فصل ۱۱ اصول و مفاهیم تحلیل

مفاهیم کلیدی (مرتب بر حروف الفبا)

استخراج نیازمندیها ، اصول تحلیل ، اصول تعیین مشخصه ها ، بازبینی مشخصه ها ، پیاده سازی دیدگاه FAST ، تجزیه و تفکیک ، دامنه اطلاعات ، دیدگاه بنیادین ، گسترش کارکرد کیفیت (QFD) ، نمونه سازی ، Use-case

KEY CONCEPTS

Analysis principles , essential view , FAST implementation view , information domain , partitioning , prototyping requirements elicitation , QFD , Specification principles , Specification review , use-case

نگاه اجمالی

نقش کلی نرم‌افزار در یک سیستم بزرگ‌تر در طول مهندسی سیستم مشخص می‌شود (فصل ۱۰). در هر حال، نگاهی عمیق‌تر به نقش نرم‌افزار ضرورت دارد ـ تا نیازمندیهای مشخص که باید برای دستیابی به نرم‌افزار با کیفیت بالا، برآورده شود، ادراک گردد. وظیفهٔ تحلیل نیازمندیهای نرم‌افزاری همین است. برای آن‌که این کار به‌خوبی انجام گیرد، شما باید از یک‌سری مفاهیم و اصول متابعت نمایید.

چه کسی کار را انجام می‌دهد؟ عموماً یک مهندس نرم‌افزار، تحلیل نیازمندیها را انجام می‌دهد. در هر صورت، برای به‌کارگیری آن در زمینهٔ کارهای تجاری پیچیده، یک "تحلیل‌گر سیستم" آموزش دیده در جنبه‌های تجاری حوزهٔ به‌کارگیری ـ می‌تواند این کار را انجام دهد.

چرا این کار اهمیت دارد؟ اگر شما تحلیل نکنید، بسیار محتمل است که نرم‌افزاری بسیار با شکوه فراهم آورید که مشکل دیگری را حل می‌کند. نتیجه آن‌که، پول و زمان هدر رفته است، دلسردی شخصی پیش آمده است و مشتریان ناراضی شده‌اند.

نقل قول

"این جمله خود را نقض می‌کند ـ نه در عمل چنین نیست." داگلاس هاف تادتر

مراحل کار چه می‌باشد؟ نیازمندیهای داده‌ها، کارکردی و رفتاری از طریق استخراج اطلاعات از مشتری، مشخص می‌شوند. نیازها پالایش شده و تحلیل می‌شوند تا وضوح، بی‌عیبی و سازگاری درونی آنها ارزیابی شود. یک روش تعیین مشخصات که مدلی از نرم‌افزار را در خود لحاظ کرده است، ایجاد شده سپس توسط مهندسین نرم‌افزار و مشتریان/ کاربران هر دو اعتبارات می‌یابد.

حاصل کار چیست؟ یک طریقه ارائه مؤثر نرم‌افزار باید به‌عنوان پیامد تحلیل نیازمندیها به‌وجود آید. مانند نیازمندیهای سیستم. نیازمندیهای نرم‌افزاری می‌توانند با استفاده از نمونهٔ اولیه، یک مشخص‌سازی یا حتی مدل نمادین ارائه شوند.

تحلیل نیازها و مشخص‌سازی می‌تواند کاری نسبتاً ساده به نظر برسد، اما گول ظاهر را نباید خورد. مضمون ارتباطی بسیار بالا است. احتمال بروز سوء تعبیر یا اطلاعات غلط فراوان است. ابهام محتمل است. معضلی که یک مهندس نرم‌افزار با آن مواجه می‌شود را می‌توان با تکرار جملهٔ یک مشتری ناشناس (به نام) ادراک کرد: "من می‌دانم که شما معتقدید آنچه را که فکر می‌کنید من گفته‌ام را فهمیده‌اید، ولی من مطمئن نیستم که شما دریافته باشید. آنچه شما شنیدید هر آنچه که منظور من بود، نبود ..."

مهندسی نیازمندیهای نرم‌افزاری، یک فرآیند کشف، پالایش، مدل‌سازی و مشخص‌سازی است. نیازمندیها سیستم و نقش واگذاری شده به نرم‌افزار ـ که ابتدا توسط مهندس سیستم پایه‌ریزی می‌شود ـ به‌طور تفصیلی پلایش شده‌اند. مدل‌های داده‌های مورد نیاز، اطلاعات، جریان کنترل و رفتار عملیاتی خلق شده‌اند. راه‌حل‌های آلترناتیو تحلیل شده و یک مدل تحلیلی کامل ایجاد شده است. دونالد ریفر[REI94][1] فرآیند مهندسی نیازهای نرم‌افزاری را به روش ذیل شرح می‌دهد:

مهندسی نیازمندیها، به‌کارگیری نظام مند اصول اثبات شده، تکنیک‌ها، زبانها و ابزارهای تحلیل هزینه مؤثر، تنظیم سند و تکامل جاری نیازهای مصرف‌کننده و مشخص ساختن رفتار برونی یک سیستم جهت جلب رضایت آن‌گروه از نیازهای کاربر می‌باشد. دقت کنید که مانند تمام اصول مهندسی، مهندسی نیازمندیها به طریقه‌ای پراکنده، تصادفی و یا اتفاقی نیست، بلکه به‌کارگیری نظام مند رهیافتهای اثبات شده است.

مشتری و مهندس نرم‌افزار هر دو نقش فعال در مهندسی نیازمندیها نرم‌افزار ایفاء می‌کنند ـ مجموعه‌ای از فعالیت‌هایی که غالباً از آنها تحت‌عنوان تحلیل یاد می‌کنیم. مشتری تلاش می‌کند تا شرح

داده‌ها، کارکرد و رفتار سیستم گاهی اوقات مبهم را دوباره فرموله کند. توسعه‌دهنده به‌عنوان پرسش‌گر، مشاور، حلال مشکلات و مذاکره‌کننده عمل می‌کند.

۱۱-۱ تحلیل نیازمندیها

تحلیل نیازمندیها[2]، وظیفهٔ مهندسی نرم‌افزار بوده که شکاف بین مهندسی نیازمندیهای سطح سیستم و طراحی نرم‌افزار را (شکل ۱۱-۱) پر می‌کند. فعالیت‌های مهندسی نیازمندیها منجر به مشخص‌سازی خصوصیات عملیاتی نرم‌افزار (کارکرد، داده‌ها و عملکرد) شده، بیانگر تعامل با دیگر عناصر سیستم بوده و

1.Reifer,D.J.

2.Requirements analysis

محدودیت‌های نرم‌افزار را تعیین می‌کند. تحلیل نیازمندیهای نرم‌افزاری می‌تواند به پنج قسمت تقسیم شود: (۱) فهم شکل، (۲) ارزیابی و تلفیق، (۳) مدل‌سازی، (۴) مشخص‌سازی و (۵) مرور.

ابتدا تحلیل‌گر[1] مشخصات سیستم[2] را مطالعه می‌کند (چنانچه وجود داشته باشد) و بعد طرح پروژه نرم‌افزار[3] را بررسی می‌کند. درک نرم‌افزار در یک بافت سیستم و بررسی حوزهٔ عمل نرم‌افزار که برای ایجاد و فراهم شدن محاسبات اولیه طراحی به‌کار رفته اهمیت دارد. سپس برای تحلیل باید ارتباطات برقرار شود تا از فهم مشکل اطمینان حاصل شود. هدف شناخت پیدا کردن از عناصر مشکل اصلی است آن‌گونه که توسط مشتری/ کاربر دریافت و استنباط می‌شود.

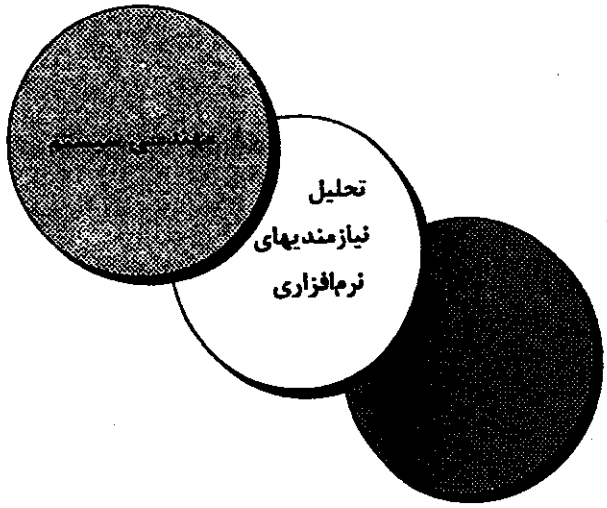
ارزیابی مشکل و یافتن راه‌حل، حوزهٔ اصلی تلاش بعدی مربوط به تحلیل است. تحلیل‌گر باید تمام جنبه‌های ملموس داده‌های قابل مشاهدهٔ برونی را تعریف کرده، جریان و مضمون اطلاعات را ارزیابی کرده، کلیه کارکردهای نرم‌افزار را تعریف کرده و در مورد آن به تفصیل شرح دهد، رفتار نرم‌افزار را در بافت حوادثی که روی سیستم اثر می‌گذارند درک نمایید؛ خصوصیات تعامل سیستم را تثبیت کرده و محدودیت‌های طراحی بیشتر را علنی نماید. هریک از این امور در خدمت تشریح مشکل هستند تا رهیافت کلی یا راه‌حل را بتوان بدست آورد.

