Arman Process Journal
(APJ)
Homepage: www.armanprocessjournal.ir

Feasibility Checking and Assessment for Software Architecture Requirements

N. Fareghzadeh[*,1]

[1] Department of Computer Science, *Khodabandeh Branch*, Islamic Azad University, Zanjan, Iran

ABSTRACT

Received: 4 June 2022
Accepted: 20 Septebber 2022

KEYWORDS:

Software Architecture,

Feasibility,

Assessment,

architectural requirements,

service quality,

[1] Corresponding author
n.fareghzadeh@iauz.ac.ir

Today, most technology-oriented applications are dependent on large and complex software systems. Software architecture provides a systematic framework for managing software complexities and dependencies. Software architecture contains key decisions about the structure of a target software system, which includes the selection of software components, modules and the relationships between these components. Moreover, the software architecture specifies the behavior of these components as the interaction they perform together to build a monolithic software system. Nowadays, one of the issues of concern to software developers is the correct and principled implementation of software architecture and accordingly, the accurate evaluation of architectural dimensions. Also, one of the main roles and functions of information technology units in different organizations is to provide assurance in terms of technical structure and software architecture. The requirement for the basic implementation of software architecture is a clear understanding of the goals and requirements of the architecture. Therefore, in order to make sure, architectural requirements should be fundamentally assessed and evaluated. Due to the importance of this issue, in this article we discuss the feasibility checking and assessment of software architecture requirements.

NUMBER OF REFERENCES	NUMBER OF FIGURES	NUMBER OF TABLES
21	0	0

نشریه تخصصی آرمان پردازش، دوره ۳، شماره ۲، تابستان ۱۴۰۱

فصلنامه تخصصی آرمان پردازش
(APJ)
Homepage: www.armanprocessjournal.ir

امکان سنجی و ارزیابی الزامات معماری نرم افزار

نفیسه فارغ زاده*[1]

[1] گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی، واحد خدابنده، زنجان، ایران

چکیده

امروزه اکثر کاربردهای فناور محور، وابسته به سیستم های نرم افزاری بزرگ و پیچیده می باشند. معماری نرم افزار قادر است چارچوبی نظام مند جهت مدیریت پیچیدگی ها و وابستگی های نرم افزار ارائه نماید. معماری نرم‌افزار حاوی تصمیمات کلیدی راجع به ساختار یک سیستم نرم‌افزاری است که شامل انتخاب اجزای سازنده نرم‌افزار و روابط بین این اجزاء است. همچنین معماری رفتار این اجزاء را براساس تعاملی که برای پیاده سازی یک سیستم نرم افزاری، انجام می‌دهند، تبیین می نماید. امروزه یکی از مسائل مورد توجه توسعه‌دهندگان نرم‌افزار، پیاده سازی صحیح و اصولی نیازمندی های معماری نرم‌افزار و به تبع آن، ارزیابی دقیق ابعاد معماری نرم افزار است. همچنین از نقشها و وظائف اصلی واحدهای فناوری اطلاعات در سازمان های مختلف، ایجاد اطمینان از لحاظ ساختار فنی و معماری نرم‌افزار می‌باشد. لازمه پیاده سازی اصولی معماری نرم افزار درک شفاف از اهداف و نیازمندی های معماری می باشد. لذا در این راستا ضروری است، نیازمندی ها و الزامات معماری بصورت اصولی امکان سنجی و ارزیابی شوند. با توجه به اهمیت این مساله، در این مقاله به امکان سنجی و ارزیابی نیازمندی ها و الزامات معماری نرم افزار می پردازیم.

