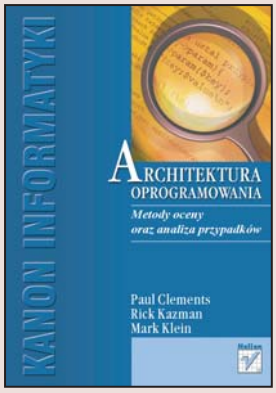
Architektura oprogramowania. Metody oceny oraz analiza przypadków



Autorzy: Paul Clements, Rick Kazman, Mark Klein
Tłumaczenie: Bartłomiej Garbacz
ISBN: 83-7197-929-0
Tytuł oryginału: Evaluating Software Architectures
Format: B5, stron: 330

Podstawą każdego systemu oprogramowania jest jego architektura, czyli sposób, w jaki oprogramowanie jest tworzone z niezależnie rozwijanych komponentów oraz mechanizmy interakcji i wzajemne zależności pomiędzy nimi. Jeśli system ma być tworzony przez więcej niż jedną osobę, właśnie architektura pozwala im na wzajemną komunikację. Choć architektura jest postrzegana jako jeden z najważniejszych aspektów rozwoju współczesnych systemów, to jej ewaluacja niemal nigdy nie staje się standardową częścią procesu rozwojowego.

Wykorzystując wyraźnie określone związki między decyzjami dotyczącymi architektury projektu a wynikającymi z nich właściwościami oprogramowania, niniejsza książka opisuje metody ewaluacji architektury oraz przypadki ich praktycznego zastosowania. Książka „Architektura oprogramowania. Metody oceny oraz analiza przypadków" prezentuje podstawową wiedzę pojęciową z zakresu metod oceny architektury i stanowi podręcznik opisujący krok po kroku proces takich ewaluacji przeprowadzanych w przypadku wielu organizacji rządowych i przemysłowych.

Architektura oprogramowania to gwałtownie rozwijająca się dziedzina badań i działań praktycznych w zakresie inżynierii oprogramowania. Książka prezentuje w szczególności trzy metody jej ewaluacji:

- metodę analizy kompromisów architektonicznych (Architecture Tradeoff Analysis Method, ATAM)
- metodę analizy architektury programowej (Software Architecture Analysis Method, SAAM)
- czynne przeglądy projektów pośrednich (Active Reviews for Intermediate Designs, ARID)

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

3

ATAM — metoda ewaluacji architektury 65

4

System kierowania polem walki — pierwsza analiza przypadku dla metody ATAM 107

5

Istota atrybutów jakościowych 129

6

Analiza przypadku wykorzystania metody ATAM 147

7

Wykorzystanie metody SAAM w ewaluacji przykładowej architektury 231

8

ARID — metoda ewaluacji architektur częściowych 259

9

Porównanie metod ewaluacji architektur oprogramowania 273

2

Ocena architektury oprogramowania

Jeśli opracowujesz swoją architekturę w pośpiechu, możesz później tego żałować.

— Barry Boehm
fragment przemówienia: And Very Few Lead Bullets Either

Kwestią o podstawowym znaczeniu jest określenie, czy dla danego oprogramowania wybrano odpowiednią architekturę. Dokonanie odpowiedniego wyboru nie doprowadzi do katastrofy, ale raczej utoruje drogę do bezproblemowego rozwoju i w efekcie opracowania udanego produktu.

Jest to złożony problem i na obserwowane efekty działań ma wpływ wiele czynników. Podstawą każdego systemu programistycznego jest jego architektura. To ona właśnie decyduje o możliwościach i ograniczeniach związanych z wszelkimi aspektami jakościowymi systemu. Modyfikowalność, wydajność, bezpieczeństwo, dostępność, niezawodność — wszystkie te cechy zostają określone po zdefiniowaniu architektury. W przypadku systemu o nieprawidłowej architekturze właściwości tych nie poprawi żadne strojenie lub zastosowanie pewnych sztuczek implementacyjnych.

Mówiąc wprost, architektura stanowi podstawę sukcesu danego systemu. Zapewne dużo lepiej jest z góry wiedzieć, czy dokonało się trafnych wyborów zamiast czekać niemal do momentu zakończenia prac nad systemem i dopiero wówczas mieć możliwość określenia, czy spełnia on stawiane wymagania. Kupując system lub płacąc za jego opracowanie z pewnością warto posiadać pewne gwarancje, że od samego początku prace będą posuwać się w odpowiednim kierunku. Natomiast architekt bez wątpienia doceni możliwość weryfikacji własnych przeczuć i doświadczeń, co pozwoli mu na nabranie przeświadczenia, że zaufanie pokładane w utworzonym projekcie opiera się na solidnych podstawach.

Do niedawna nie istniały niemal żadne ogólnoużytkowe metody służące do weryfikacji poprawności architektury oprogramowania. Stosowane sposoby oceniania, o ile w ogóle istniały, były niespójne, formułowane *ad hoc* i nie zapewniały powtarzalności wyników. Z tego względu nie były one zbyt wiarygodne. Okazuje się jednak, że można znaleźć lepsze rozwiązania.

Niniejsza książka stanowi przewodnik po metodach oceny architektur oprogramowania. Jej treść oparto na zestawie trzech metod, które zostały opracowane w Software Engineering Institute i które mogą znaleźć zastosowanie w przypadku każdego systemu intensywnie wykorzystującego oprogramowanie:

- metoda analizy kompromisów architektury (ATAM, *Architecture Tradeoff Analysis Method*);
- metoda analizy architektury oprogramowania (SAAM, *Software Architecture Analysis Method*);
- aktywne przeglądanie projektów pośrednich (ARID, *Active Reviews for Intermediate Designs*).

Wszystkie z wymienionych metod mają solidne podstawy i były stosowane przez lata w przypadku dziesiątków projektów o różnym poziomie złożoności oraz w szerokim zakresie obszarów zainteresowania. Dzięki nim jasnym się stało, że nadszedł czas, aby ewaluacja architektury oprogramowania stała się jednym ze standardowych etapów każdego paradygmatu konstrukcyjnego. Tego rodzaju oceny stanowią rozsądny sposób zmniejszania ryzyka i są względnie niedrogie. Rekompensatę za włożony wysiłek stanowi możliwość uniknięcia kosztownych błędów i nieprzespanych nocy.

W poprzednim rozdziale przedstawiono zarys pojęcia architektury oprogramowania, w niniejszym zaś zostaną zaprezentowane podstawowe wiadomości dotyczące oceny architektur. Zdefiniowano tu pojęcie architektury oprogramowania oraz omówiono właściwości, pod względem których można (lub nie można) dokonywać oceny danej architektury.

W pierwszej kolejności należy zdefiniować przedmiot oceny:

> *Architektura oprogramowania danego programu lub systemu informatycznego to struktura lub struktury systemu, które obejmują komponenty programowe, widoczne na zewnątrz właściwości tych komponentów oraz istniejące między nimi relacje.* [Bass 98]

Przez właściwości „widoczne na zewnątrz" należy tu rozumieć założenia, które mogą przyjmować inne komponenty względem danego komponentu, na przykład w kwestii oferowanych przez niego usług, charakterystyk wydajnościowych, obsługi błędów, wykorzystanie zasobów współużytkowanych i tak dalej. Powyższa definicja sugeruje, że architektura oprogramowania musi abstrahować od pewnych informacji dotyczących systemu (w przeciwnym wypadku nie miałoby sensu zajmowanie się architekturą — byłoby to po prostu badanie systemu jako całości), ale jednocześnie zapewniać taką ilość informacji, aby mogły one stanowić podstawę przeprowadzanych analiz, podejmowanych decyzji, a przez to redukcji ryzyka (patrz ramka *Co jest związane z architekturą oprogramowania?*).

Architektura definiuje komponenty (takie jak moduły, obiekty, procesy, podsystemy, jednostki kompilacji i tak dalej) oraz odpowiednie, występujące między nimi zależności (takie jak wywołania, przesyłanie danych, mechanizmy synchronizacji, użycia, zależności, konkretyzacje i wiele innych). Architektura jest rezultatem wczesnych decyzji projektowych, których podjęcie jest niezbędne, aby możliwe stało się wspólne skonstruowanie przez grupę ludzi systemu programistycznego. Im większa lub bardziej rozproszona jest owa grupa, tym większego znaczenia nabiera architektura (należy jednak podkreślić, że grupa ta wcale nie musi być bardzo duża, aby architektura oprogramowania stała się istotna).

Jedną z uwag, dotyczących architektury z rozdziału 1., którą Czytelnik musi w pełni zrozumieć, zanim metody jej oceny staną się jasne, jest poniższa:

> *Architektura decyduje o istnieniu lub braku niemal wszystkich atrybutów jakościowych danego systemu.*

Prowadzi to do sformułowania podstawowej zasady dotyczącej oceny architektur: skoro decyzje związane z architekturą determinują atrybuty jakościowe systemu, możliwa staje się ocena tych decyzji pod względem ich wpływu na owe atrybuty.

Co jest związane z architekturą?

Wcześniej czy później każdy zadaje pytanie: co należy do architektury? Niektórzy zadają sobie to pytanie z intelektualnej ciekawości, jednak w przypadku osób odpowiedzialnych za ocenianie architektur jest to raczej paląca potrzeba zrozumienia, jakie informacje stanowią dane wejściowe, a jakie wyjściowe w procesie, którym się zajmują. Pytanie to można przeformułować, na przykład do postaci przedstawionych poniżej:

- Jaka różnica istnieje między architekturą a projektem wysokiego poziomu?
- Czy szczegóły, takie jak priorytety procesów, także są związane z architekturą?
- Dlaczego kwestie implementacyjne, takie jak przepełnienia bufora, powinny być traktowane jako związane z architekturą?
- Czy interfejsy komponentów stanowią część architektury?
- Jeśli jest się w posiadaniu diagramów klas, czy jest potrzebne cokolwiek więcej?
- Czy architektura jest związana z działaniami fazy wykonania, czy też jest strukturą statyczną?
- Czy system operacyjny stanowi część architektury? A język programowania?
- Jeśli jest się zmuszonym do używania konkretnego produktu komercyjnego, to czy jest on częścią architektury? Jeśli ma się swobodę w zakresie wyboru spośród szerokiej gamy produktów komercyjnych, to czy jest to część architektury?