برای مثال، یک سیستم کنترل فهرست موجودی برای یک تولیدکنندهٔ قطعات موتور اتومبیل، مورد نیاز است. تحلیل‌گر می‌یابد که مشکلات مربوط به سیستم دستی فعلی عبارت هستند از: (۱) عدم توانایی در کسب جایگاه اجزاء به‌طور سریع؛ (۲) صرف زمان دو یا سه روزه برای روزآمد کردن یک پرونده (۳) سفارشات مجدد چندگانه به فروشنده‌ای یکسان، چرا که هیچ روشی برای مربوط ساختن فروشندگان با اجزاء و غیره وجود ندارد. زمانی‌که مشکلات تعیین شوند، تحلیل‌گر اطلاعاتی را که باید توسط سیستم جدید تولید شوند و این‌که چه داده‌هایی به سیستم ارائه خواهد شد را تعیین می‌کند. برای مثال، مشتری خواستار یک گزارش روزانهٔ بیانگر قطعات برگرفته از فهرست موجودی و این‌که چه تعداد قطعهٔ مشابه باقی‌مانده است، می‌باشد. مشتری اظهار می‌دارد که کارمندان موجودی شمارهٔ شناسایی هر قطعهٔ حذف شده از حوزهٔ فهرست موجودی را، وارد خواهند کرد.

توصیه

طی مرحله تعیین نیازمندیها، اندکی به طراحی بگذرانید و طی مرحله طراحی ، اندکی به تعیین نیازمندیها.

1.analyst
2.System Specification
3.Software Project Plan



شکل ۱۱-۱ تحلیل به منزله پلی است بین مهندسی سیستم و طراحی نرم‌افزار

پرسش

در این مرحله نخستین دغدغه ما چه باید باشد؟

بر پایهٔ ارزیابی مشکلات جاری و اطلاعات مطلوب (ورودی ـ خروجی)؛ تحلیل‌گر شروع به یافتن یک یا چند راه‌حل می‌کند. در آغاز، اشیاءداده‌ای، کارکردهای پردازشی و رفتار سیستم به‌طور تفصیلی شرح داده می‌شوند. زمانی‌که این اطلاعات بدست آمد، معماری اولیه برای پیاده‌سازی آن در نظر گرفته می‌شود. یک راهکار خادم/مخدوم مناسب به نظر می‌رسد، ولی آیا نرم‌افزار معماری آن در تمام دامنهٔ عمل ترسیم شده در طرح نرم‌افزار حمایت می کند؟ یک سیستم مدیریت استوار بر داده‌ها (سیستم مدیریت پایگاه داده ها) به نظر ضروری می رسد، ولی آیا نیاز مشتری/ کاربر برای امکان برقراری ارتباط، توجیه شده است؟ پروسهٔ ارزیابی و راه‌حل‌یابی ادامه می‌یابد تا تحلیل‌گر و مشتری هر دو احساس اطمینان کنند که نرم‌افزار را می‌توان به‌طور کامل برای مراحل توسعه بعدی مشخص نمود.

در طی ارزیابی و یافتن راه‌حل، تمرکز اصلی تحلیل‌گر بر روی "چه" تا "چگونه" است. سیستم چه داده‌هایی را تولید و مصرف می‌کند. چه کارکردهایی تعریف شده‌اند و محدودیت‌هایی اعمال می‌شوند؟[1]

در طول دورهٔ ارزیابی و فعالیت یافتن راه‌حل، تحلیل‌گر مدل‌هایی از سیستم در تلاش برای درک بهتر داده‌ها و جریان کنترل، پردازش کارکردها، رفتار عملیاتی و مضمون اطلاعات را خلق می‌کند. این مدل به‌عنوان زیربنای طراحی نرم‌افزار و پایه خلق یک مشخص‌سازی برای نرم‌افزار، خدمت خواهد کرد.

در فصل ۲ متوجه شدیم که ارائه مشخصات تفصیلی شاید در این مرحل امکان‌پذیر نباشد. مشتری می‌تواند از آنچه دقیقاً می‌خواهد مطمئن نباشد. به‌وجود آورنده شاید نسبت به کارآیی مناسب یک رهیافت معین برای به انجام رساندن کارکرد و اجرای طرح خود اطمینان نداشته باشد. بنا به این دلایل و بسیاری

دلایل دیگر، یک رهیافت جایگزین برای تحلیل نیازمندیها بنام نمونهٔ اولیه‌سازی[2] می‌تواند مورد استفاده واقع شود. ما بعداً در این فصل دربارهٔ این رهیافت بحث خواهیم کرد.

۱۱-۲ تعیین نیازمندیهای نرم افزار

قبل از آن‌که بتوان نیازمندیها را تحلیل، مدل‌سازی و یا مشخص کرد باید آنها طی یک پروسهٔ استخراج گردآوری کرد. یک مشتری مشکلی دارد که می‌تواند با یک راه‌حل کامپیوتری مرتفع شود. یک تحلیل‌گر به تقاضای مشتری جهت کمک پاسخ‌می‌گوید. ارتباط شروع شده است، ولی همان‌طور که قبلاً خاطرنشان شد، جادهٔ ادراک معمولاً پر از چاله و دست‌انداز است.

۱۱-۲-۱ راه اندازی فرآیند

رایج‌ترین تکنیک استخراج نیازمندیها برگزاری نشست یا مصاحبه است. اولین ملاقات بین مهندس نرم‌افزار (تحلیل‌گر) و مشتری را می‌توان به ناخوشایندی اولین ملاقات دو نوجوان ارتباط داد. هیچ‌کدام نمی‌دانند که باید چه گفته یا سؤال کنند، هر دو نگران هستند که آنچه می‌گویند به‌خوبی درک نشود. هر دو به سرانجام دیدار خود فکر می‌کنند (هر دو احتمالاً در این‌جا دارای انتظارات کاملاً متفاوتی هستند)، هر دو می‌خواهند که کار را تمام کرده، اما در آن واحد هر دو می‌خواهند که ملاقاتشان قرین موفقیت باشد.

با این حساب، ارتباط باید شروع شود. گاز و وینبرگ [GAU89][3]، پیشنهاد می‌دهند که تحلیل‌گر ملاقات را با طرح سؤالات رها از مضمون اصلی[4]، شروع کند. یعنی آن‌که یک‌سری سؤالات که به شناخت اساسی مشکل، افرادی که برای آن راه‌حل می‌خواند، ماهیت راه‌حل مطلوب، و کارآیی خود اولین ملاقات منتهی خواهد شد. اولین سری سؤالات رها از مضمون اصلی بر روی مشتری، اهداف و منابع کلی متمرکز می‌شوند. مثلاً، تحلیل‌گر ممکن است سؤال کند:

- چه کسی تقاضای این کار را داشته است؟
- چه کسی از این راه‌حل استفاده خواهد کرد؟
- سود اقتصادی یک راه‌حل مؤفق چه خواهد بود؟
- آیا منبع دیگری برای راه‌حل مورد نیاز شما وجود دارد؟

نقل قول

سوال صریح و پاسخ صریح بهترین طریقه برطرف کردن پیچیده گر هاست.

مارک توین

این سؤالات به شناخت سهامدارانی که در این نرم‌افزار طراحی شده، منافعی دارند کمک می‌کند. به‌علاوه، سؤالات، سود قابل محاسبهٔ پیاده‌سازی موفق و جایگزین‌های محتمل توسعه تهیهٔ نرم‌افزار را تعیین

---

۱. داویس [DAV93] معتقد است که واژه های "چه" و "چگونه" بسیار مبهم و گنگ می باشند. خوانندگان مشتاق این امر، می توانند به کتاب او مراجعه نمایند.

2.prototyping

3.Gause,D.C. and G.M.

4.context-free questions

می‌کند. سری بعدی سؤالات، تحلیل‌گر را قادر می‌کند تا درک بهتری از مشکل و مشتری پیدا کرده و نظرات خود را دربارهٔ راه‌حل به او بگوید.