واژگان کلیدی:

معماری نرم افزار،

امکان سنجی،

ارزیابی،

نیازمندی های معماری،

کیفیت سرویس،

تعداد جداول	تعداد شکل ها	تعداد مراجع
۰	۰	۲۱

مقدمه

امروزه بدون بهره گیری از دستاوردهای عظیم و شاخصه های ارزنده حوزه علمی مهندسی نرم افزار، ارائه خدمات کاربردی مختلف، به فرآیندی پیچیده، زمان بر و گاها طاقت فرسا تبدیل میگردد. اساسا نرم افزار محصول فناور محوری میباشد که به واسطه فرآیند مهندسی نرم افزار طراحی و ایجاد می شود و شامل مجموعه ای از دستورالعملها، ساختارهای داده ای، برنامه ها و اسناد و مدارک مرتبط میباشد که مبین عملکرد و نحوه ارائه سرویس و خدماتی است که کاربرد خاصی را اتوماتیک و مکانیزه می سازد و کنترل فعالیتهای سخت افزار و هدایت پردازش داده ها را بر عهده دارد. همچنین واژه مهندسی به قوانین، اصول، استانداردها و نظم دادن به روشهائی اشاره دارد که بتوان از آنها در فرایندهای تولید و توسعه بهره گرفت [۱].

معماری نرم‌افزار، مجموعه‌ای از ساختارهای لازم برای استدلال در مورد یک چارچوب یک سیستم نرم افزاری می‌باشد که شامل اجزاء و عناصر نرم‌افزاری، روابط بین آنها و ویژگی‌های مرتبط می باشد. معماری نرم‌افزار بنیانی محکم جهت ایجاد و توسعه نرم‌افزار فراهم می نماید که عموما با توجه به تعداد کاربر، زیرساخت، اهداف کسب و کار و درجه توسعه پذیری آن سیستم انتخاب می‌شود. همچنین رفتار این اجزا را به عنوان تعاملی که باهم برای ساختن یک زیر سیستم بزرگتر، انجام می‌دهند، مشخص می سازد. معماری نرم‌افزار، کارکردها، قابلیت استفاده، انعطاف پذیری، عملکرد، استفاده مجدد، قابل درک بودن، محدودیت‌های اقتصادی و فناوری، نحوه تبادل اطلاعات و جنبه‌های زیبایی سیستم نرم‌افزاری را شامل می‌شود [۲]. معماری عموما به عنوان یک نقشه و یا یک الگو برای سیستم ایفای نقش می‌کند. با استفاده از معماری می‌توانیم یک دید کلی[1] جهت مدیریت پیچیدگی‌های یک سیستم نرم افزاری ایجاد کنیم و یک چارچوب ارتباط بسیار مفید و یک هماهنگی منحصربه‌فرد را بین اجزای تشکیل دهنده یک سیستم ایجاد نماییم. معماری، قادر است یک راه حل ساختارمند را برای رسیدن به تمامی نیازمندی های فنی و عملکردی یک سیستم در دست قرار دهد [۳]. علاوه بر این؛ با استفاده از معماری می توان ویژگی های کیفی معمول از قبیل؛ عملکرد، کارائی و امنیت را تقویت و یا بهینه سازی کرد. مراحل و گام های ایجاد شمای معماری نرم افزار که از طریق آن بتوان یک محصول نرم افزاری را طراحی منطقی نموده و به شکل اصولی توسعه داد فرایند معماری نرم افزار نام دارد. همچنین، معماری شامل تصمیمات مهمی در رابطه با سازمان مربوط به توسعه نرم افزار است. هر کدام از این تصمیمات می‌توانند تاثیر چشمگیری بر روی کیفیت، قابلیت نگهداری، قابلیت عملکرد و موفقیت سراسری پروژه داشته باشند. برخی از مهم ترین این تصمیمات شامل موارد زیر هستند [۴]:

- انتخاب عناصر ساختاری و واسطهایی که با استفاده از آنها سیستم به عملکرد خود جامه عمل می پوشاند
- رفتار تعریف شده بین اجزاء معماری و همکاری‌های شکل گرفته مابین این عناصر
- ترکیب این ساختارها و رفتارهای آنها به درون یک زیر سیستم بزرگتر
- تصمیمات معماری منطبق بر اهداف کسب و کار
- سبک های معماری و تصمیمات کاربردی در برابر آنها

