Creating Software Modernization Roadmaps: The Architecture Options Workshop

Neil A. Ernst, Mary Popeck, Felix Bachmann, Patrick Donohoe
Carnegie Mellon University Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA
Email: {nernst,mpopeck,fb,pd}@sei.cmu.edu

***Abstract*—Architecture modernization requires a clear roadmap to transition to a new state. However, creating that roadmap is often difficult, particularly in complex settings. This paper investigates how one might systematically derive such roadmaps. We introduce the Architecture Options Workshop (AOWS), a systematic treatment to address the problems of moving from identified system risk themes to potential design options, and a roadmap for implementation. Many techniques present a range of options and leave it to stakeholders to select, or are tailored for detailed design processes. The Architecture Options Workshop, by contrast, is intended to resolve the question of what options to choose at a high level of abstraction. Applying a technical action research approach, we applied the AOWS to three different real-world systems. We describe the advantages—reasonably efficient, systematic architecture modernization—and some remaining questions for future research.**

I. Introduction

We describe the application of what we call the *Architecture Options Workshop (AOWS)*, an approach to help stakeholders identify the specific tasks that should be undertaken to improve or modernize the system's architecture (and related process and workflow approaches) and ultimately, meet their business goals. The output for an AOWS is a concrete roadmap for 2-3 iterations of the customer's development cycle, along with longer-term 'targets'. We specifically avoid overly constrained trajectories, instead identifying specific short-term milestones and longer-term targets, recognizing the path might well change. One difficult aspect of the modernization problem is that it has to run in parallel with evolution of the existing system, because of the time it take to modernize.

This paper describes the design science approach we used to create the AOWS, applying its techniques with three organizations. We describe its rationale, required inputs, workflow, and outputs, relating these to other design techniques. We are continuing to improve this process iteratively, and we conclude with a look at questions which have arisen after applying AOWS in realistic settings.

The contributions of this paper are

- Description of a novel technique, the AOWS, for moving qualitatively from options to roadmaps in software modernization.
- Guidance for applying the three phases of the AOWS including necessary preconditions and how to move to decisions.
- Demonstration of the AOWS in practice, in three real-world settings.

II. Research methodology

Our research methodology is technical action research [1] (TAR). TAR is a methodology that applies to the treatment validation phase of the design science lifecycle (i.e., problem investigation, treatment design, treatment validation, and treatment implementation [2]). A treatment in this context is an artifact that aims to solve a problem, which was (potentially) the subject of the problem investigation phase, producing knowledge questions and design problems. TAR in particular is a methodology—like statistical difference-making experiments—for understanding the effectiveness of the proposed treatment (AOWS, in this paper) in context. Particular to TAR is the notion that helping the client is part of the research approach. This makes it well-suited to the application of novel engineering approaches in settings where we nonetheless must deliver value to the client.

A. Treatment design

The problem the AOWS treatment addresses is modernization planning. Stakeholders have great difficulty developing specific roadmaps to get a software-intensive system from a current (typically problematic) state to a desired state. We have encountered this in many of our client projects while applying architecture techniques such as the Architecture Tradeoff Analysis Method [3] (ATAM), the Quality Attribute Workshop [4] (QAW) and the Mission Thread Workshop [5] (MTW), among others.

Certain commonalities exist across these methods:

- There is a focus on quality attributes, derived from business / mission goals, as decision drivers
- They are client and stakeholder focused, in order to let the domain experts drive the analysis;
- They are scenario-based in order to 'test' the system-to-be against actual scenarios that, if met, would allow the organization to achieve its business goals.

Many of the outputs of these techniques present stakeholders (customers) with a range of problems prioritized during the workshop. From there, the onus has been on the stakeholders to move ahead with identifying options for input into a design process, using the architecture challenges or risk themes[1] derived from quality attribute scenarios.

This is the case in modernization initiatives as well, where the system(s) are in an initial state (costly, hard to change, and otherwise unsatisfactory) and the stakeholders desire to move them to an improved target state. The Architecture Options Workshop is a systematic treatment to address the problems of moving from identified system risk themes to potential design options, and to provide a roadmap for implementation.

B. Context: Case study backgrounds

We conducted our options workshops with three cases we report on here. As is expected in technical action research, in between cases we reflected on lessons learned and incrementally improved the technique, then re-applied it to the new case. The organizations are listed in temporal order of our involvement.

Organization A is Bursatec, a financial services company aiming to build a new high-performance trade processing system, previously documented in Bachmann et al. [6], among others. This was our first application of the AOWS concept (in 2013).

Organization B is a multinational in the medical domain. The task was a modernization effort for a middleware component in a wider ecosystem of client apps and data stores, applied in 2015.

Organization C is a large governmental organization with regulatory, mission-critical information systems varying in age from 1-30 years, totaling millions of lines of code. We applied the AOWS in 2014-2016. The SEI was asked to aid in a modernization effort for several of these systems.

III. Treatment Design: Architecture Options Workshops

A. Overview and Rationale

The AOWS originated in response to problems we found in our work with modernization projects. Therefore the AOWS was iteratively designed to do the following.

1) Elaboration of a set of alternative technical architectures (defined by options derived from the risk themes, driven by business goals).
2) Create a single initial target architecture, selected using trade-off and prioritization from the set of possible architectures. In that way, you can return to the original set if things do not go as planned. This is not an end state but a 'next' state.