- چگونه خروجی "خوب" را که یک راه‌حل موفق به‌دنبال می‌آورد، تشریح می‌کنید؟
- این راه‌حل چه مشکلاتی را مورد توجه قرار می‌دهد؟
- آیا امکان دارد که محیط به‌کارگیری راه‌حل را برایم تشریح کنید؟
- آیا مباحث ویژه عملکردی یا محدودیت‌هایی در مسیر یافتن راه‌حل وجود دارند؟

آخرین سری سؤالات بر روی کارآیی جلسه، متمرکز هستند. گایز و وینبرگ [GAU89][1] به آنها فوق سؤالات گفته و لیستِ مختصر شده ذیل را پیشنهاد می‌دهند:

- آیا شما فرد مناسبی برای پاسخ‌گویی به این سؤالات هستید؟
- آیا پاسخ‌های شما "رسمی" هستند؟
- آیا سؤالات من به مشکل شما مربوط می‌شوند؟
- آیا من بیش از اندازه در حال طرح سؤالات هستم؟
- آیا فرد دیگری وجود دارد که بتواند اطلاعات اضافی در اختیار بگذارد؟
- آیا مورد دیگری وجود دارد که از شما سؤال کنم؟

این سؤالات (و دیگر سؤالات) به "شکستن یخ" کمک کرده و ارتباط را برقرار می‌کند که برای موفقیت تحلیل‌گر ضروری می‌باشد. اما یک جلسهٔ پرسش و پاسخ، رهیافتی نیست که خیلی قرین موفقیت باشد. در حقیقت، جلسهٔ پرسش و پاسخ فقط برای اولین برخورد است و سپس باید جلساتی با چهارچوب تلفیق عناصر حل مشکل، مذاکره و مشخص‌سازی همراه آن برگزار شود. روشی برای این نوع جلسات در قسمت بعدی ارائه شده است.

توصیه

اگر یک سیستم یا محصول به کاربران زیادی خدمات دهد، اطمینان حاصل کنید که نیازمندیها از طیف گسترده کاربران کسب گردیده است. تعریف تمام نیازمندیها توسط یک کاربر به منزله بالا رفتن ریسک پذیرش (بقیه کاربران) می باشد.

۱۱-۲-۲ فنون تسهیل مشخص سازی کاربردی

غالباً، مشتریان و مهندسین نرم‌افزار دارای یک ذهنیت ناخود آگاه "ما و آنها" هستند. به‌جای آن‌که به‌صورت تیم، فعالیت کنند تا نیازمندیها را تعیین و پالایش نمایند، هر یک به‌صورت حوزه‌های جداگانه قلمرو خود را از طریق یک‌سری یادداشت‌ها، اوراق وضعیت رسمی، اسناد و جلسات پرسش و پاسخ، تعیین می‌کنند. تاریخ نشان داده است که این رهیافت نتیجهٔ خوبی به‌همراه ندارد. سوءتفاهم درحد بالا، اطلاعات مهم حذف شده، و رابطهٔ کاری مؤفق هیچ‌گاه برقرار نمی‌شود.

با در نظر داشتن چنین مشکلاتی بود که تعدادی از محققین مستقل، یک رهیافت معطوف به کار تیمی را برای گردآوری نیازمندیها ابداع کرده‌اند. این روش در طول مراحل اولیهٔ تحلیل، مشخص‌سازی

ارجاع به وب

رهیافتی برای FAST به نام (JAD) " طراحی کاربرد اتصالی " وجود دارد که توصیف آن در آدرس زیر آورده شده است:
www.bee.net/blu ebird/jaddoc.htm

1.Gause ,D.C.and G.M.

به‌کار گرفته می‌شود. این روش که نام آن " فنون تسهیل مشخص‌سازی"[1] (FAST) می‌باشد، ایجاد یک تیم مشترک از مشتریان و تحلیل‌گران را تشویق می‌کند تا با یکدیگر در زمینهٔ تعیین مشکل، پیشنهاد عناصر راه‌حل، مذاکره در مورد راه‌کارهای مختلف و مشخص ساختن یک مجموعه ابتدایی نیازمندیهای راه‌حل [ZAH 90][2] همکاری کنند. FAST به‌طور عمده توسط جامعهٔ سیستم‌های اطلاعاتی به‌کار گرفته شده است، ولی این تکنیک پتانسیل بهبود روابط را در کلیهٔ زمینه‌های کاربردی ارائه می‌کند.

انواع رهیافت‌های FAST پیشنهاد شده‌اند.[3] هر کدام از این رهیافت‌ها از سناریویی با اندکی تفاوت استفاده می‌کند، اما همگی آنها تغییراتی را اعمال می‌کنند که راهنمودهای ذیل آنها را ارائه می‌کند :

- ملاقاتی در مکانی بی‌طرف برگزار و مهندس نرم‌افزار و مشتریان حضور دارند.
- قوانین برای آمادسازی و تدارک، تثبیت شده‌اند.
- دستور کاری پیشنهاد می‌شود که آن‌قدر رسمی است تا نکات مهم را پوشش دهد، ولی آن‌قدر غیر رسمی نیز هست تا جریان آزاد عقاید را امکان‌پذیری سازد.
- یک "تسهیل‌گر" (می‌تواند یک مشتری، مهندس نرم‌افزار یا فرد دیگری باشد) جلسه را کنترل می‌کند.
- یک "مکانیسم تعریف" (می‌تواند برگهٔ کاری، چارت، یا علایم نصب به دیوار یا برد الکترونیکی باشد) به‌کار می‌رود.
- هدف تعیین مشکل، پیشنهاد عناصر راه‌حل، مذاکره در مورد رهیافت‌های گوناگون و تعیین مجموعه مقدماتی نیازمندیهای راه‌حل، در یک فضای مفید برای تحقق هدف، می‌باشد.

پرسش

چه چیز "ملاقات فنون مشخصات کاربردی تسهیل شده" (FAST) را با ملاقات معمولی، متفاوت می سازد؟

برای فهم بهتر جریان امور همان‌طور که در یک جلسه ملاقات معمول FAST رخ می‌دهد ما یک سناریوی کوتاه از خطوط اصلی سلسلهٔ حوادث منتهی به جلسه ملاقات، آنچه ضمن آن رخ می‌دهد و پس از آن روی می‌دهد را ارائه می‌کنیم. ملاقات‌های اولیه بین مشتری و مهندس نرم‌افزار (بخش ۱۱-۲-۱) برگزار و پرسش و پاسخ‌های اساسی به روشن شدن دامنهٔ مشکل کمک کرده و ادراک کلی از راه‌حل به‌دست می‌آید. ماحصل این جلسات اولیه آن است که مشتری و مهندس نرم‌افزار هر دو یک گزارش یک یا دو صفحه‌ای دربارهٔ "درخواست محصول" می‌نویسند. مکان، زمان و تاریخ جلسهٔ FAST انتخاب شده و یک تسهیل‌گر نیز انتخاب می‌شود. شرکت‌کنندگان از هر دو طرف، مشتری و مهندس نرم‌افزار برای شرکت در جلسه دعوت می‌شوند. درخواست محصول قبل از برگزاری جلسات توزیع می‌شوند.

1.facilitated application specification techniques

2.Zahniser , R.A.

۳. دو گونه از رهیافت های متداول FAST یکی توسعه کاربردی اتصال (JAD) می باشد که توسط آی بی ام توسعه یافته و دیگری (METHOD) که توسط شرکت منابع کارآمد، ساخته شده است

ضمن بررسی تقاضا در روزهای پیش از ملاقات، از هر شرکت‌کنندهٔ در FAST خواسته می‌شود که لیستی از اشیاء[1] که قسمتی از محیط پیرامون سیستم، دیگر اشیاء را که سیستم باید تولید کند، اشیایی که باید سیستم برای انجام کارکردهایش به‌کار بگیرد، تهیه نماید. به‌علاوه، از هر شرکت‌کننده خواسته می‌شود تا لیست دیگری از خدماتی[2] که (فرآیندها یا کارکردها)، اشیاء را کنترل و یا با آنها تعامل می‌کند را، تهیه نماید. سرانجام فهرستهای محدودیتها[3] (مثلاً هزینه، اندازه، قوانین تجاری) و معیارهای انجام کار[4] (مثلاً، سرعت، دقت) نیز تهیه می‌شوند به شرکت‌کنندگان اطلاع داده می‌شود که لیستها نباید خسته‌کننده باشد، اما باید چشم‌انداز خود را دربارهٔ سیستم منعکس کنند.

پیش از ملاقات (FAST)، لیستی از اشیاء، موضوعات اصلی،خدمات، قیود و معیارهای عملکردی تهیه کنید.

