چگونگی شناسایی نیازمندی‌ها برای ایجاد نرم‌افزار خودانطباق در شرایط عدم اطمینان نیازمندی

رایحه معین‌فر، احمد عبداله‌زاده بارفروش و سیدمهدی تشکری هاشمی

چکیده: یکی از چالش‌های اساسی در تولید سیستم‌های نرم‌افزاری، به روز رسانی نیازمندی‌ها در مرحله تولید و اجرا است که می‌تواند ناشی از عدم اطمینان از فهم و خواسته ذی‌نفعان باشد. عدم اطمینان در نیازمندی، لزوم تولید یک معماری انعطاف‌پذیر و قابل انطباق جهت مدیریت‌کردن ریسک سیستم در مرحله اجرا را ایجاد می‌کند. مدل‌کردن عدم اطمینان در فرایند تولید نرم‌افزار و انطباق معماری نرم‌افزار با تغییر نیازمندی‌ها در زمان اجرا به صورت خودکار از جمله راه حل‌هایی هستند که در این زمینه مطرح می‌شوند. جهت پیاده‌سازی و اجرایی‌نمودن سنجش و رفع عدم اطمینان نیازمندی در مراحل تولید و اجرا از طریق مدل‌سازی و خودکارسازی آن، نیازمند کمی و محاسباتی‌نمودن نیازمندی هستیم. این مقاله ضمن تبیین منابع عدم اطمینان، به کمی‌کردن نیازمندی‌ها و ویژگی‌های توصیفی و کیفی می‌پردازد. به این ترتیب، تصمیم‌گیری در هر مرحله از فرایند تولید نرم‌افزار، مبتنی بر محاسبات عددی می‌باشد که راهی برای خودکارسازی تولید نرم‌افزار است.

کلیدواژه: **سیستم‌های خودانطباق، عدم اطمینان در نیازمندی، معماری نرم‌افزار، مهندسی نیازمندی.**

۱- مقدمه

پس از تولید یک محصول نرم‌افزاری، تغییر مسئله و به دنبال آن تغییر راه حل، حتی در مطلوب‌ترین تکرار از فرایند توسعه، یک فعالیت بسیار زمان‌بر است. پیچیدگی سیستم‌های نرم‌افزاری امروزی سبب می‌شود که منابع متعددی سبب ایجاد عدم اطمینان در مرحله‌های مختلف از تولید نرم‌افزار و به ویژه در نیازمندی‌ها شوند [۱]. گاهی عدم اطمینان از ناشناخته‌بودن ابعاد مسئله برای ذی‌نفعان ناشی می‌شود. از سوی دیگر باید با نیازهای محیط متغیر پیرامون که به سرعت و به صورت پویا در حال تحول است همگام شد. چنین تغییراتی گاه منجر به ایجاد تغییرات در ساختار ارتباطی اجزای یک سیستم طراحی‌شده نمی‌شود اما گاه مشاهده می‌شود که برای رسیدن به رضایت ذی‌نفعان، راهی به جز تطبیق معماری با وضعیت جدید وجود ندارد. علاوه بر شناخت نادرست و یا تغییر در نیازمندی‌ها، نیازهایی که بر پایه حس‌گرهایی با احتمال گزارش اطلاعات دارای خطا هستند و یا نیازهایی که مبتنی بر فرضیات می‌باشند از منابع ایجاد عدم اطمینان به شمار می‌روند [۲].

یکی از راه‌های قدیمی برای تصحیح نرم‌افزار، جستجوی ریشه مشکلات در کد اجرایی است. به این ترتیب پس از یافتن مشکل، آن را به صورت معکوس به مراحل پیشین چرخه تولید، انتشار می‌دادند. اما در روش‌های جدید، تحلیل و اصلاح از مراحل نخستین آغاز می‌شود و با استفاده از معماری مبتنی بر مدل، تحلیل و اصلاح بر روی مدل‌هایی انجام می‌شود که در سطوح مختلفی از انتزاع قرار دارند. به این ترتیب در زمان اجرا، امکان جستجو، تحلیل و دستکاری مدل‌ها وجود دارد تا نحوه تطابق با وضعیت نیازهای جدید مشخص شود. چنین امری سبب افزایش وضوح و کاهش عدم اطمینان در فرایند اعتبارسنجی و صحت‌سنجی معماری سیستم‌های نرم‌افزاری می‌شود.

اگر بخواهیم مسئله را با جزئیات بیشتری دنبال کنیم می‌توان گفت که لزوم تغییر در معماری ایجادشده به دو دلیل کلی ایجاد می‌شود. ابتدایی‌ترین دلیل برای تغییر معماری آن است که در فرایند ارزیابی معماری متوجه شویم که بعضی از نیازهایی که در مرحله تحلیل تشخیص داده شده‌اند برآورده نشده‌اند یا به کیفیت مطلوبی که برای آنها تعریف شده است نرسیده‌اند. این نتیجه ممکن است در مرحله پایانی فرایند تولید معماری و یا هر یک از مراحل میانی آن حاصل گردد. دلیل دیگری که سبب ضرورت تغییر معماری می‌شود عبارت است از این که در مرحله تحلیل، شناخت ما از نیازمندی‌های مسئله و یا ویژگی‌ها و محدودیت‌های محیط دچار نوعی عدم اطمینان باشد. عدم شناخت کامل یکی از نیازهای سیستم، منجر به عدم کیفیت در زمان اجرا می‌شود. همچنین در یک سیستم نرم‌افزاری بزرگ‌مقیاس، علاوه بر شناخت ناکامل نیازها، ممکن است بعضی از نیازها اصلاً شناخته نشوند. در نتیجه، مشکلات پیچیده‌تری در زمان اجرا پدید می‌آید که ناشی از عدم کیفیت معماری تولیدشده است. در این مقاله عنوان می‌شود که شناخت ناکامل و عدم اطمینان از نیازمندی‌ها و محدودیت‌های یک سیستم در واقع سبب می‌شود که اندازه‌گیری سیستم میسر نشود. بنابراین نیاز داریم که منابع عدم اطمینان تبیین شوند، تشخیص داده شوند و همچنین ویژگی‌های کیفی و توصیفی که در تحلیل و ارزیابی نیازمندی‌ها مؤثر هستند نیز کمی‌سازی شوند. به این ترتیب، سیستم (چه از بعد صورت مسئله و چه از بعد راه حل مسئله) اندازه‌گیری می‌شود.

به طور کلی می‌توان گفت عدم اطمینان سبب عدم انطباق نیازمندی‌های شناخته‌شده برای سیستم و نیازمندی‌های حقیقی سیستم در زمان اجرا می‌شود. در نتیجه در زمان اجرا به منظور ایجاد رضایت از خدمات سیستم باید سیستم با شرایط جدید انطباق داده شود. سیستم‌های نرم‌افزاری خودانطباق در راستای ایجاد رضایت مستمر از خدمات سیستم ایجاد شده‌اند [۳].

در این مقاله، بخش دوم به توضیح پیشینه پژوهشی و دسته‌بندی مطالعات انجام‌شده در زمینه مدیریت سیستم‌های خودانطباق می‌پردازد و شیوه‌های موجود برای حل مسئله و نقاط ضعف و قوت آنها را بررسی می‌کند. بخش سوم مسئله را به صورت دقیق تعریف می‌نماید و ویژگی‌ها

این مقاله در تاریخ ۱۴ فروردین ماه ۱۳۹۵ دریافت و در تاریخ ۷ آذر ماه ۱۳۹۶ بازنگری شد.

رایحه معین‌فر، دانشکده ریاضی و علوم کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، (email: mfar@aut.ac.ir).

احمد عبداله‌زاده بارفروش، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، (email: ahmadaku@aut.ac.ir).

سیدمهدی تشکری هاشمی، دانشکده ریاضی و علوم کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، (email: hashemi@aut.ac.ir).

شکل ۱: دسته‌بندی راهکارهای ایجاد خودانطباقی.

و مشخصات آن را بیان می‌کند. در بخش چهارم، راه حل کلی برای حل مسأله به صورت انتزاعی و ملزومات تبدیل آن به صورت یک روش کاربردی مورد بررسی قرار گرفته است. در بخش پنجم، پیاده‌سازی شیوه مطرح‌شده برای اندازه‌گیری نیازمندی‌ها به عنوان اولین گام در راستای خودکارسازی مدیریت نرم‌افزار خودانطباق انجام گرفته است. در بخش ششم روش بیان‌شده تحلیل و ارزیابی می‌شود. بخش‌های هفتم و هشتم به جمع‌بندی و معرفی منابع استفاده‌شده می‌پردازند.

۲- پیشینه پژوهشی

همان گونه که تیم پژوهشی IBM در سال ۲۰۰۳ [۴] ادعا کرده‌اند، برای تولید یک سیستم خودانطباق باید یک حلقه MAPE-K[1] تشکیل‌شده از نظارت، تحلیل، پردازش و اجرا که همگی با یک پایگاه دانش در ارتباط هستند ایجاد گردد. این ادعا با بیان‌های متفاوت و مشابه در کارهای سایر پژوهشگران نیز گزارش شده است [۱] و [۵] تا [۷]. با انجام تحقیقات کامل‌تر مشخص شد که علاوه بر حلقه نظارت و کنترل مباحث دیگری مانند بازنمایی و مدل‌سازی نیازمندی‌ها، فضای طراحی و تضمین سیستم خودانطباق نیز مطرح هستند [۸] و [۹]. مقالات مطرح در زمینه سیستم‌های خودانطباق را می‌توان به سه دسته کلی تقسیم کرد. دسته اول، مقالاتی که صرفاً به تشریح و تحلیل موضوع خودانطباق پرداخته‌اند و مطالب علمی و تئوریک پیرامون این موضوع مطرح نموده‌اند [۴]، [۸]، [۱۰] و [۱۱]. دسته دوم، مقالاتی که به پیشنهاد و پیاده‌سازی روشی برای ایجاد (بخشی از) سیستم‌های خودانطباق پرداخته‌اند [۵] تا [۷] و [۱۲]. دسته سوم، مقالاتی هستند که ضمن بررسی مقالات دسته اول و دوم، نقاط ضعف و قوت هر یک از روش‌های پیشنهادشده در مقالات دسته دوم را مشخص کرده‌اند و سپس، یک یا چند روش (و یا ابزار) تعریف‌شده برای تولید یک یا چند بخش از سیستم خودانطباق را به نحوی با هم ترکیب نموده‌اند که نقاط ضعف پوشش داده شود و از مزایای این روش (ابزار)ها استفاده شود [۱]، [۵]، [۱۳] و [۱۴]. از بعد زمانی می‌توان گفت که در سال‌های اول دهه کنونی، غالب مقالات متعلق به دسته نخست (تئوری و توصیفی) بوده‌اند و اکثر پژوهش‌های اخیر از دسته سوم هستند. با بررسی مقالات متعدد، آنچه در ایجاد یک سیستم خودانطباق باید توسط یک طراح مورد تصمیم قرار گیرد در نمودارهای شکل ۱ جمع‌بندی شده است. این نمودارها بیان‌کننده راه حل‌های شناخته‌شده برای خودانطباقی از ابعاد مختلف است.

همان گونه که در شکل ۱ نشان داده شده است برای هر یک از مسایل مطرح در زمینه سیستم‌های خودانطباق تا کنون راه حل‌هایی بیان شده است. برای توضیح بیشتر در جدول ۱، راه حل‌های بیان‌شده برای ایجاد سیستم‌های خودانطباق در مطالعات پیشین به دو صورت تفصیلی و اجمالی آمده است. در این بررسی، کلیدی‌ترین ایده‌هایی که در مقالات استفاده شده عنوان می‌شود. این ایده‌ها شامل بازنمایی فضای طراحی، تعریف معماری دولایه یا سه‌لایه، استفاده از معماری مبتنی بر مؤلفه، معماری مبتنی بر فرایند، معماری مبتنی بر عامل، افزودن پایگاه دانش و قابلیت یادگیری، استفاده از درخت تصمیم، استفاده از بیز ساده و اندازه‌گیری نیازمندی‌ها هستند. راه حل‌های عنوان‌شده به طور کلی مبتنی بر مهندسی نیازمندی‌ها یا مبتنی بر معماری هستند. بعضی از مقالات نیز به ترکیب هر دو روش پرداخته‌اند. جدول ۱ ضمن دسته‌بندی راه حل‌های بیان‌شده، به طور اجمالی نقاط ضعف و قوت هر مقاله را در قالب ویژگی‌های راه حل بیان کرده است.

