گزارش تبیین پذیری سیستم‌های نرم‌افزاری: از آنالیز نیازمندی‌ها تا ارزیابی سیستم

سودابه آشوری علیرضا سلطانی نشان ملیکا محمدی گل
a.soltani@iau-tnb.ac.ir

۹ فروردین ۱۴۰۳

۱ مقدمه و بیان مسئله

دوران الگوریتمیک یا Algorithmic age منحصر به زمان کنونی و آینده جهان است، به گونه‌ای که الگوریتم‌های گوناگونی در زندگی انسان‌ها وارد شده‌اند. از ساده‌ترین عملیات گرفته تا پیچیده‌ترین تصمیم‌گیری‌ها را می‌توان پیاده‌سازی کرد. این الگوریتم‌ها حوزه‌های مختلف از قبیل اقتصاد، بهداشت، حمل و نقل و غیره را تحت تاثیر خود قرار داده است. از انتخاب کوتاه‌ترین مسیر رانندگی گرفته تا پیشبینی و تشخیص سرطان و طراحی پروتئین و داروهای مختلف.

این برگه به روشنی از قابلیت‌های این سیستم‌ها، که تصمیم‌گیری‌های مختلف را به گونه‌ای انجام می‌دهند که کاربر نمی‌داند که این پیشبینی چگونه رخ داده است یا این ماشین با چه داده‌هایی به چنین نتیجه‌ای رسیده است را Black box systems می‌نامد. زمانی که کاربر نمی‌تواند متوجه شود که پشت صحنه این پیشبینی‌ها به چه شکلی می‌باشد می‌گوییم که این سیستم‌ها فاقد شفافیت [۱] هستند. فاقد شفافیت نگرانی‌هایی را در رابطه با پارامترهایی مانند مسئولیت‌پذیری، انصاف و پیامدهای اخلاقی ایجاد می‌کند به ویژه در حوزه‌هایی که سیستم تصمیماتی را می‌گیرد که می‌تواند روی مردم و جامعه به طور کلی تاثیر گذار باشد.

بر همین اساس، این برگه بحث‌هایی را در مورد شفافیت و اخلاق سیستم‌های مدرن مطرح کرده است. درک چگونگی ادغام این نگرانی‌ها در سیستم‌ها و در نتیجه نحوه برخورد با آنها در طول مهندسی نرم‌افزار و مهندسی نیازمندی‌ها بسیار مهم است.

[۱] Transparency

فهرست مطالب

۲ رویکرد و راه‌حل

رویکر و روش‌شناسی‌ای که این مقاله در مورد آن صحبت می‌کند تبیین‌پذیری در سیستم‌های نرم‌افزاری و حتی مدل‌های هوش مصنوعی است تا بتواند ضعف عدم شفافیت سیستم‌ها را رفع کند.

۱.۲ تبیین‌پذیری چیست؟

تبیین‌پذیری یک روش مفید است تا از نگرانی‌های اخلاقی نرم‌افزارها و مدل‌ها بکاهد. به معنای قابلیت شرح نرم‌افزار و سیستم است. وقتی یک سیستم یا مدل هوش مصنوعی تبیین‌پذیر است، به این معناست که عملکرد و تصمیمات آن قابل تفسیر و توجیه است. به عبارت دیگر، می‌توان به راحتی فهمید که یک سیستم به چه شکلی کار می‌کند و چگونه به تصمیمات خود رسیده است. تبیین‌پذیری یک ویژگی بسیار مهم در سیستم‌های نرم‌افزاری است که موجب افزایش اعتماد به آن می‌شود و ارزش‌های اخلاقی و قانونی را در رابطه با سیستم تعریف خواهد کرد. امروزه به مسئله تبیین‌پذیری سیستم‌ها بسیار اهمیت داده می‌شود و یکی از مهم‌ترین نیازمندی‌های Non-functional محسوب می‌شود. در حالتی که به کاربران این اجازه را می‌دهد که خودشان بتوانند انتخاب کنند که از این سیستم استفاده کنند یا از آن دوری کنند چرا که بر روی رابطه قابلیت اعتماد و اتکای سیستم بسیار تاثیرگذار می‌باشد.

نکته: با توجه به قدرت هوش مصنوعی در تمام حوزه‌های زندگی بشر، تبیین‌پذیری به عنوان یکی از مهم‌ترین پایه‌های اعتماد در نیازمندی‌های نرم‌افزار می‌باشد.

همچنین در این مقاله در مورد رابطه بین جنبه‌های کیفی و تبیین‌پذیری صحبت می‌شود.

