the 'conversations' which occur across each interface.

It is critical that this description, in combination with the architectural diagrams, provide a clear and unambiguous description of the product's architecture. Errors, omissions and ambiguities in this description could easily cause the development effort to be delayed or even derailed entirely.

The detailed architecture description development phase ends with a careful review of the detailed architecture description with emphasis on ensuring that the described architecture is actually capable of implementing the features defined earlier.

5. A case can be made that there comes a time when the architecture is allowed to rule out a genuinely useful or even arguably necessary feature although a better argument is probably that such a feature has been ruled out as a result of a cost-benefit analysis rather than as a direct result of the architecture.

STAGED FEATURE ROLLOUT PLANNING PHASE

The next phase is to develop a staged feature rollout plan which describes the sequence in which the products features will appear in the marketplace. Each stage of the feature rollout will support ‘new’ features⁶ which address an expanding range of market needs / opportunities⁷. Primary consideration should be given to which features provide the greatest value to customers in the marketplace.

The first shipped product version must, of course, satisfy enough of the needs/desires of the marketplace to be commercially interesting. That said, one should keep in mind that the early release of a less featureful FCS product is likely to be of considerably greater value to the company than the late release of a more fully featured FCS product.

It is generally unreasonable to assume that the second major release will represent a “great leap” beyond the FCS release as a considerable amount of effort is likely to be spent during the second major release cycle dealing with issues which only appear once the product is “in the field”.

The phased feature rollout planning phase ends with a careful review of the resulting plan. Although the entire plan must be reviewed from the perspective of feature sequencing (i.e. are dependent features available when required) and growing value to the customer, particular attention must be paid to the planned feature set for the FCS release to ensure that it is achievable (taking reasonable account of the time and resources likely to be available), meets an adequate subset of the market needs / opportunities identified earlier to be successful in the marketplace, and sets the stage for a smooth transition into later releases.

TECHNOLOGY IDENTIFICATION PHASE

The next phase of the software architecture development process is technology identification where the technologies which will provide the foundation upon which the software product will be built are identified and documented. Particular attention must be paid to foundational technologies which the company may need to develop in-house. Each technology must be investigated in sufficient detail to ensure that either the current state-of-the-art is sufficiently robust and capable or a clear understanding exists regarding how the company will close the gap between the current state-of-the-art and where the technology needs to be to satisfy the company’s requirements.

The nature of technology being what it is, the focus of this effort should be on those technologies which are required for the FCS release although effort must also be expended

6. As identified in the feature definition phase.

7. As identified in the product positioning phase.



on technologies not required until later releases to ensure that an adequate appreciation exists of the risks associated with relying on any required advances in the technologies needed for later releases being available when needed.

The technology identification phase ends with a careful review of the identified technologies with a particular emphasis on any gaps which might exist between a technology's state-of-the-art and the demands that the project will place on the technology.

FCS RELEASE DEVELOPMENT SEQUENCING PHASE

The final phase of the software architecture development process is to identify the key development deliverables leading up to the FCS release and to document the sequence in which they will be designed, developed and integrated. Attention should be paid to opportunities for parallel development while taking into account the inevitable dependencies (e.g. we need X before we can complete Y).⁸

The FCS release development sequencing phase ends with a review of the development deliverable sequencing. Consideration should be given to the makeup of a suitable software development team both in terms of eventual composition and hiring priorities.

DID WE SAY FINAL PHASE?

The role of the software architect does not end with the completion of the FCS release development sequencing plan. Even the best laid plans and schemes are unlikely to survive the cold hard realities of the software development process or the continually expanding and evolving understanding of the product's position in the marketplace⁹.

The software architect will continue to be involved both in dealing with issues as they arise and in guiding and reviewing the software development efforts. The software architect must also ensure that development team members understand (at a level suitable to their role, skills, experience and outlook) both the role that they play on the team and the role that their deliverables will play within the product.

8. One of the flaws of most project planning tools seems to be that they assume that if X is required before Y then Y may not begin until X is completed. At least in a software development context, this is rarely the case. Unfortunately, attempts to finesse one's way around these sorts of limitations usually involves having to deal with allowing subproject Y having dependencies on internal deliverables of subproject Y with the result that the plan becomes incomprehensible.

9. General and later President Eisenhower once said that while the plan which results from a planning process is rarely very useful, he found the process itself to be incredibly valuable.

ON THE IMPORTANCE OF DOCUMENTATION

Each of the phases outlined above produces deliverables in the form of diagrams and/or documents. Each of the phases also ends with a review of these deliverables, a process which almost always requires reference to the deliverables of earlier phases. It is simply not possible to complete any of the above phases unless the deliverables actually exist in diagram/document form (there is simply no such thing as an architecture which exists but has not yet been ‘written down’).

Only when the time comes to end-of-life the product and disband the software development organization is the software architect’s task complete. Given the nature of the marketplace in which software lives, a successful company is unlikely to ever reach the point where the services of a software architect are no longer required. Since even software architects don’t live forever, considerable care and attention must be paid to ensuring that the architectural diagrams and documents remain up-to-date and that the project-knowledge developed by the architect is shared/disseminated in considerable detail across a suitable range of team members.

ARCHITECTURAL VS DESIGN DIAGRAMS AND DOCUMENTS

Just as a building’s architectural diagrams and documents are not engineering blueprints, a software product’s architectural diagrams and documents are not design documents. Architectural diagrams and documents focus on form and function (i.e. the building / software product’s overall structure and what the structure does). Engineering blueprints and software design documents provide a much more detailed view of how the building / software product is actually constructed.

Although a software architecture description is, without a doubt, a technical document, it should also be a document which can, at least in theory, be placed in front of an investor or a customer with the expectation that they will be able to comprehend or at least appreciate the overall form and function of the architecture.

CONCLUSIONS

Exemplary software architectures, like exemplary building architectures, do not just “happen”. They are the result of the efforts of the system’s software architect and are reflected in the architectural diagrams and documents which flow from these efforts.