Istnieją dwa sposoby rozważania tego zagadnienia.

Po pierwsze, warto przyjrzeć się definicji architektury podanej w rozdziale 1. Parafrazując: architektura oprogramowania jest związana z ogólną organizacją systemu opisanego pod względem tworzących go komponentów, ich widocznych na zewnątrz właściwości oraz istniejących między nimi zależności. Oczywiście, definicja ta jest jak najbardziej prawidłowa, jednak nie obejmuje ona w sposób bezpośredni koncepcji kontekstu. Jeśli obszar bieżącego zainteresowania ogranicza się do podsystemu systemu, stanowiącego część systemu systemów, wówczas to, co postrzega się jako związane z architekturą będzie różne od analogicznego przedmiotu zainteresowania architekta systemu systemów. Stąd też kontekst ma ogromny wpływ na to, co jest związane z architekturą.

Po drugie, można zapytać, co *nie* jest związane z architekturą. Jak stwierdzono, algorytmy nie stanowią części architektury. Podobnie jest ze strukturami danych lub szczegółami dotyczącymi przepływu danych. Ponownie okazuje się jednak, że powyższe uwagi są tylko częściowo poprawne. Niektóre cechy algorytmów, takie jak ich złożoność, mogą mieć ogromny wpływ na wydajność. Niektóre z cech struktur danych, na przykład fakt, czy muszą umożliwiać dostęp współbieżny, w bezpośredni sposób wpływają na wydajność i niezawodność. Pewne szczegóły dotyczące przepływu

danych, takie jak zależności komponentów od komunikatów o określonym typie lub kwestia dostępu określonych komponentów do określonych typów danych, wpływają, odpowiednio, na modyfikowalność oraz poziom zabezpieczeń systemu.

Pojawia się zatem pytanie, czy istnieje ogólna reguła, którą można by wykorzystać w celu określenia elementów związanych z architekturą. Aby sformułować taką zasadę, można odwołać się do sposobów wykorzystania architektury. Przyjęte kryterium istnienia związku danego element z architekturą będzie następujące: musi być to albo komponent, albo relacja pomiędzy komponentami, albo właściwości (komponentów lub relacji), które muszą być *widoczne na zewnątrz*, aby umożliwić wartościowanie systemu pod względem spełniania stawianych wobec niego wymagań dotyczących jakości lub wspieranie dekompozycji systemu na elementy możliwe do niezależnego wdrażania. Poniżej wymieniono kilka wniosków, jakie można wysnuć na podstawie tej reguły:

- *Architektura opisuje zawartość systemu*. Po określeniu bieżącego kontekstu użytkownik określa jednocześnie granicę, która precyzuje elementy wchodzące oraz niewchodzące w skład danego systemu (który może stanowić jednocześnie podsystem innego użytkownika). Architektura opisuje te elementy, które wchodzą w skład systemu.
- *Architektura stanowi abstrakcyjny opis danego systemu*. Informacje, które niesie ze sobą architektura to najbardziej abstrakcyjny, a jednocześnie znaczący opis danej postaci systemu. Posiadanie specyfikacji danej architektury powinno eliminować konieczność sporządzania bardziej abstrakcyjnego opisu. Nie oznacza to, że wszystkie aspekty danej architektury są abstrakcyjne, ani że istnieje pewna wartość graniczna poziomu abstrakcji, którą należy przekroczyć w celu uznania danego elementu projektu za część architektury. Nie należy przejmować się faktem, że pewna architektura sięga do detalów, które inni mogą uważać za część bardziej szczegółowego projektu.
- *Elementy związane z architekturą oprogramowania powinny stanowić główne kryterium decydowania o najważniejszych wymaganiach*. Architektura stanowi pomost łączący zdefiniowane wymagania z całą resztą danego projektu. Jeśli projektant stwierdza, że pewne informacje mają podstawowe znaczenie pod względem definiowania stopnia spełniania przez system stawianych mu wymagań, wówczas stają się one częścią architektury. Architekt jest tu najlepszym sędzią. Z drugiej strony, jeśli okazuje się, że pewne szczegóły można wyeliminować, a i tak poprzez modelowanie, symulacje, przeglądy i inne podobne działania wciąż można w przekonujący sposób pokazać, że dana architektura spełnia kluczowe wymagania, wówczas szczegóły takie do architektury nie należą. Jednakże, jeśli architektura zawiera zbyt wiele szczegółów, to może okazać się, że nie spełnia kolejnej reguły.
- *Specyfikacja architektury powinna być zrozumiała*. Istota stosowania opisu systemu na poziomie globalnym polega na tym, że można go zrozumieć i wyciągać na tej podstawie pewne wnioski. Zbyt duża liczba szczegółów stanowi w tym względzie dużą przeszkodę.

- *Architektura wprowadza ograniczenia*. Nakłada ona wymagania na wszelkie niskopoziomowe specyfikacje projektowe. Warto wprowadzić w tym miejscu rozróżnienie na proces podejmowania decyzji oraz faktycznej ich realizacji. Przykładowo, w trakcie projektowania architektury można określić strategię ustalania priorytetów procesów, strategię redukcji nadmiarowości komponentów lub zbiór reguł hermetyzacji. Jednak faktycznie przez dość długi czas można by nie podejmować żadnych działań związanych z przydzieleniem priorytetów, nie zdefiniować żadnego algorytmu służącego do obliczeń określających nadmiarowość ani nie określić szczegółów interfejsu.

  Mówiąc w skrócie:

  To, co stanowi część architektury, jest najbardziej abstrakcyjnym odwzorowaniem systemu, które pozwala na wyciąganie wniosków związanych z najważniejszymi wymaganiami oraz determinuje wszelkie wprowadzane w przyszłości udoskonalenia.

Jeśli Czytelnik odnosi wrażenie, że odkrycie wszystkich cech danego systemu, które są związane z architekturą jest zadaniem niełatwym, to ma rację. Jest mało prawdopodobne, że od razu uda się odnaleźć wszystkie elementy związane z architekturą, a poza tym raczej nie warto się o to starać. Specyfikacja architektury ewoluuje z upływem czasu, co jest jedną z konsekwencji stosowania opisanych reguł w celu określenia elementów składowych architektury.

— MHK

2.1. Cele dokonywania oceny architektury

Im wcześniej udaje się znaleźć problem związany z projektem programistycznym, w tym lepszej jest się sytuacji. Koszt naprawienia błędu odkrytego w czasie definiowania wymagań lub wczesnych faz projektowania jest o rzędy wielkości niższy, niż ma to miejsce w przypadku tego samego błędu odnalezionego w trakcie testowania. Architektura oprogramowania stanowi efekt wczesnych faz projektowania i jej wpływ na system oraz projekt jest znaczny.

Nieodpowiednia architektura przyspiesza załamanie się projektu. W takiej sytuacji nie da się osiągnąć celów związanych z wydajnością. Także cele związane z zapewnieniem bezpieczeństwa nie zostaną osiągnięte. Klient zacznie się niecierpliwić, kiedy pewne schematy działania będą niedostępne, zaś dodanie ich do systemu okaże się zbyt trudne. Harmonogramy i budżety staną się nieaktualne, gdyż zespół konstrukcyjny będzie zmuszony do pokonywania licznych problemów. Po upływie miesięcy lub lat będzie trzeba zrezygnować ze zmian, które można było przewidzieć i zaplanować, ponieważ wówczas będą już wiązać się ze zbyt dużymi kosztami. Chyba trudno jest sobie wyobrazić gorszy scenariusz.

Architektura oprogramowania determinuje również strukturę projektu: biblioteki kontroli konfiguracji, harmonogramy i budżety, cele wydajnościowe, struktura zespołu, organizacja dokumentacji, a także działania związane z testowaniem oraz konserwacją — wszystkie one są zorganizowane wokół architektury. Jeśli główny schemat działania musi ulec zmianie wskutek pewnych późno odkrytych wad, to może się okazać, że w całym projekcie zapanuje chaos. Znacznie lepiej jest zmienić architekturę zanim zostanie już ona poddana procesowi realizacji.

Ocena architektury stanowi mało kosztowny sposób uniknięcia poważnych problemów. Metody opisane w niniejszej książce mają w zamierzeniu być stosowane w sytuacji, gdy architekturę stanowi tylko specyfikacja zapisana na papierze (oczywiście mogą jednak być stosowane także w późniejszym okresie) i polegają na wykonaniu serii prostych, ale przemyślanych eksperymentów. Wszystkie wymagają wspólnej pracy głównych zainteresowanych w formie przemyślanych sesji „burzy mózgów", prezentacji oraz analiz. Ogólnie rzecz biorąc dokonanie oceny architektury oznacza zazwyczaj dodanie do harmonogramu projektu nie więcej niż kilku dni.

Ujmując rzecz nieco inaczej — gdyby Czytelnik budował dom, z pewnością nie rozpoczął by odpowiednich działań konstrukcyjnych przed dokładnym zapoznaniem się z planami. Nikt nie zawaha się poświęcić pewnego dodatkowego czasu w tym celu, ponieważ jest oczywiste, że znacznie lepiej jest odkryć brak sypialni w momencie, gdy architektura opiera się wyłącznie na planach, nie zaś w dniu przeprowadzki.

2.2. Moment dokonywania oceny architektury

Klasyczny przypadek oceny architektury istnieje w sytuacji, gdy architektura została już określona, ale nie rozpoczęto jeszcze procesu jej wdrażania. Użytkownicy iteracyjnych lub przyrostowych modeli okresu użytkowania mogą oceniać decyzje dotyczące architektury podjęte w trakcie ostatniego cyklu. Jedną z istotnych cech metod oceny architektury jest jednak fakt, że można je stosować w dowolnym stadium istnienia danej architektury. Istnieją dwie użyteczne odmiany metody klasycznej: wczesna oraz późna.

Odmiana wczesna. Z oceną wcale nie trzeba czekać do momentu pełnego określenia danej architektury. Z metody tej można korzystać na dowolnym etapie procesu tworzenia architektury w celu zbadania już podjętych decyzji projektowych oraz dokonywania wyboru spośród dostępnych opcji architektonicznych, które wciąż nie zostały rozstrzygnięte. Znaczy to tyle, że w takim samym stopniu nadaje się ona do oceniania decyzji architektonicznych, które już podjęto, jak i tych, które są dopiero rozważane.