به طور کلی، نقاط ضعف مشاهده‌شده در راه حل‌ها و یا چارچوب‌های مطرح در زمینه سیستم‌های خودانطباق را می‌توان در موارد زیر خلاصه کرد:

۱) سیستم تنها در وضعیتی به طور خودکار قادر به ایجاد انطباق است که آن وضعیت قبلاً توسط طراح سیستم حدس زده شده و پیش‌بینی شده باشد. به این ترتیب، فضای طراحی یک فضای شمارا و متناهی است و به بیان دیگر، پاسخ برای محیط قطعی ارائه شده است.

۲) نقش انسان در یک و یا چند گام از تولید سیستم خودانطباق و حتی در زمان اجرای سیستم خودانطباق بسیار پررنگ و ضروری است. هنگام تعیین و بازنمایی نیازمندی‌های سیستم (مرحله تحلیل)، در هنگام پیش‌بینی فضای طراحی و تعیین رویه‌های انطباق (مرحله طراحی) و حتی در بعضی از مقالات در هنگام انتخاب رویه انطباق در شرایطی که سیستم با چند انتخاب مواجه است و یا رویه پیش‌بینی شده، فاقد اثر مطلوب ارزیابی گردیده است (زمان اجرا) سیستم قادر به مدیریت خودکار نیست و حضور انسان برای مدیریت ضروری است.

1. Monitor, Analysis, Plan, Execute, Knowledge

جدول ۱: راه حل‌های مطالعات پیشین برای ایجاد سیستم‌های خودانطباق.

دسته- سال نشر- منبع	راه حل	ویژگی‌های راه حل
مبتنی بر معماری- ۲۰۰۷- [۱۸]	این مقاله به معرفی یک معماری سه‌لایه برای انطباق خودکار در زمان اجرا می‌پردازد. یک لایه برای مدیریت اهداف، یک لایه برای برنامه‌ریزی و اجرای برنامه و لایه سوم نیز حاوی مؤلفه‌هایی است که وظایف سیستم را انجام می‌دهند. ورودی این سیستم عبارت از اهداف سیستم و یک معماری طراحی‌شده برای برآورده‌کردن این اهداف است. در صورتی که سیستم در حال اجرا نتواند یکی از اهداف را به درستی برآورده کند، لایه برنامه‌ریزی یک برنامه برای اصلاح این وضعیت بر لایه سوم پیشنهاد و اجرا می‌کند. این برنامه‌ریزی شامل اعمالی همچون افزودن و یا حذف‌کردن یک مؤلفه، مقداردهی جدید به پارامترهای مؤلفه‌ها و یا تغییر در اتصالات بین مؤلفه‌های موجود در لایه سوم است. ایده کلی: معماری سه‌لایه (مبتنی بر هدف، مبتنی بر برنامه‌ریزی، مبتنی بر مؤلفه)	جستجوی پاسخ در فضای شمارا و محیط قطعی انجام می‌شود.
مبتنی بر معماری- ۲۰۰۹- [۷]	این مقاله به پیشنهاد راه حلی می‌پردازد که با تغییر در معماری، سیستم را بر اساس وضعیت و شرایط جدید انطباق می‌دهد. معماری‌های متعددی مدل شده‌اند که هر یک از آنها یک جنبه از اهداف را برآورده می‌کنند. برای هر معماری یک یا چند نقطه شکست تعریف شده که امکان ترکیب مؤلفه‌های دو معماری از نقاط شکست با یکدیگر میسر است. این سیستم مشابه کارهای پژوهشی بسیاری که در این زمینه انجام شده است بر حسب شرایط پیش‌آمده نخست بررسی می‌کند که کدام معماری مناسب‌تر است و سپس تغییر معماری فعلی را برای رسیدن به معماری پیشنهادی می‌سنجد. به این ترتیب که باید سیستم در حال اجرا تبدیل به سیستمی با یک معماری دیگر شود و نیز تصمیمات طراحی که منجر به تولید یک معماری نشدنی[1] است را بررسی و پیش از اعمال حذف می‌کند. ورودی این راه حل، تعدادی معماری و تعدادی شرایط از پیش تعریف شده و یک معماری به عنوان نقطه شروع است. خروجی نیز یک معماری انطباق داده شده با شرایط جدید است. ایده کلی: پیش‌بینی مدل معماری‌های مختلف برای استفاده در شرایط مقتضی	جستجوی پاسخ در فضای شمارا و محیط قطعی انجام می‌شود.
مبتنی بر نیازمندی- ۲۰۱۰- [۲]	این مقاله بیان می‌کند که عدم اطمینان در اثر متغیربودن محیط و یا فهم ناصحیح نیازمندی‌ها ایجاد می‌شود. در نتیجه، مهم‌ترین منبع عدم اطمینان، نیازمندی‌هایی هستند که بر پایه ادعاهایی که قابل اثبات نیستند بنا می‌شوند. این مقاله به بیان راهکاری برای مدل‌کردن عدم قطعیت در نیازمندی می‌پردازد و بررسی می‌کند که اگر ثابت شود یک ادعا اشتباه است در زمان اجرا سیستم چگونه رفتار می‌کند. اما نحوه اصلاح رفتار سیستم بررسی نشده است. ورودی این راه حل، نیازمندی‌ها هستند. خروجی، یک مدل از نیازمندی‌ها است که نقاط عدم اطمینان و انعطاف سیستم در آن مشخص شده است. ایده کلی: مدل‌کردن نیازمندی‌ها در شرایط عدم اطمینان	نقش انسان در مرحله‌های تحلیل و طراحی بسیار پررنگ است و سیستم قادر به مدیریت خودکار نیست.
مبتنی بر معماری- ۲۰۱۰- [۶]	در این مقاله چارچوبی برای خودانطباقی معرفی شده است. این چارچوب نخست اهدافی را تعریف می‌نماید که سیستم باید برای این منظور تلاش کند که در هنگام اجرا همواره این اهداف در بالاترین سطح خود برآورده شوند. برای بررسی میزان برآورده‌شدن هر هدف مقیاس سنجش[2] و یک تابع Utility تعریف شده است. هر گاه مقدار این تابع از میزان مورد انتظار قابل قبول کمتر باشد سیستم برای انطباق تحریک می‌شود. در این هنگام یک خاصیت[3] برای انطباق انتخاب می‌شود و فعالیت‌های تعریف‌شده در این خاصیت دنبال می‌شود. یک خاصیت یک دنباله از سرویس‌ها در سیستم مبتنی بر سرویس و یک دنباله از قانون‌ها در سیستم مبتنی بر قواعد است. پس از اعمال این انطباق، مقدار تابع Utility بر اساس مقیاس سنجش دوباره اندازه‌گیری، نرمال و بررسی می‌شوند. در صورتی که مقدار واقعی این تابع با مقداری که انتظار می‌رفت متفاوت باشد این تفاوت، پایگاه دانش را برای آن خاصیت تصحیح می‌کند. ورودی این راه حل اهداف سیستم خودانطباق، مقیاس سنجش و تابع Utility برای هر یک از اهداف، خاصیت‌ها و معماری سیستم است. خروجی این چارچوب در هر گام، تعیین خاصیتی است که باید به سیستم اعمال شود و نیز تصحیح ارزش خاصیت در پایگاه دانش، در صورت لزوم است. ایده کلی: استفاده از مقیاس سنجش تابع Utility و یادگیری برای انتخاب بهترین انطباق	جستجوی پاسخ در فضای شمارا و محیط قطعی انجام می‌شود.

1. Infeasible
2. Metric
3. Feature

مبتنی بر معماری- ۲۰۱۰- [۵]	این مقاله یک شیوه معماری مبتنی بر برنامه‌ریزی[1] را برای خودانطباقی تعریف می‌کند. در یک لایه مؤلفه‌های معماری و در یک لایه دیگر مؤلفه‌هایی برای نظارت، برنامه‌ریزی و اعمال برنامه بر مؤلفه‌های لایه نخست، قرار دارند. برنامه‌ریزی به معنای یک سلسله‌مراتب از عملیاتی است که هر وضعیت ممکن برای سیستم را به یکی از اهداف متصل می‌کند. ورودی این راه حل تمام فضای حالتی که است که پیش‌بینی می‌شود در این حالات، سیستم دچار مشکلی شود و نیاز به انطباق داشته باشد. برای رفع هر مشکل نیز راه حل‌هایی به عنوان ورودی در زمان طراحی تعریف شده‌اند. تمام وضعیت‌های[2] ممکن شمرده می‌شوند و عملیاتی[3] که سبب رفتن از یک وضعیت به وضعیت دیگر می‌شوند مشخص شده‌اند. یک معماری اولیه نیز برای مؤلفه‌ها تعریف شده است. خروجی این راه حل در هر گام از برخورد با مشکل، تعویض عملیات مؤلفه‌ها با عملیات جایگزین است به گونه‌ای که سیستم در وضعیت جدیدی قرار گیرد. ایده کلی: معماری دولایه مبتنی بر برنامه‌ریزی پیش‌بینی تمام وضعیت‌های ممکن و تعریف عملیات مورد نیاز برای هر وضعیت در مؤلفه‌ها	جستجوی پاسخ در فضای شمارا و محیط قطعی انجام می‌شود.
مبتنی بر نیازمندی- ۲۰۱۱- [۱۹]	این مقاله در ارتباط با ثبت فرضیات در مهندسی نیازمندی‌ها است و عدم اطمینان را ناشی از مشخص‌نبودن تأثیر و ارزش یک استراتژی در برآورده‌کردن اهداف و نیازمندی‌های سیستم می‌داند. این مقاله شیوه‌ای برای دربرگرفتن عدم اطمینان پیشنهاد می‌کند. این شیوه مجهز به منطقی برای انتخاب یک استراتژی از بین استراتژی‌هایی است که سبب تحقق اهداف سیستم می‌شوند. ورودی این راه حل اهداف و نیازمندی‌های سیستم هستند و خروجی آن درختی است که با انتشار تأثیر اشتباه‌بودن هر نیاز بر کل سیستم، استراتژی مطلوب برای رسیدن به اهداف را مشخص می‌نماید. ایده کلی: مدل‌کردن نیازمندی‌ها و استراتژی‌های برآورده‌کردن اهداف در شرایط عدم اطمینان	نقش انسان در مرحله‌های تحلیل و طراحی بسیار پررنگ است و سیستم قادر به مدیریت خودکار نیست.
مبتنی بر نیازمندی- ۲۰۱۲- [۲۰]	این مقاله با استفاده از زبان RELAX به نمایش نیازمندی‌های سیستم می‌پردازد و از منطق مرحله‌ای برای استدلال در ارتباط با احتمال برآورده‌شدن یا نشدن هر نیازمندی در زمان اجرا استفاده می‌کند. سپس بین نیازها و استراتژی‌های برآورده‌شدن آنها اتصالاتی را ایجاد و ارزش‌دهی می‌کند. در زمان اجرا استراتژی‌های دارای ارزش بالاتر برای برآورده‌ساختن نیازها انتخاب می‌شوند. ورودی این راه حل نیازمندی‌ها و استراتژی‌هایی برای ارضای نیازها هستند و خروجی آن بهترین استراتژی برای برآورده‌کردن هر هدف در زمان اجرا است. ایده کلی: مدل‌کردن نیازمندی‌ها در شرایط عدم اطمینان با زبان RELAX و ارزش‌دهی به استراتژی‌های ارضای نیازها	نقش انسان در مرحله‌های تحلیل و طراحی بسیار پررنگ است و سیستم قادر به مدیریت خودکار نیست. پیاده‌سازی این راه حل نیاز به تسلط بر زبان RELAX دارد.
مبتنی بر نیازمندی- ۲۰۱۳- [۱۷]	این مقاله با استفاده از شبکه‌های تصمیم‌گیری پویا (DDNs) مدل‌های هدف را ایجاد می‌کند تا به این وسیله از تصمیم‌گیری در انتخاب استراتژی بهینه در سیستم‌های خودانطباق پشتیبانی نماید. ورودی این راه حل نیازمندی‌های سیستم و استراتژی‌هایی (تصمیمات طراحی) هستند که برای ارضای نیازمندی‌ها تعریف می‌شوند. همچنین بر اساس نظر خبره، وزن‌ها و احتمالاتی به اتصال‌های رابط میان نیازمندی‌ها و استراتژی‌ها نسبت داده می‌شود. خروجی این راه حل، استراتژی(های) بهینه برای برآورده‌کردن نیازمندی‌ها در زمانی است که نیاز به انطباق تحریک‌شده است. ایده کلی: مدل‌کردن هدف‌ها بر اساس روابطی که میان هدف‌ها و نیز میان هدف‌ها و تصمیمات طراحی وجود دارد در شرایط عدم اطمینان	به ازای هر هدف یک DDN تعریف می‌کند و به این دلیل به فضای حافظه زیادی احتیاج دارد. مقداردهی اولیه برای وزن‌های اتصال توسط خبره یا ذی‌نفعان انجام می‌شود که نیازمند یک روش سیستماتیک است.
مبتنی بر نیازمندی- ۲۰۱۴- [۱۰]	این مقاله مفهوم سورپرایز را به عنوان میزان انحراف یک سیستم خودانطباق از رفتار نرمال تعریف کرده و از انحراف کولبک- لیبر برای محاسبه سورپرایز به صورت عددی استفاده می‌کند. این مقاله ادعا می‌کند که اگر مقدار محاسبه‌شده برای سورپرایز عددی غیر صفر باشد احتمال نیاز به تحریک انطباق نزدیک به ۱ است. ورودی این راه حل نیازمندی‌های سیستم و استراتژی‌هایی (تصمیمات طراحی) هستند که برای ارضای نیازمندی‌ها تعریف می‌شوند. همچنین تخمین تأثیر شواهد بر هدف‌های غیر ایستا توسط خبره تعیین می‌شود. خروجی این راه حل، عدد مربوط به سورپرایز در بازه‌های زمانی مختلف است.	تحلیل‌های عددی بر اساس ارزش اختصاص داده شده به استراتژی‌های مختلف در شرایط پیش‌بینی شده متفاوتی انجام می‌شود و توسط خبره تعیین می‌شوند. تعیین این مقادیر نیازمند یک روش سیستماتیک است.