اساسا در این حوزه، نیازمندی های معماری تشریح و توصیف ویژگی‌ها و معماری سیستم نرم افزاری هدف است. نیازمندی ها انتظارات کارکردی و عملیاتی از محصول نرم افزاری را منتقل می‌کند. الزامات می تواند آشکار یا پنهان، شناخته یا ناشناخته، مورد انتظار یا غیر منتظره باشند و درچرخه تکامل معماری به مرور لحاظ بشوند. فرایند گردآوری و جمع آوری نیازهای معماری نرم افزار، تجزیه و تحلیل و مستند سازی آن ها به عنوان مهندسی نیازمندی‌های معماری شناخته می شود. هدف از مهندسی نیازمندی، توسعه و نگهداری سند سطح بالا و توصیفی خصوصیات نیازمندی های معماری سیستم هدف است. پس از نهایی شدن این مراحل، تمام نیازمندیها مستندسازی شده و برای اعتبارسنجی، تصحیح و سپس پیاده سازی معماری آماده هستند [۵]. مهندسی نیازمندیها روند بسیار ساده ای است، اما در واقع از حساس ترین فرآیندها در چرخه حیات سیستم نرم افزاری است. با توجه به ضرورت این مساله در این مقاله دیدگاه امکان سنجی و ارزیابی نیازمندی های معماری نرم افزار را توصیف نموده و ابعاد آن تشریح می گردد.

امکان سنجی و ارزیابی نیازمندی های معماری نرم افزار

بمنظور ایجاد یک دیدگاه مدون درخصوص امکان سنجی و ارزیابی نیازمندی‌های معماری نرم افزار لازم است ابتدا دید جامعی درخصوص اهداف و ابعاد معماری نرم افزار ایجاد شود. اساسا معماری نرم افزار یک برنامه یا سیستم محاسباتی، ساختار یا ساختارهای سیستم محاسباتی نرم افزاری است که خصوصیات قابل رویت از بیرون، عناصر و ارتباطات بین آنها را توصیف نموده و نمایش می دهد [۶]. درواقع معماری نرم افزار فرایند تعریف یا راه حل نرم افزاری ساختارمند است، به شکلی که بتواند کلیه نیازمندیهای فنی و عملیاتی مورد انتظار را، پوشش دهد. این فرایند تکاملی باید به

[1] Abstraction View

شکلی انجام شود که معیارهای کیفی مانند بهره دهی، امنیت، قابلیت اطمینان و مدیریت پذیری بهینه شده باشند. معماری همچنین، باید توصیفی از کارکردهای سیستم، قابلیت استفاده، انعطاف پذیری، عملکرد، استفاده مجدد، قابل درک بودن، محدودیتهای اقتصادی و فناوری، نحوه تبادل اطلاعات و جنبه های زیبایی سیستم نرم افزاری را پوشش دهد [۸-۷]. معماری نرم افزار شامل مجموعه ای از تصمیمات مهم مبتنی بر امکان سنجی در مورد سازمان مربوط به توسعه نرم افزار است و هر یک از این تصمیمات می تواند تاثیر قابل توجهی بر عملکرد و موفقیت کلی محصول نهایی داشته باشد. در این راستا، برخی از اهداف و نیازمندی‌های بنیادین معماری نرم افزار عبارتند از [۱۱-۹]:

- رده بندی اجزاء و المان های سیستم نرم افزاری براساس ویژگی های عملکردی و غیرعملکردی ضمن بیان شیوه تعامل و ارتباط؛
- در معرض دید قراردادن ساختار سیستم ضمن مخفی-سازی جزئیات اجرای آن؛
- تحقق بخشیدن به تمام موارد استفاده و سناریوهای کاربرد؛
- سعی در رسیدگی به الزامات ذینفعان مختلف؛
- رسیدگی به الزامات عملکردی و کیفیتی سیستم؛
- کاهش هدف مالکیت و بهبود موقعیت سازمان در بازار؛
- بهبود کیفیت و قابلیت های قابل ارائه توسط سیستم؛
- بهبود اعتماد خارجی به سازمان و سیستم؛
- کمک به توسعه دهندگان برای استفاده از مولفه ها یا کد؛
- حذف کدهای زائد با ساختن عناصر و کتابخانه های قابل استفاده مجدد و چند بار مصرف؛
- حل کردن مسائل مبهم و غیرمترقبه ای که در حین طراحی هر برنامه ای رخ می دهد؛
- ایجاد رضایت برای کاربران نهایی با ارائه راه حل های میانبر و Cutting-Edge ؛
- راحتی در بروزرسانی و سازگار کردن برنامه با هر گونه تغییرات در آینده؛