Oczywiście, zupełność oraz dokładność oceny są bezpośrednio zależne od zupełności i dokładności opisu architektury utworzonego przez architekta. W praktyce koszty oraz wysiłek logistyczny związane z przygotowaniem pełnej ewaluacji są rzadko podejmowane, jeśli stan przygotowania architektury oprogramowania nie uzasadnia takich działań. Nie jest po prostu opłacalne zebranie kilkunastu lub kilkudziesięciu osób bezpośrednio zainteresowanych projektem oraz analityków po to, aby dokonali oceny wstępnych notatek architekta zapisanych na skrawku papieru, nawet jeśli tego rodzaju notatki dają w rzeczywistości pogląd na wiele istotnych ścieżek projektowych — takich, które obrano oraz takich, które odrzucono.

Niektóre instytucje zalecają stosowanie tak zwanego przeglądu odkryć (ang. *discovery review*), który stanowi w istocie bardzo wczesny proces wstępnej oceny. Jego celem jest zarówno osiągnięcie porozumienia oraz ustalenie priorytetów związanych z kłopotliwymi wymaganiami, jak również dokonanie analizy utworzonego dotąd prototypu architektury. W przypadku przeglądu odkryć grupa uczestników jest mniej liczna, ale muszą do niej należeć osoby upoważnione do podejmowania decyzji związanych ze stawianymi wymaganiami. Celem takiego spotkania jest rozpatrzenie wszelkich wątpliwości, jakie może mieć architekt w związku z koniecznością zapewnienia spełniania przez daną architekturę połączonych wymagań. Wymagania te dotyczą kwestii jakościowych oraz związanych z zachowaniem systemu i są narzucane w momencie, kiedy wciąż jeszcze jest czas na to, aby złagodzić te najbardziej kłopotliwe lub najmniej istotne. Efektem przeglądu odkryć jest znacznie bardziej uściślony zbiór wymagań oraz opis wstępnych działań mających zapewnić ich spełnienie. Taki opis, po bliższym sprecyzowaniu, może stanowić w przyszłości przedmiot pełnej oceny.

Przeglądy odkryć nie zostaną w tym miejscu szczegółowo opisane, ponieważ stanowią one jedynie pewną odmianę metod oceny architektury. Jeśli działanie takie ma mieć miejsce, należy zapewnić, aby:

- odbyło się zanim stawiane wymagania zostaną definitywnie określone oraz kiedy architekt ma dobry pomysł dotyczący rozwiązania problemu;
- w grupie uczestników znajdowała się osoba upoważniona do podejmowania decyzji związanych ze stawianymi wymaganiami;
- w materiałach końcowych wyszczególniono zbiór wymagań z uwzględnieniem ich priorytetów, jeśli nie istnieje żaden oczywisty sposób zapewnienia spełnienia ich wszystkich.

W przypadku przeglądu odkryć warto także mieć w pamięci słowa słynnego projektanta samolotów, Willy'ego Messerschmitta, któremu nieobce były problemy związane ze stawianiem wymaganiami:

> *Można spełnić wszelkie życzenia Ministerstwa Lotnictwa dotyczące uwzględnienia dodatkowych możliwości, pod warunkiem jednak, że zrezygnuje się z wymagania, aby projektowany samolot latał.*

Odmiana późna. Druga odmiana znajduje zastosowanie w momencie, gdy nie tylko określono architekturę, ale zakończono także sam proces wdrażania. Przypadek taki ma miejsce w sytuacji, gdy instytucja dziedziczy pewien istniejący system. Mógł on zostać kupiony na rynku lub utworzony na podstawie własnych materiałów archiwalnych. Techniki oceniania architektury już istniejącej (ang. *legacy architecture*) nie różnią się od tych, które stosuje się w przypadku architektury nowo powstającej. Proces oceny jest pożyteczny ze względu na fakt, że pomaga nowym właścicielom systemu w jego zrozumieniu i pozwala im na sprawdzenie, czy można na nim polegać w zakresie spełnienia stawianych wymagań jakościowych i zachowaniowych.

Pojawia się w tym momencie pytanie natury ogólnej, dotyczące tego, kiedy można przeprowadzić proces oceny architektury. Najlepiej wówczas, gdy utworzona dotąd struktura architektury uzasadnia takie działanie. Różne instytucje mogą w różny sposób

definiować kryteria takiej zasadności, jednak przydatna praktyczna zasada brzmi: proces oceniania należy przeprowadzić wówczas, gdy zespół konstrukcyjny zaczyna podejmować decyzje, które uwarunkowane są architekturą, zaś koszt ewentualnego anulowania tych decyzji przewyższałby koszty związane z przeprowadzeniem oceny.

2.3. Zainteresowane strony

Można wyróżnić dwie grupy osób związanych z przeprowadzaniem oceny architektury.

(1) *Zespół oceniający*. Są to osoby odpowiedzialne za proces oceniania oraz przeprowadzenie analiz. Członkowie zespołu oraz szczegółowy opis ich zadań zostaną opisane w dalszej części niniejszego rozdziału. Na razie wystarczy powiedzieć, że reprezentują oni po prostu jedną z grup uczestników.

(2) *Główni zainteresowani*. Głównymi zainteresowanymi (ang. *stakeholders*) są osoby, które wniosły pewien kapitał w rozwój architektury oraz systemu budowanego na jej podstawie. We wszystkich trzech metodach oceniania, które opisano w niniejszej książce, osoby te mają za zadanie wyrażać określone wymagania stawiane wobec architektury. Nie dotyczy to wymagań, które definiują oczekiwany sposób działania systemu. Niektórzy zainteresowani są członkami zespołu konstrukcyjnego: koderzy, integratorzy, testerzy, konserwatorzy i tak dalej.

Z przeprowadzeniem oceny architektury w szczególny sposób są związane osoby odpowiedzialne za podejmowanie decyzji projektowych. Są to osoby zainteresowane wynikami procesu oceny. Posiadają one uprawnienia do podejmowania decyzji i mają wpływ na przyszły kształt projektu. Należą do nich architekci, projektanci komponentów oraz kierownictwo projektu. Kierownictwo jest odpowiedzialne za podejmowanie decyzji dotyczących reakcji na problemy odkryte w wyniku oceny. W przypadku pewnych ustaleń (szczególnie jeśli chodzi o projekty rządowe), osobą odpowiedzialną za podejmowanie decyzji projektowych może być również klient lub sponsor.

Zwykły zainteresowany wyraża swoje życzenia dotyczące powstającej architektury, z kolei osoba odpowiedzialna za podejmowanie decyzji ma możliwość takiego zadysponowania zasobami, aby można było je urzeczywistnić. Tak więc kierownik projektu może (jako zainteresowany) stwierdzić: „Chciałbym, aby architektura była możliwa do ponownego wykorzystania w innym podobnym projekcie, którym kieruję". Z kolei jako osoba odpowiedzialna za podejmowanie decyzji może stwierdzić: „Z moich obserwacji wynika, że zmiany, których wprowadzenie zidentyfikowano jako konieczne do zapewnienia możliwości ponownego wykorzystania architektury w moim drugim projekcie są zbyt kosztowne i dlatego nie mogę na nie pozwolić". Inna różnica polega na tym, że osoby odpowiedzialne za podejmowanie decyzji projektowych są upoważnione do autorytatywnego wypowiadania się w kwestii projektu i, przykładowo, niektóre etapy metody ATAM bezpośrednio tego od nich wymagają. Z drugiej strony, zwykli zainteresowani mogą jedynie mieć nadzieję, że będą mieć wpływ (choć na pewno nie będzie to równoznaczne z podjęciem decyzji) na projekt. Więcej informacji na ten temat zawiera ramka *Główni zainteresowani* na stronie 85. w rozdziale 3.

Beneficjentem procesu oceny architektury zwykle jest osoba odpowiedzialna za podejmowanie decyzji projektowych, bezpośrednio zainteresowana wynikami oceny oraz posiadająca pewną kontrolę nad projektem.

Czasem zespół oceniający składa się z pracowników związanych z projektem, w której to sytuacji są oni jednocześnie zainteresowanymi systemem. Nie jest to jednak zalecany schemat działania, ponieważ będzie im wówczas brak obiektywizmu w kwestii oceny architektury.

2.4. Rezultaty procesu oceny architektury

Mówiąc konkretnie, efektem procesu oceny architektury oprogramowania jest sporządzenie raportu, którego forma i zawartość zależą od wykorzystanej metody. Dodatkowym rezultatem oceny architektury jest także uzyskanie pewnych informacji. W szczególności chodzi tu o odpowiedzi na dwa rodzaje pytań.

(1) Czy dana architektura jest odpowiednia dla systemu, dla którego została zaprojektowana?
(2) Która z dwóch lub większej liczby konkurencyjnych architektur najlepiej odpowiada wymaganiom danego systemu?

Badaniu podlega przydatność architektury oprogramowania do wykonania danego zadania. Jest ona odpowiednia, jeśli spełnia dwa kryteria.

(1) System zbudowany na jej podstawie spełni stawiane mu cele jakościowe. Oznacza to, że system będzie pracował w sposób przewidywalny oraz wystarczająco szybko, aby mógł spełniać wymagania czasowe (ang. *timing*). Będzie mógł być modyfikowany w zaplanowany sposób. Będzie zgodny z nałożonymi ograniczeniami dotyczącymi zabezpieczeń. Będzie oferował wymagane funkcje definiujące sposób zachowania. Nie każda jakościowa cecha systemu stanowi bezpośredni rezultat jego architektury, ale w wiele przypadkach tak właśnie jest. Architektura jest odpowiednia wówczas, gdy zapewnia odpowiedni plan budowy systemu, który będzie się charakteryzował tymi cechami.
(2) System można zbudować z wykorzystaniem dostępnych zasobów: zespołu pracowników, budżetu, istniejącego oprogramowania (jeśli takie jest) oraz w wyznaczonym czasie. Oznacza to, że architektura umożliwia jego zbudowanie.

