پشتیبانی از تصمیم‌های معماری نرم‌افزار به کمک هستان‌شناسی

سیدمحمدمسعود صدرنژاد[۱] and میلاد شمسی جلالی[۲]

[۱] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی و علوم کامپیوتر، دانشگاه شهید بهشتی، تهران sadrnezhaad@ce.sharif.edu
[۲] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی و علوم کامپیوتر، دانشگاه شهید بهشتی، تهران m.shamsijalali@mail.sbu.ac.ir

چکیده: معماری نرم‌افزار را می‌توان مجموعه‌ای از تصمیم‌های مهم طراحی دانست. معماران نرم‌افزار طراحانی هستند که نگاهی سطح بالا به جنبه‌های فنی و حرفه‌ای و همچنین سایر مسائل مرتبط با گسترهٔ وسیعی از ذینفعان دارند و تصمیم‌های مهم طراحی را می‌گیرند. معماران نرم‌افزار در مورد اینکه از کدام سبک معماری استفاده کنیم، چگونه رابط برنامه‌نویسی کاربردی را طراحی کنیم یا چه توابعی باید در یک کلاس قرار بگیرند تصمیم می‌گیرند. هرچند که پژوهش‌های اندکی روی چگونگی تصمیم‌گیری دربارهٔ طراحی معماری توسط معماران انجام شده‌است اما همهٔ فعالیت‌های معماری نرم‌افزار شامل تصمیم‌گیری هستند. هر تصمیم معماری در راستای برآورده شدن تعدادی از دغدغه‌ها است. در هستان‌شناسی پیشنهادشده در این پژوهش، تصمیم‌ها و دغدغه‌ها نگه‌داری می‌شوند و با یک رابطه بین تصمیم‌ها و دغدغه‌ها مشخص می‌شود که هر تصمیم در راستای برآورده شدن کدام دغدغه‌ها است. همچنین رابطهٔ بده‌بستان میان دغدغه‌های مختلف مشخص می‌شود. سامانهٔ ایجاد شده در این پژوهش می‌تواند در ساخت و مستندسازی معماری، بصری‌سازی مفاهیم معماری، تعامل‌پذیری بیشتر میان طراح و سایر ذینفعان، تحلیل بده‌بستان برای هر معماری، پیشنهاد راهکنش و الگوهای مرتبط با هر دغدغه به معمار، تکامل معماری به فرد معمار یاری رساند.

کلمات کلیدی: معماری نرم‌افزار، هستان‌شناسی، سامانهٔ پشتیبان تصمیم

۱ مقدمه

معماری نرم‌افزار «مجموعه ساختارهای مورد نیاز برای استدلال در مورد سامانه شامل عناصر نرم‌افزاری، روابط بین آن‌ها و خصوصیات هر دو است». با این وجود، در اواسط دههٔ ۱۹۹۰، معماری نرم‌افزار به عنوان یک رشتهٔ وسیع‌تر ظاهر شد که شامل مطالعهٔ ساختارهای نرم‌افزاری و معماری‌ها به شیوه‌ای عمومی‌تر بود. این امر باعث ایجاد چندین مفهوم قابل توجه در مورد طراحی نرم‌افزار در سطوح مختلف انتزاع شد. برخی از این مفاهیم می‌توانند در حین طراحی معماری (به عنوان مثال، سبک‌های معماری) و همچنین در هنگام طراحی تفصیلی (به عنوان مثال الگوهای طراحی) مفید باشند. این مفاهیم طراحی همچنین می‌توانند برای طراحی خانوادهٔ برنامه‌ها که به عنوان خط تولید هم شناخته می‌شوند، استفاده شوند. بسیاری از این مفاهیم را می‌توان به عنوان تلاش برای توصیف و در نتیجه، استفادهٔ مجدد از دانش طراحی به دید آورد. [۱]

در فصل ۱ دربارهٔ ادبیات موضوع و مفاهیم به‌کاررفته در فصل‌های بعد صحبت شده‌است. در زیرفصل ۱-۱ دربارهٔ تلاش‌های صورت گرفته برای رسیدن به یک تعریف مشترک از معماری نرم‌افزار و در ۱-۲ دربارهٔ تفاوت‌ها و شباهت‌های مفاهیم معماری و طراحی نرم‌افزار و ارتباط این مفهوم با تصمیم‌گیری صحبت شده‌است. در فصل ۲ فنون و روش‌های به‌کارگرفته‌شده برای تصمیم‌گیری در زمینهٔ معماری نرم‌افزار با استفاده از هستان‌شناسی در پژوهش‌های پیشین بررسی شده‌است. در فصل ۳ در قالب طرح کردن ایده‌های این پژوهش برای ساخت یک سامانهٔ پشتیبان تصمیم معماری نرم‌افزار با استفاده از هستان‌شناسی بررسی شده‌است. در فصل ۴ خلاصه‌ای از نتایج این پژوهش جمع‌بندی شده‌است.

۱-۱ معماری نرم‌افزار

محققین و مهندسین نرم‌افزار از دیرباز تعاریف گوناگونی از معماری داشته‌اند. برای برخی معماری سازمان‌دهی اصلی یک سامانه است یا به تعبیری این‌که چگونه مولفه‌ها در بالاترین سطح به یکدیگر متصل می‌شوند. اشکال این تعریف این است که برای تشخیص این‌که چه چیزی را اساسی یا سطح بالا تعریف کنیم، هیچ راه مشهودی وجود ندارد و این ابهام را ایجاد می‌کند که مرز بین خروجی طراحی و معماری کجاست. از یک دیدگاه بهتر، معماری درک مشترکی است که برنامه‌نویسان متخصص

از طراحی یک سامانه دارند. [۲]

شیوهٔ معمول دیگر برای تعریف معماری نرم‌افزار، معماری را تصمیم‌های طراحی تعریف می‌کند که باید در اوایل یک پروژه گرفته شوند و تغییر در آن‌ها تبعات سنگینی دارد؛ ولی به این تعریف هم این نقد وارد است که به نظر می‌رسد معماری تصمیم‌هایی است که آرزو می‌شود بتوان در اوایل پروژه به درستی گرفت و تصمیم‌گیری درست در اوایل پروژه واقع‌گرایانه نیست. از این رو اگر این تصمیم‌ها را نمی‌توان از ابتدا به درستی گرفت باید تلاش کرد تا تصمیم‌ها را طوری گرفت که در آینده تغییر در آن‌ها تبعات کمتری داشته باشد. [۲]

از دو تعریف مذکور تعریف سومی منتج می‌شود. بنا به این تعریف «معماری دربارهٔ چیزهای مهم است، هر آنچه که باشد». در ابتدا شاید این تعریف سطحی به نظر برسد ولی از غنای بالایی برخوردار است. این تعریف بیان می‌کند که قلب تفکر معمارانه دربارهٔ نرم‌افزار این است که تصمیم بگیریم چه چیزی مهم است و سپس انرژی را صرف قرار دادن چیزهای مهم در شرایط مناسبشان کنیم. برنامه‌نویسی که می‌خواهد معمار شود باید بتواند عناصر مهم را تشخیص دهد و همین‌طور تشخیص دهد چه المان‌هایی در صورت عدم کنترل ممکن است باعث ایجاد مشکلات جدی شوند. [۲]

۱-۲ ارتباط میان معماری و طراحی نرم‌افزار

در نسخهٔ سوم کتاب پیکرهٔ دانش مهندسی نرم‌افزار[1]، معماری نرم‌افزار بخشی از فرایند طراحی نرم‌افزار است که در شکل ۱ نمایش داده شده‌است. هدف از طراحی نرم‌افزار پر کردن خلأ میان نیازمندی‌ها تا ساخت نرم‌افزار[2] است. [۱] بخش‌هایی از این شکل که به مباحث مطرح شده در این پژوهش مرتبط هستند با رنگ قرمز مشخص شده‌اند.