بنابر موارد عنوان شده نیازمندی معماری نرم افزار، خصوصیت ها یا ملاکهائی هستند که با هدف دستیابی به اهداف معماری نرم افزار و رفع پیچیدگی ها، افزونگی ها و معضلات موجود تعریف و تبیین می گردند [۱۲]. درحوزه امکان سنجی نیازمندی های معماری نرم‌افزار، بمنظور بیان صحیح و نظام مند نیازمندی‌های ضروری معماری، در مراجع اخیر انجام مجموعه ای از فعالیتها توصیه شده است که در این بخش ابتدا این موارد را بیان می نمائیم. در این راستا، تعیین اینکه کدام نیازمندی ها دارای اهمیت معمارانه هستند بسیار حائز اهمیت است. برای هر سیستم خاص، نیازمندی های معمارانه مهم باید با اهداف کسب و کار و مأموریت سیستم همسو باشند. برخی از ویژگی های سیستم ها علاوه بر جنبه عملکردی سیستم باید از نظر معماری نیز اهمیت داشته باشند. اساسا چهار فعالیت عمده در حوزهٔ امکان سنجی نیازمندیهای معماری نرم افزار وجود دارد [۱۵-۱۳]:

- استخراج: فعالیتی است که در آن فرد مسئول با ذینفعان پروژه ارتباط برقرار میکند و نیاز و انتظار آنان از نرم افزار و معمار که قرار است تولید شود را کشف می نماید. این فعالیت از طریق مصاحبه، مشاهده و سایر تکنیکهای ممکنه صورت میپذیرد.
- تجزیه و تحلیل: در این مرحله با تجزیه و تحلیل خروجی مرحله قبل تضادهای احتمالی بین نیازمندیهای استخراج شده از بین میرود، مرز نیازمندیهایی که باید پیاده سازی شوند مشخص شده و نیازمندیها دسته بندی میشوند.
- تهیه مشخصات: این فعالیت به زبان ساده به مستندسازی و مکتوب کردن نیازمندیها اشاره دارد، با این توضیح که بصورت نظام مند قابل بازبینی، ارزیابی و تصدیق باشند.
- تصدیق: در این فعالیت نیازمندیهای مستند شده با هدف تایید و تصدیق صحت آنها بررسی میشوند. این صحت شامل مطابقت با استانداردهای شرکت، قابل فهم بودن، سازگاری و کامل بودن ویژگیهای آنها میباشد.

مسلّما، عدم توجه به هر کدام از فعالیتها و موارد ذکر شده در قسمت بالا صحت و موفقیت پروژه نرم افزاری را با خطر جدّی مواجه میکند. امکان سنجی نیازهای معماری نرم افزار را می توان براساس الزامات زیر طبقه بندی نمود [۱۷-۱۶]:

- امکان سنجی کارکردی

این نوع امکان سنجی مشخص می دارد که معماری نرم افزار براساس نیازمندی های بنیادین "چه کاری" را باید انجام دهد و اهداف معماری نرم افزار را شفاف می سازد. امکان سنجی کارکردی، از مدل منطقی که براساس ضروریات سیستم و قابلیت های خواسته شده کاربر ساخته شده است، استنتاج می شوند و باید در برگیرنده خواص عملکردی و عملیاتی نیز باشند.

- امکان سنجی واسطه ای

این امکان سنجی مشخص می کند که سخت افزار، نرم افزار یا عناصر پایگاه داده، باید با کدام سیستم و یا کدامیک از اجزای سیستم، در ارتباط بوده و کار نماید. الزامات واسطه ای می بایست به واسط های نرم افزاری، سخت افزاری و ارتباطی طبقه بندی شوند. واسط های نرم افزاری می توانند شامل سیستم های عامل، محیط های نرم

افزاری، قالب پرونده ها، سیستم های مدیریت پایگاه داده ها و یا سایر نرم افزارهای کاربردی باشند. الزامات واسط سخت افزاری، پیکربندی سخت افزار را مشخص می کنند. الزامات ارتباطی چارچوب و نحوه ارتباط با سایر نرم افزارها و سخت افزار ها را تعیین می کنند. به عنوان مثال ممکن است استفاده از یک قرارداد[۲] خاص شبکه را ایجاب نماید. الزامات واسط خارجی می بایست یا مشخص گردند و یا به سند کنترل واسط ارجاع نمایند. نیازهای واسط کاربر می بایست با توجه به الزامات عملیاتی تعیین شوند [۱۸]. الزامات واسطه ای را می توان توسط نمودارهای بلوکی سیستم نشان داد، به عنوان مثال نمایش پیکربندی سخت افزار.