Tak zdefiniowane pojęcie odpowiedniości (ang. *suitability*) określać sferę zainteresowania dalej prezentowanych materiałów. Wynika z niego kilka istotnych implikacji. Po pierwsze, odpowiedniość ma zastosowanie tylko w kontekście określonych (i bezpośrednio wyrażonych) celów stawianych architekturze i tworzonemu na jej podstawie systemowi. Architektura, którą zaprojektowano mając na względzie głównie kwestie wydajności, może dać w efekcie system, który działa bardzo szybko, ale wymaga miesięcy pracy całych zespołów programistów w przypadku konieczności wprowadzenia do niego pewnych modyfikacji. Gdyby to właśnie modyfikowalność była dla takiego systemu ważniejsza

od wydajności, wówczas architektura ta byłaby nieodpowiednia (choć mogłaby stanowić podstawę dla jeszcze innej).

W książce *Alicja w Krainie Czarów* Alicja spotyka Kota z Cheshire i pyta go o drogę. Kot odpowiada, że zależy to od miejsca, do którego chce się udać. Alicja mówi, że nie wie, na co kot odpowiada, że w takim razie nie ma znaczenia, gdzie się skieruje. A zatem:

> *Jeśli osoba za to odpowiedzialna nie potrafi określić żadnych celów jakościowych, jakie należy postawić systemowi, wówczas zadanie to spełni dowolna architektura.*

Najważniejszą częścią procesu oceny architektury jest zidentyfikowanie i nadanie priorytetów określonym wymaganiom, które dana architektura musi spełniać. W sytuacji idealnej wszystkie wymagania zostałyby określone w odpowiednim dokumencie, jednak koncepcja taka nie sprawdza się z dwóch powodów. Po pierwsze, kompletne i aktualne dokumenty definiujące wymagania nie zawsze istnieją, a po drugie, dokumenty takie określają wymagania dotyczące systemu. W przypadku architektury oprogramowania oprócz uwzględnienia wymagań stawianych systemowi nakładane są także pewne dodatkowe wymagania (jako przykład można w tym miejscu podać możliwość zbudowania — ang. *buildability*).

Drugim wnioskiem wynikającym z oceny odpowiedniości architektury jest fakt, że odpowiedź wynikająca z przeprowadzonej ewaluacji nie jest pewnego rodzaju rezultatem skalarnym, który można by już znać na podstawie przeprowadzonych ocen innego rodzaju artefaktów programistycznych. W przeciwieństwie do, przykładowo, metryki kodu (ang. *code metrics*), w którym to przypadku poszukiwana odpowiedź mogłaby stwierdzać, że wartość 7,2 oraz każda inna większa od 6,5 jest nie do zaakceptowania, ewaluacja architektury daje w wyniku bardziej dogłębne rezultaty.

Nie istnieje potrzeba precyzyjnego charakteryzowania jakiegokolwiek atrybutu jakościowego (stosując miary, takie jak czas do awarii lub średni całościowy czas opóźnienia). Byłoby to działaniem pozbawionym sensu we wczesnej fazie projektowania, ponieważ faktyczne parametry, które determinują te wartości (na przykład faktyczny czas wykonania danego komponentu) są często zależne od implementacji. To, co należy zrobić — w atmosferze działań mających na celu uniknięcie ryzyka — to określenie punktu, w którym na badany atrybut oddziałują decyzje projektowe związane z architekturą. Ma to umożliwić dokładne przeanalizowanie tych decyzji, ich bardziej szczegółowe wymodelowanie w trakcie późniejszych analiz, a także spowodować poświęcenie większej ilości wysiłku związanego z projektowaniem, analizą oraz tworzeniem prototypów przy podejmowaniu takich decyzji.

Ocena architektury oprogramowania mówi o tym, że dana architektura okazała się być odpowiednią pod względem jednego zestawu zakładanych celów oraz problematyczna pod względem innego zestawu. Czasem cele takie są wobec siebie przeciwstawne lub przynajmniej pewne cele będą ważniejsze od innych. Kierownik projektu będzie zmuszony do podejmowania pewnych decyzji, jeśli okaże się, że ocena architektury jest pozytywna pod pewnymi względami, a pod innymi negatywna. Czy kierownik może pozwolić sobie na akceptowanie stanu charakteryzującego się określonymi słabościami? Czy można architekturę umocnić w tych obszarach? A może należy podjąć decyzję o rozpoczęciu prac nad projektem od początku? Ewaluacja pomaga w określaniu słabych punktów architektury, ale porównanie kosztów i korzyści związanych z projektem ulepszenia architektury stanowi całkowicie funkcję kontekstu danego projektu i znajduje się w rękach kierownictwa. Tak więc:

Dlaczego należy w to wierzyć?

Często osoba zajmująca się metodami oceniania jest traktowana jako outsider. Może zostać wynajęta przez lidera projektu, kierownika lub klienta w celu dokonania oceny projektu. Może to być postrzegane jako działania audytorskie lub po prostu element działań zmierzających do ulepszenia metod inżynierii programowania w danej jednostce organizacyjnej. Bez względu na przyczynę, o ile proces oceniania nie jest częścią długoterminowych założeń, specjalista nie zna zazwyczaj osobiście architekta albo głównych zainteresowanych.

Czasem taki wzajemny dystans nie stanowi problemu — uczestnicy są otwarci i podchodzą do problemu z entuzjazmem, są chętni do nauki oraz ulepszania swoich architektur. Jednak w pewnych sytuacjach można się spotkać ze swego rodzaju oporem, a nawet obawami. Główni zainteresowaniu siedzą wówczas z rękami skrzyżowanymi na piersiach, najwyraźniej rozdrażnieni, że odciąga się ich od zasadniczej pracy związanej z projektowaniem architektury i zmusza do zajmowania „tymi głupimi" ocenami nadzorowanymi przez kierownictwo. W jeszcze innych sytuacjach są nastawieni przychylnie, jednak wykazują duży sceptycyzm. Są oni wszakże ekspertami w swoich dziedzinach i nad danym zagadnieniem lub nawet konkretnym system pracują od lat.

W każdym bądź razie ich nastawienie, czy to przychylne, czy wrogie, wykazuje znaczną ilość sceptycyzmu w kwestii szans, że ewaluacja konkretnej architektury może faktycznie okazać się pomocną. W efekcie mówią oni: „Co, czego sami nie wiemy, może nam powiedzieć o naszym własnym systemie grupa specjalistów z zewnątrz?". Prawdopodobnie każdemu specjaliście ds. oceny architektur zdarzy się kiedyś zmierzyć z tego rodzaju opozycją lub oporem.

Należy pamiętać o dwóch sprawach w przypadku konieczności przeciwdziałania takiej opozycji. Po pierwsze, należy przezwyciężyć obawy. Dlatego też należy zachować spokój. Jeśli będzie się przyjaznym i pozwoli oponentom zrozumieć, że celem spotkania jest poznanie i ulepszenie architektury (nie zaś obwinianie kogokolwiek), wówczas okaże się, że ich opór szybko zacznie słabnąć. W rzeczywistości większość ludzi bawi proces ewaluacji i bardzo szybko dostrzegają oni wynikające z niego korzyści. Po drugie, należy przezwyciężyć sceptycyzm. Oczywiście członkowie zespołu są ekspertami w swojej dziedzinie. Wie o tym oceniający i wiedzą to oni sami, więc warto z góry to podkreślić. Jednak oceniający jest ekspertem w zakresie architektury oraz atrybutów jakościowych. Bez względu na dziedzinę zainteresowania, podejścia architektoniczne służące do usprawniania zarządzania oraz analizy atrybutów jakościowych nie podlegają zbytnim zmianom. Istnieje stosunkowo niewiele sposobów podejścia do zagadnień wydajności, dostępności lub bezpieczeństwa na poziomie architektonicznym. Doświadczony specjalista w zakresie ewaluacji (z pomocą społeczności specjalistów w zakresie atrybutów jakościowych) spotyka się z nimi niejednokrotnie i nie zmieniają się one zbytnio przy przechodzeniu z jednej dziedziny do drugiej.

Ponadto, jako człowiek z zewnątrz, oceniający przynosi ze sobą świeże spojrzenie i choćby tylko ten jeden fakt często może pozwolić na odkrycie nowej cechy projektu. Oceniający postępuje zgodnie z regułami procesu, który był udoskonalany w trakcie

przeprowadzania dziesiątków ewaluacji dotyczących dziesiątków różnych dziedzin wiedzy bądź techniki. Był on udoskonalany w celu umożliwienia wykorzystania doświadczeń wielu osób, w celu odkrycia, udokumentowania i skontrolowania wymagań dotyczących atrybutów jakościowych oraz informacji o architekturze. Już nawet tylko to może przynieść korzyści danemu projektowi — nie raz już się tak zdarzyło. Cały proces po prostu funkcjonuje poprawnie!

— R.K.

Ewaluacja architektury nie daje odpowiedzi typu „tak" lub „nie", „dobrze" lub „źle", czy też „6,75 z 10". Mówi raczej, gdzie czyhają pewne niebezpieczeństwa.

Ewaluacja architektury może być wykorzystana wobec pojedynczej architektury lub całej grupy współzawodniczących architektur. W tym drugim przypadku może pomóc w odkryciu silnych i słabych punktów każdej z nich. Oczywiście można założyć, że żadna architektura nie zdobędzie oceny lepszej od wszystkich innych pod każdym względem. Raczej każda z nich otrzyma lepsze oceny pod jednymi względami, zaś pod innymi gorsze w porównaniu z pozostałymi architekturami. Proces ewaluacji pozwoli na zidentyfikowanie w pierwszej kolejności interesujących obszarów, a następnie na pokazanie silnych i słabych punktów każdej z architektur w tych dziedzinach. Kierownictwo musi zdecydować która (jeśli w ogóle którakolwiek) z konkurencyjnych architektur powinna zostać wybrana lub poprawiona albo też uznać, że żadna z przedstawionych alternatyw nie jest akceptowalna i należy zaprojektować nową architekturę[*].

2.5. Właściwości, pod względem których architektura może podlegać ocenie

W tym podrozdziale w nieco dokładniejszy sposób zostanie opisane pojęcie odpowiedniości (ang. *suitability*). Nie jest prawdą, że po pobieżnym przeanalizowaniu danej architektury można określić, czy powstały na jej podstawie system będzie spełniał stawiane mu wymagania jakościowe. Z jednej strony, implementacja może na tyle odbiec od planów architektonicznych, że upadną plany jakościowe. Jednak z drugiej strony, architektura nie określa ściśle wszystkich cech systemu.