ساختار و معماری نرم‌افزار[3]، ذیل عنوان طراحی نرم‌افزار[4] مورد بررسی قرار می‌گیرد و ذیل موضوع ساختار و معماری نرم‌افزار به مفاهیم ساختارها و دیدگاه‌های معماری[5]، سبک‌های معماری[6]، الگوهای طراحی[7]، تصمیم‌های طراحی معماری[8]، خانواده‌های برنامه‌ها و چارچوب‌ها[9] پرداخته شده‌است. [۱] در این بین مفهوم تصمیم‌های طراحی معماری[10] بیش از همه به هدف این پژوهش نزدیک است و این‌چنین تعریف شده‌است:

ضمن فرایند طراحی، طراحان نرم‌افزار شماری تصمیم‌های اساسی می‌گیرند که تأثیرات ژرفی روی نرم‌افزار و فرایند ایجاد آن می‌گذارد **به همین دلیل تفکر دربارهٔ فرایند طراحی معماری از دیدگاه تصمیم‌گیری و نه از دیدگاه یک فعالیت مفید است.** معمولاً تأثیر

[1] Software Engineering Body of Knowledge (SWEBOK)
[2] software construction
[3] software structure and architecture
[4] software design
[5] architectural structures and viewpoints
[6] architectural styles
[7] design patterns
[8] architecture design decisions
[9] families of programs and frameworks
[10] architecture design decisions

روی ویژگی‌های کیفی و بده‌بستان‌ها[11] بین ویژگی‌های کیفی[12] هماورد مبنای تصمیم‌های طراحی هستند. [۱]

در یک نگاه ابتدایی، هیچ تفاوتی بین معماری و طراحی نرم‌افزار وجود ندارد. کلمهٔ «معماری» اغلب در متون برای اشاره به چیزی در سطح بالا استفاده می‌شود که از جزئیات سطح پایین جدا شده‌است، در حالی که بیشتر به نظر می‌رسد «طراحی» ساختار و تصمیم‌ها را در سطح پایین‌تر نشان می‌دهد. اما با بررسی فعالیت‌های یک معمار، ملاحظه می‌شود که این کاربرد از واژهٔ معماری پوچ است. [۳]

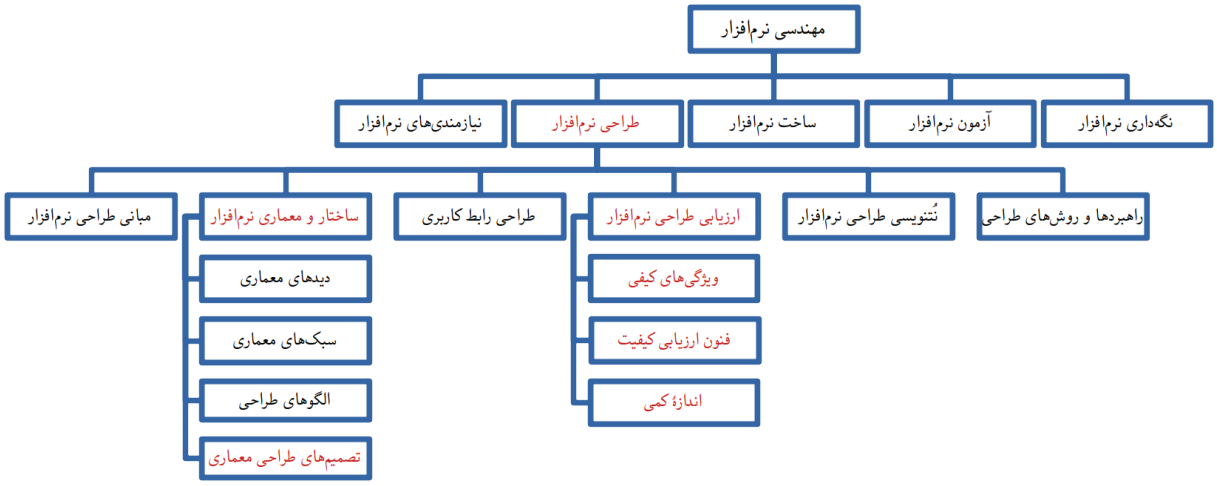
به عنوان مثال معماری یک خانه، ظاهر بیرونی آن، ارتفاعات و چیدمان فضاها و اتاق‌ها است اما با نگاه به نمودارهای تولید شده توسط معمار، تعداد بی‌نظیری از جزئیات سطح‌پایین مانند این‌که هر پریز، چراغ روشنایی و نور در کجا قرار خواهد گرفت به چشم می‌خورد. همچنین مشخص شده‌است که کدام کلید کدام چراغ را کنترل می‌کند، شومینه در کجا قرار دارد و محل قرارگیری آبگرمکن و پمپ نیز دیده می‌شود، به تفصیل نحوهٔ ساخت دیوارها، سقف‌ها و پایه‌ها دیده می‌شود و به‌طور خلاصه، تمام جزئیات کمی که تمام تصمیم‌های سطح بالا را پشتیبانی می‌کنند در آن دیده می‌شود. همچنین دیده می‌شود که آن جزئیات کم‌سطح و تصمیم‌های سطح بالا بخشی از کل طراحی خانه است. چنین وضعیتی در مورد طراحی نرم‌افزار هم وجود دارد. جزئیات سطح پایین و ساختار سطح بالا همه بخشی از یک کل هستند. آن‌ها مادهٔ پیوسته‌ای را تشکیل می‌دهند که شکل سامانه را مشخص می‌کند. هیچ‌یک از این دو بدون دیگری کامل نخواهد بود و هیچ خط واضحی آن‌ها را از هم جدا نمی‌کند. می‌توان به‌سادگی بیان کرد که یک زنجیرهٔ پیوسته‌ای از تصمیم‌ها از بالاترین تا پایین‌ترین سطح وجود دارد. [۳]

۱-۳ نیازمندی‌های خاص معماری

هدف از تصمیم‌های معماری محقق شدن یا پاسخ دادن به مجموعهٔ مشخصی از نیازها است. این نیازها، نیازمندی‌های خاص معماری[13] نام دارد. برحسب تعریف نیازمندی‌های خاص معماری، زیرمجموعه‌ای از نیازمندی‌ها است که تأثیر قابل ملاحظه و قابل اندازه‌گیری‌ای روی معماری سامانه دارند. از این رو معمار باید درگرفتن تصمیم‌های معماری توجه زیادی را معطوف به برآورده شدن این نیازها کند. [۴]

مدت‌ها، از اصطلاح نیازمندی‌های خاص معماری کمتر استفاده می‌شد چون این نیازمندی‌ها را معادل نیازمندی‌های غیروظیفه‌ای[14] یا ویژگی‌های کیفی[15] می‌دانستند. به عبارت دیگر تصور این بود که نیازمندی‌ها به دو دستهٔ وظیفه‌ای و غیروظیفه‌ای محدود می‌شود. نیازمندی‌های وظیفه‌ای کارهایی است که انتظار می‌رود سامانه انجام دهد و نیازمندی‌های غیروظیفه‌ای ویژگی‌های کیفی مورد نظر برای سامانه است ولی با گذشت زمان در تحقیقات تجربی دیده شد که بخشی از

[11] trade-offs
[12] quality attribute
[13] architecturally significant requirements
[14] non-functional requirements
[15] quality attribute



شکل ۱: جایگاه معماری نرم‌افزار در درخت موضوعی مهندسی نرم‌افزار

نیازمندی‌ها هستند که غیروظیفه‌ای هم نیستند و تأثیر قابل ملاحظه‌ای روی معماری دارند مانند محدودیت‌ها و استانداردها و قوانین و حتی برخی نیازهای وظیفه‌ای. از این رو برای اشاره به نیازهایی که مرتبط به معماری هستند از این اصطلاح استفاده شد. [۴]

شناخت درست این نیازها گام اول در اتخاذ تصمیم‌های معماری است و برای ارزیابی تصمیم‌های اتخاذ شده نیز باید میزان برآورده شدن این نیازها مورد بررسی قرار گیرد. این نیازها چند شاخصهٔ اصلی دارند:

- معمولاً تعریف و بیان آن‌ها سخت است و خیلی مبهم مطرح می‌شوند. به همین خاطر ممکن است که در اوایل پروژه مورد غفلت قرار گیرند یا در سایر نیازها مخفی شده باشند.
- بیشتر ذهنی[۱۶] هستند و متغیرند و متناسب با موقعیت قابل تعریفند. [۱۷]
- تأثیر گسترده‌ای روی سامانه دارند.
- نقاط بده‌بستان را هدف قرار می‌دهند.
- سخت‌گیرانه هستند یعنی محدودکننده هستند و قابل مصالحه نیستند.
- فرضیات را باطل می‌کنند و رسیدن به آن‌ها دشوار است.
- مخاطرات[۱۸] فنی بالایی دارند و معمولاً ارزش تجاری بالایی دارند.
- معمولاً جزو دغدغه‌های یک ذینفع خاص و تأثیرگذاری هستند.

