دانشگاه پیام نور
واحد تهران شمال

سمینار ۱ کارشناسی ارشد در رشته

مهندسی کامپیوتر

عنوان پایان نامه

روش ارزیابی جدید برای معماران مرجع به منظور طراحی یک معماری مرجع نرم‌افزار برای دستگاه‌های جابجایی مایعات

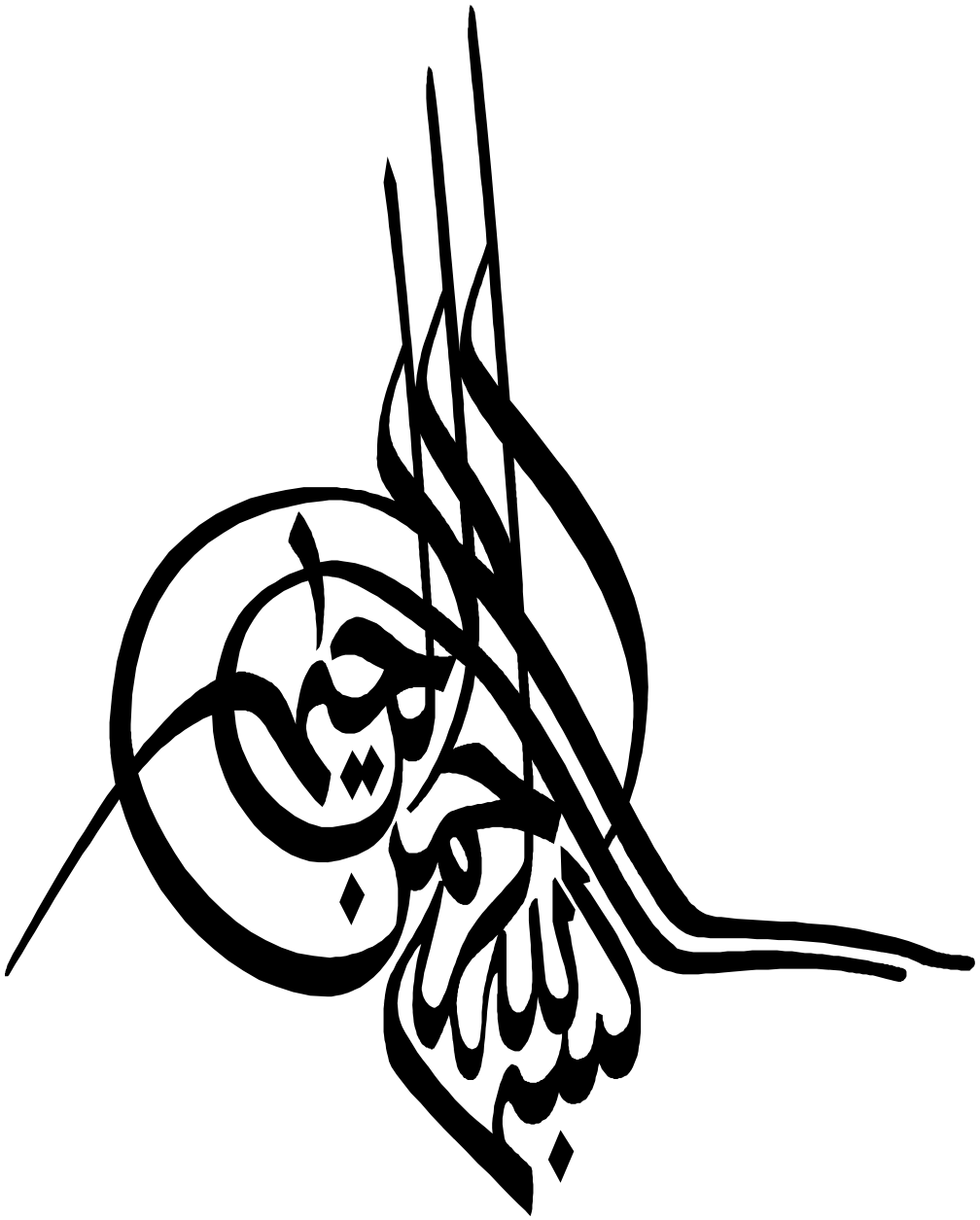
استاد راهنما

دکتر سید علی رضوی ابراهیمی

نگارنده

اکبر حمیدی

مرداد ۱۴۰۱


بسم الله الرحمن الرحيم

دانشگاه پیام نور

سمینار کارشناسی ارشد در رشته

مهندسی کامپیوتر

عنوان پایان نامه

روش ارزیابی جدید برای معماران مرجع به منظور طراحی یک معماری مرجع نرم‌افزار برای دستگاه‌های جابجایی مایعات

استاد راهنما

دکتر سید علی رضوی ابراهیمی

نگارنده

اکبر حمیدی

مرداد ۱۴۰۱

صفحه تاییدیه هیات داوران

فرم تعهد اصالت پایان نامه

اینجانب اکبر حمیدی دانش آموخته مقطع کارشناسی ارشد ناپیوسته تخصصی در رشته مهندسی کامپیوتر با که در تاریخ ..... از پایان نامه/ رساله خود تحت عنوان روش ارزیابی جدید برای معماران مرجع به منظور طراحی یک معماری مرجع نرم‌افزار برای دستگاه‌های جابجایی مایعات با کسب نمره ..... دفاع نموده‌ام بدینوسیله متعهد می شوم:

۱-این پایان‌نامه/ رساله حاصل تحقیق پژوهش انجام شده توسط اینجانب بوده و در مواردی که از دستاوردهای علمی و پژوهشی دیگران (اعم از پایان‌نامه، کتاب، مقاله و ...) استفاده نموده‌ام، مطابق ضوابط ورودیه موجود، نام منبع مورد استفاده و سایر مشخصات آن را در فهرست مربوطه ذکر و درج کرده‌ام.

۲- این پایان‌نامه/ رساله قبلاً برای دریافت هیچ مدرک تحصیلی(هم سطح، پایین تر یا بالاتر) در سایر دانشگاه ها و مؤسسات آموزش عالی ارائه نشده است.

۳- چنانچه بعد از فراغت از تحصیل، قصد استفاده و هر گونه بهره برداری اعم از چاپ کتاب، ثبت اختراع و ... از این پایان‌نامه داشته باشم، با ذکر نام استادان راهنما و مشاور و درج نام دانشگاه پیام نور اقدام خواهم کرد.
۴- چنانچه در هر مقطع زمانی خلاف موارد فوق ثابت شود، عواقب ناشی از آن را می‌پذیرم و دانشگاه پیام نور مجاز است با اینجانب مطابق ضوابط و مقررات رفتار نموده و در صورت ابطال مدرک تحصیلی‌ام هیچگونه ادعایی نخواهم داشت.

نام و نام خانوادگی

تاریخ، و امضا،

سپاسگذاری

سپاس و ستایش مر خدای را جل و جلاله که آثار قدرت او بر چهره روز روشن تابان است و انوار حکمت او در دل شب تار، درفشان.

آفریدگاری که خویشتن را به ما شناساند و درهای علم را بر ما گشود و عمری و فرصتی عطا فرمود تا بدان بنده ضعیف خویش را در طریق علم و معرفت بیازماید. اینک که پایان نامه کارشناسی ارشد خود را به پایان می رسانم جا دارد تا از زحمات استاد ارجمندم *سید علی رضوی ابراهیمی* که در تمامی مراحل این تحقیق همواره از نظرات ارزنده شان بهره بردم، تقدیر و تشکر کنم.

فهرست مطالب

فهرست اشکال

فهرست جداول

چکیده

توسعه نرم‌افزار برای اطمینان از نیازمندی‌ها و کیفیت‌های سیستم مساله بسیار مهمی است. اگر فرآیند معماری به طور ضعیف اجرا شود، آنگاه سیستم ممکن است انتظارات کاربران را برآورده نکند. یک معماری مرجع می‌تواند فرآیند دستیابی به یک معماری منسجم را تسهیل کند، زیرا رویکردهای معماری، طرح‌ها و اجزای آن را به عنوان نقطه شروع فراهم می‌کند، و این مساله باعث می‌شود که فرآیند معماری به یک پروسه‌ی قابل تکرار تبدیل شود. یک معماری مرجع می‌تواند در حوزه دستگاه آزمایشگاهی به ویژه برای دستگاه‌های جابجایی مایع[۱] به دلیل الزامات و ویژگی‌های مشترکی که این نوع دستگاه‌ها دارند، مفید واقع گردد. برای رسیدن به یک معماری مرجع، نیاز به یک ارزیابی جامع و کامل می‌باشد. باید به این نکته دقت کرد که تعداد روشهای ارزیابی در زمینه‌ی معماری‌های مرجع محدود نبوده و متاسفانه تعداد زیادی از این نوع روش‌های ارزیابی امکان پذیر می‌باشد. در نتیجه، این پایان‌نامه یک روش ارزیابی تطبیق یافته از روش تجزیه و تحلیل مصالحه‌ای معماری[۲] (ATAM) را برای ارزیابی معماری‌های مرجع ارائه می‌دهد. روش ارزیابی تطبیق یافته با استفاده از آن در دستگاه جابجایی مایع برای به دست آوردن یک معماری مرجع مورد آزمایش قرار گرفت. معماری مرجع به‌دست‌آمده نیز در این پایان‌نامه ارائه شده‌است و با استفاده از آن به عنوان یک تسهیل‌کننده برای پیاده‌سازی یک نمونه کوچک از یک دستگاه جابجایی مایع مورد آزمایش قرار گرفته‌است.

[۱] liquid handling devices
[۲] Architecture Tradeoff Analysis Method

۱ مقدمه

معماری نرم‌افزار بخش مهمی از هر سیستمی است. این کار به افراد فنی و غیر فنی کمک می‌کند تا درک کنند که چگونه یک سیستم اجرا و مستقر خواهد شد. اگر معماری نرم‌افزار به درستی طراحی نشود، تصمیمات ضعیف معماری در برآورده کردن نیازها و کیفیت‌ها پیش می‌آید. علاوه بر این، معماری نرم‌افزار اگر از ابتدای امر برای هر سیستم طراحی شود، حتی اگر سیستم‌ها یک دامنه مشابه را به اشتراک بگذارند، ممکن است فرآیند زمان‌بری باشد. بنابراین، داشتن یک معماری مرجع نرم‌افزار[۳] (SRA) برای یک دامنه خاص می‌تواند مزایایی را در زمینه‌ی کاهش زمان و تلاش مورد نیاز برای دستیابی به سیستم‌های منسجم[٤] (معین)در یک دامنه فراهم کند. آن می‌تواند به عنوان یک تسهیل‌کننده برای تسهیل فرآیند حصول معماری معینی از نرم‌افزار عمل کند، زیرا نیازمندی‌ها و کیفیت‌های مشترک از قبل توسط SRA مدیریت خواهد شد.

برای دستیابی به یک SRA خوب، یک روش ارزیابی باید مورد استفاده قرار گیرد. روش‌های ارزیابی معماری زیادی وجود دارند، شناخته‌شده‌ترین آن‌ها ATAM است. با این حال، هیچ یک از آن‌ها منحصر به یک SRA خاص نیستند، و این مساله می‌تواند منجر به بروز برخی معایب شود. در نتیجه، این مقاله یک روش ارزیابی معماری را ارائه می‌دهد که از ATAM اقتباس شده‌است. روش‌های ارزیابی سازگار[٥] معماران و تیم نرم‌افزار را برای رسیدن به یک SRA هدایت می‌کند که می‌تواند الزامات و کیفیت‌هایی که ذینفعان از یک دامنه انتظار دارند را برآورده کند.

علاوه بر این، برای اثبات اثربخشی روش ارزیابی تطبیق یافته، این روش در حوزه‌ی دستگاه‌های آزمایشگاهی به ویژه برای دستگاه‌های جابجایی مایعات استفاده شد. حاصل کار یک SRA برای دستگاه‌های جابجایی مایع شد که در این مقاله نیز ارائه شده‌است.I.DOT One یک دستگاه آزمایشگاهی است که به عنوان یک موضوع آزمون برای تست اثربخشی SRA با تلاش برای حصول معماری معینی از نرم‌افزار و اجرای یک نمونه اولیه کوچک با استفاده از SRA مورد استفاده قرار گرفت. در پایان، کارهای آینده در این تحقیق مورد بررسی مختصر قرار می‌گیرند.

[۳] Software Reference Architecture
[٤] concrete
[٥] adapted evaluation methods

۱.۱ بیان مساله

در حال حاضر یک معماری نرم‌افزاری باید از ابتدا برای هر دستگاه آزمایشگاهی ایجاد شود، حتی اگر آن‌ها برخی از سناریوهای استفاده اولیه و قابلیت‌هایی مانند اشتراک‌گذاری داده، دسترسی از راه دور و اتوماسیون را به اشتراک بگذارند. توسعه معماری نرم‌افزار برای هر سیستم در یک دامنه خاص زمانبر است زیرا ویژگی‌های نرم‌افزارهای رایج باید بیش از یک‌بار ارزیابی شوند. ویژگی‌های مشترکی مانند:

۱- چگونگی تعامل کاربر با دستگاه‌ها،
۲- اتوماسیون،
۳- امنیت،
٤- اتصال،
٥- یکپارچه‌سازی ابری[٦]،
٦- قابلیت نگهداری.

یک معماری نرم‌افزار باید برای پیاده‌سازی و انجام عملکردهای مورد نیاز، ویژگی‌های کیفیت، محدودیت‌ها و اصول طراحی شود [Bro۱۸]. اگر یکی از این الزامات در نظر گرفته نشود، ممکن است مشکلات متعددی در حین و پس از توسعه نرم افزار ایجاد شود که می تواند بر زمان پاسخگویی، امنیت و قابلیت نگهداری تأثیر بگذارد [Bro۱۸]. علاوه بر این، معماری نرم‌افزاری با طراحی ضعیف باعث می‌شود که منطق تجاری، رابط کاربری و کد سخت‌افزار به‌شدت مرتبط شوند. اتصال قوی غیرضروری، نگهداری، بهبود و درک نرم افزار را به دلیل وابستگی‌هایی که معرفی می‌کنند دشوارتر می کند.

مشکلات ناشی از طراحی ضعیف معماری نرم‌افزار را می‌توان در هر حوزه‌ای مشاهده کرد. آنها برای مثال می‌توانند بر قابلیت نگهداری که توانایی معماری برای مقابله با سناریوهای غیرمنتظره پس از طراحی یا پیاده‌سازی سیستم است، تأثیر بگذارند. به عنوان مثال، در حوزه دستگاه آزمایشگاهی، ویژگی‌های مشترک جدید باید به یک دستگاه توسعه یافته معرفی شود. ویژگی‌های جدید برای مثال دسترسی از راه دور، اتوماسیون یا یکپارچه‌سازی ابری هستند. برای فعال‌کردن این ویژگی‌ها، چندین تصمیم فنی و ریسک باید ارزیابی شوند، مانند

[٦] cloud integration

پروتکل‌های ارتباطی، مسائل امنیتی و محدودیت‌های سخت افزاری. اگر به درستی با آن برخورد نشود، تصمیمات معماری ناکافی گرفته می شود.