[1]An *architecture challenge* [5] applies to a system-to-be; a *risk theme* [3] applies to a system-in-use. We will use the term risk theme to refer to both.

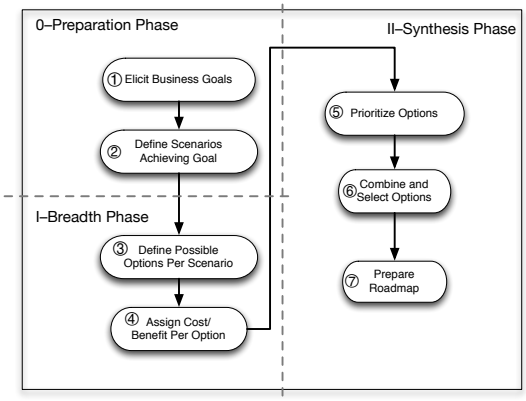

Fig. 1. The Architecture Options Workshop model

3) Documentation of rationale for most decisions using simple, qualitatively derived costs and benefits.
4) Priorities for roadmap phasing using a decision process that enforces decision making.

The Architecture Options Workshop also results in some guidance on redesigning the system architecture although it does not provide a full redesign of the architecture. The reality is that up-front, big bang modernization is rarely successful; the AOWS recognizes this by focusing on moving to an improved state and iterating from there.

Ideally, the Architecture Options Workshop will enable an Architecture Runway[7] that allows for iteratively enhancing the existing products with low impact to the business. Developers build according to the path laid out by the architects. Architects set the parameters for the developers to follow. The goal of the AOWS is to find out what those parameters are.

The AOWS approach is composed of seven steps (shown in Fig 1). These are divided into a preparation phase, a breadth phase and a synthesis phase. The preparation phase (phase 0) is a norming function to ensure no matter where the workshop starts, there is a set of scenarios to focus on in the subsequent phases.

Phase 0 - Preparation

1) Elicit the business goals that the organization is trying to attain.
2) For each business goal, define a set of scenarios that would allow that business goal to be achieved.

Phase 1 - Breadth

3) For each scenario, define a set of architecture options that would allow the scenario to be realized.
4) For each option, collect cost and benefits, to be used as the criteria for decision making.

Phase 2 - Synthesis

5) Prioritize the architecture options.
6) Combine and prioritize the high priority architecture options of all scenarios per business goal.
7) Create a roadmap by combining and prioritizing the architecture options across all selected business goals.

B. Preconditions

In any large modernization project, such as with Organizations B and C, (large meaning, among other things, multiple systems and multiple years of code), we suggest the following workflow:

1) Begin with an activity (such as a Mission Thread Workshop/MTW) that walks through a key business process fundamental to the organization's business goals. This improves understanding of the scope of the system of systems you are dealing with. This will identify architecture challenges traceable to individual systems. For organization B, this phase was done offsite. For organization C, we conducted 3 separate MTWs.
2) Perform an architecture analysis on one or more problem systems (identified in step 1). This might be an ATAM, for example. That results in risk themes partially traceable to problematic components (some are more generic, such as lack of documentation, or are process related). For Organization A, we conducted lightweight peer evaluations every iteration (see [6]). In B, we did a 5 day architecture analysis, and in C, two separate ATAMs on different systems identified as critical in the MTWs.
3) For a critical problem component, perform an AOWS, outlining solution options to resolve that risk theme (and create a roadmap for change). For Organization A, this followed on from the risks identified in the peer evaluations. Learning from those experiences, we conducted (largely in parallel) AOWS on components identified in the ATAM's for Organization B and C. For example,[2] in Organization B, a replacement system was desired and a key component was the middleware between clients and backend.
4) If necessary, (as was the case in Organization C), conduct a follow-up options workshop to identify more detailed solutions. From here one can rely on well-developed approaches for detailed design, such as Attribute-Driven Design [8].

This is a process that essentially decomposes a modernization problem into more manageable, yet high priority, targets for improvement. Note that at the first two steps, multiple problems are identified! That is acceptable: the purpose is not to boil the ocean but identify highly important (with respect to business goals) components that must be fixed. For example, the data service layer in a system of systems where many applications all write to a common database. At each level quality attribute scenarios are required to properly 'test' the systems/components under evaluation. The AOWS requires some identified set of business goals/drivers, risk themes, and quality attribute scenarios to drive the analysis.

Business goals are goals independent of the technical solution that focus the quality attribute evaluation and are critical in later prioritization of the options. An example of a business goal is "release new features within 2 weeks" or "increase market share by 10%". Without business goals, one cannot understand the context for prioritization.

Risk themes (in the modernization context) are business goal derived technical problems. For example, a risk is "deployment to production takes 10 business days", a theme is "slow deployment", impeding our goal of rapid feature release. Ideally these risk themes are derived from an explicitly architecture analysis phase, such as an ATAM, but could also be simple expressions of dissatisfaction with the current technical architecture. In a modernization context, the risks are with respect to doing nothing and continuing with the system-as-is.

Two of the instances of the AOWS we have conducted (Organization C and Organization B) have derived inputs from an earlier ATAM, in particular, the risk themes and business drivers. If those inputs do not exist, business goals in particular must be developed with the customer.