1. Plan-Based Architecture
2. State
3. Action

مبتنی بر معماری و مبتنی بر نیازمندی- ۲۰۱۴- [۱]	این مقاله روش‌های مبتنی بر نیازمندی و روش‌های مبتنی بر معماری را با یکدیگر ترکیب کرده و روشی را برای خودانطباقی پیشنهاد داده است. در این روش، اساس کار بر مبنای یک درخت تصمیم است که گره‌های آن در پایین‌ترین سطح که برگ‌های درخت هستند تصمیمات طراحی معماری است و در بالاترین سطح (سطح بعد از ریشه) اهداف است. هر هدف مسیری را تا یک یا چند برگ طی می‌کند به این معنا که یک یا چند طراحی برای برآورده‌شدن هر هدف، امکان‌پذیر است. اتصالات درخت مقداردهی می‌شوند و این مقادیر نشان‌دهنده این است که این گره فرزند چه قدر در برآورده‌شدن این هدف تأثیر مثبت یا منفی دارد. هنگام تغییر شرایط و نیازمندی‌های جدید، این درخت بررسی می‌شود و یک مجموعه جدید از تصمیمات طراحی برای برآورده‌شدن هدف جدید انتخاب می‌شود. به عنوان مثال به جای استفاده از طراحی سری از موازی استفاده می‌شود. ورودی این راه حل، درخت تصمیمی است که توسط معماری سیستم در زمان طراحی تولید شده و خروجی آن، مجموعه‌ای از تصمیمات طراحی است که منجر به بازتعریف معماری سیستم به منظور انطباق با شرایط جدید می‌شود. ایده کلی: استفاده از درخت تصمیم، وزن‌دهی به اتصالات درخت بر اساس تأثیر تصمیمات طراحی در برآورده‌شدن/ نشدن هر هدف	جستجوی پاسخ در فضای شمارا و محیط قطعی انجام گرفته است. یک ساختار ساده برای معماری در نظر گرفته شده است.
مبتنی بر معماری- ۲۰۱۵- [۱۴]	این مقاله با ترکیب مدل‌هایی که برای فرمال‌کردن نیازمندی‌ها و ایجاد ارتباط سلسله‌مراتبی بین نیازمندی‌ها، ریلکس‌کردن بعضی نیازها و تعریف هدف از روی نیازمندی‌ها تعریف شده‌اند، سعی می‌کند تا نیازهای سیستمی که بر عدم قطعیت بنا شده را دسته‌بندی و تا حد ممکن دقیق‌تر کند. اما درباره نحوه انطباق ساختار سیستم در زمان اجرا صحبت نکرده است. ورودی این راه حل نیازمندی‌های اولیه و خروجی آن مدل‌هایی است که به ساختاردهی به نیازها می‌پردازند. ایده کلی: دسته‌بندی و ساختاردهی نیازمندی‌ها	نقش انسان در مرحله‌های تحلیل و طراحی بسیار پررنگ است و سیستم قادر به مدیریت خودکار نیست.
مبتنی بر نیازمندی- ۲۰۱۵- [۱۳]	این مقاله بر اساس سیستم چندعامله یک ساختار خودانطباق را ارائه می‌دهد و یک انطباق توزیع‌شده را برای محیط قطعی ایجاد می‌کند. اساس کار این سیستم به صورتی است که هر عامل وظیفه دارد که بر کارایی سیستم نظارت و آن را اندازه‌گیری کند. اگر کارایی از عدد تعیین‌شده فاصله گرفته است ابتدا اقدام به تنظیم پارامترهای فرایندهای مرتبط می‌نماید و در صورتی که مشکل حل نشد، یک یا چند فرایند، جایگزین فرایندهای مرتبط با کارایی در آن عامل می شوند. ورودی این راه حل تعیین پارامتر کارایی و نحوه اندازه‌گیری آن توسط هر عامل است. یک معماری اولیه برای ساختاردهی به مؤلفه‌های مختلف نیز به عنوان ورودی موجود است. فرایندهای جایگزین برای هر عامل نیز مشخص شده‌اند. خروجی این راه حل عبارت است از عاملی که یک یا چند فرایند آن با فرایندهای دیگری جایگزین شده و با شرایط جدید انطباق داده شده است. ایده کلی: کنترل توزیع‌شده مبتنی بر عامل	پیاده‌سازی این راه حل به نیروی متخصص با دانش بالا در زمینه خودانطباقی نیاز دارد. جستجوی پاسخ در فضای شمارا و محیط قطعی انجام می‌شود.
مبتنی بر نیازمندی- ۲۰۱۵- [۱۵]	این مقاله بر اساس تصمیم‌گیری چندمعیاره به اندازه‌گیری ریسک و سود ناشی از هر یک از استراتژی‌ها (تصمیمات طراحی) می‌پردازد و در نتیجه تصمیماتی که نتایج بهتری در ارضای نیازمندی‌های کیفی و با اهمیت دارند را برای اجرا انتخاب می‌نماید. ورودی این راه حل نیازمندی‌های سیستم و استراتژی‌هایی (تصمیمات طراحی) هستند که برای ارضای نیازمندی‌ها تعریف می‌شوند. همچنین بر اساس نظر خبره، احتمالاتی برای تخمین میزان ارضای نیازمندی‌ها بر اساس مشاهده شواهدی از محیط و پیاده‌سازی استراتژی‌ها نسبت داده می‌شود. خروجی این راه حل، استراتژی(های) بهینه برای برآورده‌کردن نیازمندی‌ها به صورت برون‌خط است. ایده کلی: انتخاب استراتژی بهینه بر اساس مقادیر کمی	سود ناشی از هر تصمیم طراحی به صورت تأثیر هر تصمیم طراحی در کاهش ریسک معرفی شده که توسط طراح تخمین زده می‌شود. تعیین این مقادیر عددی که تأثیر زیادی در جواب نهایی دارند نیازمند یک روش سیستماتیک است.
مبتنی بر نیازمندی- ۲۰۱۶- [۱۶]	این مقاله از متد P-CNP برای مقداردهی اولیه به پیش‌فرض‌های مربوط به اولویت‌بندی نیازها از دید کاربر استفاده کرده و همچنین از شبکه‌های تصمیم‌گیری پویا برای مدل‌کردن باورها و ایجاد اتصال بین شواهد و پیش‌فرض‌ها استفاده نموده به نحوی که پیش‌فرض‌های در نظر گرفته شده توسط ذی‌نفعان مجدداً ارزیابی شود. ورودی این راه حل نیازمندی‌های سیستم و استراتژی‌هایی (تصمیمات طراحی) هستند که برای ارضای نیازمندی‌ها تعریف می‌شوند. همچنین بر اساس نظر خبره، احتمالاتی برای تخمین میزان ارضای نیازمندی‌ها بر اساس مشاهده شواهدی از محیط و پیاده‌سازی استراتژی‌ها نسبت داده می‌شود. خروجی این راه حل، پیش‌فرض‌هایی هستند که بر اساس متد تعریف‌شده، مجدداً ارزیابی شده‌اند. ایده کلی: اندازه‌گیری میزان برآورده‌شدن نیازمندی‌ها بر اساس زمان، شواهد و استراتژی‌های مختلف	به دلیل حجم زیاد جدول‌های P-CNP تنها نیازهای کاربران در نظر گرفته شده و از نیازهای سایر ذی‌نفعان صرف نظر شده است. فقط نیازهای کیفی مهم از دید کاربر بررسی شده‌اند و معیاری برای سنجش اهمیت نیازها به منظور تعیین نیازهای مهم بیان نشده است.

۳) در بعضی مقالات، راه حل و یا چارچوب ارائه‌شده تا حدی پیچیده و خاص‌منظوره است که پیاده‌سازی آن برای سایر مسایل دشوار و زمان‌بر است و به نیروی متخصص با دانش بالا در زمینه سیستم‌های خودانطباق نیاز دارد.

روش پیشنهادی در این مقاله در حوزه "خودانطباقی مبتنی بر نیازمندی" مطرح می‌شود. همان گونه که در جدول ۱ بیان شد تا کنون مطالعاتی در زمینه نحوه بازنمایی نیازمندی‌ها [۱۴] و اتخاذ تصمیم بر مبنای اندازه‌گیری نیازها [۱۵] انجام شده است. پرکاربردترین مقاله‌هایی

جدول ۲: منابع ایجاد عدم اطمینان در سیستم‌های نرم‌افزاری.

مرحله	منبع عدم اطمینان	دلیل ایجاد عدم اطمینان
شناخت نیازمندی و محیط	عدم اطمینان شناخت محیط سیستم و نیازهای ذی‌نفعان	بعضی از نیازها یا محدودیت‌ها اساساً از ابتدا تشخیص داده نمی‌شوند. بعضی از نیازها یا محدودیت‌ها درست فهمیده نمی‌شوند. بعضی از اطلاعات درباره نیازها از منابعی مانند حس‌گرها دریافت می‌شوند که دارای حواشی زاید یا خطا هستند. بعضی از نیازها نیز بر پایه فرضیات و حدسیاتی بنا می‌شوند که درستی این فرض‌ها گاهی در زمان طراحی هم قابل اثبات نیستند.
تحلیل نیازمندی و محیط	عدم اطمینان در تحلیل محیط سیستم و نیازهای ذی‌نفعان	نظرات ذی‌نفعان مختلف تا حدی با یکدیگر متفاوت است که ناشی از تفاوت در جزئیاتی است که هر یک از ذی‌نفعان در هدف خود لحاظ می‌کنند. این در حالی است که کلیت مسئله از دید همگان یکسان است. افزوده‌شدن جزئیات متعددی که در هر یک از دیدگاه‌ها متفاوت است سرمنشأ بروز عدم اطمینان است.
تمام مرحله‌ها	عدم اطمینان در صحت مدل‌های تبدیل‌شده	در معماری مبتنی بر مدل بر اساس قواعد تبدیل، مدل‌های هر سطح به سطح دیگر تبدیل می‌شوند. در ساختارهای چندمدله، امکان ناسازگاری مدل‌ها با یکدیگر وجود دارد.
مرحله طراحی	عدم اطمینان از ارزش تصمیمات طراحی	در مرحله طراحی، برای برآورده‌شدن نیازها استراتژی‌هایی تعریف می‌شوند که گاهی ارزش یک یا چند استراتژی در برآورده‌کردن یک یا چند نیاز مشخص نیست.
مرحله اجرا	عدم اطمینان از ثبات نیازمندی‌ها	علاوه بر مشکلات ناشی از عدم شناخت صحیح نیازها، مشکل تغییر در محیط یا نیازهای ذی‌نفعان در زمان اجرا مطرح می‌باشد.