به‌عنوان یک مثال،[5] در نظر بگیرید که تیم FAST که برای یک شرکت مصرف‌کننده محصولات کار می‌کند، با شرح محصول ذیل روبه‌رو شده است:

تحقیق ما نشان می‌دهد که بازار سیستم‌های ایمنی منازل با نرخ ۴۰٪ در سال، در حال رشد است. ما علاقه‌مندیم تا با تولید یک سیستم ایمنی منزل استوار بر پایه میکرو پروسسور، که در مقابل مواردی مانند موقعیت‌های نامطلوب، مانند ورود غیر قانونی، آتش‌سوزی، سیل و غیره، ساکنین منزل را مورد محافظت قرار می‌دهد، وارد این بازار شویم. این محصول که در حالت مقدماتی خود "خانهٔ امن" نام دارد از سنسورهای مناسب برای تشخیص هر موقعیت استفاده می‌کند و می‌تواند توسط مالک خانه برنامه‌ریزی شود و به‌طور خودکار به یک آژانس نظارتی هنگام تشخیص هر موقعیت تلفن می‌کند.

در واقع، اطلاعات بسیار بیشتری در این مرحله ارائه می‌شود، اما حتی با ارائه اطلاعات بیشتر، ابهام هنوز وجود دارد. احتمال حذف مواردی وجود دارد و امکان وقوع خطا دارد. فعلاً "شرح محصول" فوق کفایت می‌کند.

تیم FAST متشکل از نمایندگانی از بخش‌های بازاریابی، مهندسی نرم‌افزار و ساخت‌افزار و تولید می‌باشد. یک تسهیل‌گر خارجی نیز باید حضور داشته باشد. هر یک از اعضای تیم، فهرستهای شرح داده شدهٔ فوق را تهیه می‌کند. اشیاء تشریح شده برای "خانهٔ امن" می‌تواند اقلام دیگری را نیز در برگیرد؛ تشخیص‌گرهای دود، سنسورهای پنجره و در، تشخیص‌گرهای حرکت، زنگ خطر، یک رویداد (یک سنسور فعال شده است)، صفحهٔ کنترل، صفحه‌نمایش، شماره‌های تلفن، تماس تلفن و غیره. لیستی از خدمات می‌تواند شامل تنظیم زنگ خطر، نظارت بر سنسورها، شماره‌گیری تلفن، برنامه‌ریزی صفحهٔ کنترل، خواندن صفحه نمایش (دقت داشته باشید که خدمات روی اشیاء عمل می‌کنند).

اشیاء با خدمات اداره می شوند و باید در میان قیود و محدودیتها " زندگی " کنند و عملکرد به وسیله تیم (FAST) تعریف می شود.

1.objects

2.services

3.constraints

4.performance criteria

۵. این مثال (با ملحقات و تغییرات) برای تشریح شیوه های مهم مهندسی نرم افزار در بسیاری از فصل های بعدی دنبال شده است. به عنوان تمرین، ارزشمند است که با نشست FAST ارتباط پیدا نموده و مجموعه لیست هایی برای آن بیابید.

به طریقی مشابه هر شرکت‌کنندهٔ **FAST** لیست‌های عوامل محدودکننده خود را تهیه می‌کند (مثلاً، سیستم باید یک هزینه ساخت کمتر از ۸۰ دلار به‌همراه داشته باشد، باید به‌راحتی مورد استفادهٔ کاربر قرار بگیرد، باید مستقیماً با یک خط تلفن استاندارد ارتباط برقرار کند) و معیارهای انجام کار (مثلاً، یک رویداد سنسوری باید ظرف یک ثانیه دریافت شود، یک شمای اولویت رویداد باید پیاده‌سازی شود).

به همان ترتیب که جلسهٔ **FAST** آغاز می‌شود، اولین موضوع بحث، نیاز و توجیه تولید محصول جدید است ـ همه باید بر روی موجه بودن محصول، توافق داشته باشند. زمانی‌که توافق حاصل شد، هر شرکت‌کنندهٔ حاضر، لیست‌های مباحثه خود را ارائه می‌کند. می‌توان این لیست‌ها را به دیوار نصب کرد و یا به هر طریقهٔ دیگر در معرض دید دیگران قرار داد. ترجیح داده می‌شود که از بردهای الکترونیک برای این‌منظور استفاده شود. در هر حال این لیست‌ها باید به‌نحوی ارائه شوند که بتوان آنها را با یکدیگر ترکیب کرده، حذف کرده و یا اضافاتی در آنها انجام داد. در این مرحله، انتقاد و بحث اکیداً ممنوع است.

پس از آن‌که لیست‌های جداگانه ارائه شدند، یک لیست ترکیب توسط گروه در مورد یک موضوع، تهیه می‌شود. لیست ترکیب، لیست‌های تکراری را حذف می‌کند. نظرات جدید را به آن اضافه می‌کند، ولی هیچ چیزی را پاک نمی‌کند. پس از آن‌که لیست‌های ترکیب برای تمام موضوعات فوق گردآوری و تنظیم شدند، مباحثه که توسط تسهیل‌گر هماهنگ می‌شود، آغاز می‌گردد. لیست ترکیب کوتاه شده، بلند شده یا مجدداً نوشته می‌شود تا به‌خوبی بتواند منعکس‌کنندهٔ سیستم/ محصول پیشنهادی برای تولید باشد. هدف بدست آوردن یک لیست "اتفاق نظر"[1] در هر زمینه و موضوع (اشیاء، خدمات، محدودیت‌ها و اجرا) می‌باشد. لیست‌ها سپس برای اقدام بعدی به کناری نهاده می‌شوند.

زمانی‌که لیست‌های اتفاق نظر تکمیل شدند، تیم مربوطه به تیم‌های کوچک‌تری تقسیم می‌شود که هر یک در جهت تهیه یک مشخص‌سازی کوچک[2] برای یک یا چند مورد اضافه شده به هر یک از لیست‌ها، فعالیت خواهد کرد. مینی ـ مشخص‌سازی پرداختن معضل به یک کلمه یا عبارت به‌کار رفته در یک لیست می‌باشد.[3] مثلاً، مینی ـ مشخص‌سازی برای صفحهٔ کنترل اشیاء خانهٔ امن می‌تواند چنین باشد :

- به دیوار نصب شده باشد
- ابعاد آن حدوداً ۵×۹ اینچ بوده
- دارای ۱۲ صفحه کلید و کلیدهای ویژه باشد
- دارای صفحه نمایش **LCD** به شکل نمایش داده شده در طرح کلی (این‌جا ارائه نشده است) باشد
- تمامی تعاملات مشتری از طریق کلیدها انجام شود

1.consensus list

2.mini-specifications

۳. یک رهیافت جایگزین ایجاد Use-Case ها را دنبال می کند. برای جزئیات بیشتر به بخش ۱۱-۲-۴ مراجعه نمایید.

- برای به‌راه انداختن و از کار انداختن سیستم باشد
- نرم‌افزار راهنمایی تعامل و محاوره ارائه نماید
- به کلیه سنسورها متصل باشد

هر تیم فرعی سپس هر یک از مینی ـ مشخص‌های‌سازی خود را به کلیه شرکت‌کنندگان در **FAST** برای بحث روی آن ارائه می‌کند. اضافات، موارد حذف و توضیحات بیشتر، انجام می‌شوند. در بعضی موارد توسعهٔ مینی ـ مشخص‌سازی‌ها به اشیاء، خدمات، محدودیت‌ها یا نیازمندیهای عملکردی جدید، منتهی می‌شود که به لیست‌های اصلی اضافه خواهند شد. در طول کلیه مباحثات، تیم ممکن است مبحثی را مطرح کند که در طول جلسه به نتیجه نرسد. یک لیست مباحث برای بررسی بعدی این مباحث برای بررسی بعدی این مباحث، نگهداری خواهد شد. پس از آن‌که مینی ـ مشخص‌سازی‌ها تکمیل شد هر شرکت‌کننده در **FAST** یک لیست معیار اعتبار[1] برای محصول/ سیستم تهیه کرده و لیست خود را به تیم ارائه می‌دهد. سپس یک لیست اتفاق نظر در مورد معیار اعتبار فراهم می‌شود. سرانجام یک یا چند شرکت‌کننده (یک افراد خارجی) وظیفهٔ نوشتن پیش‌نویس کامل مشخصات را، با استفاده از کلیهٔ ورودی‌های حاصل از جلسهٔ **FAST** برعهده می‌گیرد.

FAST یک اکسیر جادویی شفابخش برای مشکلات بوجود آمده در مرحلهٔ اول استخراج نیازها نیست. اما رهیافت تیم مزیت برخوردار شدن از نقطه نظرات بسیاری را همراه با مباحث همزمان و پالایش آنها را فراهم کرده و گامی مؤثر به‌سوی توسعه مشخص‌سازی است.