Scenarios, ideally quality attribute scenarios per [4], are the 'tests' [9] used to traverse (and reduce) the option space. These quality attribute scenarios should be derived from the business goals, or at least relatable to those goals. In our case study with Organization C, we also derived the quality attribute scenarios from risk themes found in previous ATAMs on the systems involved.

Scenarios in an AOWS are slightly different, in that they must be technical enough to stimulate technical option generation. For example, a scenario from our most recent experience was "Successfully develop and deploy a new application on average within X weeks after approval of requirements, where users are able to easily use this application." From this scenario, many technical architectural options are possible, including documenting the existing structure, devising automated tests, developing more powerful middleware, and improving the application frameworks. In organization C, then, we found ourselves (as the consultants) creating scenarios based on our knowledge of the context (we had all been involved since the initial context setting in the Mission Thread Workshops).

Depending on the customer, one may make the scenarios either a precondition or a part of the workshop. If there was a pre-existing and recent ATAM or QAW, for example, it may be easier for the analysis team (e.g., the SEI) to generate these scenarios, since they will have a suitable understanding from those previous engagements about key problems. If there is sufficient time, scenario generation could be done early on in the workshop. Regardless, the following conditions are vital in the scenarios:

- Agreement on what the important scenarios are.
- Scenarios must focus on solving the problem, i.e., achieving a business goal.
- Involvement of all appropriate stakeholders. We discuss this aspect in Section V.
- Relevant to system risk themes. For example, if a scenario is about deployment, there should be a mapping

[2]some details have been changed for confidentiality

to a risk theme (and thence to a business goal) about deployment challenges with the architecture. [3]

At the end, the set of scenarios created for use during either an ATAM, a QAW, or derived from business goals, describe the desired state in which the existing architectural problems are resolved. The risk themes help to understand and prioritize the potential architecture options. The remainder of the workshop activities now focus on determining options that would make these scenarios achievable.

The AOWS is designed to fit into the existing SEI portfolio of software architecture analysis and design techniques as shown in Figure 2. An instance of this approach as applied to Organization C is shown in Fig. 3.

C. Stakeholders

As we described at the end of the previous section, it is vital that the "right" people be invited and participate in the AOWS. There should always be two groups of stakeholders present. One group represents the user community, the stakeholders who have to live with the architecture options chosen. The second group represents the stakeholders that are responsible for the product, the stakeholders who define and implement the options chosen.

The following is a rough guide to who must be there:

- Anyone who has the requisite high-level technical knowledge about both the organization and the technical infrastructure. They are the ones who know if an option will be possible. For example, what frameworks and tools are feasible given the current organizational climate and ability? *Examples: chief architects, enterprise architects.*
- People who are up to date on architectural tactics, patterns, and best practices. This could possibly be people who only know technical solutions, and don't care about organizational capability. An option could be technically possible but not organizationally feasible. *Examples: developers, testers, contractors.*
- Those with political power to approve workplans. Since the AOWS creates a roadmap, it is vital that the roadmap be realistic, and that the decisions it embodies be committed to by the organization. *Examples: program managers, technical leads, technical managers.*
- Anyone who understands the business processes and end-user concerns. They check suggested options for feasibility within the organization. *Examples: business managers, supervisors, end-users, system administrators, field service engineers*

[3]This may seem somewhat backward from the typical flow of an ATAM, for example. There one starts with business goals, and then develop scenarios and finally risk themes. The difference is that in a modernization context, we expect to have a set of all three. Without goals, we cannot prioritize; without scenarios, we cannot focus our analysis; without risk themes, we cannot understand what scenarios to focus on

This is potentially a large number of stakeholders. In our experience, what is more important is that a sufficiently representative group able to play the major roles are present. For example, with Organization C our AOWS session consisted of 10-12 people, including technical architects, contracted developers, program managers, and technical domain experts. With Organization A, options were evaluated in smaller teams, one chief architect, and 5-8 junior architects.

D. Phase 0 - Preparation

1. Elicit the business goals that the organization is trying to attain. Well-chosen goals point an organization in the desired direction and help to keep them on track. In the absence of such goals, it may be possible to use outstanding architectural problems, which are likely impeding the business from attaining their (hidden) business goals. For Organization C, these proved challenging to gather, so extra time was allocated to find them.

2. For each business goal, define a set of scenarios that would satisfy that business goal. During the AOWS, scenarios are produced for each business goal by the stakeholders that describe the desired state of the business. A diverse group of stakeholders is critical in order to ensure that at least the most important aspects of attaining the business goal are considered and how the potential solutions affect different members of the organization. For Organization A, these scenarios were vital to guide more detailed design.

Having collected goals and scenarios, we can continue to Phase 1. Note that these do not usually occur on the same day/session.

E. Phase 1 - Breadth: Option Brainstorming

3. For each scenario, elicit a set of architecture options that would allow the scenario to be realized. Architecture options are anything from a small architecture transformation (such as implementing an architecture tactic[4]) to integrating a big application or framework into existing systems. Options are solutions that could be used to achieve the scenario. Every workshop participant is encouraged to provide their options, in case their view on how to solve the problem differs from the other stakeholders. At this point in the AOWS, all options are considered no matter how unlikely they are to be implemented. That is why this phase is called the breadth phase. For Organization C, we had around 6-7 options per scenario, diminishing over time as options were applicable to multiple scenarios (e.g., document the architecture).