۲.۱ هدف

هدف اصلی این پایان‌نامه توسعه یک SRA برای یک زیرمجموعه خاص از دستگاه‌های آزمایشگاهی است که دستگاه‌های کنترل مایع نامیده می‌شوند. این SAR باید به عنوان یک تسهیل‌کننده برای حصول معماری معینی از نرم‌افزار عمل کند. به عبارت دیگر، یک تسهیل کننده SRA به عنوان پایه‌ای برای سیستم‌های آینده عمل می‌کند [CMV+۰۹]. این یک نقطه شروع برای معماران است تا از نوآوری مجدد و اعتبارسنجی مجدد سناریوهای مشترک و ویژگی‌های کیفی در یک دامنه خاص جلوگیری کنند [CMV+۰۹]. یک SRA باید تمام سناریوهای مشترک و ویژگی‌های کیفی یک زیرمجموعه دستگاه آزمایشگاهی را پوشش دهد. در صورت نیاز، باید به راحتی برای تطبیق با یک دستگاه خاص و ساخت معماری مشخص نرم‌افزاری سازگار شود .

با این حال، قبل از معرفی یک SRA برای زیرمجموعه دستگاه آزمایشگاهی، معماری‌های مختلف باید در برابر یکدیگر مقایسه شوند. این کار با استفاده از یک روش ارزیابی مشخص برای ارزیابی معماران و به دست آوردن مناسب‌ترین روش انجام خواهد شد. روش ارزیابی نشان می‌دهد که آیا یک SRA برای یک حوزه قابل‌اجرا است و آیا سناریوهای رایج و ویژگی‌های کیفیت را پوشش می‌دهد. علاوه بر این، بسته به ارزیابی SRA، ممکن است چندین SRA برای استفاده در حوزه‌های مختلف دستگاه آزمایشگاهی معرفی شوند. به دلیل نیاز به مقایسه معماری، روش ارزیابی برای SRA در این پایان‌نامه معرفی خواهد شد.

برای رسیدن به این هدف، باید به دو سوال تحقیق پاسخ داده شود. اینها عبارتند از:

- **RQ۱**: آیا یک روش ارزیابی معین وجود دارد که بتواند برای ارزیابی SRA در دستگاه‌های آزمایشگاهی استفاده شود؟ آیا روش‌های موجود می‌توانند برای داشتن تناسب با این مورد خاص سازگار شوند؟
- **RQ۲**: یک SRA مناسب برای دستگاه‌های جابجایی مایعات، که با روش ارزیابی از قبل تعیین‌شده اندازه‌گیری می‌شود، چیست؟

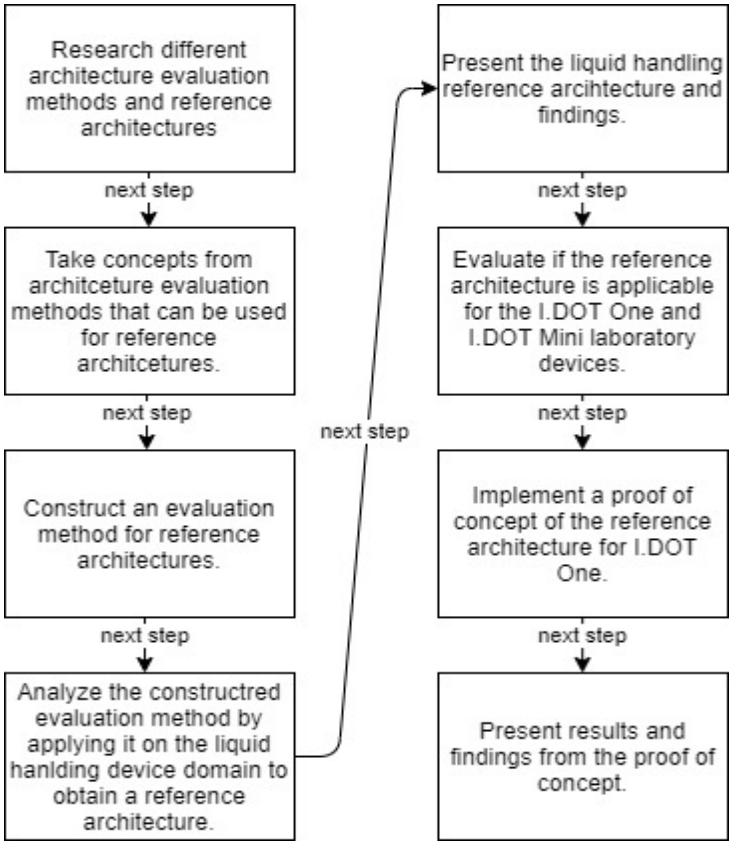
۳.۱ مشارکت و متدولوژی تحقیق

نقش این پایان‌نامه از عوامل متعددی ناشی می‌شود.

اولین مورد، حل زمان‌بر طراحی و توسعه نرم‌افزارهای دستگاه‌های آزمایشگاهی است که شرکت‌ها با آن مواجه هستند و مبنایی برای نحوه پیاده‌سازی اتوماسیون، اتصال ابری و دسترسی از راه دور برای دستگاه‌های آزمایشگاهی فراهم می‌کند. مقالات زیادی وجود ندارد که در مورد معماری نرم‌افزار دستگاه‌های آزمایشگاهی صحبت می‌کند، به‌طور خاص، صحبت‌های بسیار کمی در مورد دستگاه‌های انتقال مایعات وجود دارد.

سهم دوم پر کردن شکاف تحقیقاتی است که با روش‌های ارزیابی معماری پیدا می‌شود. مقالات زیادی در مورد چگونگی ارزیابی معماری‌های مرجع یا ارائه یک روش ارزیابی که به سمت طراحی و تجزیه و تحلیل یک معماری مرجع است، بحث نمی‌کنند..

شکل۱.۱متدولوژی تحقیق را نشان می‌دهد که قراراست دراین پایان‌نامه برای پاسخ به سوالات تحقیق دنبال شود.



شکل ۱-۱: متدولوژی تحقیق

۲ آخرین فناوری روز

یک مشکل رایج برای مساله اتوماسیون در آزمایشگاه‌ها وجود دارد [KYZC۱۲]. در این آزمایشگاه‌ها معمولا سیستم‌ها بدون در نظر گرفتن ادغام با سایر سیستم‌ها یا دستگاه‌های دیگر طراحی می‌شوند[KYZC۱۲]. این امر می‌تواند تلاش برای ادغام سیستم‌های مختلف با یکدیگر را بسیار پیچیده‌تر کند. برخی تحقیقات برای استانداردسازی ارتباطات و داده‌ها بین دستگاه‌های آزمایشگاهی انجام شده‌است [۲۰] [PSL+۲۰]. یکی از معروف‌ترین آنها SilA۲ است [SA۱۷] [۲۰] [PSL+۲۰]. کونگ و همکاران [KYZC۱۲] اشاره می‌کنند که می‌توان بسیاری از لوازم جانبی را به سیستم دستگاه جابجایی مایع اضافه کرد به نحوی که هدف برنامه‌پذیر بودن سیستم با یک رابط کاربر پسند و قابل ادغام با سایر دستگاه‌ها حاصل شود. این معماری مشخص می‌کند که چگونه دستگاه‌ها می‌توانند خودکار شده و با یکدیگر ادغام گردند. برای مثال، [KGFČ۱۵] یک معماری برای کنترل خودکار آزمایشگاه‌های راه دور ارائه می‌دهد. مثال‌های دیگر [ZSZ۱۱] و [Bis۱۳] هستند که یک معماری را برای ساخت آزمایشگاهی به صورت مجازی (از راه دور) ارائه می‌دهند. همچنین مقاله‌ای [SA۱۷] وجود دارد که یک معماری مرجع متمرکز را برای ادغام دستگاه‌های آزمایشگاهی مختلف مانند دستگاه‌های جابجایی مایعات فراهم می‌کند. معماری مرجع قرار است هنگام اضافه کردن پلتفرم‌ها در یک آزمایشگاه خودکار به مسائل فنی بپردازد [SA۱۷]. همه اجزای نرم‌افزاری دیگر مانند رابط کاربر و انتقال داده با سیستم متمرکز اصلی از طریق اتصال‌دهنده‌ها[۷] ادغام خواهند شد [SA۱۷].

متاسفانه، مستندات عمومی زیادی در مورد مشکلات پیش روی توسعه نرم‌افزار برای دستگاه‌های انتقال مایعات وجود ندارد. مشکلات معمولا در داخل شرکت‌ها رخ می‌دهد. مشکلاتی مانند اینکه چه چارچوب‌هایی، الگوهای معماری و فرمت های داده باید استفاده شوند. هیچ کدام از آن‌ها ساختار معماری نرم‌افزار دستگاه‌های جابجایی مایع را مشخص نمی‌کنند. اکثر معماری‌ها کل آزمایشگاه را بدون تمرکز بر یک دستگاه آزمایشگاهی و یا نوع خاصی از دستگاه‌های آزمایشگاهی پوشش می‌دهند. بیشتر کارها یک معماری ارائه می‌دهند که نیازمند این است که مشتری میزبان یک ماشین متمرکزسازی باشد که با چندین دستگاه ادغام شده است. این امر به کاربر پیچیدگی‌های بیشتری می‌افزاید، به خصوص اگر کاربران دستگاه‌های آزمایشگاهی زیادی نداشته باشند و و یا جنبه‌های خاصی از معماری مانند ارتباط از راه دور یا اتوماسیون را مطالبه نکنند. همچنین داشتن یک پایگاه‌داده

[۷] Connectors

برای همه دستگاه‌ها نیز می‌تواند پیچیدگی سیستم را افزایش دهد زیرا دستگاه‌های آزمایشگاهی مختلف نیاز به طرح‌های مختلفی دارند. با توجه به دستگاه آزمایشگاهی، معماری‌های ارائه‌شده کلی بوده و دیدگاه‌های متفاوتی از معماری ارائه نمی‌دهند. در نتیجه، این پایان‌نامه در نظر دارد تا این شکاف را پر کند و یک معماری مرجع دقیق برای دستگاه‌های انتقال مایع ارایه نماید.

۳ آشنایی با معماری نرم‌افزار و طراحی نرم‌افزار

قبل از معرفی یک SRA برای دستگاه‌های آزمایشگاهی، این فصل به طور خلاصه اصطلاحات معماری نرم‌افزار، طراحی نرم‌افزار و معماری مرجع را تعریف می‌کند زیرا تعریف مشترکی برای آن‌ها وجود ندارد.

۱.۳ معماری نرم‌افزار

ابتدا یک تعریف معماری نرم‌افزار ارائه می‌شود و سپس درایورهای معماری نرم‌افزار ارائه می‌شود.

۱.۱.۳ تعاریف

چندین تعریف از معماری نرم‌افزار ارائه شده‌است. شرط مشترک در میان آن‌ها این است که اطمینان حاصل شود که دیدگاه و هدف کلی سیستم ارضا شده‌است. معماری نرم‌افزار، اگر بر این اساس دنبال شود، خروجی نهایی سیستم را تعیین می‌کند. در ابتدا، چهار تعریف معماری نرم‌افزار ذکر می‌شوند، و سپس یک تعریف انتخابی براساس دانش از چهار تعریف و مطابقت آن‌ها با یک SRA دنبال می‌گردد:

- اولین تعریف از موسسه مهندسی نرم‌افزار[۸] [CBB+۱۰] است که توسط فیربنکس[۹][Fai۱۲] نیز ذکر شده‌است. او می‌گوید: "معماری نرم‌افزاری یک سیستم محاسباتی مجموعه‌ای از ساختارهای مورد نیاز برای استدلال در مورد سیستم است که عناصر نرم‌افزاری، روابط بین آن‌ها و ویژگی‌های هر دو را شامل می‌شود."

[۸] Software Engineering Institute
[۹] Fairbanks

- تعریف دوم از سولمز[۱۰] [Sol۱۲] است که در آن معماری نرم‌افزار را به عنوان "زیرساخت نرم‌افزاری" تعریف می‌کند که در آن مولفه‌های نرم‌افزار که عملکرد کاربر را فراهم می‌کنند را می‌توان مشخص، مستقر و اجرا کرد. اجزای برنامه کاربردی یا اپلیکیشن[۱۱]، همان اجزای نرم‌افزاری هستند که به نیازمندی‌های عملکردی سیستم نرم‌افزاری می‌پردازند."
- تعریف سوم، تعریف [RW۱۲] است. نویسندگان معماری نرم‌افزار را به عنوان سیستمی تعریف می‌کنند که از عناصر سیستم و روابط بین آن‌ها، ویژگی‌های سیستم که از قابلیت‌های سیستم و ویژگی‌های کیفیت تشکیل شده‌است، و اصول طراحی که به عنوان راهنمایی برای تعریف معماری عمل می‌کنند، تشکیل گردیده‌است.
- آخرین و چهارمین تعریف از [Bro۱۸] است که در آن نویسنده معماری نرم‌افزار را به عنوان "ترکیبی از معماری برنامه کاربردی و معماری سیستم، و در ارتباط با ساختار و "چشم‌انداز[۱۲]" تعریف می‌کند. معماری برنامه، چگونگی سازماندهی، طراحی و ساخت برنامه را توصیف می‌کند. معماری سیستم دیدگاه بالاتری دارد که از واحدهای قابل گسترش زیادی تشکیل شده‌است که برای درک نرم‌افزار و سخت‌افزار کنار هم قرار گرفته‌اند.

تمام تعاریف در مورد رسیدن به هدف سیستم و ارائه دیدگاه ذینفعان است. تعریف اول و سوم مناسب‌ترین تعریف برای این مقاله هستند زیرا آن‌ها تمام جنبه‌های مختلف معماری را در نظر می‌گیرند که باید در یک SRA مدل‌سازی شوند. جنبه‌های معماری عبارتند از:

۱- عناصر نرم‌افزاری،
۲- روابط بین آن‌ها
۳- ویژگی‌های هر کدام برای توصیف ساختار یک سیستم.

این جنبه‌های معماری توضیح می‌دهند که چگونه سیستم سازماندهی، طراحی و ساخته می‌شود تا عملکردهای مورد نیاز سیستم، ویژگی‌های کیفیت، سناریوها، محدودیت‌ها و اصولی که توسط سهامداران فراهم می‌شوند را ارائه دهد. نتایج قابل تحویل معماری نرم‌افزار یعنی:

۱- عملکردها(کارکردها)،
۲- ویژگی‌های کیفیت،

[۱۰] Solms
[۱۱] Application components
[۱۲] vision

۳- سناریوها،
٤- محدودیت‌ها
٥- اصول

نیز محرک‌های معماری هستند و در عمل، معماری نرم‌افزار نهایی را شکل می‌دهند.

۲.۱.۳ درایورهای معماری

محرک‌های معماری از پنج عنصر تشکیل شده‌اند که سیستم باید ارائه دهد. در زیر به اختصار توضیح داده شده است.

الزامات عملکری

موارد استفاده و الزامات عملکردی اغلب اولین جنبه‌ای هستند که مورد بحث قرار می‌گیرند زیرا درک کلی از اهداف اصلی سیستم را فراهم می‌کنند و در حقیقت، قابلیت‌های سیستم و کاربران نهایی را مشخص می‌کنند. قابلیت‌های سیستم سناریوهایی از آنچه کاربر نهایی می‌تواند با سیستم به دست آورد را فراهم می‌کند [RW۱۲] [Bro۱۸].