- امکان سنجی عملیاتی

این امکان سنجی چگونگی اجرای اجزاء و ماژول های معماری سیستم و چگونگی برقراری ارتباط سیستم با مجریان و بهره برداران را بیان می کنند. الزامات عملیاتی اجرائی شامل واسطهای معماری نرم افزار، کلیه عوامل تاثیر متقابل کاربر و سیستم نرم افزاری، نحوه استفاده و تمامی نیازهای تدارکاتی و تشکیلاتی پیاده سازی و نحوه اجرای عناصر معماری می باشند.

- امکان سنجی منابع

این امکان سنجی حد نهایی منابع فیزیکی مانند قدرت پردازش، حافظه اصلی، فضای دیسک و غیره را مشخص می سازند. تعیین این مشخصات به ویژه زمانی که توسعه بعدی توان پردازشی سخت افزار در طول چرخه حیات، مانند بسیاری از سیستم های نصب شده، خیلی گران تمام می شود، لازم و ضروری است.

- امکان سنجی وارسی و اعتبار سنجی

این امکان سنجی شرایط چگونگی وارسی قابلیت ها و کارکردهای معماری نرم افزار را مشخص کرده و چارچوب طرح وارسی و اعتبار سنجی معماری را تعیین می کنند. آنها می توانند شامل مشخصات لازم جهت انجام شبیه سازی، تقلید، آزمون های زیست با داده های شبیه سازی شده، آزمون های زیست با داده های واقعی و برقراری ارتباط با محیط آزمایش باشند [۱۹].

- امکان سنجی استانداردسازی

این امکان سنجی، نیازهای ویژه جهت مستندسازی استاندارد اجزاء معماری نرم افزار به انضمام آنچه که در این استانداردها مشخص شده اند (مانند فهرست مندرجات راهنما) را معین می کنند.

- امکان سنجی کیفی

این امکان سنجی خواصی از معماری نرم افزار را بیان می دارند که از مناسب بودن معماری و نرم افزار با اهدافی که برایش تعیین شده است اطمینان حاصل شود. این خواص به جز ویژگی های کیفی عمده مانند قابلیت اطمینان، قابلیت نگهداری و ایمنی است که همواره می بایست موجود باشند. امنیت، کارایی، قابلیت اطمینان، قابلیت تغییر و ... همه ویژگی‌های کیفیتی هستند که ممکن است بر ساختار سیستم تأثیر بگذارند. هر ویژگی کیفی دارای مجموعه‌ای از تکنیک‌های معمارانه ساختاری است که برای دستیابی به این ویژگی ها به کار می روند و در امکان سنجی باید مدنظر قرار داده شود [۲۰].

- امکان سنجی تکنولوژیکی و زیرساخت فنی

این امکان سنجی خواصی از معماری نرم افزار که مرتبط با چارچوب تکنولوژیکی نرم افزار هدف و زیرساخت فنی و تاثیرات ساختاری مرتبط باشد را مدنظر قرار می دهد. اساسا هر عملکرد خاصی بر ساختار معماری تأثیری نخواهد داشت، اما مجموعه های بزرگی از عملکردهای مشابه تأثیرگذار خواهند بود. یکی از نمونه‌های مربوط به تأثیر ساختاری، شکستن عملکرد است تا بتوان آن را به موقع اجرا کرد. مثال دیگر شناسایی اشتراکات در عملکرد برای کاهش زمان اجرا و نگهداری می باشد. همچنین هنگامی که نیازمندی های مجموعه‌ای از سیستم‌های مشابه مانند خط تولید نرم‌افزار جمع‌آوری می‌شوند، اشتراکات و تغییرات در آن سیستم‌ها ممکن است از نظر معماری قابل توجه باشند. ممکن است نیاز به استفاده از یک فناوری خاص برای یک سیستم، مانند .NET یا J2EE (Java 2 Enterprise Edition) وجود داشته باشد. این نیازمندی ها احتمالا از نظر معماری مهم هستند، زیرا بسیاری از تصمیمات بعدی با به کاربردن فناوری های خاص محدود می شوند [۲۱].