Dobrym przykładem może być w tym miejscu pojęcie użyteczności (ang. *usability*). Użyteczność stanowi miarę możliwości wykorzystania systemu przez użytkownika w sposób efektywny. Użyteczność jest istotnym celem jakościowym w przypadku wielu systemów, ale zwykle zależy ona głównie od interfejsu użytkownika. W przypadku projektów współczesnych systemów zwykle są podejmowane wysiłki mające na celu zawarcie poszczególnych aspektów interfejsu użytkownika w ramach niedużych struktur architektonicznych.

[*] W tym miejscu po raz ostatni jest poruszane zagadnienie ewaluacji więcej niż jednej architektury jednocześnie, gdyż opisywane metody mogą w obu przypadkach być stosowane w identyczny sposób.

Pobieranie i przekazywanie danych do i z interfejsu użytkownika oraz zapewnienie ich przepływu w ramach systemu, tak aby możliwe było wykonanie pracy potrzebnej do obsługi wymagań użytkownika, bez wątpienia stanowi kwestię związaną z architekturą, podobnie jak możliwość zmiany interfejsu użytkownika w razie takiej potrzeby. Jednakże wiele aspektów interfejsu użytkownika — takie kwestie, jak czerwone lub niebieskie tło widoczne dla użytkownika, przełączniki opcji lub okno dialogowe — nie jest związanych z architekturą, gdyż dotyczące ich decyzje ogólnie rzecz biorąc ograniczają się do bardzo zawężonego obszaru systemu.

Jednakże inne atrybuty jakościowe dotyczą bezpośrednio architektury. Na przykład metoda ATAM dotyczy przede wszystkim ewaluacji architektury pod względem odpowiedniości przy uwzględnieniu możliwości zapewnienia w systemie następujących atrybutów jakościowych (definicje oparte są na pozycji Bassa i in. [Bass 98]):

- *Wydajność*: wydajność (ang. *performance*) dotyczy czasu odpowiedzi systemu — czasu wymaganego do odpowiedzi na dany bodziec (zdarzenie) lub wiele zdarzeń przetwarzanych w pewnym przedziale czasu. Cechy wydajnościowe często wyraża się liczbą transakcji na jednostkę czasu lub ilością czasu potrzebnego do wykonania transakcji w systemie. Miary wydajności są często wyrażane w ramach testów wzorcowych (ang. *benchmarks*), które stanowią określone zbiory transakcji lub warunki obciążenia, przy których jest mierzona wydajność.
- *Niezawodność*: niezawodność (ang. *reliability*) jest zdolnością systemu do zachowania operatywności wraz z upływem czasu. Niezawodność jest zwykle wyrażana jako średni czas upływający między uruchomieniem systemu a zaistnieniem konieczności jego naprawy.
- *Dostępność*: dostępność (ang. *availability*) jest funkcją czasu, w którym system jest sprawny i działa. Wyraża się ją przez długość czasu pomiędzy awariami, a także przez szybkość, z jaką system jest w stanie wznowić działanie w przypadku awarii.
- *Bezpieczeństwo*: bezpieczeństwo (ang. *security*) jest miarą zdolności systemu do zapobiegania nieupoważnionym próbom jego wykorzystania oraz zapobiegania atakom zablokowania usług przy jednoczesnym oferowaniu dostępu do jego usług uprawnionym użytkownikom. Bezpieczeństwo podlega kategoryzacji pod względem typów zagrożeń, które mogą potencjalnie dotyczyć systemu.
- *Możliwość modyfikowania*: modyfikowalność (ang. *modifiability*) jest zdolnością do dokonywania w systemie zmian w sposób szybki i ekonomiczny. Jest ona mierzona przy wykorzystaniu specjalnych modyfikacji jako wskaźników oraz przez rejestrację kosztów związanych z wprowadzeniem tych zmian.
- *Przenośność*: przenośność (ang. *portability*) jest zdolnością systemu do działania w różnych środowiskach komputerowych. Środowisko takie może stanowić sprzęt, oprogramowanie lub ich kombinacja. System jest przenośny w takim stopniu, w jakim wszystkie przyjęte założenia dotyczące dowolnego środowiska komputerowego ograniczają się do jednego komponentu (lub w najgorszym przypadku — do niewielkiej liczby łatwo modyfikowalnych komponentów). Jeśli przeniesienie do nowego systemu wymaga wprowadzenia zmian, wówczas przenośność jest po prostu szczególnym przypadkiem modyfikowalności.
- *Funkcjonalność*: funkcjonalność (ang. *functionality*) jest zdolnością systemu do wykonania działań, dla których jest on przeznaczony. Wykonanie zadania wymaga, aby

wiele lub większość komponentów systemu pracowało w sposób skoordynowany w celu zakończenia pracy.

- *Zmienniczość*: zmienniczość (ang. *variability*) jest miarą tego, jak bardzo architektura może być rozszerzana lub modyfikowana w celu utworzenia nowych architektur, które różnią się w określony, z góry założony sposób. Rodzaje mechanizmów zmienniczości to mechanizmy czasu wykonania (na przykład negocjowanie protokołów na bieżąco), czasu kompilacji (na przykład ustawianie parametrów kompilacji w celu powiązania określonych zmiennych), czasu konsolidacji (na przykład zawieranie lub wykluczanie różnych komponentów lub wybieranie różnych wersji komponentu) albo mechanizmy czasu kodowania (na przykład kodowanie sterownika urządzenia dla nowego urządzenia). Zmienniczość jest istotną cechą, jeśli dana architektura ma służyć jako podstawa dla całej rodziny pokrewnych produktów, na przykład w przypadku linii produkcyjnej.
- *Podzbiorowość*: podzbiorowość (ang. *subsetability*) jest zdolnością do utworzenia podzbioru systemu. Choć może wydawać się, że jest to dość dziwaczna właściwość architektury, to w rzeczywistości jest ona jedną z najbardziej przydatnych i najczęściej niedocenianych. Podzbiorowość może wyznaczać różnice między sytuacją, gdy nie można dostarczyć niczego, gdyż nie było możliwe przestrzeganie ustalonego harmonogramu, a sytuacją, gdy w podobnych warunkach można dostarczyć niemal pełną wersję produktu. Podzbiorowość umożliwia również rozwój przyrostowy, co stanowi ważny paradygmat konstrukcyjny. Zgodnie z tym paradygmatem we wczesnej fazie buduje się minimalny, ale działający system, a następnie dodaje się do niego odpowiednie funkcje do czasu utworzenia ostatecznej wersji systemu. Podzbiorowość stanowi szczególny rodzaj zmienniczości, o której była mowa wyżej.
- *Spójność koncepcyjna*: Spójność koncepcyjna (ang. *conceptual integrity*) stanowi odpowiedni motyw lub wizję, która ujednolica projekt systemu na wszystkich jego poziomach. Architektura powinna zapewniać wykonywanie podobnych działań w podobny sposób. Spójność koncepcyjna występuje w architekturze, która wykazuje spójność, posiada niewielką liczbę danych oraz mechanizmów kontroli oraz wykorzystują niewielką liczbę wzorców w celu wykonania określonych zadań.

Z kolei metoda SAAM dotyczy szczególnie modyfikowalności w różnych jej postaciach (na przykład przenośności, podzbiorowości oraz zmienniczości) oraz funkcjonalności. Metoda ARID zapewnia wgląd w kwestie odpowiedniości danego wycinka architektury, który ma być używany przez programistów w celu wykonania powierzonych im zadań.

Jeśli istotne okażą się także inne właściwości, niż te wymienione powyżej, to opisywane metody wciąż spełniają swoje zadanie. Na przykład metoda ATAM jest budowana etapowo i niektóre etapy są uzależnione od badanej właściwości, zaś inne nie. Pierwsze etapy metody ATAM pozwalają na zdefiniowanie nowych atrybutów jakościowych przez bezpośrednie opisanie interesujących właściwości. Metoda ATAM może z łatwością zapewnić możliwość analizy nowych zależności jakościowych. Po przedstawieniu tej metody Czytelnik będzie wiedział, w którym momencie należy to zrobić. Na razie przyjmujemy jednak, że właściwości wymienione na przedstawionej powyżej liście tworzą podstawowy zestaw możliwości oferowanych przez metody, a także zawierają w sobie większość zagadnień, którymi zwykle zajmują się specjaliści dokonujący ewaluacji architektury.

2.6. Przyczyny dużej niejasności analiz atrybutów jakościowych

Atrybuty jakościowe tworzą podstawę ewaluacji architektur oprogramowania, ale samo nazwanie poszczególnych atrybutów nie stanowi wystarczającej podstawy, na której można by oprzeć oceny związane z odpowiedniością architektury. Często zapisuje się zdania określające wymagania, podobne do podanych poniżej:

- *system powinien być odporny*;
- *system powinien być łatwo modyfikowalny*;
- *system powinien być zabezpieczony przed niepowołanymi próbami uzyskania dostępu*;
- *system powinien wykazywać akceptowalną wydajność*.

Bez wdawania się w zbędne dyskusje, każde z takich zdań stanowi zwykle przedmiot różnych interpretacji i przyczynę nieporozumień. To, co projektant może uważać za przejaw odporności systemu, jego klient może postrzegać za mało potrzebną cechę — lub *vice versa*. Czasem system może z łatwością przyjmować nowe bazy danych, ale nie może adaptować się do nowych systemów operacyjnych. Czy system taki jest możliwy do konserwacji, czy też nie? Czasem system posługuje się mechanizmem zabezpieczeń opartym na wykorzystaniu haseł, co zapobiega dokonywanie włamań przez całą rzeszę nieupoważnionych użytkowników, ale jednocześnie nie posiada żadnej ochrony antywirusowej. Czy system taki jest zabezpieczony przed infiltracją, czy też nie?