که اخیراً در این زمینه مطرح شده‌اند، [۱۵] تا [۱۷] هستند که شیوه‌ای برای کمی‌کردن نیازمندی‌ها و لحاظ‌کردن عدم اطمینان در محاسبات عددی ارائه نموده‌اند. متد مطرح‌شده در این مقالات از شبکه‌های بیزین و شبکه‌های تصمیم‌گیری پویا استفاده می‌کند و بر مبنای فرضیاتی به شرح زیر است:

۱) شرایط مختلفی که ممکن است محیط در زمان اجرای سیستم داشته باشد پیش‌بینی می‌شود.
۲) نیازهای کاربردی کاملاً توسط سیستم تولیدشده برآورده می‌شود.
۳) نیازهای "کیفی" که دارای اهمیت بیشتری هستند انتخاب و روابط میان این نیازها از جمله تناقض‌ها بررسی می‌شوند.
۴) فقط نیازهای "کاربران" سیستم در نظر گرفته می‌شوند.
۵) استراتژی‌های متفاوتی که برای برآورده‌شدن هر یک از نیازها وجود دارد مشخص می‌شوند.
۶) نیازمندی‌ها بر اساس اندازه‌گیری‌های عددی ارائه‌شده توسط ذی‌نفعان اولویت‌بندی می‌شوند.
۷) میزان احتمال برآورده‌شدن هر یک از نیازها توسط ذی‌نفعان به صورت اعداد دقیق پیش‌بینی می‌شود.
۸) میزان احتمال تأثیر هر یک از استراتژی‌ها در برآورده‌شدن هر یک از نیازها توسط ذی‌نفعان به صورت اعداد دقیق پیش‌بینی می‌شوند.

فرض ما در این مطالعه بر مبنای این است که تقسیم‌بندی نیازمندی‌ها در قالب نیازهای کمی و کیفی در شرایط عدم اطمینان، دسته‌بندی مطلوبی نمی‌باشد. از این رو نیازها را در قالب دو دسته نیازهای اساسی معماری (ASR[1]ها) و سایر نیازمندی‌ها دسته‌بندی خواهیم کرد. نیازهای اساسی معماری که مبنای کار محاسباتی ما را تشکیل می‌دهند صرفاً محدود به نیازهای کاربران سیستم نبوده و نیازهای سایر ذی‌نفعان را نیز مورد محاسبه قرار می‌دهند.

آنچه در مقالات پیشین، پایه محاسبات تلقی شده است مقادیری برای اندازه‌گیری نیازمندی‌ها، اولویت‌بندی نیازها و اندازه‌گیری استراتژی‌های مختلف در برآورده‌ساختن نیازمندی‌ها است که به صورت پیش‌فرض این اندازه‌ها توسط ذی‌نفعان ارائه و در این متدها صرفاً از این اعداد استفاده می‌شود. در این مطالعه به شیوه‌ای متفاوت روش‌هایی به منظور اندازه‌گیری سیستماتیک این مقادیر ارائه شده و برای وضوح بیشتر، این روش‌ها بر روی یک سیستم هوشمند پیاده‌سازی گردیده که در فصل پنجم به آن پرداخته شده است. این مقاله ضمن تبیین منابع عدم اطمینان، به کمی‌کردن ویژگی‌های توصیفی و کیفی می‌پردازد که این ویژگی‌ها در تحلیل نیازمندی‌ها و طراحی سیستم بر اساس نیازمندی‌ها تأثیرگذار هستند. به این ترتیب، تصمیم‌گیری در هر مرحله از فرایند تولید نرم‌افزار، مبتنی بر محاسبات عددی می‌باشد که راهی در راستای خودکارسازی تولید نرم‌افزار است.

۳- تعریف مسأله و اهداف تحقیق

همان گونه که در بخش دوم بیان شد، نخستین گام برای طراحی یک سیستم نرم‌افزاری (بزرگ‌مقیاس) شناخت محیط سیستم و نیازهای ذی‌نفعان آن است. پس از تحلیل نتایج گام نخست، ساختار کلی سیستم تعیین می‌شود و طی چند مرحله، جزئیات سیستم طراحی می‌گردد. این گام تا جایی تکرار می‌شود که یک سیستم از قابلیت پیاده‌سازی برخوردار شود. از سوی دیگر، وجود عدم اطمینان در مرحله‌های مختلف تولید نرم‌افزار منجر به ایجاد نیاز به انطباق در زمان اجرا می‌شود. منابع عدم اطمینان در مرحله‌های مختلف تولید سیستم عبارتند از:

۱) عدم اطمینان (عدم امکان) شناخت محیط سیستم و نیازهای ذی‌نفعان [۱۴] و [۲۱]
۲) عدم اطمینان در تحلیل محیط سیستم و نیازهای ذی‌نفعان [۲]، [۱۴] و [۲۱]
۳) عدم اطمینان در صحت مدل‌های تبدیل‌شده [۲۲]
۴) عدم اطمینان از ارزش تصمیمات طراحی [۱]
۵) عدم اطمینان از ثبات نیازمندی‌ها [۱۰]

عدم اطمینان در هر یک از منابع فوق بر اثر دلایل متعددی ایجاد می‌شود. این دلایل با جزئیات بیشتری در جدول ۲ توضیح داده شده‌اند. نخستین ستون از جدول ۲ بیان می‌کند که عدم اطمینان در کدام یک از مرحله‌های تولید و یا اجرای سیستم‌های نرم‌افزاری ایجاد می‌شود. در

1. Architecturally Significant Requirements

مرحله نخست از تولید سیستم، علاوه بر این که ممکن است عدم اطمینان در شناخت و یا تحلیل نیازمندی‌ها و محیط ایجاد شود و به مرحله‌های بعد انتشار یابد، ممکن است در هنگام تبدیل مدل نیازمندی به مدل معماری سیستم نیز نوع دیگری از عدم اطمینان ایجاد گردد که عبارت از عدم اطمینان در صحت تبدیل مدل معماری است. به طور کلی در ساختارهای چندمدله در هر یک از مرحله‌های تولید سیستم ممکن است نوعی عدم اطمینان از صحت تبدیل مدل‌ها ایجاد شود.

بر این اساس به نظر می‌رسد که پرداختن به مهندسی نیازها به منظور شناسایی‌کردن و کاهش‌دادن عدم اطمینان از ضرورت زیادی برخوردار است. به بیان دیگر نیاز است که نه تنها عدم اطمینان در شناخت محیط و نیازها کاهش پیدا کند بلکه نیاز است که عدم اطمینان در تحلیل نیازها و ایجاد مدل‌های معماری بر مبنای نیازمندی‌ها کاهش داده شود. برای این منظور، مطلوب است که ویژگی‌های کیفی به ویژگی‌های کمی تبدیل شوند. از این رو باید در هر مرحله، تمام تصمیمات بر اساس اندازه‌های کمی اتخاذ شود و مقیاس‌هایی برای اندازه‌گیری محصولات تولیدشده در پایان هر مرحله تعریف شود. به این ترتیب به جای تصمیم‌گیری بر اساس معیارهای کیفی، اندازه‌گیری بر اساس مقیاس‌های کمی انجام می‌شود. این مقیاس‌ها در واقع ویژگی‌هایی هستند که در تحلیل نیازمندی‌ها و طراحی سیستم بر اساس نیازمندی‌ها تأثیرگذار هستند.

٤- شیوه حل مسئله و راه حل پیشنهادی

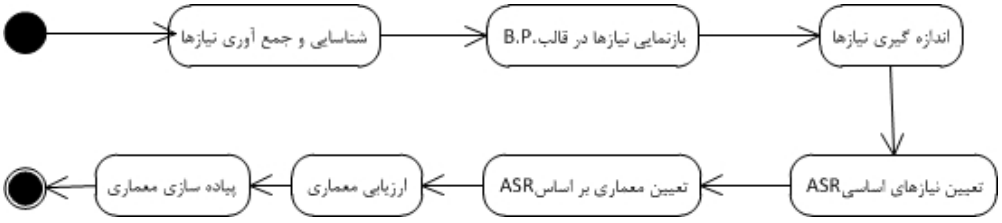
همان گونه که در بخش نخست عنوان شد تغییر و اصلاح نرم‌افزار ممکن است در سطح کد و یا در سطح طراحی انجام شود. به طور کلی، هر چه اصلاح در مراحل ابتدایی‌تری انجام شود کم هزینه‌تر است. از طرف دیگر در سیستم‌های خودانطباق باید تغییر و اصلاح سیستم در زمان اجرا میسر باشد. اما با توجه به محدودیت‌های منابعی مانند فضا و به ویژه محدودیت زمان باید امکان تغییر در زمان اجرا در مرحله‌های مختلف توسعه نرم‌افزار مانند مرحله طراحی و حتی مرحله شناخت و تحلیل نیازمندی‌ها پیش‌بینی و ملزومات مربوط به آن فراهم شده باشد. یکی از چالش‌های خودانطباقی، تصمیم‌گیری بر مبنای ویژگی‌های متعدد برای انتخاب یک مجموعه از مهم‌ترین نیازمندی‌ها و نیز انتخاب بهترین استراتژی‌ها برای برآورده‌کردن آن نیازها می‌باشد. وجود عدم اطمینان در یک یا چند ویژگی ممکن است مانع از اتخاذ بهترین تصمیم شود. روش پیشنهادی این مقاله، تبدیل ویژگی‌های کیفی به کمی است تا تصمیم‌گیری بر اساس اندازه‌گیری و محاسبات انجام شود.

در نتیجه از یک سو دقت در تصمیم‌گیری افزایش پیدا می‌کند و از سوی دیگر، اثرات ناشی از یک تصمیم در مرحله‌های بعد از آن قابل تعقیب است. به طور کلی اگر سیستم در زمان اجرا دارای عملکرد نامطلوبی باشد جستجوی دلیل آن در زمان اجرا پرهزینه است و ممکن است خارج از محدودیت‌های سیستم باشد. اما با وجود قابلیت تعقیب یک عملکرد می‌توان دلیل مشکل را با سرعت در مدل نیازمندی‌ها و یا مدل طراحی سیستم نرم‌افزاری شناسایی نمود. در این راستا در هر مرحله تمام تصمیمات بر اساس اندازه‌های کمی اتخاذ می‌شود و مقیاس‌هایی برای اندازه‌گیری محصولات تولیدشده در پایان هر مرحله تعریف می‌شود. تحلیل نیازمندی‌ها بر اساس محاسبات عددی ایده محوری در این مقاله می‌باشد. از آنجایی که تصمیمات طراحی برای تولید معماری نرم‌افزار مبتنی بر نیازمندی‌های اساسی است شناسایی نیازهای اساسی از اهمیت قابل توجهی برخوردار است.