ویژگی‌های کیفی

ویژگی‌های کیفی، که به عنوان الزامات غیر کارکردی نیز شناخته می‌شوند، مشخص می‌کنند که سیستم چه ویژگی‌هایی را باید ارائه دهد. ویژگی‌های کیفی برای مثال:

۱- عملکرد[۱۳]،
۲- امنیت[۱٤]،
۳- قابلیت نگهداری[۱٥]،
٤- قابلیت استفاده[۱٦]،
٥- ثبات[۱۷]،
٦- قابلیت درک[۱۸]

[۱۳] performance,
[۱٤] security,
[۱٥] maintainability,
[۱٦] usability,
[۱۷] stability,
[۱۸] understandability,

۷- قابلیت تغییر [۱۹]

هستند. الزامات سیستم را می توان با استفاده از تصمیمات معماری مختلف اجرا کرد. با این حال، هر تصمیم معماری، ویژگی‌های کیفی را به طور متفاوت برآورده می‌کند، زیرا این ویژگی‌ها با توجه به تبادلات و ریسک‌های مختلفی که معرفی می‌کنند، متفاوت هستند. در نتیجه، مهم است که این ویژگی‌های کیفی توسط سهامداران مورد بحث، و اولویت‌بندی قرار گیرند تا دلایلی برای پیروی از یک تصمیم ارائه شود [Fai۱۲] [RW۱۲] [Bro۱۸].

سناریوها

یک سناریو توصیف کوتاهی از تعامل ذینفعان با سیستم است[KKC۰۰]. یک معماری نرم‌افزار طراحی شده باید بر روی میزان رضایت کاربران از سناریوها ارزیابی شود. براساس نظر کازمن و همکاران[۲۰] [KKC۰۰]، سناریوها را می‌توان به سه نوع مختلف تقسیم کرد:

۱- مورد استفاده[۲۱]،
۲- رشد[۲۲]،
۳- سناریوهای اکتشافی[۲۳].

سناریوی مورد استفاده مشابه با الزامات عملکردی تعریف‌شده قبلی است. آن‌ها موارد استفاده مختلفی که کاربر نهایی می‌تواند با سیستم داشته باشد را مشخص می‌کنند. سناریوی رشد، سناریویی است که تغییرات آینده را پیش‌بینی می‌کند که ممکن است در سیستم رخ دهد. یک سناریوی اکتشافی سناریویی است که تغییرات شدید در سیستم را بررسی می‌کند. این کار برای تاکید بر سیستم و آزمایش قابلیت‌های آن تحت شرایط غیر معمول انجام می‌شود [RW۱۲].

[۱۹] changeability
[۲۰] Kazman et al.
[۲۱] Use case
[۲۲] growth,
[۲۳] exploratory scenarios

اجبار(محدودیت‌ها)

محدودیت‌ها یا Constraints، محدودیت‌های معماری سیستم را مشخص می‌کنند. آن‌ها اغلب توسط سهامداران ارائه می‌شوند، و محدودیت‌هایی را در مورد این که چگونه نیازهای عملکردی و غیر عملکردی می‌توانند به خوبی درک شوند، تعیین می‌کنند. محدودیت‌ها می‌توانند

۱- زمان،
۲- هزینه،
۳- تکنولوژی مورد استفاده،
٤- سازمان یا تیم پیاده‌سازی
۵- توسعه سیستم

باشند. محدودیت‌ها می‌تواند براساس آشنایی تیم با حوزه‌های فن‌آوری خاصی ایجاد شده باشد که گزینه‌های مورد استفاده برای پیاده‌سازی معماری را محدود می‌کند [RW۱۲] [Bro۱۸].

اصول

اصول معماری، قوانین و اصولی هستند که برای هدایت طراحی و اجرای سیستم دنبال می‌شوند. اصول می‌توانند به عنوان یک نقطه شروع خوب برای انتقال از فضای مساله به فضای راه‌حل عمل کنند در حالی که دلایل ریشه‌ای تصمیمات معماری را فراهم می‌کنند [RW۱۲]. اصول معماری می‌تواند نوعی سبک معماری باشد که باید به دقت پیگیری و دنبال شود. به عنوان مثال، یک معماری لایه‌ای N – لایه، لوله‌ها و فیلتر، یا یک معماری مرکز داده. علاوه بر این، اصول معماری همچنین می‌توانند به مفاهیمی اشاره کنند که معماری باید داشته باشد، مانند جفت شدن یا استقلال اجزا [Fai۱۲] [RW۱۲] [Bro۱۸] .

۲.۳ تمایز بین طراحی نرم‌افزار و معماری نرم‌افزار

معماری نرم‌افزار طراحی نرم‌افزار است اما برعکس این گزاره صادق نیست [Bro۱۸]. طراحی نرم‌افزار در مورد انتخاب یک راه‌حل از بسیاری از راه‌حل‌های ممکن اجرای سیستم است [Bro۱۸]. یک معمار هنگام طراحی یک معماری نرم‌افزار باید بیشتر بر ویژگی‌های کیفی تمرکز کند تا ویژگی‌ها به عنوان یک مشخصه کلی، زیرا

ویژگی‌ها را می‌توان با استفاده از روش‌های مختلف طراحی نرم‌افزار با کیفیت‌های مختلف به دست آورد[Fai۱۲]. طراحی نرم‌افزار بیشتر به حوزه سطح پایین‌تر نحوه ساخت اجزا می‌پردازد.

برای ایجاد یک خط تعریفِ روشن بین معماری نرم‌افزار و طراحی نرم‌افزار، باید به این نکته اشاره کرد که معماری نرم‌افزار باید شامل تصمیمات مهمی باشد که به راحتی در چرخه‌های بعدی عمر نرم‌افزار قابل‌تغییر نیستند [Bro۱۸]. معماری نرم‌افزار مهم‌تر از طراحی نرم‌افزار در رسیدگی به مسائل مربوط به سیستم است، مانند استفاده غیر منتظره یا تغییر سناریوها و ویژگی‌های کیفی، زیرا ساختار سیستم در این مسائل منعکس می‌شود[Fai۱۲]. معماری نرم‌افزار جزئیات مورد نیاز برای دستیابی به یک ویژگی کیفی کلی سیستم را در بر می‌گیرد، از سوی دیگر، طراحی نرم‌افزار راه‌حل پیاده‌سازی برای ارائه کارکردها و برآورده کردن ویژگی‌های کیفی است.

۳.۳ معماری مرجع نرم‌افزار

مشابه با معماری و طراحی نرم‌افزار، هیچ تعریف مشترکی در مورد SRA وجود ندارد [CMV+۰۹]. با این حال، SRA معمولا به عنوان یک استاندارد یا تسهیل‌کننده برای حصول معماری معینی از نرم‌افزار شناخته می‌شود. این معماری در سطح بالاتری از انتزاع[۲۴] در مقایسه با معماری نرم‌افزار واقعی طراحی شده‌است [AGG۱۲]. این مساله باعث می‌شود یک SRA به عنوان یک استاندارد و یا یک تسهیل‌کننده برای زمینه‌های مختلف در یک حوزه مشابه قابل‌اجرا باشد. هدف یک SRA استاندارد قابلیت هم‌کاری سیستم است در حالی که تسهیل‌کننده با هدف فراهم کردن دستورالعمل‌های طراحی معماری نرم‌افزار به شکل منسجم عمل می‌کند [AGG۱۲].

کلوتیر و همکاران [CMV+۰۹] و آنجلوف و همکاران[AGG۱۲] . SRA را به‌عنوان هدفی تعریف می‌کند که معماری‌های موجود و دانش دامنه را برای کمک به توسعه با اجرای عملکردهای مشترک و جریان داده‌های معماری نرم‌افزار جذب می‌کند. یک SRA همچنین می‌تواند به عنوان یک معماری نرم‌افزار سطح بالا دیده شود که عملکردهای مشترک، ویژگی‌های کیفیت، محدودیت‌ها و اصول را برای یک دامنه خاص ثبت می‌کند.

مزایای SRA عبارتند از:

- کاهش هزینه‌ها و زمان توسعه[MSA+۱۵] [MAFA۱۳] .
- بهبود قابلیت درک معماری و ارتباط میان تیم‌ها و ذینفعان [MAFA۱۳] [CMV+۰۹] .

[۲۴] abstraction

- کاهش ریسک [CMV+۰۹].
- افزایش کیفیت کلی نرم‌افزار [MSA+۱۵] [MAFA۱۳].
- **اعتبار مجدد و اختراع مجدد راه حل‌ها را حذف کنید زیرا استانداردی برای طراحی و توسعه تاسیسات ارائه می‌دهد [AGG۱۲] [CMV+۰۹].**
- **می‌تواند به خوبی در شرکت‌ها و سازمان‌هایی که با چندین پروژه نرم‌افزاری مشابه در حال گسترش هستند سازگار شود [MSA+۱۵].**
- **مناسب برای معماری یک دامنه کاربردی خاص برای تعریف محرک‌های معماری رایج از جمله ویژگی‌های کیفیت [MSA+۱۵].**

حذف اعتبارسنجی مجدد و ابداع مجدد راه‌حل‌ها زیرا استاندارد طراحی و توسعه تسهیلات را فراهم می‌کند و می‌تواند به خوبی در شرکت‌ها و سازمان‌هایی که در حال توسعه با چندین پروژه برنامه کاربردی نرم‌افزاری مشابه هستند، سازگار شود.

معایب SRA عبارتند از:

- یک منحنی یادگیری را برای معماران و توسعه‌دهندگان معرفی می‌کند اگر آن‌ها با SRA آشنا نباشند[MSA+۱۵] [MAFA۱۳].
- سرمایه‌گذاری زمان اولیه در ساخت SRA می‌تواند یک مشکل باشد اگر سیستم نیاز به استقرار در یک دوره زمانی کوتاه داشته باشد[MAFA۱۳]
- مسائل وابستگی در SRA به این دلیل که باید قبل از اعمال آن به سیستم‌ها تغییر کند (اگر یک محرک معماری مشترک جدید معرفی شود) [MAFA۱۳].
- انعطاف‌پذیری محدود در سیستم‌ها به این دلیل است که محدود به یک حوزه‌ی خاص است [MAFA۱۳].

دامنه هدف SRA:

ابعاد طبقه‌بندی SRA: این بُعد برای طبقه‌بندی SRA استفاده می‌شود و به پرسش‌های زیر پاسخ می‌دهد:

- " از آن در کجا استفاده خواهد شد؟ به این سوال درباره نوع سازمان مورد نظر پاسخ می‌دهد[CBS۱۷] [AGG۱۲].

- " چه کسی آن را تعریف می‌کند؟ به این سوال پاسخ می‌دهد که آیا SRA برای یک سازمان یا چند سازمان در نظر گرفته شده است [CBS۱۷] [AGG۱۲].
- " چه زمانی تعریف می‌شود؟": به این سوال پاسخ می‌دهد که آیا SRA قبل یا بعد از توسعه معماری و برنامه کاربردی نرم‌افزار معین اجرا می‌شود [AGG۱۲].
- "مرحله بلوغ دامنه چیست؟": بررسی می کند که آیا دامنه بالغ است یا خیر تا اطمینان حاصل شود که یک SRA می تواند بدون تغییرات زیادی در آینده طراحی شود [CBS۱۷].

ابعاد طراحی SRA: این بعد تعیین می‌کند که چگونه SRA استفاده و اجرا خواهد شد [AGG۱۲]. همچنین شامل چگونگی جزئیات، توصیف و نمایش SRA است[CBS۱۷] [AGG۱۲]. و به پرسش‌های زیر پاسخ می‌دهد:

- "چه توضیح داده شده است؟": توضیح می‌دهد که SRA شامل چه چیزی می‌شود[CBS۱۷]. همچنین می‌تواند عناصر اساسی را در سطح بالایی فهرست کند [AGG۱۲].
- "چگونه نمایش داده می شود؟": توضیح می دهد که آیا SRA قرار است به صورت غیررسمی، نیمه رسمی یا رسمی ارائه شود[CBS۱۷] [AGG۱۲].
- "چگونه توضیح داده شده است؟": فرمت (بنی، گرافیکی، ...) را در مورد نحوه توصیف SRA مشخص می کند [CBS۱۷] .

۴ ارزیابی معماری

ارزیابی معماری یک فرآیند یا چارچوبی برای هدایت طراحی معماری نرم‌افزار و ارائه دلایلی برای پیروی از تصمیمات خاص معماری است. این روش برای توضیح معماری، بررسی اینکه آیا تصمیمات و مبلدلات کافی گرفته شده‌است یاخیر و ارائه نقاط تصمیم‌گیری و توافق‌نامه‌های رسمی بین ذینفعان استفاده می‌شود [RW۱۲].

دامنه یک SRA : این مقاله بر روی استفاده از یک SRA بر روی یک سازمان واحد متمرکز است که به عنوان یک تسهیل‌کننده (نه یک استاندارد) برای هدایت و ساده‌سازی طراحی معماری‌های منسجم که تحت همان دامنه با SRA عمل می‌کنند، عمل می‌کند. اجزا همان اجزای نرم‌افزاری رایج و تعاملات بین آن‌ها هستند. دامنه SRA

برای محدود کردن احتمالات مختلف روش‌های ارزیابی و روش‌هایی که می‌توانند در این پایان‌نامه دنبال شوند و برای کمک به تعریف بهتر لایه انتزاعی SRA تعریف می‌شود.

۱.٤ رویکرد ارزیابی معماری

یک ارزیابی معماری می‌تواند پنج رویکرد مختلف، یا ترکیبی از چندین رویکرد را برای ارزیابی یک معماری دنبال کند:

- تجربه قبلی توسعه دهندگان و ذینفعان [PS۱۰a].
- یک مدل ریاضی برای ارزیابی مشخصه‌های کمی کیفیت مانند عملکرد، تاخیر و قابلیت اطمینان[PS۱۰a].
- یک روش شبیه‌سازی برای تقلید اجزای معماری در سطح بالا [PS۱۰a].
- یک سیستم نمونه اولیه به عنوان اثبات مفهوم [RW۱۲]
- یک رویکرد مبتنی بر سناریو برای شناسایی سناریوها به منظور تحلیل اثرات آن‌ها بر ویژگی‌های کیفی[PS۱۰a] [RW۱۲].

استفاده از تجربه قبلی به عنوان یک رویکرد ارزیابی می‌تواند برای ارزیابی اعتبار آن دشوار باشد زیرا تجربه بین افراد مساله‌ی متفاوتی است و کیفیت متفاوتی نیز دارد. علاوه بر این، استفاده از آن در صورتی که تجربه قبلی وجود نداشته باشد، غیر ممکن است. همانطور که برای مدل ریاضی، یک رویکرد مناسب برای استفاده زمانی است که ویژگی‌های کمی اولویت بالایی دارند. با این حال، این مقاله فرض می‌کند که SRA شامل جزئیات سخت‌افزار نمی‌شود که باعث می‌شود مدل ریاضی یک رویکرد ناکافی باشد.

همانند روش‌های شبیه‌سازی و نمونه اولیه، رویکردهای ارزیابی می‌توانند به عنوان اثبات مفهوم برای ارزیابی قابلیت استفاده SRA با طراحی و اجرای معماری نمونه اولیه مبتنی براساس SRA مورد استفاده قرار گیرند. رویکرد مبتنی بر سناریو، ویژگی‌ها و سناریوهای مختلف کیفیت را از ذینفعان می‌گیرد و دلایلی برای پیروی از تصمیمات یک معماری خاص فراهم می‌کند. رویکرد مبتنی بر سناریو در این مقاله استفاده خواهد شد زیرا ویژگی‌های کیفی SRA را ارزیابی می‌کند و نحوه مقابله SRA با سناریوهای مختلف در یک حوزه ذرات را ارزیابی می‌کند. در حالی که رویکرد نمونه اولیه را می‌توان برای ارزیابی قابلیت استفاده SRA در مورد اینکه چقدر طراحی چندین معماری معین را تسهیل می‌کند، استفاده کرد.