۳-۲-۱۱ تنظیم کارکرد کیفیت

گسترش(تنظیم) کارکرد کیفیت[2] (**QFD**)، تکنیک مدیریت کیفیتی است که نیازهای مشتری را به نیازهای فنی نرم‌افزار ترجمه می‌کند. در اصل این تکنیک در ژاپن بوجود آمد و برای اولین بار صنایع سنگین کشتی سازی میتسوبیشی به‌کار گرفته شد. در اوایل دهۀ ۱۹۷۰ **QFD** "برای حداکثر رضایت مشتری از فرآیند مهندسی افزار تلاش می‌کرد." **[ZUL92]**[3] برای تحقق این امر، **QFD** بر روی درک آنچه برای مشتری ارزشمند است تأکید کرد و سپس تمام این ارزش‌ها در فرآیند مهندسی پیاده‌سازی می‌کرد. **QFD** سه نوع نیازمندی را شناسایی کرده است:[ZUL92]

نیازمندیهای عادی. مقاصد و اهدافی که برای محصول یا سیستم در طول ملاقات با مشتری مطرح شده‌اند. چنانچه این نیازمندیها برآورده شوند آن‌گاه مشتری راضی خواهد بود. برای مثال، نیازهای عادی می‌توانند نمایش‌های گرافیکی، کارکردهای خاص سیستم و سطوح تعریف شدهٔ اجراء باشند.

1.validation criteria

2.Quality function deployment

3.Zultner ,R.A.

توصیه
از برخوردی محکم و بسته با ایده های مشتریان که آنها را " بسیار هزینه بر " یا " غیرعملی " می پندارید، اجتناب کنید. در مذاکرات بنا را بر پذیرش درخواست آنها بگذارید، این امر فکری باز می خواهد.

تعریف
گسترش کارکرد کیفیت (QFD) نیازمندیها را به گونه ای تعریف می کند که حداکثر رضایت مندی مشتری حاصل شود.

توصیه
همه مایلند که تعداد زیادی از نیازمندیهای تعیین شده را به دقت پیاده سازی کنند. اینکه نیازمندیها چگونه به آرامی رشد می کنند، دلیل امر مدعاست. به عبارت دیگر اغلب نیازمندیهای کسب شده، موجب پیشرفت سریع محصول خواهد شد.

نیازمندیهای مورد انتظار. این نیازمندیها می‌توانند به‌صورت تلویحی دربارهٔ محصول یا سیستم مطرح شده و به میزانی بنیادی باشند که مشتری به‌طور صریح آنها را مطرح نکند. فقدان آنها می‌تواند باعث نارضایتی شدید شود. مثال‌ها در مورد، نیازمندیهای مورد انتظار عبارت هستند از: راحتی تعامل انسان با ماشین، صحیح انجام شدن کلی عملیات و قابلیت اطمینان و سادگی نصب نرم‌افزار.

نیازمندیهای هیجان‌انگیز. این ویژگی‌ها فراتر از انتظارات مشتری رفته و زمانی‌که ارائه می‌شوند، بسیار رضایت‌بخش خواهند بود. مثلاً، نرم‌افزار پردازش واژه‌ها با ویژگی‌های استاندارد مورد تقاضا می‌باشد. محصول تحویل داده شده در بردارندهٔ تعدادی صفحات با توانایی صفحه‌آرایی است که بسیار مطلوب و غیرمنتظره است.

در واقع، **QFD** کل فرآیند مهندسی را در برمی‌گیرد [AKA90][1]. در هر حال، اکثر مفاهیم **QFD** در مورد فعالیت‌های مربوط به استخراج نیازمندیها صدق می‌کنند. ما یک چشم‌انداز که فقط این مفاهیم را (تطبیق داده شده برای نرم‌افزار کامپیوتر) بررسی می‌کند، در پاراگرافهای بعدی ارائه خواهیم کرد.

در جلسات برگزار شده با مشتری، توسعهٔ کارکرد[2] برای تعیین ارزش هر کارکرد که برای سیستم مورد نیاز است، به‌کار می‌رود. توسعهٔ اطلاعات[3]، اشیاء داده‌ای و مواردی را که سیستم باید مصرف کرده و تولید کند، تعیین می‌کند. این مسایل به کارکردها گره خورده‌اند. سرانجام پیاده‌سازی وظائف[4]، رفتار سیستم یا محصول را در بافت محیطی آن، امتحان می‌کند. تحلیل مقادیر[5]، برای تعیین اولویت نسبی نیازهای تعیین شده در طول هر یک از سه مرحلهٔ پیاده‌سازی فوق‌الذکر اجرا می‌شود.

QFD از مصاحبه‌ها، مشاهدات، پژوهش‌ها و آزمون‌های تاریخی داده‌ها (مثلاً، گزارشات مربوط به مشکلات) به‌عنوان داده‌های خام برای جمع‌آوری نیازها استفاده می‌کند. سپس این داده‌ها به جدول نیازها ـ که جدول صدای مشتری[6] نامیده می‌شود ـ ترجمه می‌شوند که از سوی مشتری مورد بررسی قرار می‌گیرند. انواع دیاگرام‌ها، پانویس‌ها و روش‌های ارزشیابی برای استخراج نیازهای مورد انتظار به‌کار می‌روند و تلاش می‌شود تا نیازمندیهای هیجان‌انگیز استخراج شوند [BOS91][7].

1. Akao , Y.
2. Function deployment
3. Information deployment
4. task deployment
5. Value analysis
6. customer voice table
7. Bossert , J.L.

۱۱-۲-۴ USE-CASE ها

Use-case ها از نقطه نظر بازیگران تعریف می شوند. یک بازیگر نقشی است (کاربر یا دستگاه) در مواجهه با نرم افزار ایفا می کنند.

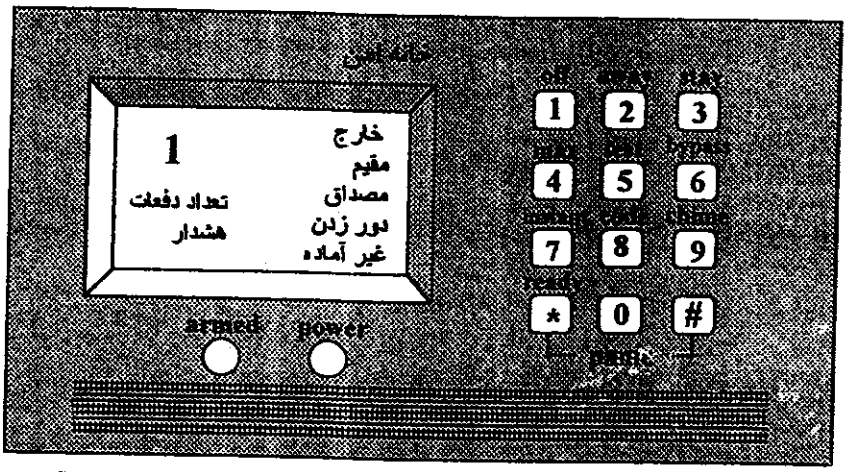
به همان ترتیب که نیازمندیها به‌عنوان بخشی از جلسات غیر رسمی **FAST** یا **QFD**، گردآوری می‌شوند، مهندس نرم‌افزار (تحلیل‌گر) می‌توانند مجموعه‌ای از سناریوهایی که یک ریسمان کاربرد سیستمی که باید ساخته شود را تعیین نمایند. این سناریوها که غالباً Use-caseها نامیده می‌شوند **[JAC92]**[1]، شرحی از چگونگی استفاده از سیستم را فراهم می‌کنند.

برای ایجاد یک Use-caseها، تحلیل‌گر باید در آغاز انواع مختلف افراد (یا ابزارها) که از سیستم یا محصول استفاده خواهند کرد، تعیین کند. این بازیگرها[2] معمولاً نقش‌هایی را افراد (یا ابزارها) به‌عنوان اپراتورهای سیستم ایفاء می‌کنند، نشان می‌دهند. یک بازیگر زمانی‌که به‌طور رسمی‌تر تعیین می‌شود همان چیزی است که با سیستم یا محصول ارتباط برقرار می‌کند و نسبت به سیستم، یک عامل خارجی محسوب می‌شود.

اهمیت دارد که خاطرنشان شود هر بازیگری و یا کاربری با یکدیگر یکسان نیستند. یک کاربر معمول می‌تواند تعداد مختلفی نقش را هنگام به‌کارگیری سیستم ایفاء نماید، در حالی‌که بازیگر نمایندهٔ گروهی از عناصر خارجی (غالباً ولی نه به‌طور همیشگی، افراد) می‌باشد که فقط ایفاگر یک نقش هستند. برای مثال، یک اپراتور ماشین (کاربر) را که با کامپیوتر کنترل برای ساخت سلولی که حاوی تعداد ربات است و ماشین‌های کنترل‌شوندهٔ عددی کار می‌کند، در نظر بگیرید. پس از بررسی دقیق نیازمندیها، نرم‌افزار برای کامپیوتر کنترلی، نیاز به چهار حالت (نقش) مختلف برای تعامل خواهد داشت: حالت برنامه‌ریزی، حالت آزمون، حالت نظارتی، و حالت برطرف‌سازی مشکلات. بنابراین چهار بازیگر را می‌توان تعریف کرد: برنامه‌ریز، آزمون‌کننده، ناظر، برطرف‌کننده مشکلات. در بعضی از موارد اپراتور ماشین می‌تواند همهٔ این نقش‌ها را ایفاء کند. در پاره‌ای دیگر از موارد، افراد مختلف باید نقش هر کدام از بازیگرها را به‌طور جداگانه ایفاء نمایند.