۲.۲ چالش‌های تبیین‌پذیری

دلایل زیر نشان‌دهنده آن است که جمع‌آوری و استخراج داده، مذاکره و اعتبارسنجی در فرایند تبیین‌پذیری با چالش‌هایی رو به رو می‌باشد:

۱.۲.۲ پیچیدگی سیستم‌ها

در سیستم‌هایی که مبنی بر هوش مصنوعی و فرایند یادگیری ماشین هستند با وجود الگوریتم‌های مختلف که وظیفه تصمیم‌گیری را در سیستم دارند، سطح پیچیدگی بسیار بالا می‌باشد. درک و توضیح این سیستم‌ها با فرایندهایشان برای کاربران مختلف به مفهوم ساده، بسیار سخت و غیرقابل درک می‌باشد.

۲.۲.۲ طبعیت Black box

از نظر کاربران، بسیاری از الگوریتم‌ها به شکل جادویی عمل می‌کنند، بدان معنا که فرایندهای داخلی این الگوریتم‌ها کاملا به صورت مات می‌باشد و توسط انسان بدون دانش قبلی به راحتی قابل درک نیست.

۳.۲.۲ زمینه‌گرایی توضیح یا Subjectivity of Explanation

زمینه‌گرایی توضیح به معنای نسبی بودن یا وابستگی توضیحات به نگرش و دیدگاه فردی است. در حالت کلی تفسیر هر چیزی توسط ذینفعان می‌تواند کاملا متفاوت از نظر معنا و دیدگاه باشد. مذاکره برای به اجماع رسیدن در سطح و نوع توضیح مورد نیاز می‌تواند چالش برانگیز باشد، به ویژه زمانی که با دیدگاها و علایق گوناگون سروکار داریم.

عدم درک مشترک [۲]

یکی دیگر از دشواری‌ها، ارتباطات مناسب در مهندسی نیازمندی است. ذینفعان بیرونی و تیم توسعه ممکن است ناخواسته از یکسری کلمات متفاوت با مفهوم یکسان استفاده کنند که در نهایت باعث ایجاد سوءتفاهم و نقص فهم مشترک بین افراد شود که در نهایت چالشی برای

[۲] Lack of shared understanding

ارتباط با یکدیگر ایجاد می‌کند. لازمه کارآمدی ارتباطات درک مشترک از مفاهیم می‌باشد که ریسک دوباره‌کاری و نارضایتی ذینفعان را کاهش می‌دهد.

رویکرد درک مشترک بین افراد

مهندسان نرم‌افزار می‌توانند مجموعه‌ای از فرآورده‌ها را ایجاد کنند که باعث ایجاد درک و فهم مشترک در پروژه‌های نرم‌افزاری می‌شود و بارها قابل استفاده مجدد و اصلاح خواهند بود تا فرآورده‌ها، محصولی از مذاکره با زبانی مشترک بین افراد باشد.

فرآورده‌ها

فرآورده‌ها هر گونه اسناد متنی و اشکال گرافیکی هستند که به دور از کدها و محصولاتی نرم‌افزاری، ابزاری برای مذاکره بین تمام افراد حاضر (چه ذینفعان چه مهندسان مختلف) می‌باشند. محتوای فرآورده‌ها معمولا اشکال، متن‌ها، مدل‌های بصری، فهرست‌ها، چارت‌ها ، چهارچوب‌ها و مدل‌های کیفیت می‌باشد. این فرآورده‌ها در شکل‌دهی ساختار پروژه بسیار کارآمد هستند به گونه‌ای که در فرایندهای مهندسی نیازمندی از قبیل، مدل‌های مفهومی [۳] کاتالوگ دانش [۴] و مدل‌های مرجع [۵] کاربرد متعددی دارند.

۴.۲.۲ تریدآف همراه با تاثیرگذاری روی عملکرد یا Trade-off with Performance

گاهی افزایش تبیین‌پذیری در یک سیستم می‌تواند به قیمت عملکرد و کارایی تمام شود. یک مهندس نیازمندی باید بتواند بین تبیین‌پذیری با سایر الزامات سیستم System requirements تعادل ایجاد کند. برای درک این چالش مثال زیر را مطالعه کنید:

تصور کنید یک شرکت در حال توسعه سیستم توصیه‌گرا برای اپلیکیشن تجاری خود می‌باشد. این سیستم الگوریتم‌های پیچیده ML را برای تحلیل رفتارها و ترجیحات [۶] کاربران استفاده می‌کند تا بتواند محصولات مشابه علاقه‌مندی آنها را به نحوی معرفی کند که کاربران انتظار داشتند. یکی از نیازمندی‌های NFR این سیستم، ارائه توضیحات برای هر کدام از نتایج محصولات توصیه شده می‌باشد تا بتواند موجب اعتماد و رضایت کاربران شود. در این صورت گنجاندن توضیحات به همراه جزئیات چرایی انتخاب این مورد (محصول) به عنوان مورد مرتبط برای این سیستم تاثیر به سزایی در عملکرد آن خواهد داشت. این عمل باعث تاخیری در تولید این موارد برای کاربران می‌شود که از نظر تجربه کاربری [۷] یک ضعف محسوب می‌شود به ویژه زمانی که کاربران انتظار دارند که تمام تقاضاهایشان از سیستم در کمتر از پنج ثانیه پاسخ داده شود.