4. For each option, collect qualitative pros and cons to be used as the criteria for decision making. Stakeholders are first asked to name all the positive attributes associated

[4]An architectural tactic is a building block for a solution, and books like Software Architecture in Practice [10] lists many possible tactics for various quality attributes. For example, one might Bound Queue Sizes to improve Performance.

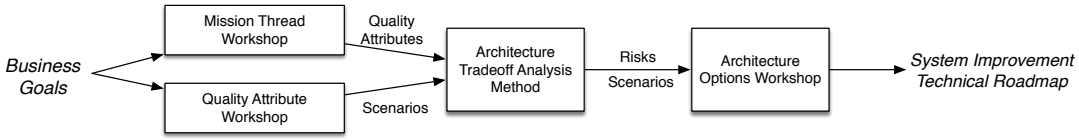

Fig. 2. One possible analysis and design approach with SEI Techniques

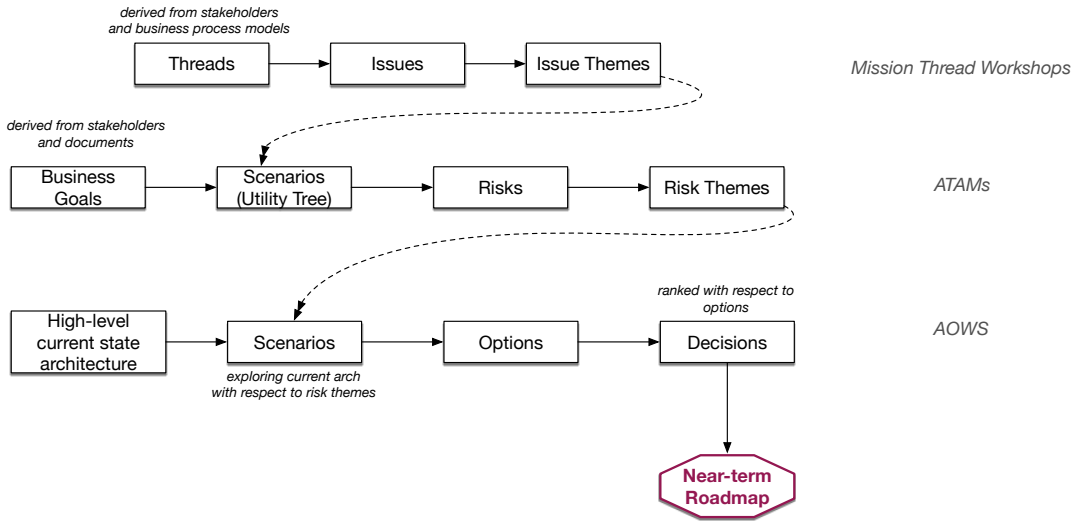

Fig. 3. Instantiation of Fig. 2

with the option being implemented. Then, they are asked to name all the negative attributes. These are enumerated for all to see and to evaluate. For example, if an option to manage the scenario of "Deploy to production in 24 hours" is to document the build process, then a possible pro is that this documentation also supports understanding the production environment.

For each option, stakeholders are asked to rate it on a High-Medium-Low scale for both Cost and Benefit. Sometimes stakeholders need to assign monetary values to the rankings (e.g., High $>$ \$1 million, Low $<$ \$100k) before being able to rate effectively. In the AOWS we ran, we did not conduct any triangulation exercise to ensure everyone was on the same page, but our stakeholders did this informally. Cost is defined as how resource intensive it will be to implement, and Benefit is defined as how much improvement the organization will see if it is implemented. For the AOWS, we deliberately keep the list of criteria on which to evaluate options simple and qualitative. We elaborate on prioritization in Section V-A.

At the end of Phase 1, we have a spreadsheet with tuples $\langle$Option, Scenario, Pro, Con, Cost, Benefit$\rangle$. For Organization C, our 1 day workshop produced 20 different options for 5 prioritized scenarios. The next phase takes this input and develops the roadmap.

F. Phase 2 - Synthesis: Roadmap Planning and Prioritization

5. Prioritize the architecture options. In Organization B and C, we struggled with prioritization. It is not a straightforward task, since options have dependencies, options apply to different parts of the system, and can be effective at different points in time. What we actually did was classify the options (future things / legacy things / supporting things), we drew dependencies (if you choose this you have to choose this too), and we tagged options according to how likely the organization is to do it.

Ultimately it was up to the analysis team to move from Options to a decision tree (described in the following section). For this tree, some decisions can be concluded quickly based on the collective prioritization. Options rated in Step 4 as High in Benefit and Low in Cost, then Medium Benefit and Low Cost, and then High Benefit and Medium Cost, tended to be fairly easy to agree to act on. Likewise, Options that rated Low in Benefit and High in Cost would be the least likely to ever be implemented and might be excluded from the roadmap entirely.

The most difficult options are the ones rated equally for benefit and Cost (Low/Low, Medium/Medium, High/High). Here no clear decision criterion might be found. This may lead to deferring decisions and adding new options like prototypes or feasibility studies to gain more insights into the problem which hopefully leads

to better decision criteria. All options are included in the AOWS documentation for completeness to show the breadth of options considered. Options not chosen may be considered in the future if technology changes result in a modification to their Cost/Benefit ratings. It is important to capture decision rationale so that if circumstances change, the decision may be revisited. For example, with Organization C, we identified a "Big Bang" option that was essentially a rewrite of their entire system. While this option was of High benefit, its cost was High++. It is important to document that this option was considered and rated, as it is tempting for new people coming onto the team and examining the roadmap to ask the question "Why can't we scrap everything and start over?"