هنگامی که یک نیازمندی‌ای ویژگی‌های کیفی یک سامانه نرم‌افزاری را مشخص می‌کند یا محدودیت‌هایی برای سامانه نرم‌افزاری معلوم می‌کند یا محیطی که سامانه در آن اجرا می‌شود را مشخص می‌کند احتمالاً یک نیازمندی خاص معماری وجود دارد. [۴]

[16] subjective
[17] situational
[18] risks

۴-۱ الگو یا سبک معماری

برای سبک معماری نرم‌افزار که به عنوان «الگوی معماری» هم شناخته می‌شود چند تعریف مرسوم وجود دارد که در ادامه مورد بررسی قرار گرفته‌اند.

- الگو در مهندسی نرم‌افزار راه‌حلی ثابت‌شده و قابل استفادهٔ مجدد برای یک مشکل تکرارشونده در یک زمینهٔ مشخص است؛ بنابراین الگوهای معماری راه‌حل‌هایی در زمینهٔ معماری نرم‌افزار هستند. پس برحسب این تعریف یک مشکل شناخته‌شده‌ای وجود دارد و به دفعات با آن مواجه می‌شویم. برای این مشکل شناخته شده راه‌حلی طراحی شده که به تجربه ثابت‌شده و آزمایش شده که راه‌حل موفقی است.
- سبک معماری مشابه معماری ساختمان، روشی خاص برای ساخت است که با ویژگی‌های برجستهٔ آن روش مشخص می‌شود.
- هر سبک معماری یعنی خانواده‌ای از سامانه‌ها از نظر سازماندهی ساختاریشان شامل تعاریف مولفه‌ها و اتصال‌دهنده‌ها و محدودیت‌هایی دربارهٔ نحوهٔ ترکیب آن‌ها. به تعبیر دیگر سبک معماری به معنی چگونگی ساختار دادن سامانه از طریق مجموعه‌ای از عناصر و اتصالات است.
- سبک معماری بسته‌های قابل استفادهٔ مجدد از تصمیم‌های طراحی و محدودیت‌های اعمال‌شده در یک معماری است.

در بیشتر موارد، یک سامانه نرم‌افزاری دامنهٔ کاربرد خاص خودش را دارد و یک سبک معماری یک انتزاع دید برای یک ساختار نرم‌افزار است که مستقل از دامنه است.

استفاده از یک سبک معماری مناسب می‌تواند تقسیم‌بندی نرم‌افزار[۱۹] را بهبود دهد؛ و استفادهٔ مجدد[۲۰] از طراحی نرم‌افزار را ترویج کند. هر

[19] software partitioning
[20] reuse

سبک معماری در محیطی خاصی ارائه می‌شود و می‌تواند مشکلات کلیدی خاصی را برطرف کند یا نیازهای خاصی را برآورده سازد. سبک معماری مناسب برای همهٔ سامانه‌ها وجود ندارد چون هر سامانه‌ای الزامات متفاوتی دارد از این رو انتخاب سبک مناسب برای سامانه سؤالی است که در هنگام طراحی برای معماران نرم‌افزار ایجاد می‌شود.

امروزه نرم‌افزارها پیچیدگی زیادی دارند و در نتیجه ممکن است در ساخت یک نرم‌افزار از فقط یک سبک یا الگوی معماری استفاده نشود بلکه برای هر زیرسامانه از یک نرم‌افزار بزرگ و پیچیده لازم باشد یک الگوی معماری جداگانه مورد استفاده قرار گیرد.

۱-۵ راهکنش‌های معماری

یک راهکنش یک تصمیم طراحی است که روی دستیابی به یک ویژگی کیفی تأثیر می‌گذارد. توجه هر راهکنش روی پاسخ به فقط یک ویژگی کیفی است. الگوهای معماری از راهکنش‌ها تشکیل شده‌اند. یک راهکنش می‌تواند نسخهٔ دقیق‌تری از راهکنش‌های دیگر باشد. هنگام انتخاب راهکنش‌ها صراحتاً به بده‌بستان‌ها میان ویژگی‌های کیفی توجه نمی‌شود و بده‌بستان‌ها باید به‌طور صریح و واضحی توسط طراح مورد توجه و کنترل قرار گیرند. از این منظر راهکنش‌های معماری با سبک‌ها یا الگوهای معماری فرق دارند. [۵]

راهکنش‌ها «بلوک‌های ساختاری» طراحی هستند و الگوهای معماری از آن‌ها ایجاد می‌شود. راهکنش‌ها مانند اتم غیرقابل تجزیه هستند و الگوها مانند مولکول هستند. بیشتر الگوها متشکل از چندین راهکنش متفاوت هستند و اگرچه همهٔ این راهکنش‌ها ممکن است در خدمت یک هدف مشترک مانند ارتقاء تغییرپذیری[۲۱] باشند، ولی بیشتر، ویژگی‌های کیفی مختلفی را افزایش می‌دهند. به عنوان مثال، ممکن است یک راهکنش برای ایمن‌تر کردن الگوی در دسترس‌پذیری[۲۲] مورد استفاده قرار گیرد، یا این که تأثیر منفی الگوی افزایش تغییرپذیری روی کارایی را کاهش دهد. [۵]

به عنوان مثال الگوی لایه‌ای که معمول‌ترین الگو در کل معماری نرم‌افزار است را در نظر بگیرید. تقریباً همهٔ سامانه‌ها از لایه‌بندی استفاده می‌کنند. الگوی لایه ای می‌تواند به‌عنوان ترکیبی از چندین راهکنش به دید آید. راهکنش‌های موجود در الگوی لایه‌ای شامل افزایش انسجام معنایی[۲۳]، خدمات مشترک انتزاعی[۲۴]، محصورسازی[۲۵]، محدود کردن مسیرهای ارتباطی[۲۶] و استفاده از یک واسطه[۲۷] است. [۵]

[21] modifiability
[22] availability
[23] increase semantic coherence
[24] abstract common services
[25] encapsulate
[26] restrict communication paths
[27] use an intermediary

۱-۶ سامانهٔ پشتیبان تصمیم

سامانه‌های پشتیبان تصمیم[۲۸] سامانه‌های تعاملی مبتنی بر رایانه هستند که به تصمیم‌گیران کمک می‌کنند که از داده‌ها و مدل‌ها برای حل مسائل عموماً غیرساخت‌یافته بهره ببرند. [۶]

برای تصمیم‌گیری فنون مختلفی به کار می‌رود. یک فن تصمیمگیری خوب باید تصمیم‌گیر را به سمت گزینه‌های بهتر و احتمالاً بهینه‌تر هدایت کند و در عین حال استفاده از آن راحت باشد. اتخاذ یک فن تصمیم‌گیری ضعیف می‌تواند دشواری‌های متعددی ایجاد کند که باعث انتخاب بدترین گزینه خواهد شد. [۷]

سامانه‌های پشتیبان تصمیم برای عملکرد صحیح به اطلاعات در سطوح مختلفی نیاز دارند و به هر میزان که اطلاعات فراهم شده برای این سامانه‌ها دقیق‌تر و کامل‌تر باشند عملکرد این سامانه‌ها نیز بهبود می‌یابد.