Istota przedstawionego problemu polega na tym, że atrybuty jakościowe nie stanowią wielkości niezmiennych — istnieją one zawsze w kontekście określonych celów. W szczególności:

- system jest modyfikowalny (lub nie) pod względem określonego rodzaju zmian;
- system jest bezpieczny (lub nie) pod względem określonego rodzaju zagrożeń;
- system jest niezawodny (lub nie) pod względem wystąpień określonego rodzaju błędów;
- system działa wydajnie (lub nie) pod względem określonych kryteriów wydajności;
- system jest odpowiedni (lub nie) dla danej linii produkcyjnej pod względem określonego zbioru lub zakresu przewidywanych produktów danej linii produkcyjnej (to znaczy, pod względem zasięgu określonej linii produkcyjnej);
- architektura jest możliwa do zbudowania (lub nie) pod względem określonych ograniczeń czasowych i budżetowych.

Jeśli rozumowanie takie nie wydaje się to dostatecznie poprawne, wystarczy zauważyć, że żaden system nie może zawsze być, na przykład, całkowicie niezawodny w dowolnych warunkach (wystarczy wyobrazić sobie awarię zasilania, tornado lub niezadowolonego operatora systemu wyposażonego w kilof). W takim razie jest zatem obowiązkiem architekta dokładne poznanie warunków, w jakich system powinien być niezawodny w celu uznania go za akceptowalny.

W sytuacji idealnej wymagania jakościowe wobec systemu byłyby całkowicie i jednoznacznie określone w dokumentacji dotyczącej takich wymagań. Sytuacje takie zdarzają się jednak bardzo rzadko. Dokumenty określające wymagania nie są pisane w ogóle, są

pisane bardzo ogólnie lub ich tworzenia nie udaje się zakończyć do momentu, w którym nadchodzi czas na rozpoczęcie tworzenia architektury. Ponadto architektury oprogramowania charakteryzują specyficzne dla nich cele, które nie są wymieniane w dokumentacji wymagań stawianych wobec projektowanego systemu: muszą one być zbudowane przy użyciu dostępnych zasobów, powinny wykazywać spójność koncepcyjną i tak dalej. Tak więc pierwszym zadaniem stawianym wobec procesu ewaluacji architektury jest wykrycie określonych celów jakościowych, pod względem których będzie dokonywana ocena architektury.

Jeśli wszystkie z tych celów zostaną bezpośrednio, jednoznacznie wyartykułowane, to bardzo dobrze. W przeciwnym wypadku trzeba poprosić głównych zainteresowanych architekturą o pomoc w ich zapisaniu podczas trwania procesu ewaluacji. Wykorzystywanym tu mechanizmem jest *scenariusz*. Scenariusz stanowi krótkie zdanie opisujące interakcję jednego z głównych zainteresowanych z systemem. Użytkownik opisałby wykorzystanie systemu w celu wykonania pewnego zadania; takie scenariusze bardzo przypominałyby przypadki użycia (ang. *use cases*) — mówiąc językiem specjalistów od systemów zorientowanych obiektowo. Osoba odpowiedzialna za konserwację opisałaby proces dokonywania zmian w systemie, takich jak aktualizowanie systemu operacyjnego w określony sposób lub dodawanie określonej nowej funkcji. Scenariusz programisty mógłby zawierać opis użycia architektury w celu zbudowania systemu lub szacowania jego przewidywanej wydajności. Scenariusz klienta mógłby opisywać architekturę w kontekście jej ponownego użycia w przypadku kolejnego produktu lub w przypadku linii produkcyjnej, albo też mógłby wymagać, aby system był możliwy do zbudowania przy zaangażowaniu określonych zasobów.

Każdy scenariusz zostaje następnie skojarzony z określoną osobą (chociaż różne osoby mogą być tak samo zainteresowani jednym scenariuszem). Każdy scenariusz jest związany ponadto z określoną właściwością systemu, ale przy zachowaniu określonych warunków. Zagadnienia dotyczące scenariuszy opisano bliżej w rozdziale 3.

2.7. Wyniki ewaluacji architektury oprogramowania

2.7.1. Wyniki metod ATAM, SAAM oraz ARID

Ewaluacja architektury daje w wyniku pewne informacje oraz wgląd w samą architekturę. Każda z metod — ATAM, SAAM oraz ARID — daje wyniki opisane poniżej.

Opatrzone priorytetami zestawienie wymagań atrybutów jakościowych

Ewaluacja architektury może się odbyć tylko wtedy, gdy są znane kryteria odpowiedniości. Stąd jednym z głównych jej etapów jest rozpoznanie atrybutów jakościowych, wobec których architektura podlega ewaluacji. Jednak żadna architektura nie może

spełniać nieograniczonej listy atrybutów jakościowych, a przez to opisywane metody wykorzystują schemat nadawania priorytetów oparty na konsensusie. Utworzenie opatrzonego priorytetami zestawienia atrybutów jakościowych służy jako doskonały fragment dokumentacji, która towarzyszy każdej architekturze i kieruje jej rozwojem. Wszystkie trzy metody tworzą takie zestawienie w formie zbioru scenariuszy dla atrybutów jakościowych.

Odwzorowanie metod postępowania na atrybuty jakościowe

Odpowiedzi na pytania związane z przeprowadzaną analizą dają pewne odwzorowanie. Daje ono informację, w jaki sposób zależne od architektury metody postępowania umożliwiają osiąganie (lub w jaki sposób nie udaje się osiągnąć) pożądanych atrybutów jakościowych. Odwzorowanie to pozwala na określenie racjonalnego uzasadnienia dla założeń architektonicznych. Uzasadnienie takie powinno stanowić część pracy każdego architekta, ale wielu nie starcza na to czasu. Odwzorowanie metod postępowania na atrybuty może dać spory fragment takiego opisu.

Decyzje ryzykowne i nieryzykowne

Decyzje ryzykowne są decyzjami architektonicznymi potencjalnie problematycznymi. Decyzje nieryzykowne są z kolei decyzjami poprawnymi, których podjęcie osadza się na założeniu, że często stanowią nieodłączony element architektury. Obydwa rodzaje decyzji powinny być dobrze zrozumiane i jawnie zanotowane[*].

Dokumentowanie decyzji ryzykownych oraz nieryzykownych składa się z:

- wyartykułowania podjętej decyzji architektonicznej (lub decyzji, która nie została podjęta);
- określonej reakcji atrybutu jakościowego, na który ma wpływ dana decyzja wraz z konsekwencjami związanymi z przewidywanym poziomem odpowiedzi;
- racjonalnego uzasadnienia pozytywnego lub negatywnego wpływu, jaki dana decyzja ma na kwestię spełnienia wymagań atrybutów jakościowych.

Przykład decyzji ryzykownej

Reguły określające tworzenie modułów zasad biznesowych w ramach drugiej warstwy trójwarstwowej struktury klient-serwer nie zostały w jasny sposób określone (*przykład decyzji, której nie podjęto*). Może to spowodować powtarzanie elementów funkcjonalnych, a przez to znacznie ograniczyć modyfikowalność warstwy trzeciej (*odpowiedź atrybutu jakościowego oraz jej konsekwencje*). Nieokreślone wyraźnie reguły tworzenia zasad biznesowych mogą powodować niecelowe i niepożądane sprzężenia komponentów (*racjonalne uzasadnienie efektu negatywnego*).

Przykład decyzji nieryzykownej

Zakładając częstotliwość odbierania komunikatów jako jeden na sekundę, czas przetwarzania na mniej niż 30 milisekund oraz istnienie jednego procesu o wyższym

[*] Decyzje ryzykowne mogą pochodzić również z innych, niezwiązanych z architekturą źródeł. Na przykład posiadanie struktury zarządzania niedopasowanej do struktury architektonicznej może stanowić ryzyko organizacyjne. Niedostateczna komunikacja między grupami zainteresowanych a architektem często stanowi rodzaj ryzyka zarządczego.

priorytecie (*decyzje architektoniczne*), rozsądną decyzją wydaje się być przyjęcie jednosekundowego czasu opóźnienia (*odpowiedź atrybutu jakościowego i jej konsekwencje*), gdyż częstotliwość docierania komunikatów jest ograniczona, zaś wywłaszczający wpływ procesów o wyższym priorytecie jest znany i może być dostosowywany (*racjonalne uzasadnienie*).

Aby decyzja nieryzykowna taką pozostała, przyjmowane założenia nie mogą ulec zmianie (lub przynajmniej, jeśli ulegną one zmianie, określenie braku ryzyka będzie wymagało ponownej oceny). Na przykład, jeśli częstotliwość docierania komunikatów, czas przetwarzania lub liczba procesów o wyższym priorytecie zmieni się w przykładzie przedstawionym powyżej, to określenie braku ryzyka również może się zmienić.

2.7.2. Wyniki związane tylko z metodą ATAM

Oprócz informacji już opisanych, metoda ATAM daje również dodatkowy zbiór rezultatów opisanych poniżej.

Katalog wykorzystywanych architektonicznych metod postępowania

Każdy architekt stosuje pewne określone strategie oraz metody postępowania w celu rozwiązania napotykanych problemów. Czasem metody takie są dobrze znane i stanowią element ustalonego poziomu wiedzy w zakresie danej dziedziny, czasem jednak są one unikalne i innowacyjne w przypadku budowanego systemu. W obu przypadkach stanowią one klucz do zrozumienia, czy dana architektura spełni swoje cele i wymagania. Metoda ATAM zawiera etap, w którym wykorzystywane metody podlegają skatalogowaniu, a katalog taki może później służyć jako wstępne źródło informacji o architekturze dla osób, które muszą się z nią zapoznać, na przykład przyszłych architektów lub zarządców systemu.

Pytania analityczne charakterystyczne dla metody postępowania oraz atrybutów jakościowych

Metodę ATAM charakteryzują pewne pytania analityczne oparte na szukanych atrybutach oraz metodach postępowania obranych przez architekta. Wraz z rozwojem architektury pytania te mogą być wykorzystywane w przyszłych miniewaluacjach w celu upewnienia się, że obrana droga nie prowadzi architektury w złym kierunku.

Punkty wrażliwości oraz punkty kompromisowe

Kluczowe decyzje architektoniczne będziemy określali mianem punktów wrażliwości (ang. *sensitive points*) oraz punktów kompromisowych (ang. *tradeoff points*). Punkt wrażliwości można zdefiniować jako właściwość jednego lub większej liczby komponentów

(i/lub relacji zachodzących pomiędzy komponentami), która ma podstawowe znaczenie dla otrzymania odpowiedzi określonego atrybutu jakościowego. Na przykład:

- poziom zaufania wobec wirtualnej sieci prywatnej może ściśle zależeć od używanej liczby bitów klucza szyfrowania;
- opóźnienie związane z przetworzeniem ważnego komunikatu może być wrażliwe z powodu zbyt małej wartości priorytetu najniższego pod tym względem procesu, który jest związany z obsługą tego komunikatu;
- średnia liczba osobodni wymaganych do konserwowania systemu może być ściśle zależna od stopnia hermetyzacji jego protokołów komunikacyjnych oraz formatów plików.