6. Combine and prioritize the high priority architecture options of all scenarios per business goal. In choosing the "best" option or set of options to achieve each scenario, the tasks to be performed are narrowed down. This makes it easier to move forward in meeting the business goals. If there is no obvious "best" option, then a feasibility study may be needed to gather more information leading to a decision point regarding a set of options, some or all of which might be selected. These tasks and decision points all help to form a roadmap.

For example, in Organization C, a decision point was introduced to choose between inserting a service layer and/or extracting embedded business logic from the data layer. The high priority tactics of all the scenarios pertaining to one business goal are combined. This is done for each business goal. Note that in some cases, an option supports more than one business goal, and this would improve its ranking. Even a high priority option may not be selected if, for example, doing so depends on selecting a lower-ranked option being chosen, or if it is exploratory and outside the time horizon of the roadmap.

7. Create a roadmap by combining and prioritizing the architecture options across all selected business goals. Roadmaps should span two years or less. If the roadmap needs to be longer, then it should be iterative. How much is accomplished in each iteration will need to be determined and shorter term goals will need to be set. Finally, for each milestone on the roadmap, a champion must be identified. The champion should then define a timeline and measurements that illustrate whether or not the task was successful at implementing the option. The stakeholders should all buy-in to the idea that the roadmap is credible and that once all the milestones are completed the business goals will be achieved.

In the Organization A (Bursatec) case, this entailed handing the chosen scenarios and chosen options to a team of designers, who then created a detailed UML-notated design. In Organization C, the follow-up to the roadmap is more concrete design workshops on certain chosen options (e.g., how to extract business logic).

G. Decision Tree

Performing Step 6—combining the options for all scenarios per business goal—will be different for every AOWS. However, in all cases some type of decision tree is needed. This is a directed acyclic graph with high-level, most constraining decisions at the top. For Organization C, these high-level decisions were whether to go with replacement or incremental re-architecting. From there, the design tree is conditional on higher-level choices, e.g., given incremental was chosen, what are the remaining options?

Options are also related, including dependency and exclusion relationships. Finding these relationships is clearly amenable to some form of automation. Creating the decision tree is non-trivial, and although the search itself is naturally logarithmic in the size of the options, enumerating the tree is not, and we are not yet able to guarantee the chosen solution set (i.e., the chosen options) are Pareto-optimal with respect to our criteria, e.g., quality attribute scenarios, cost/benefit. We have currently used three approaches for creating this tree manually.

Approach 1: Sort options into exploratory and technical tasks. Exploratory tasks could be grouped together.

Approach 2: Break decisions down by system component. If a set of options are all related to the user interface, while other options are related to the data layer, user interface and data layer are natural groupings. Be careful to keep the cross-cutting nature of the scenarios (which will almost certainly traverse layers) separate from the more hierarchical nature of options.

Approach 3: Group options by level of detail. This is the approach we took with Organization C. Our first decision point was the general approach to modernization: rewrite or refactor? Once that decision is made, some of the other options are precluded. For example, if the stakeholders choose to not do big-bang rewrites, options about starting from scratch for individual systems no longer apply. We found that for a one day workshop (that is, the decision phase (2) of the AOWS), fifteen or so decisions was tractable.

It is obvious how to create a decision tree when the decision is easy. Where stakeholders need help is when it is a tough decision (i.e., very similar pros and cons). In this case, an approach that was useful in Organization B and C was to create a separate option that conducts a feasibility study or bakeoff to gather more data to simplify the decision. For a short workshop such as the AOWS, going into more detail than "costs/benefits" does not scale. However, the presence of too many middle-range options (e.g., M/M) in a single decision suggests that more investigation is necessary.

In this case a larger set of specific criteria can be useful. For example, our colleague William Wood's approach [11] used 23 individual criteria (more precisely, risk factors) for ranking options, based on OMB's Exhibit 300 [12]. These criteria include the cost to back out a change, the ability to

manage the investment, and security risks. Other criteria may be relevant depending on context.

IV. Outcomes

By the end, the stakeholders should all be committed to a roadmap that is achievable, measurable, and will allow the organization to attain its business goals. The facilitator should have a sense for commitment during the workshop, in particular, whether any factions are opposed to it. A facilitator might consider conducting a paper survey to see if there are any stakeholders in disagreement or hold one on one discussions with various stakeholders to ascertain commitment.

The goal of the Architecture Options Workshop is to create a reasonable vision for the evolution of the systems towards mitigating the important risks uncovered during an ATAM and allowing an organization to attain its business goals. This vision includes a roadmap with identified releases for the near term and a set of features and capabilities to implement for the long term.

A. The Roadmap

Preforming Step 7—Create a roadmap—will also vary across AOWS applications. Roadmaps that are only working to achieve one business goal, which can be accomplished in a shorter timeframe, will be easier to construct than roadmaps supporting multiple business goals over longer timeframes. Checkpoints should be inserted in the roadmap to review progress. If you are not making progress, then you should stop executing the roadmap. Timeframes should be based on organizational cadences, but going beyond 2 years is probably of little value, since things will have changed substantially at the AOWS level of detail. Some problems may be larger than the timeline can cover. Create options that decompose that problem into smaller steps. Some possible techniques for "slicing" the problem can be found in [13].