اطلاعات فراهم شده برای سامانه‌های پشتیبان تصمیم با دو رویکرد کلی بالا به پایین و پایین به بالا قابل گردآوری است. در روش پایین به بالا با استفاده از داده‌های تاریخی موجود در سامانه، الگوهای موجود در داده‌ها کشف می‌شوند و با استفاده از الگوهای کشف شده می‌توان گزینه‌های تصمیم‌گیری را به کاربران سامانه ارائه کرد. در رویکرد بالا به پایین که موضوع مورد بحث این پژوهش نیز است دانش مورد نیاز سامانه در قالب یک هستان‌شناسی به سامانه داده می‌شود و سامانه با بررسی داده‌های ورودی و استنتاج بر اساس قوانین موجود در هستان‌شناسی، گزینه‌های تصمیم‌گیری را به کاربران ارائه می‌دهد. [۸]

۱-۷ هستان‌شناسی

برحسب تعریف، هستان‌شناسی[۲۹] یک توصیف صریح و رسمی[۳۰] از مفاهیم که با نام کلاس هم شناخته می‌شوند و روابط میان آن‌ها است. کلاس‌ها دارای ویژگی‌ها هستند و ویژگی‌ها یک کلاس را توصیف می‌کنند. هستان‌شناسی‌ها در حوزه‌های مختلف کاربرد برای تسهیل درک مشترک از ساختارهای موجود میان اطلاعات آن دامنه و برای استفادهٔ مجدد از دانش آن دامنه مورد استفاده قرار می‌گیرند. هستان‌شناسی سازوکار قدرتمندی است که می‌تواند همین نقش را در دنیای معماری نرم‌افزار ایفا کند. یک هستان‌شناسی معماری واژگان مشترکی را ارائه می‌دهد که سطح دقت لازم برای تصمیم‌گیری‌های مؤثر در زمینهٔ معماری را ممکن می‌سازد.

هستان‌شناسی در مستندسازی نرم‌افزار می‌تواند بسیار مفید واقع شود؛ و اطلاعات بیشتری را نسبت به روش مبتنی‌بر فایل در اختیار ذینفعان معماری نرم‌افزار قرار دهد. به عنوان مثال روابط بین نیازمندی‌ها و تصمیم‌های معماری نرم‌افزار در هستان‌شناسی می‌تواند مشخص شود؛ و ذینفعان می‌توانند با مراجعه به هستان‌شناسی اطلاعات مرتبط بیشتری را دریافت کنند. [۹]

[28] Decision Support System (DSS)
[29] ontology
[30] formal

۲ بررسی کارهای پیشین

در مقاله [۱۰] رویکردی برای توسعهٔ نرم‌افزار، با تمرکز بر تصمیم‌های معماری معرفی شده‌است که از هستان‌شناسی استفاده می‌کند. هستان‌شناسی ارائه شده در این مقاله شامل چهار مولفهٔ اصلی است: دارایی‌های معماری، تصمیم‌های معماری، دغدغه‌های ذینفعان و یک نقشهٔ راه معماری. در مقاله [۱۱] سامانه توصیه‌گیری ارائه شده‌است که به‌طور خودکار عناصر معماری را در اسناد معماری تشخیص می‌دهد و حاشیه‌نویسی می‌کند. در مقاله [۱۲] ارزیابی معماری نرم‌افزار با رویکردی مبتنی بر هستان‌شناسی و استفادهٔ مجدد از دانش انجام شده‌است. در این پژوهش دو هستان‌شناسی مورد استفاده قرار گرفته‌است. این دو هستان‌شناسی روی نقش الگوهای معماری مبتنی بر ویژگی تمرکز دارند. در پژوهش [۱۳] تلاش شده‌است تا مفاهیم موجود در حوزهٔ معماری نرم‌افزار به عنوان فرادانش گردآوری گردد. این مفاهیم با روش‌های دستی و جستجو در کتب اصلی این حوزه و با استفاده از یک روش نیمه خودکار برای استخراج اطلاعات از ویکی‌پدیا گردآوری شده‌اند.

متدولوژی ارائه‌شده در پژوهش [۱۴] برای تسهیل نمونه‌گیری از معماری مرجع هستان‌شناسی استفاده می‌کند. در پژوهش [۱۵] از هستان‌شناسی به منظور مستندسازی جنبه‌های رفتاری نرم‌افزار و رسمی‌سازی فرایند توسعهٔ معماری و امکان کنترل خودکار جامعیت استفاده شده‌است.

در پژوهش [۹] یک هستان‌شناسی برای معماری نرم‌افزار ایجاد شده‌است. این هستان‌شناسی اطلاعات مرتبط با الگوهای معماری را که در منابع مختلف با روش‌های متفاوتی دسته‌بندی شده‌اند را به شیوه‌ای یکپارچه ارائه می‌کند و با اجرای پرسمان روی هستان‌شناسی ایجاد شده می‌توان به اطلاعات هر الگو و همچنین اطلاعات جانبی مانند صفحهٔ DBpedia آن‌ها دست یافت.

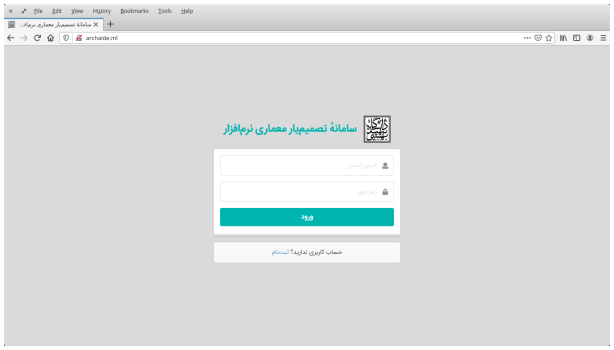
۳ سامانهٔ پشتیبان تصمیم پیشنهادشده

کد سامانهٔ پشتیبان تصمیم پیشنهادشده در این پژوهش در یک مخزن گیت در نشانی زیر در دسترس است:

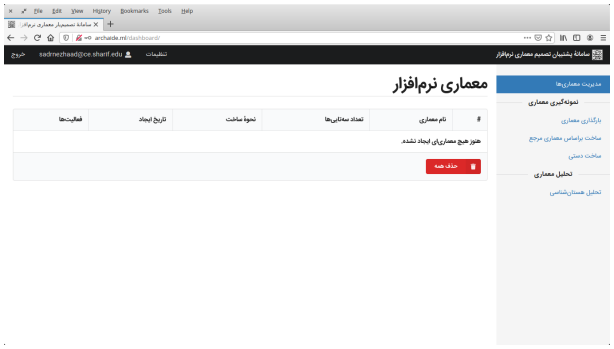
gitlab.com/smmsadrnezh/software_architect_aide

همچنین این برنامه در نشانی اینترنتی archaide.ml مستقر شده‌است و درصورت تمایل به کار با این سامانه باید ابتدا وارد صفحهٔ ثبت‌نام شده و یک حساب کاربری در آن ایجاد کنید. سپس می‌توانید از طریق صفحهٔ ورود به سامانه مطابق تصویر ۲ وارد حساب کاربری خود شوید.

پس از ورود به سامانه وارد داشبورد مدیریت معماری‌ها در تصویر ۳ می‌شوید که در ابتدا چون هنوز هیچ معماری‌ای ایجاد نکردید خالی است.