Punkty *wrażliwości* wskazują projektantowi lub analitykowi miejsca, na które powinien zwrócić uwagę próbując zrozumieć osiągnięcia zapewniane przez dany cel jakościowy. Służą one jako swego rodzaju żółte flagi: *trzeba zachować ostrożność zmieniając daną właściwość architektury*. Pewne wartości punktów wrażliwości mogą stać się źródłami ryzyka w momencie realizacji danej architektury. Warto rozpatrzyć podane powyżej przykłady. Konkretna wartość określająca siłę szyfrowania — na przykład szyfrowanie 32-bitowe — może stanowić dla architektury ryzyko. Podobnie posiadanie procesu o bardzo niskim priorytecie w potoku odpowiedzialnym za przetwarzanie ważnego komunikatu może stać się ryzykowne dla architektury.

Punkt *kompromisowy* z kolei można określić jako właściwość wpływającą na więcej niż jeden atrybut, a jednocześnie punkt wrażliwości dla więcej niż jednego atrybutu. Przykładowo, zmiana poziomu szyfrowania może mieć ogromny wpływ zarówno na bezpieczeństwo, jak i wydajność systemu. Jego zwiększanie usprawnia przewidywany poziom bezpieczeństwa, ale wymaga dłuższego czasu przetwarzania. Jeśli przetwarzanie tajnej wiadomości jest związane z surowymi wymaganiami opóźnienia, umożliwiającego pracę w czasie rzeczywistym, wówczas poziom szyfrowania może stanowić punkt kompromisu. Punkty kompromisowe stanowią najważniejsze decyzje podejmowane w związku z daną architekturą, co wyjaśnia dlaczego poświęcamy im tyle uwagi.

Na zakończenie warto podkreślić, że nie jest rzadkością sytuacja, w której architekt odpowiada na pytania mające na celu odkrycie istotnych spraw mówiąc: „Nie podjęliśmy jeszcze decyzji". W takim przypadku nie da się wskazać na dany komponent lub właściwość architektury i nazwać je punktem wrażliwości, ponieważ komponent ten lub właściwość mogą w ogóle jeszcze nie istnieć. Jednak ważną sprawą jest oznaczenie w pewien sposób kluczowych decyzji, które podjęto, ale również tych, których wciąż jeszcze nie podjęto.

2.8. Korzyści oraz koszta związane z przeprowadzaniem ewaluacji architektury

Główną, i oczywistą korzyścią wynikającą z ewaluacji architektury jest niewątpliwie fakt, że umożliwia ona wykrycie problemów, których naprawa w razie ich pozostawienia byłaby w późniejszym okresie o wiele rzędów wielkości kosztowniejsza. W skrócie,

ewaluacja architektur umożliwia w wyniku otrzymanie lepszych architektur. Nawet jeśli ewaluacja nie odkryje żadnych godnych uwagi problemów, to w każdym razie zwiększy poziom pokładanego w niej zaufania.

Osiąga się jednak także inne korzyści. Niektóre z nich dość trudno mierzyć, ale wszystkie one mają swój udział w sukcesie projektu oraz dają w wyniku bardziej przemyślaną organizację. Nie wszystkie z nich muszą wystąpić w przypadku każdej ewaluacji. Poniżej zamieszczono listę korzyści, które można zaobserwować najczęściej.

Umieszczenie głównych zainteresowanych w jednym pokoju

Ewaluacja architektury jest często pierwszą okazją, przy której może się spotkać wielu głównych zainteresowanych nią osób; czasem jest to pierwsze spotkanie z nimi architekta. Pojawia się wówczas efekt dynamizmu grupowego, a w takich warunkach wszyscy zainteresowani postrzegają się nawzajem jako osoby dążące do tego samego: zbudowania udanego systemu. Wcześniej ich cele mogły być względem siebie konkurencyjne (i w rzeczywistości może tak pozostać), zaś teraz są oni w stanie wyjaśnić swoje cele oraz motywacje, tak aby możliwe się stało wzajemne zrozumienie. W atmosferze takiej można dochodzić do kompromisów lub opracowywać nowatorskie rozwiązania w świetle lepszego zrozumienia. Niemal zawsze okazuje się, że potem różne osoby wymieniają się numerami telefonów i adresami poczty elektronicznej oraz otwierają kanały komunikacji, która wykracza często poza okres samej ewaluacji.

Wymuszenie określenia szczególnych celów jakościowych

Zadaniem głównych zainteresowanych jest określenie celów jakościowych, które powinna spełniać architektura, aby mogła okazać się udaną. Cele te często nie zostają uwzględnione w dokumentacji dotyczącej wymagań, a przynajmniej nie są definiowane w sposób jednoznaczny i nie wykraczają poza bezproduktywne truizmy dotyczące niezawodności i modyfikowalności. Scenariusze zapewniają jawne testowanie cech jakościowych.

Ustalenie priorytetów konfliktowych celów

Konflikty, jakie mogą powstawać pomiędzy celami określanymi przez różnych zainteresowanych zostają zdefiniowane. Każda metoda uwzględnia etap, w którym celom tym są nadawane przez całą grupę pewne priorytety. Jeśli architekt nie może usatysfakcjonować wszystkich współzawodniczących stron, otrzyma przynajmniej wyraźne wskazówki co do tego, które cele są najważniejsze (oczywiście, zawsze istnieje możliwość znaczącej ingerencji kierownika projektu, kiedy to odrzuca on lub dostosowuje określone gremialnie priorytety — zwykle będzie on postrzegał pewne osoby oraz ich cele za „równiejsze" niż inne — ale tylko wówczas, gdy kwestia ta zostanie w ogóle poruszona).

Wymuszenie zrozumiałego objaśnienia architektury

Architekt jest zmuszony do takiego przedstawienia grupie osób szczegółów dotyczących procesu tworzenia architektury, aby był on dla nich zrozumiały. Oprócz innych celów ma to również służyć jako forma próby generalnej przed wyjaśnieniem tych szczegółów innym

projektantom, programistom i testerom. Im wcześniej takie wyjaśnienia mają miejsce, tym większe korzyści czerpie z tego projekt.

Zwiększenie jakości dokumentacji architektonicznej

Ewaluacja często wymaga utworzenia dotąd nieprzygotowywanej dokumentacji. Na przykład zbadanie kwestii związanych z wydajnością wygeneruje potrzebę sporządzenia dokumentacji opisującej sposoby, w jakie architektura obsługuje mechanizm współdziałania zadań lub procesów w czasie wykonania. Jeśli ewaluacja tego wymaga, wówczas jest niemal pewne, że któryś z członków zespołu projektowego (w tym przypadku inżynier ds. wydajności) także będzie tego potrzebował. Projekt odnosi korzyści, ponieważ etap konstruowania zostaje lepiej przygotowany.

Odkrycie możliwości między projektowego ponownego wykorzystania

Główni zainteresowani oraz zespół ewaluacyjny pochodzą spoza zespołu konstrukcyjnego, ale często pracują nad podobnymi projektami lub znają inne projekty tworzone w ramach tej samej, macierzystej jednostki organizacyjnej. Jako takie, obie grupy znajdują się na dobrej pozycji albo dla zidentyfikowania komponentów, które można użyć ponownie w przypadku innych projektów, albo dla zidentyfikowania komponentów (lub innych zasobów), które już istnieją i prawdopodobnie mogą zostać wykorzystane w ramach bieżącego projektu.

Ulepszenie praktycznych technik tworzenia architektury

Jednostki organizacyjne, w których ewaluacja stanowi standardowy etap procesu konstrukcyjnego, donoszą o poprawie jakości architektur, które podlegają takim ocenom. Kiedy jednostki organizacyjne uczą się rozpoznawania zadawanych pytań, poruszanych kwestii oraz rodzajów dokumentów, które będą potrzebne dla celów ewaluacji, wówczas w naturalny sposób maksymalizują swoją wydajność w trakcie procesu przeprowadzania ewaluacji. Ewaluacje umożliwiają podniesienie jakości architektury, a proces ten będzie możliwy do zaobserwowania nie tylko po fakcie, ale także w trakcie dokonywania samej oceny. Z czasem w ramach danej jednostki rozwija się pewna kultura promująca dobre projekty architektoniczne.

Nie wszystkie korzyści muszą być zawsze osiągane. Jeśli dana jednostka organizacyjna jest mała, być może wszyscy główni zainteresowani znają się nawzajem i często ze sobą rozmawiają. Niektóre jednostki mogą mieć duże doświadczenie w kwestiach opracowywania wymagań stawianych systemowi i do czasu, gdy prace nad architekturą dobiegną końca, wymagania te nie będą już stanowiły żadnego problemu, ponieważ każdy będzie dokładnie wiedział, jaką mają postać. Jest to bardzo korzystna sytuacja. Jednak w przypadku wielu jednostek organizacyjnych, w których Autorzy przeprowadzali ewaluacje architektur, nie było tak dobrze i zawsze pojawiały się pewne problemy dotyczące określania wymagań, które rozważano (i rozwiązywano) w momencie bliższego badania architektury.

Można również wyróżnić korzyści związane z przyszłymi projektami w ramach tej samej jednostki. Najważniejszy etap metody ATAM stanowi proces badania architektury z wykorzystaniem zestawu pytań analitycznych związanych z jakością. Ani sama metoda, ani lista tych pytań nie stanowią tajemnicy. Architekt ma całkowitą swobodę w przygotowaniu się do procesu ewaluacji poprzez upewnienie się, że architektura spełnia wymagania określone w odpowiednich pytaniach. Przypomina to sytuację osiągania dobrego wyniku testu, z którego pytaniami można się było wcześniej zapoznać. W tym konkretnym przypadku nie jest to oszukiwanie — to profesjonalizm.