- امکان سنجی استقرار و اجرا

نیازمندی هایی که استراتژی های مربوط به استقرار پیش‌بینی‌شده و نحوه ی عملکرد سیستم را توصیف می‌کنند، از نظر معماری مهم هستند زیرا ممکن است موجودیت‌های ساختاری، ملزم به حمایت از استقرار یا ملاحظات عملیاتی باشند.

لازم به ذکر است، نیازهای معماری نرم افزار می بایست با دقت بسیار امکان سنجی و تشریح شوند. همچنانکه نیازها گردآوری می شوند، باید در برگیرنده شناسه ها، مراجع، مقادیر، اولویتها و میزان ثبات باشند. نیازها باید کامل و نامتناقض بوده و از تکرار آنها اجتناب شود.

ارزیابی

پس از انجام انواع امکان سنجی براساس دیدگاه پیشنهادی لازم است ارزیابی از معماری و بروندادهای فاز امکان سنجی صورت پذیرد. ارزیابی معماری نرم افزار تکنیک یا روشی است که ویژگی ها، نقاط قوت و ضعف معماری نرم افزار، سبک معماری نرم افزار یا الگوی

[2] Protocol

طراحی را تعیین می کند. ارزیابی معماری نرم افزار به توسعه - دهندگان اطمینان می دهد که معماری انتخابی آنها هم نیازمندی های عملکردی و هم غیرعملکردی و کیفی را برآورده می کند. سایر مزایای چنین ارزیابی به شرح زیر می باشد:

- تخمین و تعیین میزان دستیابی یک معماری به اهداف کیفی
- نمایش تعاملات این اهداف کیفی و خروجی های ارزیابی
- ارائه دقیق برآورد فنی از ابعاد تحلیلی معماری
- تفصیل امکان سنجی اهداف کسب و کار
- تحلیل و برآورد انواع امکان سنجی نیازمندی ها به صورت مجموعه هایی از سناریوها و مجموعه ای از نقاط حساس و پایاپای
- نگاشت تصمیمات معماری به الزامات و فرایند اجرای معماری

لازم به ذکر است، ارزیابی نیازمندی های معماری باید مزایای بیشتری نسبت به هزینه ی انجام آن داشته باشد. چنین ارزیابی افزایش درک از سیستم و مستندسازی آن، تشخیص مشکلات موجود در معماری و یادگیری سازمانی را تضمین می کند. در مراجع جدید چندین روش و تکنیک برای ارزیابی معماری نرم افزار پیشنهاد شده است. در میان آنها، رویکردهای مبتنی بر سناریو کاملاً بالغ در نظر گرفته می شوند. همچنین روش های دیگری بر مبنای مدل ویژگی و مدل های کمی برای ارزیابی معماری نرم افزار نیز وجود دارند و روش‌های دیگری برای توجیه سیستماتیک ویژگی‌های سبک‌های معماری و الگوهای طراحی ایجاد شده‌اند. روش های ارزیابی زودهنگام پیش از اجرا در معماری نرم افزار اعمال می شوند. روش‌های ارزیابی دیرهنگام نیز تفاوت بین معماری واقعی و معماری برنامه‌ریزی شده را مشخص می‌کنند. این روش ها دستورالعمل های مفیدی را در مورد چگونگی بازسازی معماری واقعی ارائه می دهند تا با معماری برنامه ریزی شده مطابقت داشته باشد.