شکل ۲: صفحهٔ اول ورود به سامانه



شکل ۳: داشبورد کاربر پس از ورود به سامانه

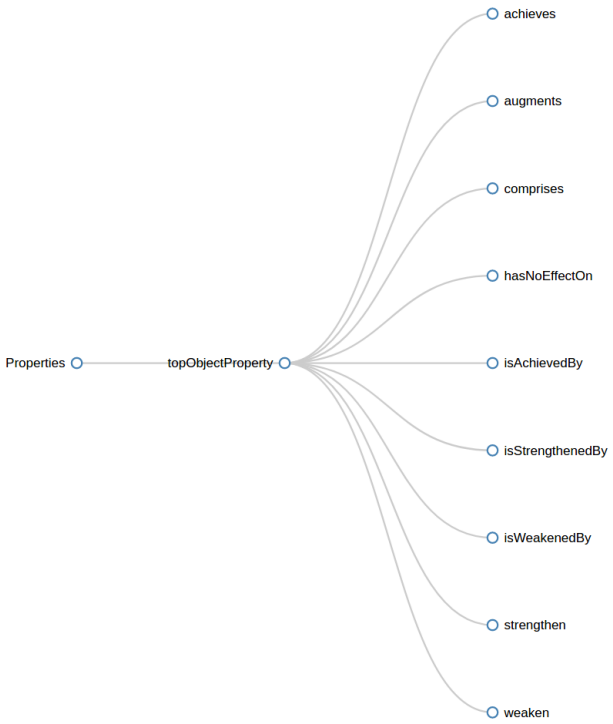
۳-۱ هستان‌شناسی معماری نرم‌افزار

همان‌طور که در بخش ۲-۱ بیان شد هر معماری نرم‌افزار شامل مجموعه‌ای از تصمیم‌ها است که هر کدام از این تصمیم‌ها برای نائل شدن یک یا تعدادی از دغدغه‌ها گرفته می‌شوند؛ بنابراین برای ساخت هستان‌شناسی معماری نرم‌افزار دو مفهوم اصلی «تصمیم» و «دغدغه» وجود دارند. این دغدغه‌ها همان نیازمندی‌های خاص معماری هستند که در بخش ۳-۱ معرفی شدند.

۳-۱-۱ روابط غیرطبقه‌ای

رابطهٔ بین تصمیم و دغدغه در معماری، رابطهٔ «نائل شدن» نام دارد. درواقع هر دغدغهٔ خاص معماری توسط یک یا تعدادی تصمیم نائل می‌شود که این رابطه در هستان‌شناسی با برچسب is achieved by مشخص شده‌است و هر تصمیم معماری یک یا تعدادی دغدغه را نائل می‌کند که این رابطه برعکس رابطهٔ قبل بوده و با برچسب achieves مشخص شده‌است. به این ترتیب یک رابطهٔ چندبه‌چند بین تصمیم و دغدغه وجود دارد. به بیان دیگر هدف از اتخاذ هر تصمیم، نائل شدن چند دغدغه است و هر دغدغه، توسط چند تصمیم نائل می‌شود.

رابطهٔ بین الگوهای معماری و راهکنش‌های معماری رابطهٔ شمول است به این ترتیب هر الگوی معماری شامل تعدادی راهکنش می‌شود و با برچسب comprises مشخص می‌شود و هر راهکنش در چند الگو مورد استفاده قرار می‌گیرد که با برچسب comprised by مشخص



شکل ۴: روابط غیرطبقه‌ای به‌کار رفته در هستان‌شناسی طراحی‌شده

می‌شود و چندی رابطهٔ میان الگوها و راهکنش‌ها چندبه‌چند است.

هم رابطهٔ «نائل می‌شود توسط» میان دغدغه‌ها و تصمیم‌ها و هم رابطهٔ «شامل می‌شود» میان الگوها و راهکنش‌ها خاصیت تراگذری[۳۱] دارند. به عبارت دیگر اگر شی الف توسط شی ب نائل و شی ب توسط شی پ نائل شود می‌توان گفت که شی الف توسط شی پ نائل می‌شود یا اگر شی الف شامل شی ب باشد و شی ب شامل شی پ باشد می‌توان گفت که شی الف شامل شی پ است.

روابط غیرطبقه‌ای مذکور در شکل ۴ نمایش داده شده‌است.

۳-۱-۲ روابط طبقه‌ای

در یک نگاه ساده، برای هر دغدغهٔ خاص معماری سه حالت می‌توان مطرح کرد. هر دغدغه، کاهش یک مخاطره[۳۲]، رفع یک نیاز حرفه[۳۳] یا دستیابی به یک ویژگی کیفی[۳۴] است. همچنین تصمیم‌ها در هستان‌شناسی مورد استفاده برای معماری نرم‌افزار از دو نوع انتخاب راهکنش‌های معماری[۳۵] و انتخاب الگوهای معماری[۳۶] هستند. به عبارت دیگر هر راهکنش با هدف کاهش تعدادی مخاطره، دستیابی به تعدادی ویژگی کیفی یا رفع برخی نیازمندی‌های حرفه انتخاب می‌شوند.

روابط طبقه‌ای یا ارث‌بری مذکور در شکل ۵ نمایش داده شده‌است.

[31] transitive
[32] risk mitigation
[33] business requirement
[34] quality attribute
[35] architecture tactic
[36] architecture pattern

۳-۲ مستندسازی معماری نرم‌افزار

معمار نرم‌افزار از راهکنش‌های متفاوتی برای پوشش نیازمندی‌های پروژه استفاده می‌کند اما تلاش برای توصیف[۳۷] دانشی که در ذهن معمار وجود دارد مسئله‌ای مهم است. دانش معماری مستندشده[۳۸] نیست. توصیف و ثبت این دانش لازمهٔ استفادهٔ مجدد[۳۹] از دانش طراحی است. تاکنون قالب‌های گوناگونی برای ساخت این مستندات یا توصیف‌های نسبتاً رسمی پیشنهاد شده‌است. برای نمونه، در یکی از این قالب‌ها، هر تصمیم معماری با موارد زیر بازنمایی می‌شود: [۱۶]

- عنوانی کوتاه برای مسئلهٔ حل‌شده و راه‌حل آن
- وضعیت تصمیم: پیشنهادشده، رد شده، تأیید شده، تاریخ گذشته و ...
- تصمیم‌گیرندگان: فهرستی از همهٔ افراد درگیر در تصمیم
- تاریخ: آخرین تاریخ به‌روزرسانی تصمیم
- داستان فنی: توصیف یا لینک مشکل[۴۰] ثبت شده
- جملهٔ توصیف‌کنندهٔ مشکل: در قالب دو یا سه جمله بدون هیچ قالب مشخص مشکل بیان شود
- فهرست دغدغه‌ها و انگیزه‌ها برای اخذ تصمیم
- گزینه‌هایی که مورد بررسی قرار گرفتند
- خروجی تصمیم اخذشده
- تبعات مثبت تصمیم: مثلاً بهبود ویژگی‌های کیفی یا ...
- تبعات منفی تصمیم: مثلاً از دست دادن ویژگی‌های کیفی یا ...
- نکات مثبت و منفی هر گزینه
- پیوندهای مرتبط