۲.٤ روش‌های ارزیابی معماری

چندین روش ارزیابی سناریو، عمدتا برای ارزیابی معماری نرم‌افزار پیشنهاد شده‌اند. با این حال، روش‌های ارزیابی معماری خاصی نسبت به SRAs وجود ندارد. در نتیجه، این مقاله یک روش ارزیابی را تعیین می‌کند که جهت‌گیری بیشتری دارد و بر ارزیابی SRAs متمرکز است. روش ارزیابی براساس مقایسه معماری با توجه به اینکه چگونه SRAs محرک‌های معماری مختلف سیستم را برآورده می‌کند، خواهد بود.

برای تعیین یک روش ارزیابی برای SRA، رویکردهای ارزیابی قبلی باید مورد مطالعه قرار گیرند و مشخص شود که آیا آن‌ها می‌توانند به عنوان یک منبع بالقوه در نظر گرفته شوند. جدول A.۱ در ضمیمه A شامل روش‌های ارزیابی است که ممکن است به عنوان یک منبع برای تعیین یک روش ارزیابی برای SRA استفاده شود. از سوی دیگر، جدول A.۲ شامل روش‌های ارزیابی است که نمی‌تواند برای رسیدن به یک روش ارزیابی برای SRA استفاده شود.

ATAM و روش تجزیه و تحلیل معماری نرم‌افزار (SAAM) رایج‌ترین روش‌های ارزیابی مورد استفاده هستند و هر دو با بیش از ۸۰۰ مقاله مورد مطالعه قرار گرفته‌اند. ATAM یک بهبود در SAAM است و فرآیندی است که توضیح می‌دهد چگونه یک معماری می‌تواند براساس ویژگی‌های کیفیت و سناریوهای ثبت شده از طرف ذینفعان ارزیابی کرد. ارزیابی، ریسک‌ها، حساسیت‌ها و مبادلاتی را که هر تصمیم معماری با توجه به یک ویژگی یا سناریوی با کیفیت خاص معرفی می‌کند، تعیین می‌کند. ATAM یک روش کامل است که ساختار روشنی را در مورد چگونگی ارزیابی معماری فراهم می‌کند و به طور گسترده توسط بسیاری از سازمان‌های مختلف مورد استفاده قرار می‌گیرد. از میان تمام روش‌های ارزیابی قابل‌اجرا که در این پایان‌نامه ذکر شده‌اند، ATAM به عنوان پایه‌ای برای تطبیق آن برای ارزیابی SRA به جای معماری معینی از نرم‌افزار استفاده خواهد شد.

۳.٤ مشکلات ATAM برای معماران مرجع نرم‌افزار

ATAM یک روش ارزیابی مستند شده کامل است که به منظور ارزیابی و ارائه دلایلی برای تصمیمات معماری خاص، یک معماری منسجم را ارائه می‌دهد. در این بخش، معایب ذکر شده و سپس یک روش توسعه‌یافته ATAM برای رفع معایب ارائه خواهد شد. اشکالات براساس استفاده از ATAM برای ارزیابی یک SRA است که دامنه آن تعریف شده‌است:

- ❖ عیب اول: به دست آوردن ویژگی‌ها و سناریوهای کیفی برای یک SRA مشخص نشده است [ATG۰۸][ATG۱٤]. سهامداران برای فراهم کردن کارکردها، ویژگی‌های کیفی و سناریوها براساس یک سیستم خاص را استفاده می‌کنند. این امر می‌تواند برای سهامداران چالش برانگیز باشد اگر آن‌ها هر زمینه در دسترس را در نظر بگیرند، آنگاه SRA می‌تواند اعمال شود. این امر منجر به تکرار سناریو و افزایش ناخواسته در سناریوهایی می‌شود که برای یک سیستم قابل‌اجرا هستند[ATG۰۸][ATG۱٤] .
- ❖ عیب دوم: روش روشنی برای ارزیابی ویژگی‌های کیفی برای یک معماری مرجع فراهم نمی‌کند[ATG۰۸][ATG۱٤]. SRA در سطح بالاتری از انتزاع در مقایسه با معماری منسجم قرار دارد، و ATAM برای تعریف جزئیات سطح پایین هر ویژگی کیفیت به عنوان شکلی از سناریوها استفاده می‌شود. به عنوان مثال، برای ویژگی کیفیت عملکرد، ممکن است سناریویی وجود داشته باشد که در آن آستانه‌ی زمان اجرا نباید توسط سیستم افزایش یابد.
- ❖ عیب سوم: اگر SRA برای سازمان‌های مختلف در یک دامنه یک‌سان اعمال شود، تعداد ذینفعان می‌تواند بسیار افزایش یابد[ATG۰۸][ATG۱٤][GVV۰٥] . ATAM پیشنهاد می‌کند تا جای ممکن سهامداران بیشتری را درگیر کنند. با این حال، مشارکت ذینفعان از سازمان‌های مختلف برای لیست کردن و اولویت‌بندی ویژگی‌ها و سناریوهای کیفی ممکن است غیر ممکن باشد[ATG۱٤][GVV۰٥] [ATG۰۸].
- ❖ عیب چهارم: ATAM از مقایسه معماری‌هابا یکدیگر برای رسیدن به معماری بهینه پشتیبانی نمی‌کند [BRST۰٥]. ATAM برای ارزیابی تعادل یک معماری واحد، بدون در نظر گرفتن سایر روش‌های معماری طراحی شده‌است.

٤.٤ روش ارزیابی معماری برای معماری مرجع نرم‌افزار

روش ارزیابی برای پاسخ به RQ۱ استفاده خواهد شد. به طور دقیق‌تر، روش ارزیابی در حال تلاش برای پاسخ به سوال زیر برای از بین بردن معایب ATAM در هنگام اعمال آن به SRAs است:

سوال ۱: چگونه می توان ویژگی‌ها و سناریوهای کیفی را برای یک SRA به دست آورد؟

سوال ۲: چه روشی را می توان برای ارزیابی ویژگی‌ها و سناریوهای کیفیت SRA دنبال کرد؟

سوال ۳: اگر یک SRA به چندین سازمان اعمال شود، سهامداران از یک سازمان چگونه می‌توانند سازمان‌های دیگر را نمایندگی کنند؟

- ❖ آیا می‌توان سهامداران را از چندین سازمان، بدون تکرار ویژگی‌ها و سناریوهای کیفی بین سازمان‌ها متحد کرد؟

سوال ٤: چگونه می توان SRA های مختلف را براساس ویژگی‌ها و سناریوهای کیفی با یکدیگر مقایسه کرد؟

در این پایان‌نامه، ATAM به عنوان یک پایه در نظر گرفته‌شده و معایب آن براساس روش‌های ارزیابی مختلف و براساس راه‌حل‌های پیشنهادی مقاله مورد بررسی قرار گرفته‌است. در بخش‌های زیر، مراحل ATAM برای یافتن یک روش ارزیابی برای SRA ها خلاصه، بحث و اصلاح می‌شوند. گام‌های ATAM معمولا در یک رویکرد سلسله‌مراتبی دنبال می‌شوند و گامها را می‌توان به ترتیب‌های مختلف و در چندین تکرار دنبال کرد [KKC۰۰]. در این مراحل، به یک معمار برای انجام برخی وظایف اشاره شده‌است. معمار می‌تواند یک تیم / شخص نرم‌افزار، یک ارزیاب نرم‌افزار، یا هر شخص دیگری باشد که فرآیند ارزیابی را اجرا می‌کند. برای سادگی، عبارت معمار در سراسر مقاله استفاده شده‌است.

ATAM شامل نه مرحله است:

۱- ارائه ATAM
۲- دستورالعمل‌های کسب‌وکار موجود
۳- معماری موجود
٤- شناسایی رویکردهای معماری
۵- ایجاد مزیت ویژگی کیفیت
٦- تحلیل معماری
۷- طوفان مغزی و اولویت‌بندی سناریوها
۸- تحلیل رویکردهای معماری
۹- ارائه نتایج

۱.٤.٤ مرحله ۱: ATAM را ارائه کنید.

خلاصه ATAM

این مرحله به طور خلاصه فرآیند ارزیابی را به یک تیم کوچک از سهامداران ارائه می‌دهد. یک تیم بزرگ از سهامداران در مرحله ۲ شناسایی خواهند شد. این ارائه همچنین شامل تکنیک‌هایی است که برای تجزیه و تحلیل و ارزیابی معماران و خروجی نهایی فرآیند ارزیابی مورد استفاده قرار می‌گیرند. هدف از این مرحله پاسخ به سوالات سهامداران و اطمینان از این است که آنها همه مسئولیت‌های خود را می‌دانند [KKC۰۰].

بحث

یک رویکرد مشابه با ATAM را می توان دنبال کرد که هدف آن اطلاع دادن به تیم کوچک اولیه سهامداران در مورد فرآیند ارزیابی و مراحل بعدی است.

اصلاحات

هیچ تغییری در این مرحله ایجاد نشده است.

۲.٤.٤. مرحله ۲: ارائه محرک‌های کسب‌وکار

خلاصه ATAM

یک تصویر سطح بالا[۲۵] از سیستم به ذینفعان ارائه خواهد شد. این سیستم باید توسط همه شرکت کنندگان، سهامداران و کاربران درک شود. به عنوان مثال، ارائه‌ی سیستم باید شامل الزامات سطح بالا، اهداف و محدودیت‌های کسب‌وکار، محدودیت‌های فنی، و بررسی سهامداران اصلی باشد که باید در مراحل آینده (تیم بزرگ‌تر ذینفعان) مشارکت داشته باشند[KKC۰۰].

بحث

ATAM در این مرحله به ارائه محرک‌های کسب‌وکار سیستم تمرکز می‌کند. این روش به بیان جزئیات در مورد تعریف دامنه، که مرزهای تعریف بهتر SRA و مقایسه‌ی روش‌های معماری مختلف را شامل می‌شود، نمی‌پردازد[LBK۹۷]. یک حوزه به خوبی تعریف‌شده[۲۶]، اعتبار سنجی مجدد و ابداع مجدد راه‌حل‌های از قبل حل شده را حذف می‌کند [AGG۱۲] [CMV+۰۹]. معمار و ذینفعان باید یک نقطه متوازن برای تعریف دامنه پیدا کنند. اگر دامنه خیلی خاص باشد، پس SRA محدود خواهد بود و اعمال آن برای سیستم‌های مشابه دشوار

[۲۵] high-level picture
[۲۶] well-defined domain

خواهد بود. اگر دامنه بیش از حد عمومی باشد که هیچ اطلاعات ارزشمندی را ثبت نکند، مشکلات مشابهی را نیز می‌توان مشاهده کرد.

علاوه بر این، روش ATAM بیشتر به سمت ثبت و دستیابی به اهدافی که به یک سیستم مربوط هستند، متمرکز است. برای توضیح بهتر هدف پشت سر یک SRA می توان اطلاعات بیشتری ارائه کرد.

اصلاحات

برخی موارد اضافی باید برای این مرحله انجام شود زیرا یک SRA به شدت به دامنه تعریف‌شده بستگی دارد. دامنه باید با تیم سهامداران کوچک مورد بحث قرار گیرد. بهتر است براساس دانش معمار و ذینفعان و براساس تجربه قبلی سیستم‌های توسعه‌یافته در آن حوزه مشخص شود. یک راه خوب برای مشخص کردن دامنه، تنظیم ابعاد مختلف یک SRA است.

اصلاح دوم درک بهتر و ارائه اهداف SRA و شناسایی هدف اصلی استفاده از یک فرآیند ارزیابی معماری خاص است. مثال‌هایی برای چنین اهدافی عبارتند از:

۱- ساخت یک سیستم جدید،
۲- گسترش یک سیستم موجود،
۳- ارزیابی ریسک‌های سیستم،
٤- پیش بینی هزینه های تعمیر و نگهداری ،
٥- رسیدن به مناسب‌ترین معماری [LBVB۰۰] [KKC۰۰] [Dol۰۱].

معمار باید هدف را با ذینفعان بحث کند زیرا به تعریف هدف اصلی پشت یک تسهیل‌کننده SRA کمک می‌کند. اهداف بالقوه می‌توانند به شرح زیر باشند:

- بهبود زمان بازاریابی برای سیستم‌های جدید در دامنه SRA.
- طراحی آینده معماری‌های معین را تسهیل می‌کند.
- به اعضای غیر فنی کمک کنید تا دامنه سیستم‌ها را بهتر درک کنند.
- ویژگی‌های کیفیت مشترک مورد نظر را شرح دهید.
- برای جلوگیری از اعتبارسنجی مجدد و ابداع مجدد، الزامات رایج را دریافت کنید.

۳.٤.٤ مرحله ۳: معماری کنونی

خلاصه ATAM

معمار یک معماری سطح بالا را در سطح مناسبی از جزئیات ارائه خواهد داد. سطح جزئیات به عوامل متعددی مانند اطلاعات و زمان در دسترس بستگی دارد. معماری سطح بالا شامل اطلاعاتی مانند تعاملات مختلف سیستم، محدودیت‌های فنی و سناریوهای کاربردی مهم است. این مرحله، مرحله‌ی مهمی است زیرا بر کیفیت تحلیل تاثیر می‌گذارد [KKC۰۰].

بحث

این مرحله اطلاعات معماری موجود فعلی را نشان می‌دهد که بر سناریوها و تحلیل‌های بدست‌آمده در مراحل ارزیابی آینده تاثیر می‌گذارد. با این حال، ارائه یک معماری سطح بالا از یک SRA می‌تواند گیج‌کننده باشد زیرا SRA در حال حاضر در سطح بالاتری از انتزاع در مقایسه با یک معماری واقعی قرار دارد. در نتیجه، اطلاعات بیشتر در مورد سطح مناسبی از جزئیات باید ارائه شود.

آنچه برای SRA ها مهم است الزامات معمول برای یک دامنه خاص است (نه الزاماتی که برای یک سیستم خاص قابل‌اجرا هستند). ATAM توضیح نمی‌دهد که چگونه الزامات سیستم در حال ضبط شدن هستند. سهامداران به احتمال زیاد برای گرفتن الزامات در مورد یک سیستم خاص و نه یک دامنه استفاده می‌شوند. الزامات معمول در سطح بالاتری از انتزاع نسبت به الزامات نرمال سیستم قرار دارند. برای مثال:

- نقاط دسترسی مختلف کاربر (دوردست، محلی، ...) به یک سیستم در یک دامنه
- مدل‌های کسب‌وکار رایج
- یکپارچه‌سازی ابر به سیستم
- کنترل و هدایت دستگاهی بطور خودکار
- یک روش برای ثبت نیازهای مشترک جهت تسهیل فرآیند

اصلاحات

معماری سطح بالا، معمولا یک معماری بسیار دقیق نمی‌باشد. این معماری بیشتر از یک نمودار است که تصویر کلی دامنه را نشان می‌دهد که در مرحله ۲ تعریف شده‌است و باید نشان دهد که SRA در کجا استفاده

خواهد شد و چه اطلاعات سیستم مشترکی را توصیف خواهد کرد. اگر اطلاعات خاصی مانند محدودیت‌های فنی یا تعاملات مختلف سیستم شناخته شوند، پس باید در معماری سطح بالا نشان داده شوند.