For Organization C, doing this results in a roadmap like that in Figure 4. Note that specific milestones are dependent on stakeholders to schedule. They must consider the following: what resources, fiscal year deadlines, available funds. For example, Organization C had little capacity to perform most of the tasks in their remaining fiscal year budget. Some tasks had to be delayed until the next fiscal year. Earlier we mentioned that a difficulty in modernization is synchronizing that work with ongoing maintenance of the existing system. In Organization C, this has meant creating separate task orders. In the future, a roadmap alignment with existing IT strategy would be helpful (currently this remains tacit).

In Figure 4, M stands for milestone. There is a champion, due date and measure associated with each of the 11 milestones shown in Figure 4. It should also be noted that some of the early milestones are for the gathering of data and comparing of different approaches to determine how to perform the tasks necessary to achieve later milestones.

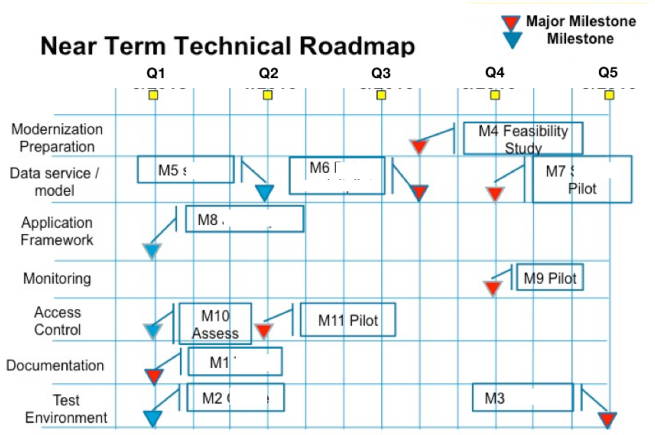

Fig. 4. Sample Technical Roadmap for Organization C. Some details hidden for confidentiality.

The implication of this roadmap is that in the short-term, achieving these milestones will move the organization closer to a more mature architectural approach.

V. AOWS: Observations and Future Improvements

As part of our design science approach, we continue to improve the AOWS treatment. We remark on issues that appeared in all three AOWS we conducted, as well as needed improvements.

A. Measuring Certainty

Each option is given two ratings. The first, "Cost", is the stakeholders' collective interpretation of the amount of resource effort required to implement that option. This includes time, money, and socio-political challenges. The second, "Benefit" is the perceived advantages, in terms of satisfying business goals. Each is rated as High, Medium or Low (HML). In practice, we see stakeholders acknowledging that a HML does not necessarily capture the true range of values for comparing options. For example, while the Cost may well be "High", in practice sometimes it is so high that it means "Never Happen", versus another option that is "High" in Cost but might be done if sufficient resources are found.

The importance of these rankings is in deciding on a course of action. In particular, things with High benefit and Low cost will always be chosen. In our experience, of course, clients do not need help understanding these decisions. Help is needed when costs are Medium/High and benefits are Medium/High, because that is much less obvious. In practice the decisions about these conflicted options is organizationally dependent. In some places, High Benefit is sufficient to motivate an organization to pursue at least gathering more information. In other (risk-averse) places, High Cost is sufficient to preclude this option, even if the benefit is High.

The cost/benefit considerations could clearly be much more nuanced (e.g., with probability measures). However, in the context of a 2-day workshop, gathering more detail did not seem justified in terms of decision-making (most of the Organization C options, for example, are high-level and require more knowledge).

B. Agile and Iterative Architecture and Options

An important consideration in building an architecture roadmap is the length and frequency of delivery iterations. In a high-maturity organization, completion of an increment may force an update to the information in the roadmap. For example, a team will report that a particular option will not work in practice due to previously unseen obstacles. In this sense, benefits and costs do not take into account incrementality; we develop our prioritization with the knowledge at the time of the workshop. This implies three things.

1) The roadmap must be updated as knowledge is gained about option-specific information.
2) Roadmaps should make clear what tasks are information-gathering and may change the roadmap (e.g., an architecture spike, outlined in [13]).
3) Roadmap developers should be aware of "unknown unknowns". A clear lesson of agile development is that prescriptive and overly detailed roadmaps never work. We tried to describe destinations instead of specific routes.

A roadmap should include a *loopback task*. If a roadmap milestone is not met, or if it fails, then you must go back to the AOWS and revisit the options, i.e., if measures show that improvement was not attained. Therefore, information must be preserved, and there needs to be a trigger when something fails or a milestone cannot be met that forces the organization to go back and repeat. Similarly, an important cost to keep in mind when doing the decision prioritization is the "back-out cost", that is, the cost to undo the work of implementing that option. In Organization C, this would include the cost of canceling a maintenance contract with a framework provider. This loopback task is particularly important when iterations are long and infrequent, which is common in government organizations.

C. Involving Stakeholders

Qualitative approaches, such as the AOWS or the ATAM, have outcomes that may vary based on who is present on the assessment team and facilitators, the technical competence of the staff, and the internal dynamics of the individuals. To address this we are working on ways to record conversations, find the right level of detail, and organizational structures.