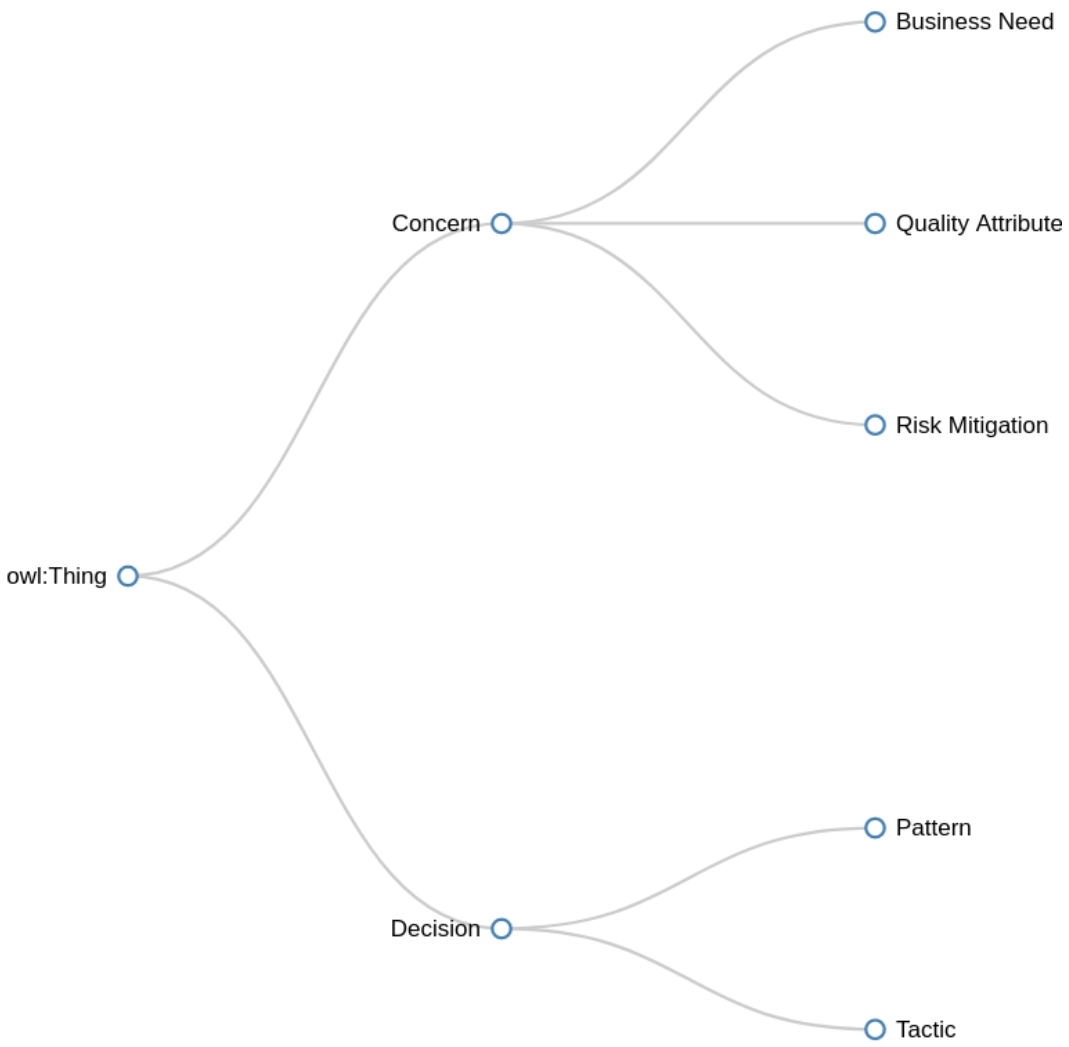
همچنین با ثبت شدن توجیه تصمیم‌ها، آن‌ها دیگر در طی زمان فراموش نمی‌شوند. فراموش شدن دلایل هر تصمیم اتفاقی است که در عمل زیاد رخ می‌دهد. مثلاً ممکن است یک معماری سال‌ها مورد استفاده قرار گیرد و بعد مدت‌ها کسی به خاطر نیاورد که چه‌شد که از ابتدا این معماری استفاده شد.

۳-۳ استفاده از هستان‌شناسی برای مستندسازی

نسخهٔ کامل هستان‌شناسی مورد استفاده برای معماری نرم‌افزار در شکل ۶ نمایش داده شده‌است.

یکی از راهکارهای پشتیبانی از تصمیم‌های معماری نرم‌افزار استفاده از هستان‌شناسی[۴۱] حوزهٔ معماری نرم‌افزار است. [۱۰] هستان‌شناسی‌هایی برای معماری نرم‌افزار وجود دارند که تمام مفاهیم این حوزه و روابط میان آن‌ها را مطابق با استانداردهای موجود برای بازنمایی معماری نرم‌افزار تعریف کرده‌اند.

[37] specification
[38] documented
[39] reuse
[40] issue
[41] ontology



شکل ۵: روابط طبقه‌ای میان مفاهیم معماری نرم‌افزار در هستان‌شناسی طراحی‌شده

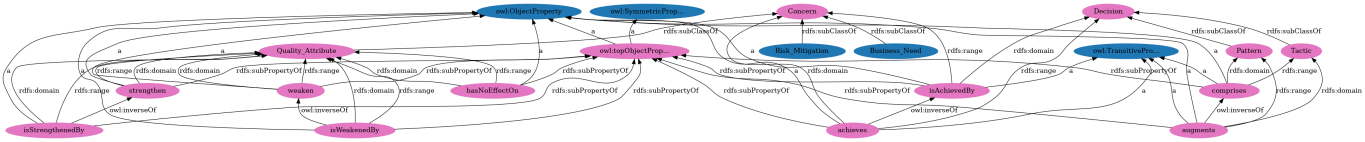
در هستان‌شناسی مورد استفاده در این پژوهش هر تصمیم معماری در راستای برآورده شدن تعدادی از دغدغه‌ها[۴۲] است. در هستان‌شناسی تصمیم‌ها و دغدغه‌ها نگه‌داری می‌شوند و با یک رابطه بین تصمیم‌ها و دغدغه‌ها مشخص می‌شود که هر تصمیم در راستای برآورده شدن کدام دغدغه‌ها است. در این روش هر معماری نرم‌افزار یک نمونه[۴۳] از هستان‌شناسی ساخته شده برای معماری نرم‌افزار است که فقط حاوی مفاهیم و کلاس‌ها و روابط کلی میان خود کلاس‌ها است و نمونه حاوی نمونه‌های ساخته شده از کلاس‌ها و روابط میان خود نمونه‌ها است. بنابراین برای ساخت معماری درواقع باید از یک هستان‌شناسی نمونه‌گیری انجام گیرد و نمونهٔ تهیه شده از آن هستان‌شناسی همان معماری نرم‌افزار ساخته‌شده خواهد بود.

همچنین در ادامه می‌توان از موتور استنتاج برای استنتاج[۴۴] و رسیدن به روابطی که در ابتدا در هستان‌شناسی ساخته شده مشهود نیستند استفاده کرد. [۱۰]

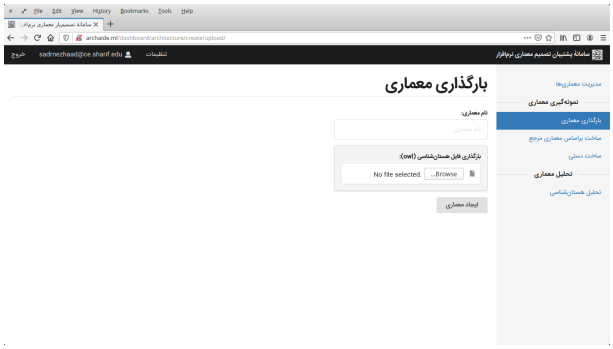
در سامانهٔ archaide سه روش برای ساخت یک معماری جدید مشخص شده‌است:

- **بارگذاری معماری:** در روش بارگذاری معماری کاربر یک فایل با پسوند owl که مربوط به هستان‌شناسی معماری ایجادشده در گذشته است را در سامانه بارگذاری می‌کند و می‌تواند در ادامه از سایر امکانات سامانه مانند بصری‌سازی، تکامل و تحلیل بده‌بستان برای آن استفاده کند. صفحهٔ مربوط به بارگذاری معماری در تصویر ۷ نمایش داده شده‌است.
- **ساخت براساس معماری مرجع**[۴۵]: در حالت ساخت براساس معماری مرجع کاربر در چند مرحله اطلاعات معماری مرجع که قبلا به سامانه داده‌شده را می‌بیند و می‌تواند دغدغه‌ها و راهکنش‌های مورد استفادهٔ خود و روابط میان آن‌ها را براساس معماری مرجع پیشنهادشده انتخاب کند. سپس از روی معماری مرجع ساخته