همانند الزامات معمول، این مقاله نشان می‌دهد که آن‌ها را به صورت جدولی نشان می‌دهد. ایده این جدول از روش ارزیابی معماری خانواده (FAAM) و تحلیل قابلیت استفاده سطح معماری مبتنی بر سناریو (SALUTA) گرفته شده‌است[FGB٠٤] [Dol٠١]. این جدول شامل نقش‌ها، نیازهای مشترک آن‌ها، و نوع نیازها است. نوع نیاز اصولا به معنی تعیین نکته است که که آیا نیاز مورد بررسی یک نیاز مشترک هسته‌ای و مرکزی است یا یک نیاز اضافی. الزامات اصلی الزاماتی هستند که باید برای هر سیستم در دامنه‌ی مشخص در دسترس باشند و باید از "SRA" استخراج گردند. همانند الزامات اضافی، الزامات اصلی، الزاماتی هستند که می‌توانند به سیستمی که در دامنه مشخص‌شده و در حال توسعه است اضافه شوند و می‌توانند از SRA استخراج شوند و یا نشوند. جزئیات بیشتر در مورد نیازهای مشترک در مرحله ۵ در زمان ثبت سناریوهای هر یک از آن‌ها بررسی خواهد شد. یک مثال از جدول نیاز مشترک در جدول ۲-۱ نشان‌داده شده‌است.

جدول ٤-١: مثال جدول الزامات رایج

Roles	Common Requirements	Requirement Type
Admin, End-user	The device execution can be automated remotely	Core
Admin	System files can be saved and retrieved from the cloud	Additional
End-user	System must provide undo-redo capability	Additional
End-user	Log-in capability	Core

هدف از جدول با FAAM و SALUTA متفاوت است. در ابتدا، معمار از آن برای ارائه نیازهای مشترک به ذینفعان برای شکل دادن طرز فکر خود برای تمرکز بر تفکر در مورد نیازهای مشترک که می‌تواند در دامنه باشد به جای یک نیاز خاص سیستم استفاده خواهد کرد. الزامات معمار ارائه‌شده ممکن است نادرست باشند، اما هدف اصلی آن‌ها عمل کردن به عنوان یک نقطه شروع برای اجازه دادن به ذینفعان برای تفکر در مورد نیازهای مشترک است. جلسات با سهامداران را می توان در جلسات متعدد تکرار کرد تا زمانی که همه نیازهای مشترک در جدول ثبت شوند و سهامداران از نتایج راضی باشند.

مزایای اضافی برای به دست آوردن نیازهای مشترک را می توان در زمانی مشاهده کرد که ذینفعان دارای دانش و تجربه قبلی در سیستم‌هایی هستند که در محدوده دامنه‌ی تعیین‌شده قرار می‌گیرند. اگر ذینفعان هنوز

مشکلاتی برای به دست آوردن نیازهای مشترک دارند، پس معمار می‌تواند برخی از نیازهای خاص سیستم موجود خود را اتخاذ کند و آن‌ها را به سطح بالاتری از انتزاع در طول جلسه منتقل کند تا به ذینفعان اجازه دهد هدف را بهتر درک کنند. علاوه بر این، تنظیم الزامات مشترک با ذینفعان در پاسخ به بخش‌های سوال ۱ کمک می‌کند زیرا به دست آوردن ویژگی‌های کیفی و سناریوهای یک SRA براساس الزامات مشترک است.

٤.٤.٤ مرحله ٤: شناسایی رویکردهای معماری

خلاصه ATAM

معمار، رویکردهای معماری مختلفی که می‌تواند مورد استفاده قرار گیرد را شناسایی خواهد کرد. معمار تنها آن‌ها را شناسایی خواهد کرد و تجزیه و تحلیل نخواهد کرد. در اینجا مهم است که هیچ رویکرد معماری را نادیده نگیرید زیرا آن‌ها بعدا ارزیابی خواهند شد. معماری، ساختار سیستم و چگونگی مقابله با تغییرات و ادغام با سیستم‌های دیگر را تعریف می‌کند. معماری در حال رسیدگی به الزامات و بالاترین اولویت ویژگی‌ها و سناریوهای کیفیت است [KKC۰۰].

بحث

این مرحله نیاز به شناسایی رویکردهای مختلف معماری را برآورده می‌کند که می‌تواند برای رسیدن به SRA دنبال شود. با این حال، الزامات مشترکی که در مرحله ۳ به دست آمده‌اند باید بیشتر برای شناسایی رویکردهای معماری مناسب‌تر به تفصیل شرح داده شوند. ATAM شناسایی رویکردهای معماری قبل از به دست آوردن ویژگی‌های کیفی و سناریوها را پیشنهاد می‌کند.

اصلاحات

این مقاله پیشنهاد می‌کند که این مرحله بعد از مرحله ۵ اجرا شود زیرا ایجاد یک درخت کاربردی ویژگی کیفیت به اصلاح بهتر و جزئیات الزامات مشترکی که SRA باید ارائه دهد کمک می‌کند، که منجر به رویکردهای معماری بهتر می‌شود.

٥.٤.٤ مرحله ٥: درخت کاربردی ویژگی کیفیت را تولید کنید.

خلاصه ATAM

این مرحله در مورد شناسایی ویژگی‌های کیفی و سناریوها برای تولید یک درخت مطلوبیت ویژگی کیفیت[۲۷] است. ویژگی‌های کیفی و سناریوها در بخش‌های قبل تعریف شده‌اند. درخت کاربردی یک نمودار درختی است که ویژگی‌های کیفی را به جزئیات ظریف تقسیم می‌کند تا زمانی که به یک سناریو برسد که معماری باید آن را پوشش دهد[KKC۰۰]. این سناریوها با توجه به اهمیت آن‌ها و سهولت دستیابی به آن‌ها اولویت‌بندی می‌شوند[KKC۰۰]. درخت سودمندی برای راهنمایی تیم در طول تجزیه و تحلیل زمانی که خطرات، معاملات و نقاط حساسیت باید ثبت شوند، مورد استفاده قرار می‌گیرد[KKC۰۰]. فهرستی از اکثر ویژگی‌های کیفی مورد استفاده در استاندارد ISO / IEC ۲۵۰۱۰[ISO۱۱] و در مقاله‌ی آروانیتوو و همکاران[۲۸] موجود است[AAC+۱۷]. از این لیست، ایده‌هایی در مورد اینکه چه ویژگی‌های کیفیتی باید در نظر گرفته شوند، استخراج می‌شود. برخی از ویژگی‌های کیفیت استاندارد عبارتند از:

۱- قابلیت نگهداری،
۲- قابلیت اصلاح،
۳- پایداری،
٤- قابلیت تحلیل،
۵- قابلیت تغییر،
٦- قابلیت استفاده مجدد[۲۹]،
۷- قابلیت درک [ISO۱۱].

بحث

همانطور که قبلا ذکر شد، شناسایی سناریوهای مشترک عمومی برای یک دامنه دشوار است. آنجلو و همکاران[۳۰] [ATG۱٤] با شناسایی زمینه‌های مختلف که SRA میتواند، راه حلی را برای این مشکل پیشنهاد می‌کنند که در آن یک درخت منفعت[۳۱] برای هر زمینه از ذینفعان مختلف در چندین سازمان به دست می‌آید [ATG۱٤]. در نهایت، درخت‌های منفعت با هم ادغام می‌شوند تا ویژگی‌های کیفیت مشترک و سناریوها را برای SRA به دست آورند[ATG۱٤]. این رویکرد وقت گیر است و باعث می‌شود تعداد ذینفعان زیاد شود، همچنین، این رویکرد فرض می‌کند که SRA برای چندین سازمان به کار گرفته خواهد شد، ولی همیشه این گونه نیست.

[۲۷] quality attribute utility tree
[۲۸] Arvanitou et al.
[۲۹] reusability
[۳۰] Angelov et al.
[۳۱] utility tree

علاوه بر این، ATAM پیشنهاد می‌کند که معمار ابتدا درخت منفعت را تولید می‌کند. سپس این سناریوها در مرحله ۷ با سایر ذینفعان برای به دست آوردن و اولویت‌بندی سناریوهای بیشتر در یک فرآیند طوفان فکری[۳۲] قرار می‌گیرند. سپس درخت منفعت از تمام منابع مختلف به روز رسانی می‌شود[KKC۰۰]. برای ارزیابی SRAها، ممکن است معمار با مشکلاتی در ایجاد یک درخت کاربردی به دلیل طیف گسترده‌ای از زمینه‌ها که یک دامنه می‌تواند پوشش دهد، مواجه گردد. از سوی دیگر، متخصصان حوزه، دیدگاه روشن تری در مورد آنچه که باید در این حوزه گنجانده شود، دارند.

اصلاحات

تجزیه و تحلیل معماری نرم‌افزار برای تکامل و قابلیت استفاده مجدد (SAAMER)، سناریوهایی را براساس اهدافی که از سهامداران گرفته شده‌اند، تولید می‌کند[LBK۹۷]. سناریوها تا زمانی ایجاد می‌شوند که متخصصان دامنه و سهامداران متقاعد شوند که سناریوها به خوبی اهداف را پوشش می‌دهند[LBK۹۷]. یک روش مشابه در این مقاله برای به دست آوردن ویژگی‌ها و سناریوهای کیفیت مشترک براساس نیازهای مشترک شناسایی‌شده در مرحله ۳ دنبال می‌شود. علاوه بر این، این مقاله دو رویکرد زیر را پیشنهاد می‌کند که در تجزیه و تحلیل تغییر سطح معماری (ALMA) برای به دست آوردن سناریوهای مشترک توصیف شده‌اند. آن‌ها به این دلیل استفاده می‌شوند که به ذینفعان کمک می‌کنند تا درخت منفعت را از دو چشم‌انداز ممکن تولید کنند[LBVB۰۰]:

۱- یک رویکرد بالا به پایین [۳۳]
۲- یک رویکرد پایین به بالا[۳۴]

رویکرد بالا به پایین

ابتدا ویژگی‌های کیفی (یا زیر مجموعه‌ای از ویژگی‌های کیفی) را یادداشت می‌کند و سپس سناریوها را، با سهامداران، که مربوط به ویژگی‌های کیفی هستند، شناسایی می‌کند[LBVB۰۰].

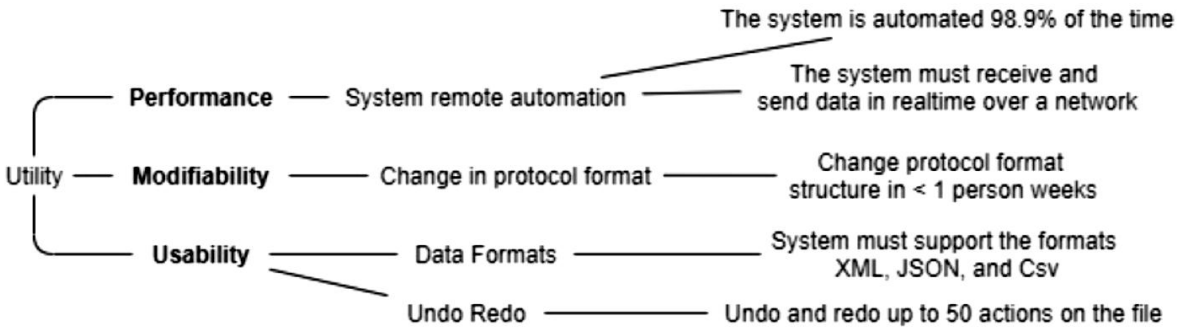
رویکرد پایین به بالا

این روش برعکس رویکرد بالا به پایین می‌باشد. در ابتدا به ذینفعان اجازه می‌دهد تا سناریوها را ذکر کنند و سپس سناریوها را به ویژگی کیفی مربوطه خود ترسیم کنند[LBVB۰۰]. اگر یک سناریو را بتوان به بیش از یک

[۳۲] brainstormed
[۳۳] top-down
[۳۴] bottom-up approach

ویژگی کیفی نگاشت کرد، پس سناریو باید بر این اساس تقسیم شود به طوری که تنها با یک ویژگی کیفی متناسب باشد.

رویکردهای بالا به پایین و پایین به بالا با یکدیگر ادغام می‌شوند. مشابه مرحله ۳، گرفتن ویژگی‌های کیفیت مشترک و سناریوها می‌تواند در چندین جلسه تا زمانی که ذینفعان با درخت منفعت راضی شوند، انجام شود. نمونه‌ای از یک درخت منفعت کوتاه در شکل ۲-۱ نشان‌داده شده‌است. این درخت را می توان با درخت منفعت کوتاه مقایسه کرد. جدول ۲-۱ برای مشاهده اینکه چگونه الزامات معمول برای اتوماسیون و اجرای پروتکل بیشتر در درخت صنایع تعریف می‌شود.



شکل ٤-١: یک مثال درخت بهره‌وری مختصر

درخت ابزار ATAM در راستای واقعی کردن سناریوها عمل می‌کند[KKC۰۰]. با این حال، به دست آوردن یک مقدار واقعی برای سناریوهای معمول می‌تواند دشوار باشد. سهامداران باید نهایت تلاش خود را برای اصلاح سناریوها تا زمانی که به یک ارزش واقعی برسند، انجام دهند. با دنبال کردن مرحله ۳ و این مرحله، سوال ۱ پاسخ داده می‌شود زیرا آن‌ها ساختاری را برای دنبال کردن الزامات، ویژگی‌های کیفی، و سناریوهای یک SRA ارائه می‌دهند.

پس از ساخت درخت منفعت، سناریوها (گره‌های برگ) توسط همه ذینفعان وزن دهی می‌شوند. سناریوهای وزن‌دهی کننده ویژگی‌های کیفی مهمی را به معمار خواهند گفت که بعدا در طول مرحله تجزیه و تحلیل (مرحله ۵) برای به دست آوردن امتیاز هر معماری مورد استفاده قرار خواهند گرفت. آن‌ها براساس اهمیت وزن گذاری می‌شوند. از آنجا که اولویت‌بندی را می توان به طور متفاوت بین ذینفعان درک کرد و گاهی اولویت‌بندی

برخی از سناریوها نسبت به سایر سناریوها دشوار است [KKC۰۰] بنابراین، اولویت‌بندی براساس سه سطح است: بالا، متوسط، و پایین.