It is critical to create templates in advance to record decisions and propose roadmaps. The important thing during a workshop is to use a template to accurately record what participants are saying. We project the notes on a whiteboard so that the stakeholders can confirm that what is being recorded is what they intended. Often, this scribing generates new discussion as differences in meaning come to the surface. Different templates are used to present the option decisions and roadmap outcomes to the stakeholders at the end or after the workshop. Even so, there are plenty of non-verbal cues that can be picked up only by detailed observation, such as facial cues and deferential behaviour. We would like to incorporate more knowledge on group decision-making [14] in order to ensure the best conversations arise.

Hitting the right level of technical detail can be a challenge. In particular, it can be difficult to find the correct level of technical abstraction to discuss options. If the technical detail is very deep, non-technical stakeholders will lose interest; if it is very high, then the option will not be refined sufficiently. At the same time, detailed discussions of task orders and funding is not interesting to people who have no say in such things.

Which stakeholders should be in the room? Small groups can get more done faster, but if key stakeholders are missing, the final options/roadmap will be deficient and lack buy-in. We are working on techniques to pre-seed discussions with ground-truthed, distributed analysis of criteria spaces. For example, a phase 0 task to collect each individual's understanding of key attributes (cost, schedule, security, viability, ...). Other issues include groupthink, Conway's Law ("organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"), and lack of common terminology (e.g., layers vs. tiers).

D. Pareto-optimality

The outcome of the AOWS is a single roadmap, with parallel task and milestones, that nonetheless converge on a specific solution. It is not clear that this is the optimal solution, depending as it does on the skill of the analyst team and the stakeholders who are involved. Other approaches for identifying the Pareto frontier of dominant solutions could be useful.

E. Treatment Validation

The TAR methodology suggests one of the next research phases is treatment validation. To some extent this is implicit in using the AOWS in actual client projects. To date our focus is iterating the treatment (AOWS) design. We have done validation implicitly, by informal conversation with our project partners (who seem satisfied) but a more in-depth analysis (e.g., Did the roadmap work? What options were omitted that became important?) would be useful from a generalizability point of view. We did not poll our participants on these meta-questions about the process in the moment, aside from some rebasing questions (e.g., "Is everyone happy with how things are progressing"). Our internal validity is threatened by a few confounders. One is we believe participants are contributing in good faith.

There are no obvious other factors that could have caused our results, but our model does not explicitly consider social factors, which certainly affect how our participants interact and respond. Finally, our external validity is strengthened by the three distinct cases; that said, making analogic inferences to other cases requires at least a case with a modernization problem of similar scale.

VI. Related Work

We consider two categories of related work. One category is focused on architecture design support, but ignores larger-scale modernization issues. The other category describes specific analytic approaches for generating modernization options, but does little to support roadmapping (i.e., selecting among those options). Our position is that the AOWS is a necessary intermediate step between analysis and more detailed design.

A. Architecture Design Support

There are numerous proposals for doing disciplined design. The SEI's own Attribute-Driven Design (ADD) technique [8] creates architectural solutions to quality attribute scenarios. It is most appropriate for taking the output of the architecture options workshop (AOWS), i.e., a specific option, and designing a less-abstract solution. For example, for Organization C, one option is to migrate to a 3-layer architecture. The specifics of how to design that new architecture, including which frameworks to use, are well-suited to an ADD-style approach [15].

Similarly, specific guidance is available as books on architecture patterns and tactics, such as the seminal Pattern-Oriented Software Architecture series [16], or somewhat more generally, introductory books such as Shaw and Garlan [17].

There are also methodologies for designing greenfield systems. Academically, Tropos [18] is a methodology for moving from early requirements to agent-oriented software implementations. Design is supported by assigning requirements to agents in the model. It is however committed to a specific architectural approach, namely agent-oriented platforms. Industry standards such as AADL [19] likewise provide assistance in creating robust architectures for particular sets of design challenges (primarily safety-related).

It is important to emphasize that the contribution of the AOWS is not in how to choose between various architectural approaches—there are existing tools for doing this—but rather, the process for moving from quality attribute scenarios to implementable roadmaps.

B. Architecture Modernization Analysis

Several SEI technologies, such as the Mission Thread Workshop and Architecture Tradeoff Analysis Method, fit well here (and we made use of them in various ways, described above).

In the early phases of system design, systems engineering approaches such as tradespace analysis [20] can be helpful in understanding high level quality attributes and business goals. Other systems engineering approaches explore the issue of options optimization in more mathematically rigorous fashion [21]. Work on software migration patterns hold promise for generating options necessary in Step 3 of the AOWS, such as the Dublo pattern [22]. Similarly, there is much useful early design analysis research in the requirements engineering field. i* for organizational dependency modeling is one such approach [23].

Tools such as Lattix[5], which provide insight into dependency, and by extension, modifiability properties of the architecture, support analysis and modernization as well, and this is perhaps where industry is most concentrated in conducting modernization analysis. We have stayed away from tools since the AOWS is more concerned with implementation independent options, although the properties of the implementation (such as vendor choices) clearly constrain the solution space.

Finally, the OMG's Architecture-driven Modernization initiative (http://adm.omg.org) is an attempt to apply model-driven software engineering to modernization problems. It seems like a promising approach, particularly for organizations that maintain reasonably complete architecture models or do architecture recovery. This was not the case in our three case studies, however.