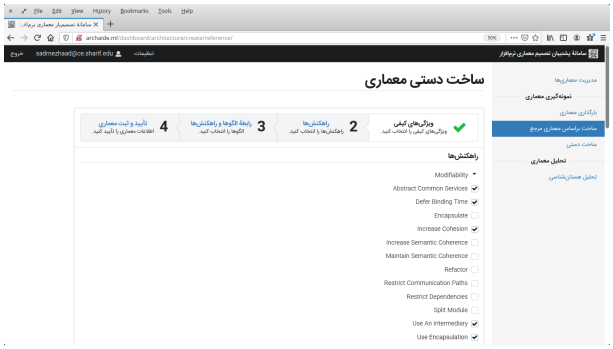
[42] concern
[43] instance
[44] inference
[45] reference architecture



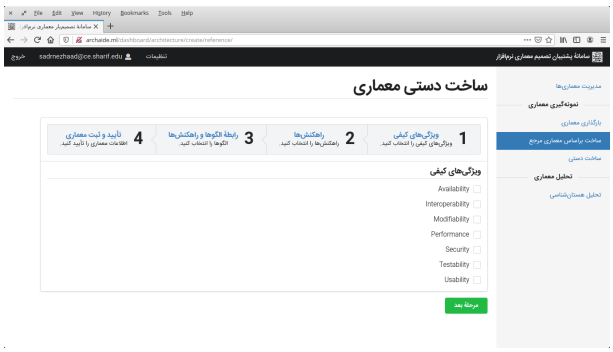
شکل ۶: هستان‌شناسی کامل معماری نرم‌افزار مورد استفاده



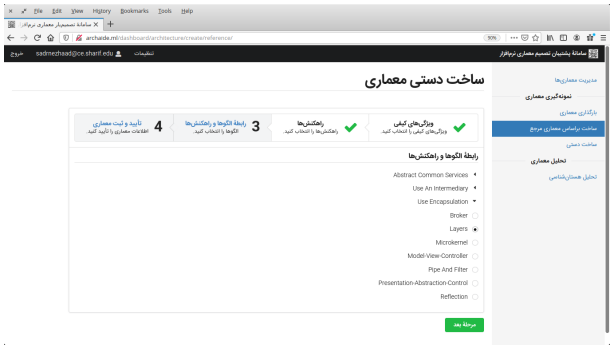
شکل ۷: ساخت معماری نرم‌افزار با بارگذاری هستان‌شناسی



شکل ۹: مرحلهٔ دوم ساخت معماری نرم‌افزار براساس معماری مرجع



شکل ۸: مرحلهٔ اول ساخت معماری نرم‌افزار براساس معماری مرجع



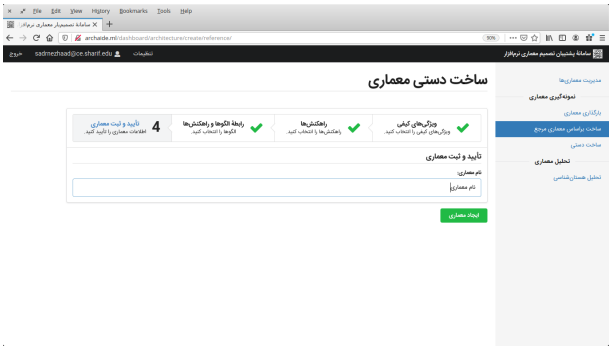
شکل ۱۰: مرحلهٔ سوم ساخت معماری نرم‌افزار براساس معماری مرجع

می‌شود. همانطور که در تصویر ۸ مشخص است، در مرحلهٔ اول این روش معمار فهرستی از صفات کیفی را مشاهده می‌کند. این صفحات کیفی از منابع معتبر حوزهٔ معماری نرم‌افزار مانند کتاب [۵] استخراج شده‌اند. در این مرحله معمار تعدادی از این صفات کیفی را به عنوان دغدغه‌ها معماری مورد نظر برای معماری در حال طراحی انتخاب می‌کند. در مرحلهٔ بعد کاربر فهرستی از راهکنش‌های مرتبط با هر ویژگی کیفی را مشاهده می‌کند و برای نائل شدن هر ویژگی کیفی تعدادی راهکنش را انتخاب می‌کند. مرحلهٔ دوم در تصویر ۹ نمایش داده شده‌است. در مرحلهٔ سوم فهرستی از الگوهای معماری را می‌بینند که حاوی راهکنش‌های انتخاب شده در مرحلهٔ قبل هستند و برای هر راهکنش می‌تواند یک الگو را انتخاب کند تا با استفاده از آن الگو در معماری راهکنش انتخاب شده نیز مورد استفاده قرار گیرد. تصویر مرحلهٔ سوم در تصویر ۱۰ نمایش داده شده‌است. در مرحلهٔ آخر هم فقط یک نام برای معماری طراحی شده انتخاب می‌کند.

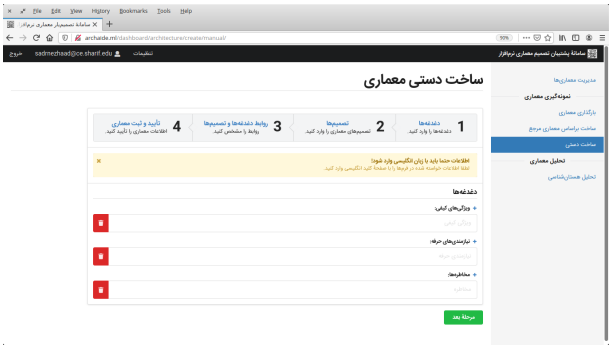
- **ساخت دستی:** در روش ساخت دستی برخلاف روش ساخت براساس معماری مرجع، معمار می‌داند که چه معماری‌ای را می‌خواهد طراحی کند و هدفش صرفا ثبت این دانش و مستندسازی آن است. در این بخش نیز معمار اطلاعات مربوط به دغدغه‌ها و تصمیم‌ها را این‌بار خودش وارد می‌کند و سامانه به ایشان هیچ کمکی در ورود و ثبت این داده‌ها نمی‌کند. مرحلهٔ اول ساخت دستی در شکل ۱۲ ملاحظه می‌شود.

۴-۳ تحلیل بده‌بستان میان دغدغه‌ها

استفاده از هر راهکنش تبعاتی دارد و دغدغه‌های جدیدی ایجاد می‌کند که با ویژگی‌های کیفی نرم‌افزار ممکن است در تضاد باشد. در این شرایط معمار نرم‌افزار باید بین دغدغه‌های ایجاد شده مواردی که اهمیت بیشتری دارد را انتخاب کند و با اتخاذ یک راهکنش جدید آن را رفع



شکل ۱۱: مرحلهٔ چهارم ساخت معماری نرم‌افزار براساس معماری مرجع



شکل ۱۲: مرحلهٔ اول ساخت دستی معماری نرم‌افزار

کند. راهکنش جدید انتخاب شده نیز به نوبه خود باعث دغدغه‌های جدیدی می‌شود و این فرایند تا جایی ادامه پیدا می‌کند که دغدغه‌های ایجاد شده قابل چشم‌پوشی باشند. برای این منظور ایدهٔ اولیه این است که هستان‌شناسی‌ای ساخته شود که همهٔ الگوها و راهکنش‌های معماری نرم‌افزار را دربرگرفته و دغدغه‌های ایجاد شده توسط هر راهکنش را در بر بگیرد. همچنین راهکنش‌ها باید مشکلی که حل می‌کنند را نیز شرح داده و نکات مثبت و منفی هر راهکنش مشخص باشد. حال با توجه به اولویت‌بندی‌ای که معمار به سامانه می‌دهد باید راهکنش‌ها را مرحله به مرحله به معمار پیشنهاد دهد و در صورت انتخاب هر راهکنش توسط معمار دغدغه‌های مرتبط را نمایش دهد و راهکنش‌های موجود برای حل این دغدغه‌ها را نیز نمایش دهد. در نهایت از ترکیب این راهکنش‌ها با هم معماری مطلوب و بهینه به دست می‌آید.