Na koszta związane z ewaluacją architektury oprogramowania składają się wszelkie koszta personalne oraz koszta dodatkowe, związane z uczestniczeniem danego personelu w ewaluacji zamiast wykonywania innych obowiązków. Z łatwością można je oszacować. W tabeli 2.1 zaprezentowano przykładową ocenę kosztów ewaluacji opartej na zastosowaniu metody ATAM. W pierwszej kolumnie licząc od lewej strony określono fazy metody ATAM (zostanie ona opisana w kolejnych rozdziałach). W pozostałych kolumnach zaprezentowano rozkład kosztów między uczestniczącymi grupami. Z łatwością można utworzyć podobne tabele dla innych metod.

W tabeli 2.1 przedstawiono wartości charakterystyczne dla przeciętnego wysiłku ewaluacyjnego. Choć wartość 70 osobodni wydaje się być wystarczającą, jednak w rzeczywistości nie zawsze tak jest. Po pierwsze, do harmonogramu projektu należy na pewno dodać czas wolnych dni kalendarzowych. Na harmonogram nie powinien mieć żadnego wpływu okres przygotowawczy ani działania uzupełniające. Czynności tych nie trzeba wykonywać na forum publicznym. Fazy środkowe tworzą faktyczny czas projektowy, zwykle około trzech dni. Po drugie, prace nad projektem zazwyczaj nie oznaczają konieczności płacenia za wszystkie 70 osobodni. Wielu głównych zainteresowanych pracuje w innych wydziałach lub w ogóle w innych jednostkach organizacyjnych niż grupa konstrukcyjna. Główni zainteresowani z definicji okazują tworzonemu systemowi wiele uwagi i często są bardziej niż chętni do poświęcenia swojego czasu dla dobra produktu, który ma charakteryzować się wysoką jakością.

Oczywiście, z łatwością można wyobrazić sobie zarówno większe, jak i mniejsze wymagania, niż przedstawione w tabeli 2.1. Czytelnik przekona się podczas lektury niniejszej książki, że wszystkie opisywane metody są elastyczne, zbudowane w sposób umożliwiający iteracyjne przechodzenie do takiego poziomu szczegółowości, jakiego życzą sobie oceniający oraz klient przeprowadzanego procesu ewaluacji. Pobieżne ewaluacje można wykonać w ciągu jednego dnia, bardzo szczegółowe ewaluacje mogą trwać tygodniami. Jednakże wartości podane w tabeli 2.1 reprezentują teoretyczne przypadki wykorzystywania metody ATAM. W przypadku niedużych projektów wartości te można zmniejszyć. Sposób zmniejszenia tych wartości zaprezentowano w tabeli 2.2.

Jeśli dana grupa dokonuje ewaluacji wielu systemów w tym samym środowisku pracy lub przy tych samych celach stawianych architekturze, wówczas istnieje dodatkowy sposób na zredukowanie kosztów ewaluacji. Należy zbierać i zapisywać scenariusze używane w przypadku obu ewaluacji. Po pewnym czasie okaże się, że zestawy scenariuszy zaczną być do siebie podobne. Po przeprowadzeniu kilku takich bardzo podobnych ewaluacji można będzie utworzyć „wzorcowy" zestaw scenariuszy opierający się na zdobytych doświadczeniach. W tym momencie scenariusze te w gruncie rzeczy rozwiną się na tyle, by stanowić listę kontrolną i można będzie pomijać dużą część pracy związanej z tworzeniem scenariuszy. Pozwala to na zaoszczędzenie około

TABELA 2.1.
Przybliżony koszt ewaluacji architektury średniej wielkości za pomocą metody ATAM

Fazy metody ATAM	Zespół ewaluacyjny (5 członków)	Główni zainteresowani	
		Osoby podejmujące decyzje projektowe (architekt, kierownik projektu, klient)	Inni zainteresowani (8 członków)
Faza 0.: Przygotowanie	1 osobodzień dla lidera zespołu	1 osobodzień	0
Faza 1.: Ewaluacja wstępna (1 dzień)	5 osobodni	3 osobodni	0
Faza 2.: Ewaluacja całkowita (3 dni)	15 osobodni	9 osobodni + 2 osobodni na przygotowanie	16 osobodni (większość osób jest obecna tylko przez 2 dni)
Faza 3.: Działania uzupełniające	15 osobodni	3 osobodni na przeczytanie i analizę raportu	0
SUMA	36 osobodni	18 osobodni	16 osobodni

TABELA 2.2.
Przybliżony koszt ewaluacji niedużej architektury za pomocą metody ATAM

Faza metody ATAM	Zespół ewaluacyjny (2 członków)	Główni zainteresowani	
		Osoby podejmujące decyzje projektowe (architekt, kierownik projektu)	Inni zainteresowani (3 członków)
Faza 0.: Przygotowanie	1 osobodzień dla lidera zespołu	1 osobodzień	0
Faza 1.: Ewaluacja wstępna (1 dzień)	2 osobodni	2 osobodni	0
Faza 2.: Ewaluacja całkowita (2 dni)	4 osobodni	4 osobodni + 2 osobodni na przygotowanie	6 osobodni
Faza 3.: Działania uzupełniające	8 osobodni	2 osobodni na przeczytanie i analizę raportu	0
SUMA	15 osobodni	11 osobodni	6 osobodni

jednego dnia. Z uwagi na fakt, że generowanie scenariuszy stanowi przede wszystkim zadanie głównych zainteresowanych, możliwe okaże się zaoszczędzenie ich czasu, co obniży koszta jeszcze bardziej.

Wciąż może jednak okazać się wskazana obecność kilku kluczowych osób, w tym klienta, którego zadaniem będzie potwierdzenie adekwatności listy kontrolnej dla ocenianego systemu. Można zredukować liczebność zespołu, gdyż nie jest konieczne zapisywanie scenariuszy. Czas przygotowania architekta powinien być minimalny, gdyż lista kontrolna będzie dostępna publicznie nawet kiedy rozpocznie on już swoje prace nad architekturą.

W tabeli 2.3 przedstawiono koszt ewaluacji architektury średniego rozmiaru, bazującej na liście kontrolnej oraz z wykorzystaniem metody ATAM. Okazuje się, że koszt bieżący stanowi około 4/7 wartości kosztu ewaluacji opartej na scenariuszach z tabeli 2.1.

TABELA 2.3.
Przybliżony koszt ewaluacji architektury średniej wielkości za pomocą metody ATAM opartej na liście kontrolnej

Faza metody ATAM	Zespół ewaluacyjny (5 członków)	Główni zainteresowani	
		Osoby podejmujące decyzje projektowe (architekt, kierownik projektu, klient)	Inni zainteresowani (8 członków)
Faza 0.: Przygotowanie	1 osobodzień dla lidera zespołu	1 osobodzień	0
Faza 1.: Ewaluacja wstępna (1 dzień)	4 osobodni	3 osobodni	0
Faza 2.: Ewaluacja całkowita (2 dni)	8 osobodni	6 osobodni	2 osobodni
Faza 3.: Działania uzupełniające	12 osobodni	3 osobodni na przeczytanie i analizę raportu	0
SUMA	25 osobodni	13 osobodni	2 osobodni

W kolejnym rozdziale przedstawiono wprowadzenie do pierwszej z trzech opisywanych w niniejszej książce metod ewaluacji architektur: metodę analizy kompromisów architektonicznych.

2.9. Dalsza lektura

Podrozdział *Dalsza lektura* w rozdziale 9. *Porównanie metod ewaluacji architektur oprogramowania* zawiera listę godnych polecenia pozycji związanych z różnymi metodami ewaluacji architektur.

Zhao skompilował przydatny zbiór informacji o zasobach piśmienniczych dotyczących analiz architektur oprogramowania [Zhao 99].

Warto w tym miejscu zastanowić się, czy jeśli w wyniku ewaluacji architektury udało się zidentyfikować konieczne do wprowadzenia zmiany, to w jaki sposób należy ustalać ich priorytety? Opracowywane są pewne metody, których zastosowanie może być pomocne architektowi lub kierownikowi projektu w kwestii określenia kosztów oraz korzyści związanych z podejmowanymi decyzjami architektonicznymi [Kazman 01].

2.10. Pytania dyskusyjne

(1) W jaki sposób jednostka organizacyjna, z którą Czytelnik jest związany, decyduje obecnie o tym, czy proponowana architektura oprogramowania powinna zostać wykorzystana lub nie? Jak podejmuje się decyzje o tym, kiedy architektura oprogramowania przestaje być użyteczna i powinna zostać zastąpiona inną?

(2) Należy opracować plan biznesowy, określony dla danej jednostki organizacyjnej, który odpowie na pytanie, czy przeprowadzenie ewaluacji architektury oprogramowania byłoby opłacalne. Należy przyjąć oszacowania kosztów podane w niniejszym rozdziale lub wykorzystać własne.

(3) Czy Czytelnik zna przypadek, kiedy niepoprawna architektura oprogramowania doprowadziła do fiaska lub opóźnienia systemu lub projektu programistycznego? Należy przeanalizować przyczyny powstałych problemów oraz ocenić, czy dokonanie ewaluacji architektury oprogramowania mogło zapobiec takiemu niepowodzeniu.

(4) Które atrybuty jakościowe wykazują tendencje do bycia uznanymi za najważniejsze w przypadku systemów w jednostce organizacyjnej Czytelnika? W jaki sposób określa się te atrybuty? Skąd architekt wie, o które z nich chodzi, co oznaczają oraz jaki dokładnie poziom każdego nich jest wymagany?

(5) Dla każdego atrybutu jakościowego omówionego w niniejszym rozdziale — lub dla każdego, który Czytelnik określił w odpowiedzi na poprzednie pytanie — należy podjąć trzy różne, hipotetyczne decyzje architektoniczne, które miałyby wpływ na dany atrybut. Na przykład decyzja o obsługiwaniu zapasowej bazy danych prawdopodobnie zwiększy poziom dostępności systemu.

(6) Należy dokonać wyboru trzech lub czterech par atrybutów jakościowych. Dla każdej pary (należy wziąć tutaj pod uwagę pewne kompromisy) należy podjąć hipotetyczne decyzje architektoniczne, które zwiększałyby jakość pierwszego atrybutu kosztem drugiego. Następnie podjąć inne hipotetyczne decyzje architektoniczne, które podniosą jakość drugiego atrybutu, ale obniżą jakość pierwszego.