1.Jacobson,I.

2.actors



شکل ۱۱ـ۲ پانل کنترل خانه امن

یک Usecase سناریو ای است که تشریح می کند در شرایط مشخص نرم افزار چگونه به کار می آید.

از آن‌جا که استخراج نیازمندیهای یک فعالیت رو به تکامل است، تمام بازیگرها در ظرف اولین تکرار مشخص نمی‌شوند. امکان دارد که بازیگران اولیه را [JAC92][1] را در اولین تکرار تعیین کرده و بازیگران ثانویه را پس از شناخت بیشتر سیستم معین کرد. بازیگران اولیه برای دستیابی به کارکرد مطلوب سیستم وارد تعامل می‌شود و سود موردنظر از سیستم را حاصل می‌کند. آنها مستقیماً و اغلب با نرم‌افزار کار می‌کنند. بازیگران ثانویه برای حمایت سیستم حضور دارند تا بازیگران اولیه بتوانند در کار خود مؤفق شوند. زمانی‌که بازیگران تعیین شدند، Use-case می‌تواند توسعه یابد.

Use-case طریقهٔ تعامل یک عمل‌کننده با سیستم را تشریح می‌کند.

یاکوبسن [JAC92]، پیشنهاد می‌دهد که سؤالات ذیل برای توسعهٔ Use-caseها پاسخ داده شوند:

- وظایف و کارکردهای اصلی که یک بازیگرباید انجام دهد، چیست؟
- چگونه سیستم اطلاعات توسط بازیگر تولید تغییر یافته یا بدست می‌آید؟
- آیا بازیگر باید سیستم را در مورد تغییرات در محیط خارجی آن مطلع کند؟
- چه اطلاعاتی را بازیگر از سیستم می‌طلبد؟
- آیا بازیگر علاقه‌مند به مطلع شدن دربارهٔ تغییرات غیرمنتظره هست؟

به‌طور کلی یک Use-case ها یک شرح روایتی ساده است که نقش هر بازیگر را ضمن تعامل با سیستم، روشن می‌کند.

با یادآوری نیازهای اساسی "خانهٔ امن" (بخش ۱۱-۲-۲)، می‌توانیم سه بازیگر را تعریف نماییم: مالک خانه (کاربر)، سنسورها (ابزارهای متصل به سیستم) و سیستم فرعی نظارتی و پاسخ‌دهی (ایستگاه مرکزی که خانهٔ امن را مانیتور می‌کند).

در رابطه با این مثال، ما فقط **مالک خانه** را به‌عنوان بازیگر در نظر می‌گیریم.

---

1. Jacobson,I.

مالک خانه با محصول به طرق مختلف تعامل می‌کند:

- اسم رمزی را وارد می‌کند تا امکان کلیهٔ تعامل‌ها به‌وجود آید.
- در مورد امنیت برقرار شده در منطقه دستگاه سؤال می‌کند.
- در مورد وضعیت سنسور سؤال می‌کند.
- دگمهٔ Panic (#, *)را در حالت اضطراری فشار می‌دهد.
- سیستم ایمنی را فعال/ غیر فعال می‌کند.

یک Use-case برای فعال‌سازی سیستم[1] به‌شرح ذیل است:

۱- مالک خانه صفحهٔ کنترل خانهٔ امن نمونه را مشاهده می‌کند (شکل۱۱-۲) تا آمادگی سیستم برای ورودی گرفتن را تعیین کند. اگر سیستم آماده نباشد، مالک خانه باید به‌طور فیزیکی پنجره‌ها/ درها را ببندد تا نشانگر آماده، حضور روشن شود. (یک نشانگر غیر آماده به‌معنای باز بودن سنسور است یا به‌عبارت دیگر باز بودن در یا پنجره را اعلام می‌کند)

۲- مالک خانه از صفحه کلید برای وارد کردن یک اسم رمز چهار رقمی استفاده می‌کند. این اسم رمز با اسم رمز معتبر ذخیره شده در این سیستم مقایسه می‌شود. اگر اسم رمز صحیح نباشد، صفحه کنترل یک‌بار صدای بوق را تولید کرده و برای ورودی بعدی خود را دوباره تنظیم می‌کند. اگر اسم رمز صحیح باشد، صفحه کنترل در انتظار اقدام بعدی خواهد ماند.

۳- مالک خانه وضعیت "مقیم بودن"[2] یا "رفتن"[3] را انتخاب می‌کند (به شکل ۱۱-۲ رجوع کنید) تا سیستم را فعال سازد. "مقیم بودن" فقط سنسورهای محیطی را فعال می‌کند. (در درون آن سنسورهای تشخیص تحرک غیر فعال می‌شوند). "رفتن" تمام سنسورها را فعال می‌کند.

۴- زمانی‌که فعال‌سازی رخ می‌دهد، یک لامپ هشدار قرمز، توسط مالک خانه مشاهده می‌شود.

هر Use-case یک سناریوی بدون ابهام از تعامل بین یک بازیگرو نرم‌افزار فراهم می‌کند که می‌تواند علاوه بر آن به‌عنوان یک تعیین‌کنندهٔ نیازمندیهای زمانی یا دیگر محدودیت‌های سناریو مورد استفاده قرار گیرد. برای مثال، در رابطه با Use-caseهای فوق‌الذکر، نیازمندیها، بیان‌گر آن هستند که فعال‌سازی ۳۰ ثانیه بعد از فشرده شدن کلید "مقیم" یا "رفتن" رخ داده است. این اطلاعات می توانند به Use-case اضافه شوند.

Use-case ها، سناریوهایی را شرح می‌دهند که توسط بازیگرهای مختلف به‌صورت متفاوت فهمیده می‌شوند. وایدر[WYD 96][4] پیشنهاد می‌کند که پیاده‌سازی کارکرد کیفیت می‌تواند برای توسعهٔ یک

1.system activation
2.stay
3.away
4.Wyder,T.

مقدار اولویت، ارزیابی شده در رابطه با هر Use-case به‌کار گرفته شود. برای تحقق این امر، Use-caseها از نقطه‌نظر تمام بازیگران تعریف برای سیستم، ارزشیابی می‌شوند. یک مقدار اولویت توسط هر بازیگر برای هر Use-case تعیین می‌شود (مثلاً، مقداری از یک تا ده).[1] یک اولویت میانگین سپس محاسبه شده و اهمیت ادراک شدهٔ هر یک از Use-caseها مشخص می‌شود. زمانی‌که یک مدل فرآیند تکرار برای مهندسی نرم‌افزار معطوف به اشیاء به‌کار می‌رود، اولویت‌ها می‌توانند بر روی تعیین اولین قابلیت کارکردی سیستم که می‌تواند ارائه شود، تأثیر بگذارند.

۱۱-۳ اصول تحلیل

اصولی که در کار تحلیل راهنما می باشند، کدامند؟

طی دو دههٔ گذشته، تعداد زیادی روش‌های مدل‌سازی تحلیل به‌وجود آمده‌اند. پژوهشگران مشکلات تحلیل را تعیین کرده با تشخیص دلایل به‌وجود آمدن آنها انواع راه‌های مدل‌سازی و مجموعه‌های راه‌گشای مؤثر در برطرف کردن آنها را ارائه داده‌اند. هر مدل تحلیل دارای یک نکتهٔ منحصربه‌فرد است. در هر صورت تمام مدل‌های تحلیل توسط مجموعه‌ای از اصول عملیاتی با یکدیگر مرتبط هستند:

۱- حوزهٔ اطلاعات یک مسئله باید بازنمایی و ادراک شود.

۲- کارکردهایی که نرم‌افزار باید انجام دهد، مشخص شوند.

۳- رفتار نرم‌افزار (به‌عنوان پیامد رخدادهای خارجی) باید بازنمایی شود.

۴- مدل‌هایی که نمایشگر اطلاعات، کارکرد، و رفتار که باید به روشی تقسیم‌بندی شوند که جزییات را به طریقی لایه‌بندی شده (یا سلسله مراتبی) آشکار نماید.

۵- فرآیند تحلیل باید از اطلاعات اساسی به‌سوی جزییات پیاده‌سازی حرکت کند.

با به‌کارگیری این اصول، تحلیل‌گر یک مشکل را به‌طور نظام مند بررسی می‌کند.