۳-۵ تعامل‌پذیری میان ذینفعان درگیر در طراحی معماری

در پروژه‌های پیچیده، طراحان مختلف منافع یا دغدغه‌های متفاوتی دارند و به همین خاطر دیدهای متفاوتی نسبت به سامانه دارند و در نتیجه از واژگان متفاوتی استفاده می‌کنند. مثلاً ممکن است برای تعریف یک صفت کیفی همیشه از کلمهٔ کارایی[۴۶] استفاده کنند درحالیکه این کلمه برای آن‌ها معنای مشخصی مانند تأخیر در بدترین حالت[۴۷] را داشته باشد ولی برای یک ذینفع دیگر معنای دیگری داشته باشد. مثلاً یک مدیر سامانه کارایی را به معنای استفادهٔ بهینه از منابع سامانه می‌داند و یک کاربر نهایی ممکن است کارایی را معادل زمانی که طول می‌کشد تا سامانه را یاد بگیرد بداند. این دشواری‌ها در تفسیر ممکن است باعث انتخاب یک گزینهٔ نادرست شود که این خود باعث عدم رضایت مشتری و دوباره‌کاری برای ارضای نیازهای واقعی ذینفعان می‌شود. [۷]

مسئلهٔ هستان‌شناسی‌ها نیز بحث تعامل‌پذیری است. استفاده از یک هستان‌شناسی مشترک میان ذینفعان می‌تواند باعث کاهش مشکلات این‌چنین شود.

۳-۶ بررسی تأثیر تغییر در تصمیم‌های معماری روی دغدغه‌ها در جریان تکامل معماری

با در اختیار داشتن هستان‌شناسی معرفی شده در بخش‌های پیشین، در جریان تکامل معماری هر تصمیمی که تغییر کرد به معمار نشان داده می‌شود که این دغدغه‌ها با آن تصمیم مرتبط بوده و همچنین نشان داده می‌شود که کدام تصمیم‌های دیگر هم با دغدغه‌های این تصمیم تغییر یافته مرتبط بوده‌اند. به این ترتیب یک تعداد گروه تصمیم داریم که هر گروه تصمیم دغدغه‌های مشترکی دارند و درصورت تغییر هر کدام از آن تصمیم‌ها باید به سایر تصمیم‌های آن گروه توجه کرد. [۱۰]

همچنین در هستان‌شناسی مورد استفاده راهکنش‌ها و ویژگی‌های کیفی و تأثیر هر فن بر هر ویژگی کیفی با یک رابطه مشخص می‌شود. با توجه به نیازمندی‌های پروژه، معمار نرم‌افزار از راهکنش‌ها متفاوتی برای پوشش نیازمندی‌ها استفاده می‌کند. استفاده از هر فن تبعاتی دارد و دغدغه‌هایی با ویژگی‌های کیفی نرم‌افزار مرتبط است. در این شرایط معمار نرم‌افزار باید بین دغدغه‌های ایجاد شده مواردی که اهمیت بیشتری دارد را انتخاب کند و با اتخاذ یک فن جدید آن را رفع کند. فن جدید انتخاب‌شده نیز به نوبهٔ خود باعث دغدغه‌های جدیدی می‌شود و این فرایند تا جایی ادامه پیدا می‌کند که دغدغه‌های ایجاد شده قابل چشم‌پوشی باشند. [۱۲]

۴ نتیجه‌گیری

طراحی معماری نرم‌افزار کاری بسیار ذهنی[۴۸] و متأثر از تجربهٔ معمار و کیفیت مهندسی نیازمندی‌ها است. دانش معماری معمولاً دانش ضمنی معماران یا سایر ذینفعان محسوب می‌شود [۳] و به‌طور مستند نیست و چون ثبت نمی‌شود سرانجام از بین می‌رود. [۶] استفاده از هستان‌شناسی و سامانهٔ تصمیم‌یار ایجاد شده در این پژوهش می‌تواند به کاهش این مشکلات کمک کند.

برای این منظور ایدهٔ این پژوهش این است که هستان‌شناسی‌ای ساخته شود که تمامی الگوها و راهکنش‌های معماری نرم‌افزار را دربرگرفته و دغدغه‌های ایجاد شده توسط هر فن را در بر بگیرد. همچنین راهکنش‌ها باید مشکلی که حل می‌کنند را شرح داده و نکات مثبت و منفی هر فن مشخص باشد. حال با توجه به اولویت‌بندی‌ای که معمار به سامانهٔ

[46] performance
[47] worst-case latency
[48] subjective

پشتیبان تصمیم می‌دهد سامانه باید راهکنش‌ها را مرحله‌به‌مرحله به معمار پیشنهاد دهد و در صورت انتخاب هر فن توسط معمار دغدغه‌های مرتبط را نمایش دهد و راهکنش‌ها موجود برای حل این دغدغه‌ها را نمایش دهد.

سپاس‌گزاری

بدینوسیله از سرکار خانم دکتر شمس‌فرد که با ارائهٔ درس مهندسی هستان‌شناسی زمینهٔ انجام این پژوهش را فراهم نمودند و با بازبینی علمی و متنی مقاله ما را در انجام این تحقیق یاری نمودند صمیمانه تشکر می‌کنیم.

مراجع

[1] P. Bourque, R. E. Fairley, *et al.* *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

[2] M. Fowler, "Software Architecture," https://martinfowler.com/architecture/. [Online; accessed 30-May-2020].

[3] R. C. Martin. *Clean architecture: a craftsman's guide to software structure and design*. Prentice Hall Press, 2017.

[4] L. Chen, M. A. Babar, and B. Nuseibeh, "Characterizing architecturally significant requirements," *IEEE software*, vol.30, no.2, pp.38–45, 2012.

[5] L. Bass, P. Clements, and R. Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2012.

[6] S. Moaven, J. Habibi, H. Ahmadi, and A. Kamandi, "A decision support system for software architecture-style selection," in *2008 Sixth International Conference on Software Engineering Research, Management and Applications*, pp.213–220, IEEE, 2008.

[7] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design: A comparative survey," *ACM Computing Surveys (CSUR)*, vol.43, no.4, pp.1–28, 2011.

[8] V. L. Sauter. *Decision support systems for business intelligence*. John Wiley & Sons, 2014.

[9] W. Pinnoo, *Development of an ontology for the problem space of architectural design*. Ph.D. thesis, Ghent University, 2016.

[10] A. Akerman and J. Tyree, "Using ontology to support development of software architectures," *IBM Systems Journal*, vol.45, no.4, pp.813–825, 2006.

[11] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, M. Hassel, and F. Matthes, "An ontology-based approach for software architecture recommendations," 2017.

[12] A. Erfanian and F. Shams Aliee, "An ontology-driven software architecture evaluation method," in *Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge*, pp.79–86, 2008.

[13] M. S. Ramaiah, T. Prabhakar, D. Rambabu, *et al.*, "Archvoc–towards an ontology for software architecture," in *Second Workshop on Sharing and Reusing Architectural Knowledge-Architecture, Rationale, and Design Intent (SHARK/ADI'07: ICSE Workshops 2007)*, pp.5–5, IEEE, 2007.

[14] Z. J. F. Al-Bayati *et al.*, *Coupling ontology with reference architectures to facilitate the instantiation process of software system architectures*. Ph.D. thesis, University of Salford, 2019.

[۱۵] عرفانیان، آیدا و کریم پوردرو، نیما، "کاربرد هستان‌شناسی درجامعیت مستندات معماری نرم‌افزار،" در *دومین همایش فناوری اطلاعات، حال، آینده*، دانشگاه آزاد اسلامی واحد مشهد، ۱۳۹۰.

[16] A. D. Records, "Markdown Architectural Decision Records," https://github.com/adr/madr. [Online; accessed 30-May-2020].