این ارزیابی در جهت اطمینان از صحت و پوشش تمامی نیازمندی های امکان سنجی شده ذینفعان سیستم انجام میشود. با توجه به الزامات قابل توجه معماری که توسط تجزیه و تحلیل، وضعیت فعلی طرح و نتایج فعالیت های ارزیابی تعیین می شود، طراحی معماری ایجاد و بهبود می یابد . نهایتا ارزیابی معماری، فرایند تعیین میزان خوب بودن طراحی فعلی یا بخشی از آن نیازهای حاصل از تجزیه و تحلیل را برآورده می کند. پالایش و ارزیابی باید نگاشت های ایجاد شده را با اهداف سیستم تطابق داده و بازخوردهای مناسب را منعکس نماید . همچنین این پالایش و ارزیابی باید به دنبال کوچکترین تعداد مولفه هایی باشد که با پیمانه بندی اثربخش سازگار باشد و نیز به دنبال ساختمان های داده ای با کمترین پیچیدگی باشد که خواسته های اطلاعاتی را به طرزی مناسب پاسخگو باشد. ارزیابی و تحلیلی از تصمیمات معماری نرم افزار، قبل از ایجاد و ساخت سیستم و همچنین برای ایجاد تغییرات در سیستم، می تواند باعث کاهش ریسک، هزینه، زمان و دوباره کاری افراد در سازمان ها باشد.

لازم به ذکر است، در انتهای فرایند امکان سنجی و ارزیابی لازم است نیازمندی های امکان سنجی شده بصورت مدون مستندسازی و تجمیع گردند. اهمیت مستندسازی این نیازمندیها رابطه مستقیم با پیچیدگی اجزاء معماری، تأثیر محصول و امید به بقای نرم افزار است. اگر نرم افزار بسیار پیچیده باشد یا توسط تعداد زیادی از افراد ساخته شده باشد، مستندسازی نیازمندیها کمک میکند تا با آنچه که قصد داریم به آن برسیم بهتر ارتباط برقرار کنیم. به عبارت دیگر دورنمای بهتری از معماری محصول نرم افزاری نهایی داشته باشیم. همچنین سایر تاثیرات مثبت اتخاذ چنین رویکردی عبارتند از:

- افزایش اثربخشی تحلیل و طراحی در برآورده نیازمندی ها و تثبیت چارچوب معماری
- ایجاد راه حل های جایگزین و آلترناتیوهایی برای نیازمندی های معماری درمراحل اولیه ساخت نرم افزار
- کاهش ریسک ها و خطرات مرتبط با ساخت سیستم نرم افزاری
- مدلسازی ساخت یافته طراحی معماری نرم
- برقراری ارتباط بین طرفهای ذینفع در توسعه سیستم
- ایجاد قابلیت تصمیم گیری درخصوص نقایص و ریسک ها
- تدوین مدل قابل درک از چگونگی ساخت یافتگی و همکاری مولفه های معماری براساس نیازمندی ها

در پایان تاکید می گردد، امکان سنجی اصولی و درک صحیح نیازمندی ها و اهداف معماری نرم افزار قادر است با ایجاد درک عمیق و دیدگاه مشترک بین تیم های مختلف کاری تعاملات بین تیمی را بنحوی مدیریت نماید که ابهامات کاهش یافته و تیم های کاری پروژه بتوانند هم افزائی عملیاتی بیشتری ارائه نمایند.

نتیجه گیری

اساسا معماری نرم افزار یک برنامه یا سیستم محاسباتی، ساختار یا ساختارهای آن سیستم محاسباتی است که خصوصیات قابل رویت از بیرون، عناصر و ارتباطات بین آنها را توصیف نموده و نمایش میدهد. درواقع معماری نرم افزار فرایند تعریف یا راه حل نرم افزاری ساختارمند است، به شکلی که بتواند کلیه نیازمندیهای فنی و عملیاتی مورد انتظار را، پوشش دهد. بمنظور ایجاد یک دیدگاه جامع درخصوص امکان سنجی و ارزیابی نیازمندی های معماری نرم افزار لازم است ابتدا دید جامعی درخصوص اهداف و ابعاد معماری نرم افزار ایجاد شود. با توجه به اهمیت شناخت اهداف کسب و کار و نقش و ماموریت آنها در تحلیل، امکان سنجی، توجیه و ارزیابی معماری نرم افزار در سازمان ها و بمنظور تسهیل دستیابی به اهداف

معماری نرم افزار و شفاف سازی ارتباط اجزاء معماری با نیازمندی های سیستمی این مقاله دیدگاه امکان سنجی و ارزیابی نیازمندی های معماری نرم افزار را پیشنهاد می نماید. بکارگیری چنین رویکردی سازمان ها را قادر می سازد تا اهداف کسب وکار و نیازمندی های خود از سیستم هدف را متمرکزتر و جامع تر ارائه دهند و درجهت بهینه سازی و بقای سیستم نرم افزاری گام های موثرتری بردارند.