٦.٤.٤ مرحله ٦: تجزیه و تحلیل روش‌های معماری

خلاصه ATAM

این مرحله در مورد کسب اطلاعات کافی در مورد رویکردهای مختلف معماری برای ارزیابی آن‌ها است. خروجی این مرحله ارائه دلیل در مورد تصمیمات و کمک به رسیدن به الزامات و ویژگی‌های کیفی سیستم نهایی است. ابتدا، یک سناریو از درخت مطلوبیت برای شناسایی ریسک‌ها، حساسیت‌ها و موازنه برای هر تصمیم معماری در نظر گرفته می‌شود. این سناریوها با پرسیدن سوالاتی در مورد چگونگی ارتباط رویکردها با ویژگی‌های کیفی شناسایی می‌شوند. این سوالات براساس تجربه تیم نرم‌افزار، مطالب دانشگاهی نرم‌افزار و تجربه مستند می‌شوند. سپس نقاط حساسیت و موازنه بدست‌آمده به دو دسته ریسک و غیر ریسک تقسیم می‌شوند. در پایان، تیم باید فهرستی از ریسک‌ها و ایده‌ای در مورد جنبه‌های مهم معماری داشته باشد و اینکه کدام رویکردها باید بر سایر رویکردها غلبه کنند [KKC۰۰]

بحث

رویکرد تحلیل ATAM یک روش مناسب برای شناسایی تفاوت‌های بین رویکردهای معماری است. با این حال، ATAM بر تحلیل یک معماری سیستم واحد در یک زمان تمرکز می‌کند[BRST۰۵]. کل معماری مشخص می‌کند که چگونه اجزای معماری مختلف می‌توانند بر یکدیگر تاثیر بگذارند. نمونه‌هایی از روش‌های مقایسه معماری عبارتند از ALMA، SALUTA و مدل مقایسه معماری نرم‌افزار خاص دامنه[۳۵] (DoSAM). ALMA نیاز به لیست کردن اجزای تحت‌تاثیر قرار گرفته برای ارزیابی قابلیت اصلاح معماری دارد[LBVB۰۰]. SALUTA و DoSAM پیشنهاد می‌کنند که یک امتیاز برای هر معماری براساس میزان تحقق ویژگی‌های کیفی آن‌ها در صورتی که هدف مقایسه نامزدهای معماری مختلف باشد، داده شود [BRST۰۵] [FGB۰٤].

علاوه بر این، سطح بالایی از انتزاع برای ارزیابی SRA در نظر گرفته می‌شود. بنابراین، برخی از ویژگی‌های کیفیت با احتمال کم‌تری مهم هستند، یا تجزیه و تحلیل آن‌ها برای یک SRA دشوار است. به عنوان مثال، قابلیت استفاده و ویژگی‌های عملکردی کیفیت به جای خود معماری، به پیاده‌سازی منسجم بستگی دارد.

اصلاحات

[۳۵] Domain-Specific Software Architecture Comparison Model

یک روش نمره‌دهی وزنی برای مقایسه SRA های مختلف و رویکردهای معماری که در ابتدا در مرحله ٤ شناسایی شدند، معرفی شده‌است. این مرحله به هفت مرحله فرعی تقسیم می‌شود:

- مرحله تجزیه و تحلیل۱: در این مرحله، مصالحه، ریسک‌ها و نقاط حساسیت برای هر رویکرد معماری که سناریوهای متناظر با آن را پشتیبانی می‌کند، به دست می‌آید. این کار با پیروی از روش ATAM انجام می‌شود. رویکردهای معماری در ابتدا در مرحله ٤ شناسایی شدند. علاوه بر این، باید به مولفه‌های معماری که یا تحقق می‌یابند و یا باعث ایجاد ریسک برای سناریوها می‌شوند، توجه کرد[LBVB۰۰]. این امر به شناسایی مولفه‌هایی که نیاز به تجزیه و تحلیل بیشتر در هنگام طراحی SRAها در مرحله تجزیه و تحلیل دارند، کمک خواهد کرد.
- مرحله تجزیه و تحلیل۲: براساس موازنه، ریسک‌ها و نقاط حساسیت به‌دست‌آمده، معمار امتیازی از ۱ تا ۵ را در مورد میزان پشتیبانی هر رویکرد معماری از سناریوهای متناظر خود ارائه می‌دهد[FGB۰٤]. در صورتی که رویکرد معماری از سناریو پشتیبانی کند اما نقاط تعادل، ریسک و حساسیت بالایی داشته باشد، در این صورت امتیاز پایین است و بالعکس. امتیاز ۱ تا ۵ برای سادگی انتخاب شده است. ایده سناریوهای امتیاز دهی از SALUTA گرفته شده‌است[FGB۰٤].SALUTA امتیازی را برای هر سناریو در مورد میزان پشتیبانی معماری یا سیستم تعیین می‌کند. این روش امتیازدهی ساختاری برای پاسخ به سوال ۲ در مورد چگونگی ارزیابی ویژگی‌ها و سناریوهای کیفی SRA می‌دهد. این روش عمدتا به تجربه و دانش معمار در شناسایی مصالحه، ریسک و نقاط حساسیت و امتیازدهی آن‌ها بستگی دارد.
- مرحله تجزیه و تحلیل۳: هنگامی که امتیاز هر رویکرد معماری از تعیین سناریوهای متناظر پشتیبانی می‌کند، امتیاز باید در اولویت سناریو ضرب شود تا وزن به دست آید. رویکرد معماری با بالاترین وزن برای ساخت SRA های مختلف براساس الگوهای معماری مختلف انتخاب خواهد شد. به عنوان مثال، یک SRA می‌تواند با استفاده از یک الگوی معماری سرور–مشتری[۳٦] طراحی شود در حالی که یک SRA دیگر می‌تواند با استفاده از یک الگوی معماری خوشه–سرور[۳۷] طراحی شود. بالاترین رویکردهای معماری وزن دار باید در SRA ها گنجانده شوند. به عنوان مثال، نوع پایگاه‌داده می‌تواند یک رویکرد معماری باشد که در SRAها نشان داده خواهد شد.
- مرحله تجزیه و تحلیل ٤: پس از ساخت SRA های مختلف، معمار از ۱ تا ۵ امتیاز می‌گیرد که چگونه هر SRA کلی هر سناریو را برآورده می‌کند. کل معماری هنگام امتیاز دهی در نظر گرفته می‌شود زیرا

[۳٦] client server
[۳۷] server cluster

قابلیت اندازه‌گیری برخی سناریوها به چگونگی عملکرد چندین مولفه و تعامل با یکدیگر بستگی دارد. مشابه با مرحله تجزیه و تحلیل ۳، زمانی که امتیاز هر SRA برای هر سناریو تنظیم می‌شود، امتیاز باید با اولویت سناریو ضرب شود تا وزن به دست آید. سپس تمام وزن سناریوها که تحت همان ویژگی کیفی هستند با هم جمع می‌شوند تا امتیاز وزنی نهایی معماری را با توجه به هر ویژگی کیفی بدست آورند.

- مرحله تجزیه و تحلیل ٥: این مرحله تلاش می‌کند تا به SRA اصلاح‌شده براساس امتیازات وزنی ویژگی‌های کیفی به‌دست‌آمده دست یابد. برای هر ویژگی کیفیت، مفاهیم و مولفه‌های SRA با بالاترین امتیاز وزنی باید گرفته شود. معمار باید سعی در ترکیب، حذف، اضافه کردن یا ترکیب اجزا و تصمیمات معماری برای دستیابی به SRA های ممکن جدید داشته باشد.
- مرحله تجزیه و تحلیل ٦: مرحله تجزیه و تحلیل تکرار مرحله ٤ اما برای SRA جدید برای تشخیص اینکه آیا امتیاز وزنی بهتری دریافت خواهند کرد یا خیر. این مرحله را می‌توان چندین بار تکرار کرد تا زمانی که هیچ SRA دیگری را نمی‌توان طراحی کرد و یا اگر SRA جدید امتیاز وزنی کمتری نسبت به SRA قبلی داشته باشد.
- مرحله تجزیه و تحلیل ۷: SRA با بالاترین امتیاز وزنی انتخاب می‌شود.

این رویکرد چارچوبی ارائه می‌دهد که می‌تواند برای پاسخ به سوال ٤ دنبال شود. این روش امتیاز دهی وزنی را فراهم می‌کند که امکان مقایسه SRAها و ارائه دلیل برای انتخاب یک SRA نسبت به گزینه‌های دیگر را فراهم می‌کند.

۷.٤.٤ مرحله ۷: طوفان مغزی و اولویت‌بندی سناریوها

خلاصه ATAM

این مرحله زمانی است که سهامداران برای تفکر و شناسایی سناریوها گرد هم می‌آیند. هیچ ایده یا نظری در این مرحله نادیده گرفته نمی‌شود. سناریوهای شناسایی‌شده عبارتند از:

۱- سناریوهای مورد استفاده،
۲- رشد
۳- سناریوهای اکتشافی.

سناریوهای مورد استفاده سناریوهایی هستند که در آن‌ها کاربر نهایی با سیستم تعامل خواهد داشت. سناریوهای رشد تغییراتی هستند که ممکن است برای معماری یا سیستم رخ دهند. سناریوهای اکتشافی سناریوهای غیر معمولی هستند که برای اعمال محدودیت بر سیستم به کار می‌روند. آن‌ها می‌توانند به شکل تغییرات عمده در سیستم یا بارهای بالا و تست‌های تنش غیر معمول باشند. پس از شناسایی سناریوها، ذینفعان آن‌ها را اولویت‌بندی می‌کنند. سپس سناریوهای اولویت‌بندی شده با نمودار مطلوبیت مقایسه می‌شوند تا ببینند آیا با هم مطابقت دارند یا خیر. هر گونه اختلاف باید آشتی کند و توضیح داده شود. در نهایت، درخت منفعت اولیه از مرحله ۵ و سناریوهای شناسایی‌شده در این مرحله با هم ادغام می‌شوند تا یک درخت منفعت واحد به دست آید[KKC۰۰].

بحث

ATAM برای تولید یک درخت منفعت در مرحله ۵ به *معمار* وابسته است. پس از آن، سناریوهای شناسایی‌شده از ذینفعان در این مرحله در درخت منفعت ادغام می‌شوند[KKC۰۰]. برای یک SRA، یک معمار می‌تواند در شناسایی اکثر سناریوهای یک دامنه خاص مشکل پیدا کند. علاوه بر این، سهامداران مختلف می‌توانند سناریوهای مختلفی برای یک دامنه یک‌سان داشته باشند.

اصلاحات

این مرحله با مرحله ۵ ادغام شد تا از تجزیه و تحلیل ناخواسته برای رویکردهای معماری جلوگیری شود زیرا همه سناریوها هنوز استخراج نشده اند. برای یک SRA بهتر است قبل از شناسایی رویکردهای معماری مختلف، درخت مطلوبیت با سهامداران را شناسایی کنیم.

۸.٤.٤ مرحله ۸: رویکردهای معماری مبتنی بر تجزیه و تحلیل

خلاصه ATAM

اگر سناریوهای جدیدی در مرحله ۷ شناسایی شده باشد، این مرحله، مرحله ٦ را تکرار می‌کند. اگر هیچ تغییری در درخت منفعت در مرحله ۷ ایجاد نشود، این مرحله یک فعالیت آزمایشی برای کشف هر گونه اطلاعات جدید است. اگر اطلاعات جدید کشف شود، تیم باید به مرحله ٤ بازگردد[KKC۰۰].

بحث

این همان گام مرحله ٦ است. با توجه به ATAM، تمام مراحل را می‌توان به صورت تکراری و به ترتیب مختلف انجام داد.

اصلاحات

اصلاحات این مرحله قبلا در مرحله ٦ توضیح داده شده‌اند.

۹.٤.٤ مرحله ۹: ارائه نتایج

خلاصه ATAM

گام نهایی مستند سازی نتایج به منظور ارائه دلایلی در مورد اینکه چرا یک رویکرد معماری خاص نسبت به دیگران انتخاب شده‌است، می‌باشد. ATAM پیشنهاد می‌کند که همه چیز را به ذینفعان ارائه دهد اما مهم‌ترین بخشها، خروجی‌های ATAM مانند درخت سودمند و تحلیل‌های معماری هستند[KKC٠٠].

بحث

مستند سازی باید براساس هدف تجزیه و تحلیل و الزامات تعیین‌شده صورت گیرد[FGB٠٤]. علاوه بر این، اگر سیستم‌های قبلی در دامنه وجود داشته باشند، مستندسازی این که چگونه معماری منسجم می‌تواند با SRA سازگار شود نیز مفید خواهد بود.

اصلاحات

هیچ تغییری در این مرحله ایجاد نشده است.

۱۰.٤.٤ مرحله ۱۰: خلاصه

اولین قدم ارائه فرآیند ارزیابی به تیم کوچک اولیه سهامداران است. این مرحله با روشی که ATAM بدون هیچ تغییری پیشنهاد کرده‌است، دنبال خواهد شد.

مرحله دوم ارائه محرک‌های کسب‌وکار به تیم کوچک اولیه سهامداران و شناسایی سهامداران اصلی است که در مراحل بعدی مشارکت خواهند داشت. در این مرحله، ATAM مشخص نمی‌کند که یک دامنه باید تعریف شود و بر دستیابی به اهداف سیستم تمرکز کند. بنابراین، دو اصلاح در این مرحله انجام شد. اولین اصلاح، تعریف

دامنه با پاسخ به سوالات است که در بخش ۳ ارائه شدند. دومین اصلاح، درک بهتر اهداف SRA با لیست کردن برخی از مزایای SRA ها بود. نکته قابل‌توجه این است که طراحی آینده معماری‌های منسجم برای سیستم‌های چندگانه درون دامنه تسهیل شود.

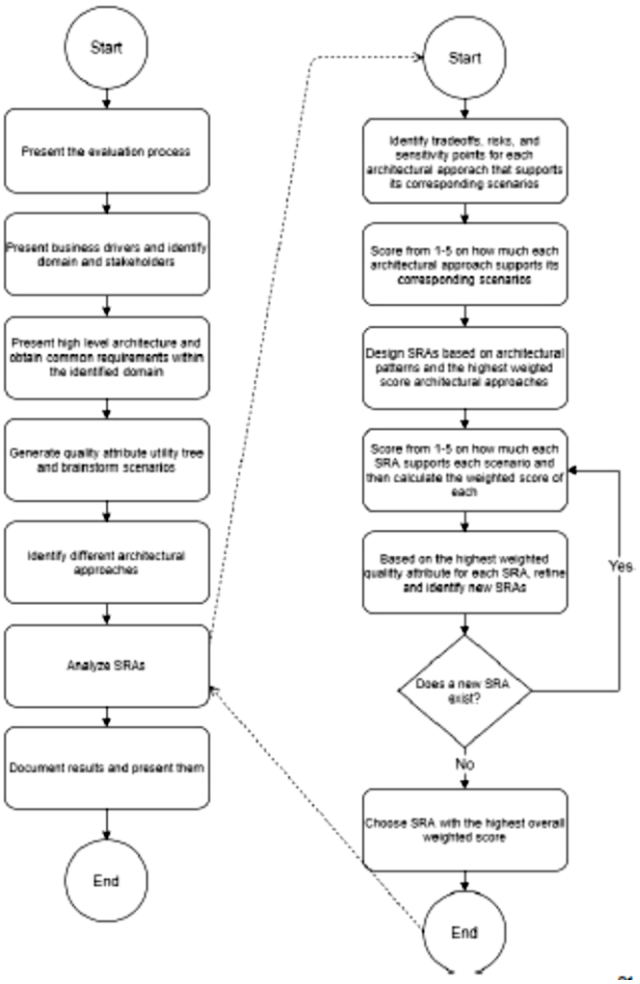
مرحله سوم ارائه معماری سطح بالا به ذینفعان اصلی است. برای SRAها، جزئیات در مورد چگونگی تعریف یک معماری سطح بالا مورد نیاز است، زیرا یک SRA در حال حاضر در سطح بالاتری از انتزاع در مقایسه با معماری واقعی است. بنابراین، معماری سطح بالا برای یک SRA یک نمودار غیر دقیق خواهد بود که دامنه تعریف‌شده را نشان می‌دهد. علاوه بر این، ATAM راهی برای ثبت نیازهای مشترک فراهم نمی‌کند. به این ترتیب، یک رسم جدولی پیشنهاد شد. این جدول در ابتدا توسط معمار برای ارائه یک مجموعه اولیه از نیازهای مشترک برای شکل دادن طرز فکر ذینفعان به سمت تمرکز بر روی دامنه برای دستیابی به نیازهای مشترک استفاده خواهد شد.