احتمالاً برای این مثال راهکارهای زیر در نظر گرفته می‌شود تا ضمن تبیین‌پذیری سیستم، عملکرد سیستم نیز مانند سابق با سرعت بالا حفظ شود:

۱. کاهش پیچیدگی توضیحات: به جای آنکه توضیحات کاملی در مورد عملکرد الگوریتم‌های هوش مصنوعی به ازای هر مورد فراهم شود، سیستم می‌تواند بسیار ساده با ارائه خلاصه‌ای مفید، فاکتورهای مهم و اساسی دلیل انتخاب موارد به عنوان توصیه کاربر را مشخص کند.

۲. استفاده از متدهای فنی در مهندسی نرم‌افزار مانند فرایندهای کش کردن انتخاب‌های کاربر (براساس کلیک‌های مختلف روی محصولات یا مدت زمانی که روی محصول مورد نظر کاربر مطالعه داشته) محاسبات از پیش تعیین شده‌ای در مورد چرایی انتخاب محصول به عنوان توصیه را مشخص کند.

انتخاب استراتژی مناسب برای حفظ تبیین‌پذیری به همراه سرعت و کارایی بالا در عملکرد سیستم توصیه‌گرا، یک تریدآفی است که وظیفه آن بر عهده تیم توسعه، طراح و معماری نرم‌افزار می‌باشد.

---

[۳] Conceptual models
[۴] Knowledge catelogues
[۵] Reference models
[۶] Preferences
[۷] User experience

۵.۲.۲ سیستم‌های پویا و در حال تکامل

یکی از چالش‌های مهم تبیین‌پذیری سیستم‌هایی است که در طول زمان دچار تغییرات کلی به ویژه در نیازمندی‌ها می‌شوند. اینکه توضیحات متناسب با تغییر سیستم‌ها به روز شود چالشی مهم است.

۶.۲.۲ اعتبارسنجی و اعتماد

اعتبار بخشیدن به توضیحات ارائه شده توسط یک سیستم می تواند دشوار باشد، به ویژه زمانی که آنها شامل فرآیندها یا داده های پیچیده باشند. ایجاد اعتماد در این توضیحات مستلزم روش‌های اعتبارسنجی قوی و شفافیت در فرآیند تولید توضیح است.

همچنین اشاره می‌کند که مهندسی نیازمندی فرایند ساده برای شناسایی و مشخص کردن نیازمندی‌ها نیست، بلکه فرایندی جهت حمایت از ارتباطات کارآمد این نیازمندی‌ها بین ذینفعان مختلف می‌باشد.

۳ سابقه دانشی

برای اجماع فهم مشترک بر مسائل مختلف این حوزه از مهندسی نرم‌افزار، خواننده نیاز دارد که با مفاهیم زیر به صورت کلی آشنا باشد تا بتواند:

۱. از دانشی فراتر از زمینه‌های خود استفاده کنند و از این دانش برای رفع نیازهای یک پروژه خاص (جاری یا جدید) استفاده کنند.

۲. دستیابی به درکی مشترک که منجر به ارتباطات بهتر و تعریف نیازمندی‌های سیستم به شکل «درست» می‌شود.

۱.۳ تعاریف یا Definitions

تعاریف در SE و RE، راهنمایی تقریبی برای مهندسان نرم‌افزار در مورد دامنه، عناصر و هر چیز دیگری را ارائه می‌دهند. به این صورت که یکی از مهم‌ترین مراحل تسهیل ارتباطات برای یک موضوع یا یک مفهوم می‌باشند. وقتی در مورد تعریف جنبه‌های کیفی یا Quality aspects صحبت می‌شود در حقیقت منظور همان راهنمایی‌ها برای مهندسان نرم‌افزار است که به آنها در فهمیدن روند مهندسی نیازمندی به خصوص تضمین کیفیت یا Quality assurance کمک می‌کند. عدم اجماع حاضرین بر سر مفاهیم و تعاریف می‌تواند سبب ایجاد نتیجه نامناسبی در مشخصات و یکپارچگی نیازمندی‌های غلط شود. برای مثال وقتی در جلسات در مورد Usability صبحت می‌شود، برخی از افراد توسعه دهنده منظور را در رابطه با استفاده و بهره‌وری بلند مدت [۸] می‌دانند و برخی از افراد منظور را در استفاده آسان محصول [۹] می‌دانند.