منابع

[1] Vogelsang, A., Eder, S., Hackenberg, G., Junker, M., & Teufl, S. (2014, January). Supporting concurrent development of requirements and architecture: A model-based approach. In *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)* (pp. 587-595). IEEE.

[2] Obergfell, P., Kugele, S., & Sax, E. (2019, September). Model-based resource analysis and synthesis of service-oriented automotive software architectures. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)* (pp. 128-138). IEEE.

[3] Shahin, M., Liang, P., & Babar, M. A. (2014). A systematic review of software architecture visualization techniques. *Journal of Systems and Software*, *94*, 161-185.

[4] do Carmo Machado, I., McGregor, J. D., Cavalcanti, Y. C., & De Almeida, E. S. (2014). On strategies for testing software product lines: A systematic literature review. *Information and Software Technology*, *56*(10), 1183-1199.

[5] Heinrich, R., Van Hoorn, A., Knoche, H., Li, F., Lwakatare, L. E., Pahl, C., ... & Wettinger, J. (2017, April). Performance engineering for microservices: research challenges and directions. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion* (pp. 223-226).

[6] Faisal, M. A., Aung, Z., Williams, J. R., & Sanchez, A. (2014). Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study. *IEEE Systems journal*, *9*(1), 31-44.

[7] Mahdavi-Hezavehi, S., Avgeriou, P., & Weyns, D. (2017). A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In *Managing Trade-Offs in Adaptable Software Architectures* (pp. 45-77). Morgan Kaufmann.

[8] Brosch, F. (2014). *Integrated software architecture-based reliability prediction for IT systems* (Vol. 9). KIT Scientific Publishing.

[9] Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortés, A., & Hinchey, M. (2014). An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry. *Journal of Systems and Software*, *91*, 3-23.

[10] Ding, W., Liang, P., Tang, A., & Van Vliet, H. (2014). Knowledge-based approaches in software documentation: A systematic literature review. *Information and Software Technology*, *56*(6), 545-567.

[11] Alégroth, E., Feldt, R., & Kolström, P. (2016). Maintenance of automated test suites in industry: An empirical study on Visual GUI Testing. *Information and Software Technology*, *73*, 66-80.

[12] Buede, D. M., & Miller, W. D. (2016). The engineering design of systems: models and methods.

[13] Guo, J., Jiang, Y., Zhao, Y., Chen, Q., & Sun, J. (2018, October). Dlfuzz: Differential fuzzing testing of deep

learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 739-743).

[14] Ahmad, A., & Babar, M. A. (2016). Software architectures for robotic systems: A systematic mapping study. *Journal of Systems and Software*, *122*, 16-39.

[15] Garcia, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., & Penabad, M. (2017). A framework for gamification in software engineering. *Journal of Systems and Software*, *132*, 21-40.

[16] Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, *116*, 191-205.

[17] Yang, C., Liang, P., & Avgeriou, P. (2016). A systematic mapping study on the combination of software architecture and agile development. *Journal of Systems and Software*, *111*, 157-184.

[18] Graham, D., Black, R., & Van Veenendaal, E. (2021). *Foundations of software testing ISTQB Certification*. Cengage Learning.

[19] Morschheuser, B., Hassan, L., Werder, K., & Hamari, J. (2018). How to design gamification? A method for engineering gamified software. *Information and Software Technology*, *95*, 219-237.

[20] Huber, S., Wiemer, H., Schneider, D., & Ihlenfeldt, S. (2019). DMME: Data mining methodology for engineering applications–a holistic extension to the CRISP-DM model. *Procedia Cirp*, *79*, 403-408.

[21] Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176-189.