ATAM پیشنهاد می‌کند که رویکردهای معماری در مرحله چهارم شناسایی شوند و یک درخت منفعت را در مرحله پنجم تولید کنند. این مقاله اجرای آن‌ها را به صورت معکوس پیشنهاد می‌کند. ابتدا درخت منفعت ساخته خواهد شد و سپس رویکردهای معماری مختلف شناسایی خواهند شد. این امر به اصلاح بهتر الزامات مشترک جذب‌شده کمک می‌کند که منجر به معماری اولیه بهتر می‌شود. همچنین، جزئیات بیشتر در مورد چگونگی به دست آوردن سناریوهای مشترک و ویژگی‌های کیفی برای SRA ها براساس رویکرد ALMA در شناسایی سناریوها ارائه شد.

مرحله ششم تجزیه و تحلیل SRA های مختلف با توجه به درخت منفعت است. یک روش تجزیه و تحلیل اصلاح‌شده از ATAM معرفی شد زیرا ATAM بر تجزیه و تحلیل یک معماری سیستم واحد در یک زمان تمرکز می‌کند. روش تجزیه و تحلیل شامل هفت مرحله است. اولین مرحله تجزیه و تحلیل، شناسایی مصالحه، نقاط حساسیت و ریسک‌ها برای هر رویکرد معماری است. مرحله تجزیه و تحلیل دوم، امتیاز دهی به رویکردهای معماری در مورد میزان پشتیبانی آن‌ها از سناریوهای متناظر براساس مصالحه، ریسک‌ها و نقاط حساسیت شناسایی‌شده، می‌باشد. مرحله سوم تجزیه و تحلیل، بالاترین رویکردهای معماری وزن دار را برای طراحی SRA های مختلف براساس الگوهای معماری مختلف دریافت می‌کند. در مرحله تجزیه و تحلیل چهارم، معمار میزان رضایت هر SRA از هر سناریو را برای به دست آوردن امتیاز وزن دهی شده کلی SRA امتیاز می‌دهد. مرحله پنجم تجزیه و تحلیل تلاش می‌کند تا به SRA اصلاح‌شده براساس امتیاز وزنی ویژگی‌های

کیفیت به‌دست‌آمده با ترکیب، اصلاح، اضافه کردن یا حذف اجزا از SRA های مختلف دست یابد. مرحله ششم تکرار تجزیه و تحلیل مرحله چهارم اما برای SRA اصلاح‌شده است. در نهایت، در مرحله تجزیه و تحلیل هفتم، SRA با بالاترین امتیاز وزنی انتخاب خواهد شد.

مرحله هفتم و نهایی، فرآیند ارزیابی مستند سازی نتایج و ارائه آن‌ها به سهامداران است. این مرحله مشابه با آنچه ATAM پیشنهاد می‌کند، انجام می‌شود.



شکل ٤-٢: فلوچارت مراحل ارزیابی

۵ معماری مرجع نرم افزار برای دستگاه های آزمایشگاهی

در این فصل، این مقاله مثالی را ارائه می‌دهد که در آن روش ارزیابی برای به دست آوردن یک SRA برای دستگاه‌های آزمایشگاهی در دستگاه‌های کنترل مایع خاص اعمال شده‌است. این فصل نشان می‌دهد که سهامداران چگونه هر روش ارزیابی و نتایج هر کدام را انجام می‌دهند. همچنین چالش‌ها یا عدم قطعیت‌های مواجه شده، و تغییرات اندکی در ارزیابی ارائه‌شده توسط سهامداران یا معمار هنگام اعمال مراحل ارزیابی ایجاد می‌کند.

به طور کلی، اثبات شد که روش ارزیابی پیشنهادی برای SRA مفید است زیرا معمار قادر به رسیدن به یک SRA برای دستگاه‌های حمل و نقل مایع براساس نیازها و الزامات ذینفعان بود. تغییرات فضایی در برخی از مراحل ارزیابی برای تسهیل فرآیند انجام شد، اما این ممکن است برای سایر ذینفعان و معماران که از روش ارزیابی استفاده می‌کنند متفاوت باشد.

۱.۵ معرفی فرآیند ارزیابی

۱.۱.۵ نتایج مرحله ارزیابی

معماری نرم‌افزار و مشکلاتی که در بخش قبل ذکر شد به تیم اولیه سهامداران ارائه می‌شود. سپس هدف اصلی طراحی یک تسهیل‌کننده SRA همراه با مزایای آن برای تسهیل توسعه دستگاه‌های حمل و نقل مایع ارائه می‌شود. بعد، فرآیند ارزیابی به طور خلاصه توضیح داده شد به طوری که هر ذی‌نفع مسئولیت‌های خود را برای مراحل بعدی بداند. در نهایت، سوالات سهامداران پاسخ داده می‌شوند.

۲.۱.۵ نظرات در مورد مرحله ارزیابی

معمار و تیم کوچک اولیه سهامداران هیچ مشکلی در این مرحله نداشتند.

۲.۵ بحث در مورد محرک‌های کسب‌وکار

در این مرحله سهامداران اولیه دامنه را تعریف کرده و سهامداران ثانویه را شناسایی می‌کنند که می‌توانند در طی فرآیند برای مراحل زیر مشارکت داشته باشند.

۱.۲.۵ نتایج دامنه

این تیم دامنه‌ی زیر را براساس سوالات مربوط به دامنه ذکر شده در انتهای بخش قبل تعریف کرده‌است.

ابعاد طبقه بندی SRA

- از SAR در کجا می‌توان استفاده کرد؟ SAR به عنوان یک تسهیل‌کننده برای رسیدن به معماری واقعی در سازمان‌هایی که نرم‌افزار را برای دستگاه‌های آزمایشگاهی توسعه می‌دهند، استفاده خواهد شد. از آنجا که طیف گسترده‌ای از دستگاه‌های آزمایشگاهی وجود دارد، دامنه دستگاه‌های آزمایشگاهی به دستگاه‌های حمل و نقل مایع برای داشتن یک SRA دقیق‌تر محدود می‌شود. نتیجه‌ی دستگاه‌های جابجایی مایع می‌تواند به شکل فیزیکی باشد، برای مثال مخلوط کردن مایعات، یا خروجی داده‌ها (برای مثال اندازه‌گیری حجم‌های مایع). پروتکل دستگاه آزمایشگاهی مجموعه‌ای از دستورالعمل‌ها است که توسط کاربر تنظیم شده‌است. این شامل گام‌هایی است که دستگاه باید برای رسیدن به نتیجه دنبال کند. پروتکل‌های دستگاه آزمایشگاهی معمولا با استفاده از یک برنامه نرم‌افزاری که بر روی دستگاه مستقر شده‌است، ایجاد و اجرا می‌شوند. یک مثال از یک دستگاه آزمایشگاهی در این حوزه، I.DOT One است که در شکل ۳.۲ نشان‌داده شده‌است. I.DOT One وسیله‌ای است که برای از بین بردن مایعات از یک آزمایشگاه استفاده می‌شود. دستورالعمل‌های پروتکل این دستگاه، مراحل توزیع مایع هستند. برنامه‌ای که برای ایجاد و اجرای پروتکل‌ها مورد استفاده قرار می‌گیرد، بر روی لوحی مستقر شده‌است که همانطور که در شکل ۵-۱ نشان‌داده شده‌است، بر روی دستگاه نصب شده‌است.
- سهامداران همچنین محدوده قیمت دستگاه را مشخص کردند.
- چه کسی SAR را تعریف می‌کند؟ SRA برای یک سازمان واحد در نظر گرفته شده‌است.
- SAR چه زمانی تعریف می‌شود؟ از آنجا که سازمان در حال حاضر سیستم‌هایی دارد که در آن دامنه ساخته شده‌اند، SRA پس از توسعه معماری نرم‌افزار واقعی تعریف خواهد شد.
- مرحله بلوغ دامنه در SAR چیست؟ دامنه کاملا بالغ است اما ایده‌های جدید در مورد پروتکل‌های ارتباطی دستگاه آزمایشگاهی، استانداردها، اتصال ابری و اتوماسیون هنوز در حال بحث هستند و به طور متفاوت بین سازمان‌ها اجرا می‌شوند.

❖ شکل ۵-۱ : افزاره آزمایشگاهی I.DOT One

ابعاد طراحی SRA

- SAR چه چیزی را توصیف می‌کند؟ SRA باید عناصر سیستم و روابط بین آن‌ها را توصیف کند. همچنین شامل ویژگی‌های سیستم و پروتکل‌های ارتباطی باشد که در صورت وجود باید دنبال شوند.
- SAR چگونه نمایش داده می‌شود؟ SRA به شیوه‌ای نیمه رسمی نشان داده خواهد شد.
- SAR چگونه توصیف شده‌است؟ SRA توسط ترکیبی از نشانه‌های گرافیکی و بافتی[۳۸] توصیف خواهد شد. نماد گرافیکی، زبان مدل‌سازی یکپارچه[۳۹] (UML) مانند نمادی است در این مورد که برای نشان دادن مولفه‌های کلیدی سیستم استفاده خواهد شد. نماد بافتی[۴۰] جزئیات نهایی در مورد مولفه‌های گرافیکی را توضیح خواهد داد.

۲.۲.۵ نظرات در مورد دامنه

ذکر این نکته الزامی است که ATAM فقط محرک‌های کسب‌وکار سیستم را ارائه می‌دهد و دامنه را مشخص نمی‌کند. در حالی که در این مورد، سهامداران هیچ مشکلی در شناسایی دامنه نداشتند و پی بردن به سوالات که متعاقبا پیش می‌آید، آسانتر می‌شود.

[۳۸] textural
[۳۹] Unified Modeling Language
[۴۰] Textural Notation

۳.۲.۵ نتایج سهامداران

از آنجا که دامنه برای یک سازمان واحد تعریف شده‌است، ذینفعان در یک سازمان مشخص شده‌اند. سهامداران بخش‌های مختلف برای استخراج دیدگاه‌های مختلف در مورد دستگاه‌های حمل و نقل مایع انتخاب شدند که شامل مهندسین سخت‌افزار، توسعه دهندگان نرم‌افزار، صاحبان محصول و دانشمندان نرم‌افزار بودند.

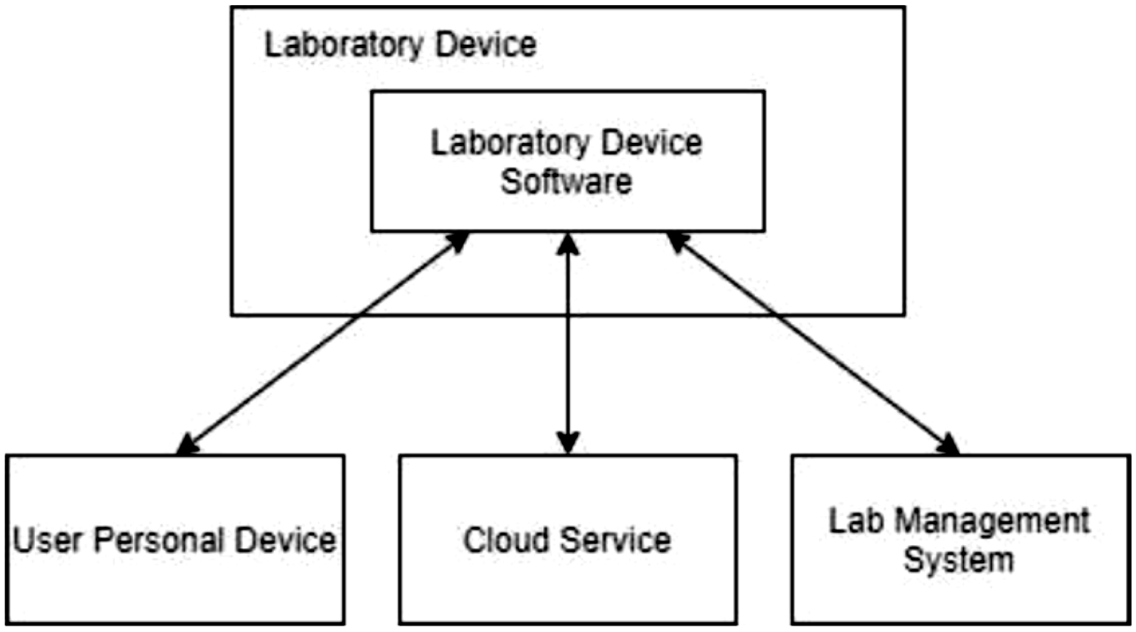
٤.۲.۵ نظرات در مورد سهامداران

شناسایی ذینفعان ساده بود زیرا سازمان مورد بررسی، یک سازمان با مقیاس کوچک تا متوسط است. ارتباطات با استفاده از پست الکترونیکی به دلیل دشواری حضور همه سهامداران برای یک جلسه به دلیل زمان‌بندی‌های مختلف، مورد استفاده قرار گرفت.

۳.۵ شناسایی الزامات مشترک

۱.۳.۵ نتایج مرحله ارزیابی

شکل ۵-۲ نمودار سطح بالایی را نشان می‌دهد که اجزای سیستم خارجی را نشان می‌دهد که در حال تعامل با نرم‌افزار دستگاه آزمایشگاهی هستند. در این دستگاه، از طریق نرم‌افزاری که بر روی دستگاه مستقر شده‌است، دستگاه‌های شخصی کاربر خارجی، سرویس ابری برای ذخیره و به اشتراک گذاری داده‌های دستگاه، سیستم مدیریت آزمایشگاهی برای اتوماسیون قابل دسترسی است.



شکل ۵-۲: نمودار سطح بالا

الزامات مشترک براساس دانش سهامداران در مورد دامنه و مستندات قبلی برای برخی از بخش‌های دستگاه‌های حمل و نقل مایع بدست آمد. اول، نیازهای مشترک توسط معمار شناسایی شدند و سپس با ذینفعان به اشتراک گذاشته شدند. هر ذی‌نفع با اصلاح، پیشنهاد، و اضافه کردن الزامات براساس دانش خود به تکامل سیستم کمک می‌کند.

نظرات در مورد مرحله ارزیابی

سهامداران قادر به ارائه الزامات مشترک ارزشمند برای دامنه به جای یک سیستم خاص بودند. چیزی که به آن‌ها کمک کرد تجربه قبلی آن‌ها در دامنه، نمودار سطح بالای معماری در شکل ۲.۵، و الزامات مشترک اولیه ارائه‌شده توسط معمار است. برخلاف ATAM، این امر به ذینفعان یک ساختار برای شروع به منظور شناسایی نیازهای مشترک می‌دهد.

انتخاب ذینفعان از نقش‌های مختلف نیز مزیت خود را در به دست آوردن تمام جنبه‌های عملکردی مختلف نشان می‌دهد. برخی از سهامداران نمی‌توانند به برخی از الزامات سیستم کمک کنند زیرا نقش یا دانش مرتبط در آن زمینه را ندارند. بنابراین، سهامداران همچنین پیشنهاد می‌کنند که این نکته را نیز تعیین نمایند که آیا یک نیاز

خاص به یک گروه خاص از ذینفعان مرتبط است یا خیر. این امر تضمین می‌کند که تیم نرم‌افزار یا معمار در صورت نیاز به شفاف‌سازی بیشتر در مورد یک نیاز، از گروه ذینفعان مناسب سوال خواهد کند.