روش پیشنهادی در این مقاله به این صورت است که در نخستین گام، نیازهای سیستم جمع‌آوری می‌شوند. این نیازمندی‌ها بر اساس معیارهای مختلفی از دید ذی‌نفعان متفاوت از جمله صاحب سیستم، کاربران نهایی، تحلیلگر سیستم، طراح سیستم، برنامه‌نویسان و تیم پیاده‌سازی و نصب از ارزش‌های مختلفی برخوردار هستند. به این معنا که ممکن است یک نیاز از دید کاربران نهایی نیاز کم‌اهمیتی باشد در حالی که از دید صاحبان سیستم، بسیار مهم و توجیه‌کننده جنبه صرفه اقتصادی سیستم باشد و همچنین نیازی است که پیاده‌سازی آن از دید تحلیلگران سیستم، پیچیده ارزیابی می‌شود. بنابراین این نیاز باید به عنوان یکی از نیازهای اساسی سیستم مورد توجه و ارزیابی ویژه‌ای قرار بگیرد.

به طور دقیق‌تر به دنبال یک روش دقیق جهت شناسایی نیازمندی‌های اساسی هستیم که معماری سیستم را متأثر می‌کنند. در این مقاله پس از جمع‌آوری نیازها، از قالب boiler-plate جهت بازنمایی نیازها استفاده می‌شود. سپس بر اساس یک سلسله محاسبات عددی، ASRها تشخیص داده می‌شوند. چگونگی انجام این محاسبات به صورت مبسوط در ادامه تشریح شده است.

بر مبنای توضیحات فوق، به طور کلی لازم است که نیازمندی‌ها اولویت‌بندی شوند. این اولویت‌بندی تحت تأثیر فاکتورهای متعددی می‌باشد که هر یک یا چند فاکتور، منعکس‌کننده نظرات یکی از ذی‌نفعان است. در نتیجه، میزان اهمیت هر کدام از فاکتورهای مؤثر بر اهمیت یک نیازمندی نیز به نوبه خود باید در مقایسه با اهمیت سایر فاکتورها بررسی شود. در این مقاله اهمیت فاکتورها به صورت کمّی و عددی مشخص می‌شوند و در محاسبات مربوط به اولویت‌بندی نیازمندی‌ها به صورت ضرایب وزنی به کار برده می‌شوند. به این ترتیب بر خلاف مقالاتی که تنها نیازها از دید کاربران مورد بررسی قرار می‌گیرند، در این روش، تحلیل فراگیر و جامعی بر روی نیازمندی انجام می‌شود. یک راه برای پیاده‌سازی روش پیشنهادی، استفاده از شیوه‌های تحلیل تصمیم‌گیری چندمعیاره است [۱۶]. هنگام تصمیم‌گیری در شیوه‌های MCDA، انتخاب‌های مختلف ارزیابی می‌شوند و بر اساس معیارهای معین‌شده، یک انتخاب یا یک زیرمجموعه از انتخاب‌ها معرفی می‌شوند. شیوه‌های MCDA روش‌هایی برای مطمئن‌شدن از اتخاذ یک تصمیم قابل اعتماد هستند و بر اساس trade-off بین فاکتورهای متعددی عمل می‌کنند که از شرایط خارجی متأثر شده‌اند. یکی از شیوه‌های پرکاربرد در MCDA، فرایند تحلیل سلسله‌مراتبی است که برای استخراج مقیاس‌هایی برای اولویت‌بندی انتخاب‌ها به کار برده می‌شود. در حوزه مهندسی نیازمندی‌ها، AHP برای تعیین پیش‌فرض‌های ویژگی‌های کیفیتی به صورت عددی و نیز استدلال در زمان اجرا بر اساس اولویت‌بندی مجموعه‌ای از تصمیم‌ها قابل استفاده است. به عنوان مثال در یک مقاله مشابه در این حوزه، با استفاده از AHP بر روی انتشار ویدئو به روی یک شبکه ادهاک در حال پرواز- در زمان اجرا- یک پروتکل مسیریابی پیاده‌سازی شده است [۲۳]. این روش، نیازهای کیفی متعددی مانند کیفیت اتصال، انرژی باقیمانده، وضعیت بافر، اطلاعات جغرافیایی و حرکت نود در یک محیط سه‌بعدی را مورد محاسبه قرار می‌دهد. این مقاله از ترکیب شبکه‌های بیزین و AHP برای متعادل‌کردن اولویت NFRها بر اساس مقادیر آنی به دست آمده در طول عملیات سیستم استفاده می‌کند. پایه روش‌های زیرمجموعه MCDA مانند AHP یا P-CNP مقایسه جفتی اعدادی است که بر مبنای نظر خبرگان تعیین شده‌اند. نتیجه حاصل از این مقایسه نشان می‌دهد که با چه نسبتی یک مؤلفه (انتخاب) بر سایر مؤلفه‌ها (انتخاب‌ها) بر مبنای یک معیار مشخص غلبه دارد. از آنجا که از نظرات خبرگان برای مقداردهی اولیه به اعداد مربوط به هر مؤلفه استفاده می‌شود، احتمال



شکل ۲: نمودار فعالیت برای پیاده‌سازی سیستم‌های نرم‌افزاری.

جدول ۳: نیازمندی‌های سیستم هوشمند مجازی مرکز اتومبیل کرایه در قالب BOILER-PLATE.

قالب boiler-plate	نوع هدف	شناسه
راننده/مسافر باید قادر به ثبت اطلاعات خود در سیستم باشد.	Usability	BP۰۱
مسافر باید قادر به ثبت سفارش با حداکثر به اندازه کوتاه‌ترین فاصله زمانی رسیدن به مقصد باشد.	Usability	BP۰۲
راننده نباید قادر به امتیازدهی به رانندگان باشد.	Usability	BP۰۳
راننده باید قادر به ثبت حضور خود در سیستم، حداقل یک بار در روز باشد.	Usability	BP۰۴
سیستم باید حداکثر پس از یک ثانیه، مسیریابی را انجام دهد.	Performance	BP۰۵
سیستم باید بر اساس اولویت‌بندی به رانندگان، سرویس تخصیص دهد.	Performance	BP۰۶
سیستم باید به صورت ۲۴ ساعت در روز و ۷ روز هفته در دسترس باشد.	Availability	BP۰۷
سیستم باید به هر تعداد کاربری که حتی در ساعات اوج روز، لاگین می‌کنند سرویس بدهد.	Availability	BP۰۸
سیستم باید به هر تعداد کاربری که حتی در ساعات اوج روز، لاگین می‌کنند سرویس بدهد.	Reliability	BP۰۹
سیستم باید به ازای هر مبدأ و هر مقصد، کوتاه‌ترین، ارزان‌ترین و سریع‌ترین مسیر را پیدا کند.	Reliability	BP۱۰

ناسازگاربودن اعداد وجود دارد. بنابراین زمانی نتایج حاصل از AHP قابل اتکا هستند که ناسازگاری ماتریس نهایی بررسی شود و سازگاری اعداد تضمین گردد. بنابراین پیش از استفاده از اعداد نهایی حاصل از محاسبات AHP، سازگاری اعداد بر اساس نرخ سازگاری [۲۴] بررسی و تأیید می‌شوند. بخش پنجم با شبیه‌سازی سیستم هوشمند مجازی "مراکز اتومبیل کرایه" با اندازه‌گیری نیازمندی‌های سیستم بر اساس مشخصات کمی به تحلیل نیازمندی‌ها و تعیین نیازهای اساسی سیستم می‌پردازد. جزئیات مربوط به پیاده‌سازی روش توضیح داده شده در این بخش، در بخش بعدی به صورت مفصل تشریح شده است.

۵- نتایج شبیه‌سازی

این بخش حاوی گزارشی است از نتایج پیاده‌سازی روشی که در بخش‌های پیش توصیف شده است. مسأله مورد بررسی عبارت از شبیه‌سازی سیستم هوشمند مجازی "مراکز اتومبیل کرایه" می‌باشد. در این مسأله، یک سیستم مدیریتیِ فراگیر ایجاد می‌شود که دربرگیرنده زیرسیستم‌های مدیریتی به صورت محلی است. با توجه به مجازی‌بودنِ چنین سیستمی، مکان استقرار نیروهای خدماتی مانند رانندگان، یک مکان ثابت و هماهنگ نیست و بر اساس درخواست هر راننده مکان استقرار و زمان‌های ارائه خدمات توسط وی تعیین می‌گردد. به این معنا که هر راننده می‌تواند بدون مراجعه به مکان ثابت مرکز، از مکان مورد درخواست خود که می‌تواند منزل و یا هر محل دیگری باشد به نقطه خدمت اعزام گردد مشروط بر آن که این راننده در زمان ثبت نام در سیستم، محل حضور خود را ثبت کرده باشد. چنین قابلیتی نیاز به یک مکان برای توقف خودروهای متعدد را در یک مرکز محلی برطرف می‌نماید. برای جذب مشتری بیشتر لازم است که سیستم در تمام روزهای هفته و هر ساعت از روز خدمت‌رسانی به کاربران را انجام دهد.

در نخستین گام، نیازهای سیستم جمع‌آوری می‌شوند. پس از آن، نیازها به صورت جزئی‌تر بررسی می‌گردند و در قالب boiler-plate قرار می‌گیرند تا ASRها تشخیص داده شوند. جدول ۳ بازنمایی نیازمندی‌ها در قالب boiler-plate را نمایش می‌دهد. پس از آن، نیازها را اندازه‌گیری می‌نماییم. اعدادی که به عنوان اندازه هر نیازمندی ثبت می‌شوند بر اساس میزان اهمیت یا کاربرد آن نیاز برای مشتری، دشواری پیاده‌سازی و پیچیدگی آن نیازمندی، نسبت داده می‌شود. شکل ۲ نمودار فعالیت از مرحله جمع‌آوری نیازمندی‌ها تا مرحله پیاده‌سازی را نمایش می‌دهد.

در جدول ۴ همان گونه که در بخش قبل عنوان شد اولویت‌بندی نیازمندی‌ها بر اساس معیارهایی انجام می‌شود. در این مقاله معیارهای مقایسه نیازمندی‌ها عبارت از سه معیار میزان کاربرد برای ذی‌نفعان (صاحب سیستم- راننده- مسافر)، پیچیدگی پیاده‌سازی و دشواری پیاده‌سازی می‌باشد. بنابراین در این بخش کار به سه بخش تقسیم می‌شود. بخش اول، تشکیل ماتریس ۳×۳ برای تعیین اهمیت هر یک از این سه معیار بر اساس روش AHP، بخش دوم، بررسی سازگاری ضرایب و بخش سوم، به ازای هر معیار یک ماتریس ۱۰×۱۰ برای اولویت‌بندی نیازمندی‌های منتخب نوشته‌شده در جدول ۳ رسم می‌کنیم.

در بخش اول، مطابق جدول ۴ در سطرها و ستون‌های ماتریس، فاکتورها نوشته شده‌اند. بر اساس نظر کاربران و ذی‌نفعان دیگر سیستم (صاحب سیستم، تحلیل‌گر و طراح سیستم) مقادیر اولیه را اختصاص می‌دهیم. به دلیل حجم زیاد محاسبات انجام‌شده و تعداد زیاد جدول‌ها، از ارائه جزئیات محاسبات خودداری شده است لیکن شیوه کار به این صورت است که هر خانه از جدول بر جمع ستون تقسیم می‌شود.

از آنجا که مقداردهی اولیه به اعداد مربوط به هر مؤلفه مبتنی بر نظرات خبرگان (کاربران، صاحب سیستم، تحلیل‌گر و طراح سیستم، توسعه‌دهنده سیستم) است احتمال ناسازگاربودن اعداد وجود دارد. بنابراین زمانی نتایج به دست آمده در جدول ۴ قابل اتکا است که از جنبه ناسازگاری بررسی شود و سازگاری اعداد تضمین گردد. بنابراین پیش از استفاده از اعداد نهایی حاصل از محاسبات AHP، لازم است که سازگاری را تأیید کنیم.

جدول ۴: ماتریس AHP تعیین ضریب اهمیت فاکتورها در اولویت‌بندی نیازها.

	میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی		میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی
میزان کاربرد برای ذی‌نفعان	۱	۱٫۵	۱٫۴	میزان کاربرد برای ذی‌نفعان	۱	۰٫۲	۰٫۲۵
پیچیدگی پیاده‌سازی	۵	۱	۲	پیچیدگی پیاده‌سازی	۵	۱	۲
دشواری پیاده‌سازی	۴	۱٫۲	۱	دشواری پیاده‌سازی	۴	۰٫۵	۱

	میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی		میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی	اولویت
میزان کاربرد برای ذی‌نفعان	۱	۰٫۲	۰٫۲۵	میزان کاربرد برای ذی‌نفعان	۰٫۰۱	۰٫۱۱۷	۰٫۰۷۶	۰٫۰۹۵
پیچیدگی پیاده‌سازی	۵	۱	۲	پیچیدگی پیاده‌سازی	۰٫۵	۰٫۵۸۸	۰٫۶۱۵	۰٫۵۶
دشواری پیاده‌سازی	۴	۰٫۵	۱	دشواری پیاده‌سازی	۰٫۴	۰٫۵	۰٫۳۰۷	۰٫۴
مجموع	۱۰	۱٫۷	۳٫۲۵					

جدول ۵: ماتریس بررسی سازگاری مقادیر جدول ۴.

	میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی		میزان کاربرد برای ذی‌نفعان	پیچیدگی پیاده‌سازی	دشواری پیاده‌سازی
اولویت	۰٫۰۹۵	۰٫۵۶	۰٫۴	اولویت	۰٫۰۹۵	۰٫۵۶	۰٫۴
میزان کاربرد برای ذی‌نفعان	۱	۰٫۲	۰٫۲۵	میزان کاربرد برای ذینفعان	۰٫۹۵	۰٫۱۱۲	۰٫۱
پیچیدگی پیاده‌سازی	۵	۱	۲	پیچیدگی پیاده‌سازی	۰٫۴۷۵	۰٫۵۶	۰٫۸
دشواری پیاده‌سازی	۴	۰٫۵	۱	دشواری پیاده‌سازی	۰٫۳۸	۰٫۲۸	۰٫۴

جدول ۶: ماتریس محاسبه λ_{max}.

مجموع وزن‌دار	اولویت	نتیجه
۰٫۳۰۷	۰٫۰۹۵	۳٫۲۳
۱٫۸۳۵	۰٫۵۶	۳٫۲۷
۱٫۶	۰٫۴	۲٫۶۵
	مجموع	۹٫۱۵
	میانگین λ_{max}	۳٫۰۵

برای بررسی میزان سازگاری مطابق مراحل زیر عمل می‌شود [۲۴]:

- اعداد اولویت به جای ستون آخر در سطر اول نوشته می‌شوند.
- اعداد هر خانه در اولویت همان ستون ضرب می‌شود.
- سپس مجموع اعداد هر سطر محاسبه می‌شود.
- اعداد به دست آمده در صورت کسر و اولویت‌های اولیه به دست آمده در مخرج کسر قرار می‌گیرند و به ازای هر معیار یک λ محاسبه می‌شود.
- در آخر، میانگین λها محاسبه می‌شود (λ_{max}).

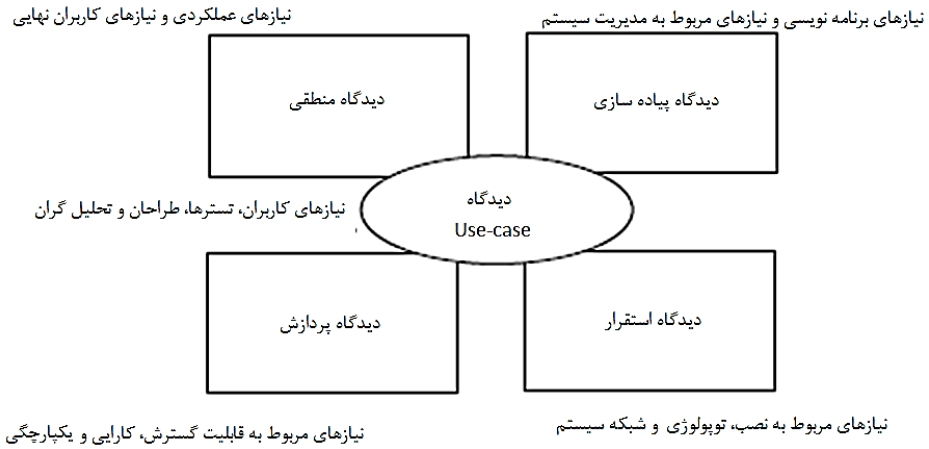
این محاسبات در جدول‌های ۵ و ۶ آمده است.

سپس اندیس سازگاری با استفاده از (۱) محاسبه می‌شود. در صورتی که نسبت اندیس سازگاری به اندیس سازگاری تصادفی عددی کوچک‌تر از ۰٫۱ باشد در این صورت اعداد ماتریس اولیه سازگار هستند و در غیر این صورت ناسازگار خواهند بود. مقدار اندیس سازگاری تصادفی به ازای ماتریس‌های AHP با ابعاد مختلف در جدول‌های [۲۴] موجود است

$$CI = (\lambda_{max} - n)(n-1) \rightarrow$$
$$CI = (۳٫۰۵ - ۳)(۳-۱) = ۰٫۰۲۵ < ۰٫۱ \quad (۱)$$

پس از جمع‌آوری نیازها و مستندسازی آنها بر اساس قالب boiler-plate، بر اساس میزان کاربرد هر نیاز برای هر ذی‌نفع و نیز پیچیدگی و دشواری پیاده‌سازی آن نیاز امتیازاتی به نیازمندی‌ها اختصاص داده می‌شود. بر اساس اهمیت خواست هر یک از ذی‌نفعان، ضرایب متفاوتی به نیازمندی مربوط به آنها نسبت داده می‌شود. پیچیدگی هر نیازمندی بر اساس میزان عدم اطمینان در شناخت ابعاد آن نیاز تعیین می‌شود. دشواری پیاده‌سازی هر نیازمندی به تجربه تیم طراحی و پیاده‌سازی برای اجرای آن نیازمندی بستگی دارد. همان گونه که در فصل چهارم توضیح داده شد بر اساس هر یک از فاکتورهای اولویت‌بندی، یک بار نیازمندی‌ها بررسی و امتیازدهی می‌شوند. جدول ۷ به نمایش ماتریس AHP مربوط به تعیین امتیازات هر نیازمندی نسبت به ۹ نیازمندی منتخب دیگر بر اساس فاکتور کاربرد برای ذی‌نفعان می‌پردازد. به این معنا که به ازای هر کدام از سه فاکتور نیاز ذی‌نفعان، پیچیدگی پیاده‌سازی و دشواری پیاده‌سازی ماتریسی ۱۰×۱۰ رسم نموده و مراحل AHP را انجام می‌دهیم. در ستون دوم جدول ۸ اولویت نهایی هر نیازمندی به دست آمده از جدول ۷ نشان داده شده است. به دلیل حجم زیاد جدول‌ها، تنها به نمایش اولویت‌های به دست آمده برای نیازمندی‌ها بر اساس فاکتورهای پیچیدگی پیاده‌سازی و دشواری پیاده‌سازی در جدول اکتفا می‌کنیم.

به ستون دوم در جدول ۳ توجه کنید که نوع هر نیازمندی در آن مشخص شده است. به بیان دیگر، نوع هر نیازمندی بیانگر این است که هر نیازمندی در راستای برآورده‌شدن کدام یک از هدف‌های سیستم



شکل ۳: دیدگاه‌های مختلف در مدل ۱+۴ view.

جدول ۷: ماتریس تعیین امتیازات هر نیازمندی بر اساس فاکتور کاربرد برای ذی‌نفعان.

میزان کاربرد برای ذی‌نفعان	BP۰۱	BP۰۲	BP۰۳	BP۰۴	BP۰۵	BP۰۶	BP۰۷	BP۰۸	BP۰۹	BP۱۰
BP۰۱	۱	۰٫۵	۱	۱	۰٫۵	۱	۰٫۲۵	۰٫۲۵	۰٫۲۵	۰٫۵
BP۰۲	۲	۱	۲	۲	۱	۲	۰٫۳۳	۰٫۳۳	۰٫۳۳	۱
BP۰۳	۱	۰٫۵	۱	۱	۰٫۵	۱	۰٫۲	۰٫۲	۰٫۲	۰٫۵
BP۰۴	۱	۰٫۵	۱	۱	۰٫۵	۱	۰٫۲	۰٫۲۵	۰٫۲۵	۰٫۵
BP۰۵	۲	۱	۲	۲	۱	۱	۰٫۳۳	۰٫۵	۰٫۵	۱
BP۰۶	۱	۰٫۵	۱	۱	۱	۱	۰٫۲۵	۰٫۳۳	۰٫۳۳	۰٫۵
BP۰۷	۴	۳	۵	۵	۳	۴	۱	۱	۲	۳
BP۰۸	۴	۳	۵	۴	۲	۳	۱	۱	۱	۲
BP۰۹	۴	۳	۵	۴	۲	۳	۱	۱	۱	۲
BP۱۰	۲	۱	۲	۲	۱	۲	۰٫۳۳	۰٫۵	۰٫۵	۱

جدول ۸: استخراج اولویت نهایی هر نیازمندی بر اساس فاکتورهای میزان کاربرد برای ذی‌نفعان، پیچیدگی پیاده‌سازی و دشواری پیاده‌سازی.

	اولویت میزان کاربرد برای ذی‌نفعان	اولویت پیچیدگی پیاده‌سازی	اولویت دشواری پیاده‌سازی
BP۰۱	۰٫۰۴	۰٫۰۶	۰٫۰۶
BP۰۲	۰٫۰۸	۰٫۱۳	۰٫۱
BP۰۳	۰٫۰۴	۰٫۰۴	۰٫۰۴
BP۰۴	۰٫۰۴	۰٫۰۷	۰٫۰۳
BP۰۵	۰٫۰۸	۰٫۰۷	۰٫۰۴
BP۰۶	۰٫۰۵	۰٫۰۵	۰٫۰۸
BP۰۷	۰٫۱۹	۰٫۱	۰٫۱۹
BP۰۸	۰٫۱۸	۰٫۲	۰٫۱۹
BP۰۹	۰٫۱۸	۰٫۲	۰٫۱۹
BP۱۰	۰٫۰۹	۰٫۰۶	۰٫۰۵

تعریف شده است. نیازمندی‌های منتخبی که در این جدول بررسی شده‌اند در قالب یکی از چهار هدف قابلیت استفاده، کارایی، در دسترس بودن و قابلیت اعتماد دسته‌بندی شده‌اند. بر اساس اطلاعات جدول ۸ و نوع هر نیازمندی، نتیجه نهایی اندازه‌گیری نیازمندی‌ها جمع‌بندی و محاسبه می‌شود. تفسیر این اعداد نشان می‌دهد که در سیستم هوشمند مجازی مراکز اتومبیل کرایه، مهم‌ترین هدف، جذب هرچه بیشتر مشتری برای سیستم می‌باشد. برآورده‌شدن این هدف در گرو آن است که مسافران و رانندگان بتوانند به سهولت از سیستم استفاده کنند و سیستم تقریباً در تمام ایام هفته و تمام ساعات روز در دسترس باشد. به این معنا که قابلیت استفاده و در دسترس بودن سیستم از قابلیت اعتماد به سیستم مهم‌تر است. به همین ترتیب قابلیت اعتماد به سیستم در قیاس با کارایی سیستم از اهمیت بالاتری برخوردار می‌باشد.