VII. Conclusions

Technical action research is about discovering the mechanisms at work in the context, or in other words, understanding the relationship artifact x context => effects by mechanisms [2]. In this paper:

- Our **artifact** (treatment) was the AOWS. We introduced a new way to identify architectural options during system engineering or modernization activities, filling the gap between early design analysis and late-stage design assistance.
- Our **contexts** were the three organizations to which we applied AOWS.
- The **effect** we observed in each setting was a roadmap that had the approval and understanding of key stakeholders.
- Finally, our **mechanisms** include the three-phased AOWS approach, the necessary inputs and outputs it produces.

The Architecture Options Workshop (AOWS) is a structured approach to understand what design options will satisfy business goals, and results in an architecture roadmap that will, all else being equal, bring the system towards the desired target architecture. We continue to refine our approach as we learn about what works and what does not.

VIII. Acknowledgments

This material is based upon work funded and supported by the Department of Defense under Contract

[5] www.lattix.com

No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. [Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. Architecture Tradeoff Analysis Method® and ATAM® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. DM-0003209

References

[1] R. Wieringa and A. Morali, "Technical action research as a validation method in information systems design science," in *Design Science Research in Information Systems*, Las Vegas, 2012, pp. 220–238.

[2] R. Wieringa, *Design Science Methodology: For Information Systems and Software Engineering*. Springer, 2014.

[3] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *International Conference on Engineering of Complex Computer Systems*, Aug. 1998, pp. 68–78. DOI: 10.1109/ICECCS.1998.706657.

[4] M. R. Barbacci, R. J. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, and W. G. Wood, "Quality attribute workshops (QAWs)," Software Engineering Institute, Technical Report CMU/SEI-2003-TR-016, 2003.

[5] M. Gagliardi, W. Wood, and T. Morrow, "Introduction to the mission thread workshop," SEI-CMU, Tech. Rep. CMU/SEI-2013-TR-003, 2013.

[6] F. H. Bachmann, L. Carballo, J. McHale, and R. L. Nord, "Integrate end to end early and often," *IEEE Software*, vol. 30, no. 4, pp. 9–14, 2013. DOI: 10.1109/MS.2013.77.

[7] D. Leffingwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional, 2011, ISBN: 978-0321635846.

[8] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, and W. Wood, "Attribute-driven design (ADD), version 2.0," Software Engineering Institute, Tech. Rep. CMU/SEI-2006-TR-023, 2006.

[9] F. Bachmann, "Give the stakeholders what they want: Design peer reviews the ATAM style," *Crosstalk: Journal of Defence Software Engineering*, Nov. 2011.

[10] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed., ser. SEI Series in Software Engineering. Addison-Wesley Professional, 2012, ISBN: 0321815734.

[11] W. Wood, *Information technology systems modernization*, Blog post, Apr. 2015. [Online]. Available: http://blog.sei.cmu.edu/post.cfm/information-technology-systems-modernization.

[12] Office of Management and Budget, "Instructions for completing the OMB Exhibit 300, capital asset plan and business case," Office of Management and Budget, Tech. Rep., 2006.

[13] N. A. Ernst, S. Bellomo, R. L. Nord, and I. Ozkaya, "Enabling incremental iterative development at scale: Quality attribute refinement and allocation in practice," Software Engineering Institute, Tech. Rep. CMU/SEI-2015-TR-008, 2015.

[14] V. Smrithi Rekha and H. Muccini, "A study on group decision-making in software architecture," in *IEEE/IFIP Conference on Software Architecture*, Apr. 2014, pp. 185–194. DOI: 10.1109/WICSA.2014.15.

[15] H. Cervantes, P. Velasco-Elizondo, and R. Kazman, "A principled way to use frameworks in architecture design," *IEEE Software*, vol. 30, no. 2, pp. 46–53, 2013. DOI: 10.1109/MS.2012.175.

[16] F. Buschmann, K. Henney, and D. Schmidt, *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*. Wiley, 2007, vol. 4.

[17] M. Shaw and D. Garlan, *Software architecture: Perspectives on an emerging discipline*. Prentice Hall, 1996.

[18] P. Giorgini, P. Bresciani, F. Giunchiglia, J. Mylopoulos, and A. Perini, "Tropos: an agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, pp. 203–236, 2004. DOI: 10.1023/B:AGNT.0000018806.20944.ef.

[19] P. H. Feiler, B. A. Lewis, and S. Vestal, "The SAE architecture analysis & design language (AADL): A standard for engineering performance critical systems," in *International Symposium on Computer Aided Control System Design*, 2006, pp. 1206–1211.

[20] H. Bagheri, C. Tang, and K. Sullivan, "Trademaker: Automated dynamic analysis of synthesized tradespaces," in *International Conference on Software Engineering*, 2014, pp. 106–116.

[21] A. Engel and T. R. Browning, "Designing systems for adaptability by means of architecture options," *Systems Engineering*, vol. 11, no. 2, pp. 125–146, 2008. DOI: 10.1002/sys.20090.

[22] W. Hasselbring, R. Reussner, H. Jaekel, J. Schlegelmilch, T. Teschke, and S. Krieghoff, "The Dublo architecture pattern for smooth migration of business information systems: An experience report," May 2004, pp. 117–126. DOI: 10.1109/ICSE.2004.1317434.

[23] E. S. K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *International Conference on Requirements Engineering*, Annapolis, Maryland, 1997, pp. 226–235.