حوزهٔ اطلاعات بررسی شده تا کارکرد به‌طور کامل‌تری ادراک شود. مدل‌هایی به‌کار می‌روند تا خصوصیات کارکرد و رفتار به طریقه‌ای فشرده تبادل شوند. تقسیم‌بندی برای کاهش پیچیدگی‌ها انجام می‌شود. نقطه نظرات ضروری و مربوط پیاده‌سازی نرم‌افزار برای تعیین نحوهٔ برخورد با محدودیت‌های فیزیکی تحمیل شده توسط عناصر سیستم را نیز مدنظر قرار می‌دهد.

علاوه بر اصول تحلیل عملیاتی فوق‌الذکر، دیویس [DAV950][2]، یک مجموعه از اصول راهنما[3] برای مهندسی نیازها را پیشنهاد می‌کند:

---

۱. ایده آل آن است که این ارزیابی که توسط افرادی از سازمان یا کارکرد تجاری انجام شده است، با یک بازیگربازنمایی شود.

2.Davis,A.

۳. تنها زیر مجموعه کوچکی از اصول مهندسی نیازمندیهای دیویس در اینجا گردآورده شده است. برای اطلاعات بیشتر به [DAV95a] مراجعه کنید.

- **شکل را قبل از شروع به ایجاد مدل تحلیل درک کنید.** گرایش به یافتن سریع راه‌حل وجود دارد، حتی قبل از آن‌که شکل ادراک شده باشد.[1] این امر غالباً به ایجاد نرم‌افزاری عالی که مشکل دیگری را حل می‌کند، منجر می‌شود!
- **نمونه‌های اولیه‌ای را ایجاد کنید که کاربر را قادر می‌سازد تا چگونگی تعامل کاربر ـ ماشین را ادراک نماید.**[2] از آن‌جا که فهم کیفیت نرم‌افزار غالباً بر پایه فهم "دوستانه" بودن رابط است، نمونهٔ اولیه ساختن (و تکرار آن نتایج) شدیداً توصیه می‌شود.
- **ریشه و دلیل هر نیازمندی را ثبت کنید،** این کار اولین گام در جهت ایجاد شناسایی نیازهای مشتری است.[3]
- **از نظرات مختلف دربارهٔ نیازمندیها استفاده کنید،** ایجاد مدل‌های داده‌ها، کارکرد و رفتار سه نقطه نظر را در اختیار مهندس نرم‌افزار قرار می‌دهد.[4] این امر احتمال آن‌که چیزی فراموش شده، کاهش داده و احتمال آن‌که عدم سازگاری‌ها کشف شوند را افزایش می‌دهد.
- **طبقه‌بندی نیازمندیها، موعدهای زمانی می‌توانند پیاده‌سازی هر نیازمندی نرم‌افزاری را متوقف کنند.**[5] اگر یک مدل فرآیند افزایشی به‌کار گرفته شود، (فصل ۲) این نیازمندیها که باید در اولین مورد پیشرفت تحویل داده شوند، باید مشخص شوند.
- **بر روی رفع ابهام‌ها کار کنید.**[6] از آن‌جا که اکثر نیازمندیها به زبانی طبیعی شرح داده شده‌اند، فرصت برای ایجاد ابهام به‌وجود می‌آید. استفاده از بررسی‌های فنی رسمی یکی از راه‌های آشکارسازی و رفع ابهام است.

یک مهندس نرم‌افزار که این اصول را مدنظر قرار می‌دهد، در توسعهٔ مشخص‌سازی نرم‌افزاری که بهترین زیربنا را برای طراحی نرم‌افزار ارائه می‌کند، موفق‌تر خواهد بود.

حوزه اطلاعات یک مسئله مشتمل است بر اقلام داده ها یا اشیاء که عبارت خواهند بود از: اعداد، متون، تصاویر، صوتها، فیلم یا هر ترکیبی از اینها.

۱۱-۳-۱ میدان اطلاعات

تمام کاربردهای نرم‌افزار را می‌توان به‌صورت گروهی، پردازش داده‌ها[7] نامید. جالب است که این واژه در بردارندهٔ کلیه فهم ما از نیازمندیهای نرم‌افزار است. نرم‌افزار برای پردازش داده‌ها، تغییر آنها از شکلی به

1.understand the problem before you begin to create the analysis model
2.Develop prototypes that enable a user to understand how human/machine interaction will occur
3.Record the origin of and the reason for every requirement
4.Use multiple views of requirements
5.Rank requirements
6.Work to eliminate ambiguity
7.data processing

شکل دیگر به‌وجود می‌آید. به‌عبارت دیگر پذیرش ورودی، کنترل و پردازش آن و سپس بیرون دادن آن است.

این توصیف بنیادین از هدف، درست است و فرق نمی‌کند که آیا نرم‌افزار برای سیستم پرداخت حقوق، یا برای کنترل جریان سوخت در پمپ بنزین باشد. باید دقت کنیم که نرم‌افزار رخدادها را نیز پردازش می‌کند. یک واقعه نمایندهٔ جنبه‌ای از کنترل سیستم است که در واقع چیزی بیش از داده‌های صفر و یک نیست ـ یا روشن است یا خاموش، یا صحیح است یا غلط است، آن‌جا هست یا نیست. برای مثال، یک سنسور فشار، تشخیص می‌دهد که آیا فشار از حد مجاز عبور کرده است یا خیر و سیگنال هشدار را به نرم‌افزار ناظر ارسال می‌کند. سیگنال هشداردهنده یک رخداد است که رفتار سیستم را کنترل می‌کند. بنابراین، داده‌ها (شماره‌ها، متن، تصاویر، صداها، ویدئو و غیره) و کنترل (رخدادها) هر دو در میدان اطلاعات یک مسئله قرار می‌گیرند.

اولین اصل تحلیل عملیاتی نیازمند یک میدان اطلاعات و ایجاد یک مدل داده‌ها[1] می‌باشد. میدان اطلاعات در بردارندهٔ سه نقطه‌نظر در رابطه با داده‌ها و کنترل است. همان‌گونه هر یک توسط برنامهٔ کامپیوتر، پردازش می‌شوند: (۱) مضمون اطلاعات و روابط (مدل داده‌ها)، (۲) جریان اطلاعات، (۳) ساختار اطلاعات برای درک کامل میدان اطلاعات، هر یک از این نقطه‌نظرات باید در نظر گرفته شوند.

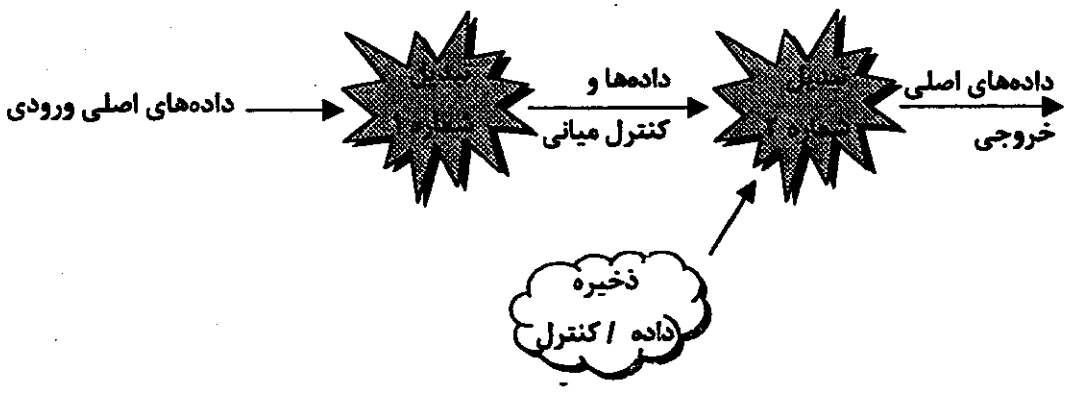
مضمون اطلاعات[2] نمایندهٔ داده‌های جداگانه و اشیاء کنترل است که مجموعه‌ای بزرگ‌تر از اطلاعات تغییر یافته توسط نرم‌افزار را تشکیل می‌دهند. مثلاً، شیء داده‌ای کنترل و پرداخت، پرداخت ناخالص، تخفیف‌ها و غیره. بنابراین، مضمون کنترل پرداخت توسط سندهایی که باید ایجاد شوند، مشخص می‌شود. به‌طور مشابه، مضمون یک شی کنترل به‌نام وضعیت سیستم را می‌توان به‌صورت زنجیرهٔ تکه‌ها تعریف کرد. هر تکه نمایندهٔ یک قلم جداگانه اطلاعات است که روشن می‌کند، آیا یک دستگاه خاص روشن است یا خاموش.