در نهایت با تحلیل جدول ۹ نتیجه می‌گیریم که نیازمندی‌های BP۰۱ – BP۰۲ – BP۰۳ – BP۰۴ – BP۰۷ – BP۰۸ دارای بیشترین امتیاز هستند و بنابراین این نیازها به عنوان نیازهای اساسی تشخیص داده شده‌اند. در ادامه نشان داده می‌شود که چگونه سیستم طراحی می‌شود و انتخاب این نیازها به عنوان نیازهای اساسی سیستم چگونه بر پیاده‌سازی نهایی تأثیرگذار است. پس از تحلیل نیازمندی‌ها، در راستای انتخاب یک سبک برای معماری باید به انتخاب یک رویکرد بپردازیم. انتخاب رویکرد یک فرایند است که ورودی این فرایند عبارت است از نیازمندی‌های سیستم که به ۲ دسته کلی نیازهای اساسی و سایر نیازمندی‌ها تقسیم‌بندی شده‌اند. پردازش عبارت است از انتخاب یک رویکرد بر اساس این نیازها و خروجی عبارت از رویکرد نهایی برای انتخاب سبک معماری می‌باشد. حال با انتخاب مدل ۱+۴ view، به انتخاب رویکردهای موجود در این مدل می‌پردازیم و مطابق شکل ۳ در نهایت، دیدگاه Use-case به عنوان یک دیدگاه مشترک، نتیجه‌گیری می‌شود.

طراحی سیستم بر اساس نیازهای اساسی به این شکل انجام می‌شود که به عنوان مثال، یکی از نیازهای اساسی سیستم این است که سیستم باید از قابلیت دسترسی بالایی برخوردار باشد. برای طراحی معماری از تاکتیک‌های معرفی‌شده [۲۵] برای برآورده‌کردن قابلیت دسترسی می‌توان استفاده کرد که این تاکتیک‌ها عبارت از تشخیص خطا، بازیابی سیستم در هنگام خطا و جلوگیری از بروز خطا هستند. فرض کنید که بخواهیم در فرایند بازیابی سیستم در هنگام خطا از روش رأی‌گیری استفاده نماییم. به

جدول ۹: تعیین ضریب اهمیت اهداف بر اساس اندازه‌گیری نیازمندی‌ها.

نیازها	نوع هدف	جمع‌بندی اولویت‌ها	ضریب اهمیت اهداف
BP۰۱	Usability	۰٫۰۶۱۴	۰٫۲۷۹
BP۰۲	Usability	۰٫۱۲۰۴	
BP۰۳	Usability	۰٫۰۴۲۲	
BP۰۴	Usability	۰٫۰۵۵	
BP۰۵	Performance	۰٫۰۷۰۸	۰٫۱۳۵۵۵
BP۰۶	Performance	۰٫۰۶۴۷۵	
BP۰۷	Availability	۰٫۱۵۰۰۵	۰٫۳۵۵۱۵
BP۰۸	Availability	۰٫۲۰۵۱	
BP۰۹	Reliability	۰٫۲۰۵۱	۰٫۲۶۷۲۵
BP۱۰	Reliability	۰٫۰۶۲۱۵	

این ترتیب باید سیستم به گونه‌ای طراحی شود که دارای پردازشگرهای چندگانه باشد. به این صورت در هر تراکنش آن پردازشگری که بر مبنای روش رأی‌گیری انتخاب می‌شود پاسخ می‌دهد و برای این منظور نیاز به توزیع‌شدگی داریم. از سوی دیگر، یکی دیگر از نیازهای ذی‌نفعان این است که به جای کنترل متمرکز از کنترل غیر متمرکز استفاده کنیم. از بین سبک‌های تعریف‌شده برای معماری، Component Independent به منظور برآورده‌ساختن هدف کنترل غیر متمرکز و همبندی توزیع‌شده، مناسب است. در این سبک، مؤلفه‌ها با یک همبندی در کنار هم قرار می‌گیرند و یک معماری را تشکیل می‌دهند. همبندی مشتری- کارگزار[1] با نیازهای سیستم مطابقت دارد و نحوه ارتباط بین این مؤلفه‌ها می‌تواند به صورت مبتنی بر رویداد انجام شود به این صورت که مشتری، هر یک از مسافران یا رانندگان هستند. کارگزار نیز به صورت توزیع‌شده برای هر نقطه از شهر تهران، به صورت واحد در نظر گرفته شده است.

٦- ارزیابی

در بخش ۵ بیان شد که طراحی سیستم بر مبنای تاکتیک‌هایی انجام می‌شود که در راستای برآورده‌کردن نیازهای اساسی سیستم هستند. تعیین نیازهای اساسی سیستم نقش قابل توجهی در طراحی و پیاده‌سازی سیستم دارد. این مقاله به بیان روشی برای تبدیل ویژگی‌های "کیفی و توصیفی" مؤثر در تعیین نیازهای اساسی سیستم به "مقیاس‌های کمی" پرداخته است. در این بخش به ارزیابی معماری تعیین‌شده می‌پردازیم تا مشخص شود که روش پیشنهادی تا چه حد در راستای تعیین یک معماری کارا مؤثر عمل کرده است. این ارزیابی بر مبنای آخرین نسخه ATAM انجام شده و در آن با تولید چند سناریو که بر اساس مشخصات کارکرد سیستم تهیه شده‌اند، بررسی گردیده که آیا نیازهای اساسی و دارای اولویت بالا در معماری لحاظ شده‌اند یا خیر. برای این منظور، اهداف سطح بالا، نیازمندی‌های اساسی و ویژگی‌های کیفی سیستم مجدداً مرور می‌شوند و سپس بر اساس آنها، معماری ارائه‌شده بررسی می‌شود. یک فهرست از تصمیمات معماری و تاکتیک‌هایی که برای برآورده‌کردن هر یک از اهداف اصلی و کیفیت‌های مورد نظر به کار برده شده است استخراج می‌شود. بعد از این مرحله یک درخت سودمندی رسم می‌شود که نشان می‌دهد در این سیستم، هر یک از اهداف و ویژگی‌های کیفی از چه جزئیاتی برخوردار هستند. به عنوان مثال، اهداف و ویژگی‌هایی مانند در دسترس‌بودن یا قابلیت تغییر در این سیستم دارای چه جزئیات و ملاحظاتی هستند. در درخت سودمندی، سناریو(ها)ی مورد نظر برای برآورده‌شدن هر یک از این اهداف و ویژگی‌ها با جزئیات شرح داده می‌شوند. پس از این مرحله، سناریوهایی که تولید شده‌اند با معماری ارائه‌شده مقایسه می‌گردد و بررسی می‌شود که معماری تا چه حد می‌تواند پاسخ‌گوی هر یک از این سناریوها باشد. در این مقاله در بخش شبیه‌سازی، به دلیل محدودیت فضا تنها ۱۰ مورد از نیازهای سیستم به منظور توضیح فرایند انجام روش پیشنهادی بیان شدند و مورد بررسی قرار گرفتند.

اما جهت انجام یک ارزیابی فراگیر، درخت سودمندی در مرحله ارزیابی مجموعه تمام نیازمندی‌ها را در نظر گرفته و معماری را بر اساس ویژگی‌های کیفی متعددی مورد ارزیابی قرار داده است. به دلیل تعدد سناریوهای تولیدشده و محدودیت فضا تنها چند سناریو از میان سناریوهای تولیدشده در درخت سودمندی نمایش داده شده در جدول ۱۰ به عنوان نمونه بیان شده‌اند. پس از استخراج درخت سودمندی، سناریوهای تعریف‌شده در این جدول بررسی می‌شوند. این سناریوها با تصمیمات معماری و تاکتیک‌های معماری برای پیاده‌سازی سیستم مقایسه می‌شوند. نقاط حساس، نقاط ریسک و غیر ریسک تشخیص داده شده و معماری از نظر پوشش‌دهی شرایط مختلف بررسی می‌شود. جدول ۱۱ نتایج حاصل از این بررسی را نشان می‌دهد.

۷- نتیجه‌گیری

در ارزیابی‌های انجام‌شده، اثر تشخیص صحیح نیازهای سیستم بر روی معماری بررسی شده است. این بررسی‌ها نشان می‌دهد که اگر نیازها به جای دسته‌بندی‌های رایج (مانند دسته‌بندی کیفی/ عملکردی) بر اساس محاسبات عددی تحلیل و دسته‌بندی شوند نه تنها معماری نهایی از کیفیت بالایی برخوردار خواهد بود بلکه قابلیت ردگیری هر یک از تصمیمات طراحی تا سطح نیازمندی‌های سیستم به صورت واضح ایجاد می‌شود. این قابلیت ردگیری سبب می‌شود که ارزیابی سیستم با سهولت و وضوح بیشتری انجام شود و اگر در زمان اجرا لزوم تغییر یک نیازمندی ایجاد شود، اثرات ناشی از آن نیازمندی در سیستم پیاده‌سازی شده قابل تشخیص باشد. به این صورت، شاخه مربوط به آن نیازمندی از سطح تحلیل نیاز تا سطح طراحی و پیاده‌سازی برای تغییر مشخص می‌شود.

در این مطالعه، معیار دسته‌بندی نیازها و اولویت تأثیر آنها در معماری بر اساس امتیاز و ضریب اهمیتی است که برای نیازها تعریف شده است. ابزاری که از آن با عنوان قالب boiler-plate یاد کرده‌ایم قالبی استاندارد برای جمع‌آوری، دسته‌بندی و به طور کلی ساختاربندی نیازمندی‌ها ارائه می‌کند و با استفاده از این قالب، دسته‌بندی مشخصی برای نیازها ارائه می‌شود. به این ترتیب، میزان پوشش نیازهای بررسی‌شده در مقایسه با نیازهای واقعی قابل مشاهده و ردگیری است به گونه‌ای که اگر نیازهای مربوط به یکی از عناوین، پوشش داده نشده باشد و یا بیش از حد به آن پرداخته شده باشد، به وضوح مشخص خواهد بود. در نهایت با ارزیابی معماری که مبتنی بر روش ATAM انجام شده است مشاهده می‌شود که انتخاب بعضی از نیازها به عنوان نیازهای اساسی، در اتخاذ تصمیمات معماری بسیار مؤثر عمل کرده است به نحوی که در نقاط حساس و نقاط ریسک در سناریوهای متفاوت بهترین تصمیم گرفته می‌شود.

این مقاله به بررسی روشی برای تشخیص، مستندسازی، اندازه‌گیری و تحلیل نیازمندی‌های سیستم‌های نرم‌افزاری می‌پردازد. استفاده از این روش در سیستم‌های خودانطباق سبب می‌شود که تشخیص تغییرات لازم برای ایجاد انطباق در زمان اجرا، ناقض محدودیت منابع و زمان- که مهم‌ترین چالش‌های انطباق در زمان اجرا هستند- نباشد.

1. Client-Server

جدول ۱۰: درخت سودمندی در ارزیابی مبتنی بر روش ATAM.

۱	ویژگی کیفی	Availability
	جزئیات	تمام دستورات توسط سیستم هوشمند مجازی "مراکز اتومبیل کرایه" اجرا می‌شوند.
	سناریوها	سناریوی ۱) زمانی که در یک مؤلفه یکی از دستورات در اثر بروز خطا اجرا نمی‌شود، یک دستور دیگر با عملکرد همان دستور جایگزین و اجرا می‌شود. سناریوی ۲) زمانی که در یک مؤلفه یکی از دستورات در اثر بروز خطا اجرا نمی‌شود، یک مؤلفه جایگزین آن دستور را با مقادیر نزدیک به مقادیر واقعی اجرا می‌کند.
۲	ویژگی کیفی	Maintability, Modifiability
	جزئیات	سیستم هوشمند مجازی "مراکز اتومبیل کرایه" باید از قابلیت بالایی برای تغییر و اصلاح برخوردار باشد.
	سناریو	سناریوی ۱) برنامه سیستم مبتنی بر مؤلفه‌های مستقل نوشته شود به نحوی که تغییر یک مؤلفه مستقل از سایر مؤلفه‌ها و به سادگی انجام شود. سناریوی ۲) هر مؤلفه از برنامه سیستم بر اساس استانداردها و بسته‌های آماده موجود در بازار نوشته شود تا با تغییر شرایط، یک مؤلفه استاندارد دیگر جایگزین آن شود.
۳	ویژگی کیفی	Availability, Configurability
	جزئیات	در هنگام اعمال تغییرات در آدرس یک شخص، تغییرات در کمتر از ۱ دقیقه به تمام زیرسیستم‌ها اعلام می‌شود.
	سناریو	سناریوی ۱) یک مؤلفه مرکزی مسئول اطلاع‌رسانی رویداد تغییر هنگام تولید به تمام زیرسیستم‌ها است. سناریوی ۲) چند مؤلفه که با هم Synch هستند اما یکسان نیستند مسئول اطلاع‌رسانی رویداد تغییر هنگام تولید به زیرسیستم‌های خود هستند.
۴	ویژگی کیفی	Portability, Modifiability
	جزئیات	سیستم هوشمند مجازی "مراکز اتومبیل کرایه" به دلیل استفاده فراگیر، باید دارای قابلیت اجرا بر روی سیستم عامل‌های مختلف باشد.
	سناریو	سناریوی ۱) معماری سیستم بر اساس لایه‌بندی و استفاده از پارتیشن‌بندی برای مخفی‌کردن لایه‌های گرافیکی سطح بالا انجام شود. سناریوی ۲) اطلاعات به صورت متنی و تقسیم‌شده به بخش‌های کم‌حجم به دستگاه‌های گیرنده اطلاعات فرستاده شود.
۵	ویژگی کیفی	Portability, Scalability, Reusability
	جزئیات	سیستم هوشمند مجازی "مراکز اتومبیل کرایه" باید دارای قابلیت استفاده مجدد توسط سایر شهرهای کشور باشد.
	سناریو	سناریوی ۱) سیستم دارای یک هسته مرکزی باشد. این هسته مرکزی عبارت است از یک template برای سیستم که با توجه به نیاز هر شهر پیاده‌سازی می‌شود. سناریوی ۲) هر مؤلفه از برنامه سیستم بر اساس استانداردها و بسته‌های آماده موجود در بازار نوشته شود تا با تغییر شرایط، قابل شخصی‌سازی باشد و یا یک مؤلفه استاندارد دیگر جایگزین آن شود.