۲.۳ مدل‌ها یا Models

یک مدل در بالاترین سطح تجرید در مورد کارایی سیستم تمرکز دارد که بتواند تمام جنبه‌های سیستم را به سادگی و به دور از جزئیات نمایش دهد. هیچ وقت یک مدل به تنهایی، تمام سیستم را تشریح نمی‌کند. مدل‌ها در طیف گسترده‌ای از جنبه‌های مختلف یک سیستم استفاده می‌شوند. مدل‌ها می‌توانند برای اهداف توسعه نرم‌افزار و توسعه کسب و کار استفاده شوند. از نظر نرم‌افزاری می‌توانند نقش مهمی در تعریف ساختار نرم‌افزار یا پیکربندی‌ها داشته باشند و از سوی دیگر برای توصیف و بهینه‌سازی نگرانی‌های سازمانی مانند فرآیندها و حوزه‌های تجاری می‌توانند بسیار مفید باشند. از انواع مدل‌ها می‌توان به موارد زیر اشاره کرد:

۱.۲.۳ مدل‌های مفهومی یا Conceptual models

مدل‌هایی هستند که برای تعریف و توصیف یک مفهوم استفاده می‌شوند. افراد را قادر می‌سازند که بتوانند ساختار و ویژگی‌های یک جنبه کیفی مشخص را در طول فرآیند تحلیل نیازمندی درک کنند. دانش مورد نیاز برای توسعه مدل‌های مفهومی معمولاً از Literature، تجارب قبلی مشابه و حوزه تخصصی استخراج می‌شوند.

[۸] Long-term efficiency
[۹] Ease of Use (EoU)

۲.۲.۳ مدل‌های کیفی یا Quality models

مدل‌هایی که صفات کیفی و ویژگی‌های نیازمندی نرم‌افزار را تعریف و مشخص می‌کنند. این مدل‌ها معمولاً روش‌های سیستماتیکی را برای ارزیابی و اطمینان کیفی نیازمندی‌ها بر اساس استانداردی معتبر بکار می‌گیرند. این مدل‌ها به ذینفعان کمک می‌کنند تا درک کنند که لازمه بالا بودن کیفیت خدمات چیست و دستورالعمل‌هایی را برای ایجاد، تجزیه و تحلیل و اعتبارسنجی نیازمندی‌ها ارائه می‌دهد.

۱. استاندارد ISO/IEC 25010 (SQuaRE): یک استاندارد جامع برای نیازمندی‌ها و ارزیابی کیفیت نرم‌افزار است. ویژگی‌های مورد بررسی آن از قبیل، عملکرد، قابلیت استفاده، کارایی، قابلیت نگهداری و غیره می‌باشند.

۲. مدل Quality Function Deployment (QFD): رویکردی مشتری محور است که مهم‌ترین وظیفه آن ترجمه نیازهای مشتری به نیازمندی‌های محصول خاص است. این مدل به مهندسان مخصوصاً مهندسان نیازمندی کمک می‌کند تا بتوانند تطابق نیازمندی‌های پیاده‌سازی شده را با انتظارات و ترجیحات مشتریان بررسی کنند.

۳. McCall's Quality Model: این مدل توسط جان.دی.مک‌کال مطرح شده است و ۱۱ عامل کیفی را که در سه دسته: عملکرد محصول، بازبینی محصول و جا به جایی و انتقال محصول قرار دارند، را شناسایی می‌کند. این عوامل شامل قابلیت اعتماد، قابلیت استفاده، کارایی و ویژگی‌های مشابه با استاندارد SQuaRE هستند.

۴. IEEE 730 Standard: این استاندارد فرآیندهای اطمینان کیفیت را برای پروژه‌های توسعه نرم‌افزار از جمله مهندسی نیازمندی‌ها را تعریف می‌کند. فعالیت‌های مرتبط با برنامه‌ریزی کیفیت، اطمینان کیفیت و کنترل کیفیت را در طول چرخه‌ی عمر توسعه نرم‌افزار شامل می‌شود.

۳.۲.۳ مدهای مرجع یا Reference models

۳.۳ کاتالوگ‌ها یا Catalogues

کاتالوگ دانش مجموعه‌ای سازمان‌دهی شده از منابع دانشی است که درون یک سازمان وجود دارد. این منابع می‌توانند شامل انواع دانش‌ها مانند اسناد، گزارش‌ها، روش‌های توسعه و بهترین رویکردهای حل مسئله، مواد آموزشی و موارد دیگر باشند. هدف اصلی از کاتالوگ‌ها تسهیل در توسعه، به اشتراک‌گذاری و استفاده مجدد از منابع دانش در یک سازمان در پروژه‌های مشابه می‌باشد.