براساس سناریوها و الزامات مشترک شناسایی‌شده، معمار رویکردهای معماری مختلفی را پیشنهاد کرد. در این مرحله هیچ رویکردی نادیده گرفته نشد و هیچ یک از رویکردها تحلیل نشدند. علاوه بر این، معمار کشف کرد که شناسایی رویکردهای معماری از طریق خوشه‌بندی سناریوها و الزامات براساس این که آیا خوشه تحت‌تاثیر نوع رویکرد مشابه قرار می‌گیرد یا خیر، آسان‌تر است. به عنوان مثال، سناریوها و الزاماتی که مربوط به ثبت وقایع حسابرسی داده‌ها هستند، با هم گروه‌بندی شدند زیرا نوع پایگاه‌داده آن‌ها را تحت‌تاثیر قرار می‌دهد.

معمار سناریوهای خوشه‌بندی را براساس نوع رویکرد معماری پیشنهاد کرد. اگر سناریوهای متعدد تحت‌تاثیر یک رویکرد معماری قرار گیرند، در این صورت باید با هم خوشه‌بندی شوند. این امر شناسایی رویکردهای معماری و امتیاز دهی به آن‌ها را برای معمار آسان‌تر می‌کند. علاوه بر این، این مرحله توسط معمار به تنهایی و فقط با گرفتن کمی بازخورد از تیم نرم‌افزار انجام شد.

اولین گام شناسایی نقاط تعادل و حساسیت برای هر رویکرد معماری با توجه به سناریوهایی بود که رویکرد معماری پشتیبانی می‌کرد. با این حال، تیم نرم‌افزار فکر کردن به مزایا و معایب هر رویکرد را راهکار آسان‌تری یافت. شناخت مزایا و معایب این عبارات برای تیم در مقایسه با عبارات موازنه و نقاط حساسیت آسان‌تر است.

سپس تیم نرم‌افزار هر رویکرد معماری را از حیث اینکه چقدر می‌تواند سناریو متناظر خود را پشتیبانی کند، از ۱ تا ۵ رتبه‌بندی کرد. رویکردهای معماری با بیش‌ترین وزن به تیم ایده‌ی روشنی از اجزای بالقوه SRAs شامل نوع پایگاه‌داده، فرمت های فایل و چارچوب‌ها داد.

مراجع

[٢٠] “Supporting standards: SOPHIA KTORI DISCUSSES THE IMPLEMENTATION OF DATA STANDARDS FOR LABORATORY INFORMATICS.” In: ١٧٠ (٢٠٢٠).url:https://link.gale.com/apps/doc/A٦٢١٥٨٠٩٠٤/AONE?u=anon~ac٧٧٥٨٨a&sid=googleScholar&xid=a٩a٤٣e٤a (cit. on p. ١١).

[AAC+١٧] E. M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, M. Galster, P. Avgeriou. “A mapping study on design-time quality attributes and metrics”. In: *Journal of Systems and Software* ١٢٧ (٢٠١٧), pp. ٥٢–٧٧ (cit. on p. ٢٤).

[AGG١٢] S. Angelov, P. Grefen, D. Greefhorst. “A framework for analysis and design of software reference architectures”. In: (٢٠١٢). doi: ١٠.١٠١٦/j.infsof.٢٠١١.١١.٠٠٩ (cit. on pp. ١٤–١٦, ٢١).

[ATG٠٨] S. Angelov, J. J. Trienekens, P. Grefen. “Towards a method for the evaluation of reference architectures: Experiences from a case”. In: *European Conference on Software Architecture*. Springer. ٢٠٠٨, pp. ٢٢٥–٢٤٠ (cit. on pp. ١٨, ١٩).

[ATG١٤] S. Angelov, J. J. Trienekens, P. Grefen. “Extending and Adapting the Architecture Tradeoff Analysis Method for the Evaluation of Software Reference Architectures”. In: (٢٠١٤) (cit. on pp. ١٨, ١٩, ٢٤).

[Bis١٣] P. Bistؘk. “Advanced remote laboratory for control systems based on Matlab and .NET platform”. In: ٢٠١٣ *IEEE ١١th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. ٢٠١٣, pp. ٣٥–٣٩. doi: ١٠.١١٠٩/ICETA.٢٠١٣. ٦٦٧٤٤٠٠ (cit. on p. ١١).

[Bro١٨] S. Brown. *Technical leadership and the balance with agility: Software Architecture for Developers - Volume ١*. ٢٠١٨ (cit. on pp. ٨, ١٢–١٤).

[BRST٠٥] K. Bergner, A. Rausch, M. Sihling, T. Ternité. *DoSAM – Domain-Specific Software Architecture Comparison Model*. Springer Berlin Heidelberg NewYork, ٢٠٠٥, pp. ٤– ٢٠. isbn: ٣٥٤٠٢٩٠٣٣٨ (cit. on pp. ١٩, ٢٣, ٢٦).

[BZJ] M. A. Babar, L. Zhu, R. Jeffery. “A Framework for Classifying and Comparing Software Architecture Evaluation Methods”. In: (). doi: ١٠.١٧٤٨٥/ijst/٢٠١٦/v٩i٣٠/٩٦٦٥٣ (cit. on p. ١٨).

[CBB+١٠] P. Clements, F. Backmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford. *Documenting Software Architectures: Views and Beyond*. ٢nd edition. Addison-Wesley Educational Publishers Inc, ٢٠١٠. isbn: ٠٣٢١٥٥٢٦٨٧ (cit. on p. ١٢).

[CBS١٧] M. A. Chauhan, M. A. Babar, Q. Z. Sheng. “A reference architecture for provisioning of tools as a service: meta-model, ontologies and design elements”. In: *Future Generation Computer Systems* ٦٩ (٢٠١٧), pp. ٤١–٦٥ (cit. on pp. ١٥, ١٦).

[CMV+٠٩] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, M. Bone. “The Concept of Reference Architectures”. In: (٢٠٠٩). doi: ١٠.١٠٠٢/sys.٢٠١٢٩ (cit. on pp. ٩, ١٤, ١٥,٢١).

[Dol٠١] T. J. Dolan. “Architecture Assessment of Information-System Families : a practical perspective”. In: (٢٠٠١) (cit. on pp. ٢١–٢٣).

[Fai١٢] G. Fairbanks. *Just Enough Software Architecture: A Risk-Driven Approach*. Marshall Brainerd, ٢٠١٢. isbn: ٩٨٧-٠-٩٨٤٦١٨١-٠-١ (cit. on pp. ١٢–١٤).

[FGB٠٤] E. Folmer, J. van Gurp, J. Bosch. *Software Architecture Analysis of Usability*. Springer,٢٠٠٤, pp. ٣٨–٥٨. isbn: ٩٧٨٣٥٤٠٢٦٠٩٧٤ (cit. on pp. ٢٢, ٢٣, ٢٦, ٢٧, ٢٩).

[GVV٠٥] B. Graaf, H. Van Dijk, A. Van Deursen. “Evaluating an embedded software reference architecture-industrial experience report”. In: *Ninth European Conference on Software Maintenance and Reengineering*. IEEE. ٢٠٠٥, pp. ٣٥٤–٣٦٣ (cit. on p. ١٩).

[ISO١١] ISO/IEC. “ISO/IEC ٢٥٠١٠:٢٠١١ Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models”. In: (٢٠١١) (cit. on pp. ٢٤, ٣٨).

[KGFČ١٥] M. Kal̊z, J. Garcيa-Zubيa, M. Fikar, Ľ. Čirka. “A Flexible and Configurable Architecture for Automatic Control Remote Laboratories”. In: *IEEE Transactions on Learning Technologies* ٨.٣ (٢٠١٥), pp. ٢٩٩–٣١٠. doi: ١٠.١١٠٩/TLT.٢٠١٥.٢٣٨٩٢٥١ (cit. on p. ١١).

[KKC٠٠] R. Kazman, M. Klein, P. Celements. “ATAM: Method for Architecture Evaluation”.In:(٢٠٠٠).url:https://resources.sei.cmu.edu/asset_files/TechnicalReport/٢٠٠٠_٠٠٥_٠٠١_١٣٧٠٦.pdf (cit. on pp. ١٣, ١٩–٢١, ٢٣–٢٦, ٢٨, ٢٩).

[KYZC١٢] F. Kong, L. Yuan, Y. F. Zheng,W. Chen. “Automatic Liquid Handling for Life Science:A Critical Review of the Current State of the Art”. In: *Journal of Laboratory Automation*١٧.٣ (٢٠١٢). PMID: ٢٢٣٥٧٥٦٨, pp. ١٦٩–١٨٥. doi: ١٠.١١٧٧/٢٢١١٠٦٨٢١١٤٣٥٣٠٢.eprint: https://doi.org/١٠.١١٧٧/٢٢١١٠٦٨٢١١٤٣٥٣٠٢. url: https://doi.org/١٠.١١٧٧/٢٢١١٠٦٨٢١١٤٣٥٣٠٢ (cit. on p. ١١).

[LBK٩٧] C.-H. Lung, S. Bot, K. Kalaichelvan. “An Approach to Software Architecture Analysis for Evolution and Reusability”. In: (١٩٩٧). url: https://resources.sei.cmu.edu/asset_files/WhitePaper/١٩٩٧_٠١٩_٠٠١_٢٩٧٧٥.pdf (cit. on pp. ٢١, ٢٣, ٢٤).

[LBVB٠٠] N. Lassing, P. Bengtsson, H. van Vliet, J. Bosch. “Analyzing Software Architectures for Modifiability”. In: (٢٠٠٠). url: https://www.researchgate.net/publication/٣٠٤٩٩١٦٤_Analyzing_Software_Architectures_for_Modifiability (cit. on pp. ٢١, ٢٣,٢٥, ٢٦).

[MAFA١٣] S. Martيnez-Fernلndez, C. Ayala, X. Franch, D. Anmeller. “A Framework for Software Reference Architecture Analysis and Review”. In: (٢٠١٣). url:https://upcommons.upc.edu/bitstream/handle/٢١١٧/٢٤٠٤٠/eselaw٢٠١٣_submission_١٤٪٢٠٪٢٨٢٪٢٩.pdf?sequence=١&isAllowed=y (cit. on p. ١٥).

[MGM٠٦] M. Mattsson, H. Grahn, F. Mٞrtensson. “Software architecture evaluation methods for performance, maintainability, testability, and portability”. In: *Second International Conference on the Quality of Software Architectures*. Citeseer. ٢٠٠٦ (cit. on p. ١٨).

[MSA+١٥] S. Martٟnez-Fernلndez, P. S. M. dos Santos, C. P. Ayala, X. Franch, G. H. Travassos. “Aggregating Empirical Evidence about the Benefits and Drawbacks of Software Reference Architectures”. In: (٢٠١٥). doi: ١٠.١١٠٩/ESEM.٢٠١٥.٧٣٢١١٨٤ (cit. on p. ١٥).

[PS١٥a] A. Patidar, U. Suman. “A Survey on Software Architecture Evaluation Methods”. In: (٢٠١٥) (cit. on p. ١٧).

[PS١٥b] A. Patidar, U. Suman. “A survey on software architecture evaluation methods”. In: ٢٠١٥ ٢nd *International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE. ٢٠١٥, pp. ٩٦٧–٩٧٢ (cit. on p. ١٨).

[PSL+٢٠] M. Porr, S. Schwarz, F. Lange, L. Niemeyer, T. Hentrop, D. Marquard, P. Lindner, T. Scheper, S. Beutel. “Bringing IoT to the Lab: SiLA٢ and Open-Source-Powered Gateway Module for Integrating Legacy Devices into the Digital Laboratory”. In: *HardwareX* ٨ (٢٠٢٠), e٠٠١١٨. issn: ٢٤٦٨-٠٦٧٢. doi: https://doi.org/١٠.١٠١٦/j.ohx.٢٠٢٠.e٠٠١١٨.url:https://www.sciencedirect.com/science/article/pii/S٢٤٦٨٠٦٧٢٢٠٣٠٠٢٧٤ (cit. on p. ١١).

[RW١٢] N. Rozanski, E. Woods. *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. ٢nd edition. Pearson Education, Inc., ٢٠١٢. isbn: ٩٧٨٠٣٢١٧١٨٣٣٤ (cit. on pp. ١٢–١٤, ١٧).

[RZRK١٩] A. Raza, S. Zafar, S.U. Rehman, U. Khattak. “Software Architecture Evaluation Methods: A Comparative Study”. In: *International Journal of Computing and Communication Networks* ١.٢ (٢٠١٩), pp. ١–٩ (cit. on p. ١٨).

[SA١٧] I. Schmid, J. Aschoff. “A scalable software framework for data integration in bioprocess development”. In: *Engineering in Life Sciences* ١٧.١١ (٢٠١٧), pp. ١١٥٩–١١٦٥.doi:https://doi.org/١٠.١٠٠٢/elsc.٢٠١٦٠٠٠٠٨.eprint:https://onlinelibrary.wiley.com/doi/pdf/١٠.١٠٠٢/elsc.٢٠١٦٠٠٠٠٨.url:https://onlinelibrary.wiley.com/doi/abs/١٠.١٠٠٢/ elsc.٢٠١٦٠٠٠٠٨ (cit. on p. ١١).

[Sol١٢] F. Solms. *What is Software Architecture?* Association for Computing Machinery, ٢٠١٢, pp. ٣٦٣–٣٧٣. isbn: ٩٧٨١٤٥٠٣١٣٠٨٧ (cit. on p. ١٢).

[SS١٢] P. Shanmugapriya, R. Suresh. “Software architecture evaluation methods-a survey”. In: *International Journal of Computer Applications* ٤٩.١٦ (٢٠١٢) (cit. on p. ١٨).

[ZSZ١١] S. Zhao, Z. Shi, S. Zhu. “A virtual laboratory architecture for engineering education”. In: ٢٠١١ *IEEE* ٣rd *International Conference on Communication Software and Networks*. ٢٠١١, pp. ٥٦٠–٥٦٣. doi: ١٠.١١٠٩/ICCSN.٢٠١١.٦٠١٣٨٩٥ (cit. on p. ١١).

Abstract

Software development is important to ensure the system's requirements and qualities. If the architecture is implemented poorly, then the system might not be fulfill the users' expectations. A reference architecture can facilitate the process of achieving a concrete architecture because it provides architectural approaches, designs, and components as starting point that can be followed. A reference architecture can be helpful in the laboratory device domain especially for liquid handling devices because of the common requirements and qualities that the devices share. To reach a reference architecture, it needs to be evaluated. Unfortunately, there is not an evaluation method that is oriented towards evaluating reference architectures. As a result, this thesis presents an adapted evaluation method from Architecture Tradeoff Analysis Method (ATAM) to evaluate reference architectures. The adapted evaluation method was tested by applying it to the liquid handling device domain to obtain a reference architecture. The obtained reference architecture is also presented in this thesis and is tested by using it as a facilitator to implement a small prototype of a liquid handling device.

Payame Noor University

Department of North Tehran

Thesis Submitted in Partial Fulfillment of the requirement for the Degree of M.Sc In software

Title:

New evaluation method for reference architectures utilized to design a software reference architecture for liquid handling devices

Supervisor:

Dr.Ali Razavi Ebrahimi

Advisor:

Dr. Ali Razavi Ebrahimi

By:

Akbar Hamidi

٨,٢٠٢٢