جدول ۱۱: مقایسه سناریوهای درخت سودمندی با تصمیمات معماری.

اندیس در درخت سودمندی		سناریوی ۱	سناریوی ۲
۱	حساسیت	وجود خطا در مقدار متغیرهای ورودی	تغییر در پاسخ به دلیل تغییر در مقدار متغیرها
	ریسک	اجرانشدن دستور	تولید پاسخ اشتباه
۲	حساسیت	دشواربودن و زمان‌بربودن پیاده‌سازی	گران‌بودن پیاده‌سازی
	ریسک	بیشترشدن زمان پیاده‌سازی از برنامه زمان‌بندی پروژه	بیشترشدن هزینه پیاده‌سازی از برنامه زمان‌بندی پروژه
۳	حساسیت	مؤلفه مرکزی و duplication مؤلفه مرکزی	تضمین ارتباطات دایمی میان مؤلفه‌ها
	ریسک	ایجاد مشکل برای مؤلفه مرکزی	ایجاد مشکل در ارتباطات میان مؤلفه‌ها
۴	حساسیت	انجام صحیح لایه‌بندی و پارتیشن‌بندی	تقسیم‌بندی اطلاعات به بخش‌های کم‌حجم و قابل فهم توسط کاربر
	ریسک	اجرانشدن برنامه یا کند اجراشدن برنامه بر روی بعضی سیستم عامل‌ها	قابل درک‌نبودن بعضی از پیغام‌ها برای کاربران
۵	حساسیت	خاص‌بودن یا کلی‌بودن template تهیه‌شده بیشتر از حد نیاز	گران‌بودن پیاده‌سازی
	ریسک	عدم امکان پیاده‌سازی بخشی از template که به اندازه کافی کلی نیست و خاص‌منظوره طراحی شده است. بیشترشدن زمان پیاده‌سازی از برنامه زمان‌بندی پروژه در صورت کلی‌بودن template	بیشترشدن هزینه پیاده‌سازی از برنامه زمان‌بندی پروژه

مراجع

[1] B. Chen, X. Peng, Y. Yu B. Nuseibeh, and W. Zhao, "Self-adaptation through incremental generative model transformations at runtime," in *Proc. 36th ACM/IEEE Int. Conf. on Software Engineering*, 12 pp., Hyderabad, India, 31 May-7 Jun. 2014.

[2] K. Welsh, R. Casallas, and P. Sawyer, "Understanding the scope of uncertainty in dynamically adaptive systems," In: R. Wieringa and A. Persson, eds., *Requirements Engineering: Foundation for Software Quality,* Lecture Notes in Computer Science, vol. 6182, pp. 2-16, 2010.

[3] N. M. Villegas, H. A. Muller, and G. Tamura, "Optimizing run-time SOA governance through context-driven SLAs and dynamic monitoring," in *Proc. IEEE Int. Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA'11*, 10 pp., Williamsburg, VA, USA, 26-26 Sept. 2011.

[4] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41-50, Jan. 2003.

[5] H. Tajalli, J. Garcia, and G. Edwards, "PLASMA: a plan-based layered architecture for software model-driven adaptation," in *Proc. of the IEEE/ACM Int. Conf. on Automated Software Engineering, ASE'10*, pp. 467-476, Antwerp, Belgium, 20-24 Sept. 2010.

[6] A. Elkhodary, N. Esfahani, and S. Malek, "FUSION: a framework for engineering self-tuning self-adaptive software systems," *ACM SIGSOFT Int. Symp. on Foundations of Software Engineering, FSE'10*, pp. 7-16, Santa Fe, NM, USA, 7-11 Nov. 2010.

[7] B. Morin, O. Barais, G. Nain, and J. M. Jezequel, "Taming dynamically adaptive systems using models and aspects," in *Proc. Int Conf. on Software Engineering, ICSE'09*, pp. 122-132, Vancouver, BC, Canada, 16-24 May 2009.

[8] R. Lemos, *et al.*, "Software engineering for self-adaptive systems: a second research roadmap," In R. de Lemos et al., eds., *Self-Adaptive Systems,* LNCS 7475, Springer-Verlag, Berlin, pp. 1-32, 2011.
[9] B. H. Cheng, *et al.*, "Software engineering for self-adaptive systems: a research roadmap," In: B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J.Magee, eds., *Software Engineering for Self-Adaptive Systems*, LNCS 5525, p. 1-26, Springer-Verlag, 2009.
[10] N. Bencomo and A. Blaggoun, "A world full of surprise: bayesian theory of surprise to quantify degrees of uncertainty," in *Companion Proc. of the 36th Int. Conf. on Software Engineering*, pp. 460-463, Hyderabad, India, 31 May-7 Jun. 2014.
[11] M. Shaw, *The Role of Design Spaces*, IEEE Computer Society, 2012.
[12] N. Bencomo, A. Blaggoun, and V. Issarny, "Dynamic decision networks for decision-making in self-adaptive systems: a case study," in *Proc. of the 8th Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems, SEAMS'13*, pp. 113-122, San Francisco, CA, USA, 20-21 May 2013.
[13] T. Preisler and W. Renz, "Structural adaptations for self-organizing multi-agent systems," *International Journal on Advances in Intelligent Systems*, vol. 8, no. 3-4, 8 pp., 2015.
[14] M. Ahmad, N. Belloir, and J. Brue, "Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems," *J. of Software and Systems*, vol. 107, pp. 50-70, Sept. 2015.
[15] S. Hassan, N. Bencomo, and R. Bahsoon, "Minimizing nasty surprises with better informed decision-making in self-adaptive systems," in *Proc. of the 10th Int. Symp. on Software Engineering for Adaptive and Self-Managing System,s SEAM'15*, pp. 134-144, Florence, Italy, 16-24 May 2015.
[16] J. L. H. Garcia Paucar, N. Bencomo, and K. Kam Fung Yuen, "Runtime models based on dynamic decision networks: enhancing the decision-making in the domain of ambient assisted living applications," in *Proc. of the 11th Int. Workshop on MoDELS,* pp. 9-17, Saint Malo, France, 4-4 Oct. 2016.
[17] N. Bencomo and A. Blaggoun, "Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks," In J. Doerr, A. L. Opdahl, eds., *Requirements Engineering: Foundation for Software Quality,* LNC7830. Springer, Berlin, 2013.
[18] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Proc. of Future of Software Engineering, FOSE'07*, pp. 259-268, 23-25 May 2007.
[19] K. Welsh, P. Sawyer, and N. Bencomo, "Towards requirements aware systems run-time resolution of design-time assumption," in *Proc. of the 26th IEEE/ACM Int. Conf. on Automated Software Engineering, ASE'11*, pp. 560-563, Washington, DC, USA, 6-10 Nov. 2011.
[20] A. J. Ramirez, *et al.*, "Relaxing claims coping with uncertainty while evaluating assumptions at runtime," In R. B.: France, J. Kazmeier, R. Breu, C. Atkinson, eds., LNCS 7590, pp. 53-69, 30 Sept.-5 Oct. 2012.
[21] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. Bruel, "RELAX: incorporating uncertainty into the specification of self-adaptive systems," in *Proc. of the 2009 17th IEEE Int. Requirements Engineering Conf., RE'09*, pp. 79-78, Atlanta, GA, USA, 31 Aug.-4 Sept. 2009.
[22] S. J. I. Herzig and C. J. J. Paredis, "Bayesian reasoning over models," in *Proc. Workshop on Model-Driven Engineering, Verification, and Validation*, pp. 69-80, Valencia, Spain, 30 Sept. 2014.
[23] L. Pimentel, D. Ros, and M. Seruffo, "Wired/wireless internet communications," in *Proc. Int. Conf. on Wired/Wireless Internet Communication,* Malaga, Spain, 25-27 May, 2015.
[24] T. L. Saaty, "Decision making for leaders: the analytic hierarchy process for decisions in a complex world," *International J. of Services Sciences*, vol. 1, no. 1, pp. 83-98, 2008.
[25] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice (3rd Edition)*, SEI Series in Software Engineering, Addison-Wesley Professional, 2012.

رایحه معین‌فر تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد در رشته علوم کامپیوتر – سیستم‌های هوشمند به‌ترتیب در سال‌های ۱۳۸۷ و ۱۳۸۹ از دانشگاه صنعتی امیرکبیر به پایان رسانده است و از سال ۱۳۹۱ تاکنون در مقاطع دکتری علوم کامپیوتر – مهندسی نرم افزار در دانشگاه صنعتی امیرکبیر مشغول به تحصیل می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: هوش مصنوعی، مهندسی نرم افزار، مهندسی نیازمندی‌ها، کیفیت نرم‌افزار، سیستم‌های نرم‌افزاری خودانطباق.

احمد عبداله‌زاده بارفروش در سال ۱۳۵۴ مدرک کارشناسی حسابداری مهاد ریاضی خود را از دانشگاه تهران و در سال ۱۳۵۹ مدرک کارشناسی ارشد مهندسی کامپیوتر – برنامه‌نویسی خود را از دانشگاه وست‌کاست در لس‌آنجلس آمریکا دریافت نمود. در سال ۱۳۶۹ موفق به اخذ درجه دکترا در مهندسی کامپیوتر – سیستم های هوشمند از دانشگاه بریستول انگلستان گردید. در همان سال به ایران بازگشت و عضو هیأت علمی دانشگاه صنعتی امیرکبیر گردید. وی دبیر کمیسیون اقتصاد دیجیتال کمیته ایرانی اتاق بازرگانی بین‌المللی، عضو هیئت مدیره انجمن تجارت الکترونیک ایران و عضو هیأت مؤسس انجمن تجارت الکترونیک ایران می‌باشد. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعات هوش مصنوعی توزیع شده، سیستم‌های چندعامله، مهندسی نیازمندی‌ها، کیفیت نرم‌افزار و هوش تجاری می‌باشد.

سیدمهدی تشکری هاشمی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد در رشته ریاضی به‌ترتیب در سال‌های ۱۳۶۷ و ۱۳۶۹ از دانشگاه مشهد به پایان رسانده است. در سال ۱۳۷۷ موفق به اخذ درجه دکترا در رشته علوم کامپیوتر- ریاضیات کاربردی از دانشگاه اوتاوا در کانادا شد. در همان سال به ایران بازگشت و عضو هیات علمی دانشگاه صنعتی امیرکبیر گردید. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعات مهندسی نرم‌افزار، بهینه‌سازی ترکیبیاتی و شبکه‌های حمل‌ونقل هوشمند می‌باشد.