برای آغاز درک حوزه اطلاعات،سوال نخست باید این باشد : " کدام اطلاعات است که این سیستم تولید می کند یا به عنوان خروجی ارائه می دارد؟ "

اشیاء داده‌ای و کنترل را می‌توان به اشیاء داده‌ای و کنترل دیگر ربط داد. مثلاً، شیء داده‌ای کنترل پرداخت دارای یک یا چند رابطه با اشیاء کارت ورود و خروج، پرسنل، بانک و دیگر موارد دارد. طی تحلیل میدان اطلاعات، این روابط باید مشخص شوند.

1.data model

2.Information content



شکل ۱۱ـ۳جریان اطلاعات و تبدیلات

جریان اطلاعات[1] نمایندهٔ طریقهٔ تغییر داده‌ها و کنترل به‌گونه‌ای است که هر یک در سیستم حرکت می‌کنند. عطف به شکل ۱۱-۳، اشیاء ورودی به اطلاعات میانی (داده‌ها و/ یا کنترل) تغییر شکل می‌دهند که درنهایت به خروجی تبدیل می‌شوند. در راستای این تغییر شکل مسیر (یا مسیرها)، اطلاعات اضافی می‌توانند از داده‌های موجود ذخیره شده، معرفی شوند. (مثلاً، یک فایل روی دیسک یا بافر حافظه). تغییر شکل‌هایی که در مورد داده‌ها اعمال می‌شوند کارکردها و یا زیرکارکردهایی هستند که یک برنامه باید اجرا نماید. داده‌ها و کنترلی که بین دو تغییر شکل (کارکردها) حرکت می‌کنند، تعامل مربوط به هر کارکرد را مشخص می‌کنند.

ساختار اطلاعات[1] نمایندهٔ سازمان درونی انواع داده‌ها و قلم‌های کنترل است. آیا داده‌ها یا قلم‌های کنترل باید به‌عنوان یک جدول n بعدی یا یک ساختار درختی سلسله مراتبی، سازمان‌دهی شوند؟ در چارچوب بافت ساختار مربوطه چه اطلاعاتی با اطلاعات دیگر مرتبط هستند؟ آیا تمام اطلاعات در یک ساختار منفرد جای می‌گیرند یا باید ساختارهای جداگانه را به‌کار گرفت؟ چگونه اطلاعات در یک ساختار اطلاعات به اطلاعات در ساختاری دیگر مربوط می‌شوند؟ این سؤالات توسط بررسی ساختار اطلاعات پاسخ گفته می‌شوند. باید توجه کرد که ساختار داده‌ها، یک مفهوم قابل بررسی در فصول بعدی به طراحی و پیاده‌سازی ساختار اطلاعات در نرم‌افزار ارجاع می‌دهد.

۱۱-۳-۲مدل سازی

ما مدل‌های کارکردی را برای درک بهتر آنچه در عمل باید ساخته شود، خلق می‌کنیم. زمانی‌که این مسئله یک مورد فیزیکی است (یک ساختمان، هواپیما یا ماشین) می‌توانیم مدلی ایجاد کنیم که از نظر شکل و فرم یکسان باشند. ولی از نظر مقیاس کوچکتر باشد. در هر صورت زمانی‌که این مسئله موردنظر

1.Information flow

نرم‌افزار باشد، مدل ما باید شکل متفاوتی پیدا کند و باید نمایندهٔ اطلاعاتی باشد که نرم‌افزار تغییر شکل می‌دهد. کارکردها (و کارکردهای فرعی) که این تغییر شکل را ممکن می‌سازند و رفتاری که سیستم ضمن تغییر شکل از خود نشان می‌دهد.

اصول دوم و سوم تحلیل عملیاتی نیازمند ساخت مدل‌های کارکرد و رفتار است.

پرسش

کدام نوع مدلها طی مرحله تحلیل نیازمندیها باید ساخته شوند؟

مدل‌های کارکردی نرم‌افزار اطلاعات را تغییر شکل می‌دهد و بدین‌منظور باید حداقل سه کارکرد عمومی را انجام دهد: ورودی، پردازش و خروجی. زمانی‌که مدل‌های کارکردی یک کاربرد خلق می‌شوند، مهندس نرم‌افزار بر روی کارکردهای ویژهٔ مسئله متمرکز می‌شود. مدل کارکردی با یک مدل در سطح بافت منفرد (به‌عبارت دیگر نام نرم‌افزاری‌که باید فراهم شود) شروع می‌شود. طی یکسری تکرارها، جزییات کارکردی هر چه بیشتری فراهم می‌شود تا این‌که یک تصویری از تمام قابلیت‌های کارکردی سیستم ارائه شود.

پرسش

چگونه مدلهایی که طی مرحله تعیین نیازمندیها ساخته ایم، مورد استفاده قرار می دهیم؟

مدل‌های رفتاری. اکثر نرم‌افزارها به رخدادها از جهان بیرونی پاسخ می‌دهند. این خصوصیت محرک ـ پاسخ، اساس مدل رفتاری را تشکیل می‌دهد. یک برنامهٔ کامپیوتری همیشه در برخی وضعیت‌ها قرار دارد ـ یک حالت رفتاری بیرونی که قابل مشاهده است. (مثلاً، انتظار، محاسبات، چاپ، چرخش و بررسی*) که فقط زمانی تغییر می‌کنند که رویداد یا رخدادی رخ بدهد. برای مثال، نرم‌افزار در حالت انتظار باقی می‌ماند تا زمانی‌که:

۱) یک ساعت درونی مشخص می‌کند که یک مقطع زمانی گذشته است. ۲) رویدادی خارجی رخ داده است، ۳) یک سیستم برونی به نرم‌افزار سیگنالی می‌دهد تا به طریقه‌ای متفاوت عمل کند. یک مدل رفتاری، نمایشی از وضعیت‌های نرم‌افزار را ارائه داده و حوادث عامل تغییر در نرم‌افزار را نیز، نمایش می‌دهد.

مدل‌های فراهم شده در طول تحلیل نیازمندیها، چند وظیفه و نقش را برعهده دارند:

- مدل، به تحلیل‌گر کمک می‌کند تا اطلاعات، کارکرد و رفتار سیستم را درک کرده و از این طریق امر تحلیل نیازها را ساده‌تر کرده و بیشتر نظام مند نماید.
- مدل، نقطه کانون توجه می‌شود و بنابراین کلیدی برای تعیین کامل بودن سازگاری و دقت مشخص‌سازی خواهد بود.
- مدل، پایه طراحی واقع شده و برای طراح، شناختی اساسی از نرم‌افزار فراهم می‌کند تا بتواند برای پیاده‌سازی آن در بافت مربوطه "نقشه" تهیه نماید.

---

1 Information structure

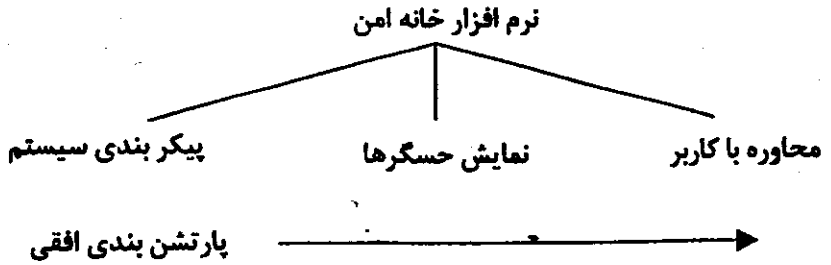
* Polling

مدل‌های تحلیل که در فصل ۱۲ و ۲۱ بحث شده‌اند روش‌های مدل‌سازی واقعی هستند. هر چند روش مدل‌سازی که ارائه شده اغلب یک مورد شخصی (یا سازمانی) برای انتخاب شده است. امر مدل‌سازی برای کار تحلیل خوب، اهمیت بنیادی دارد.

تجزیه و تفکیک فرآیندی است که نتیجه آن در تهیه کامل داده ها، کارکردها یا رفتار مشهود است. آنرا می توان به طور افقی یا عمودی انجام داد

۱۱-۳-۳ افراز (تجزیه)

مسائل اغلب بیش از اندازه بزرگ و پیچیده هستند تا بتوان به‌عنوان یک کل آنها را ادراک کرد. به‌خاطر این مسئله ما تمایل داریم تا این مسائل را تقسیم‌بندی یا (تقسیم) به قسمتهایی نماییم که به راحتی درک شده و بین قسمتها تعاملی برقرار کند که کارکرد کلی به مورد اجراء گذاشته شود. اصول تحلیل چهارم عملیاتی، چنین نظر می‌دهد که اطلاعات، کارکرد و حوزهٔ رفتاری نرم‌افزار تقسیم‌بندی شوند.



شکل ۱۱ـ۴ پارتیشن بندی افقی کارکرد خانه امن

ماهیتاً، تقسیم‌بندی[1] یک مسئله را به اجزای آن تقسیم می‌کند. به‌طور مفهولی ما یک نمایش سلسله مراتبی از کارکرد یا اطلاعات را ارائه کرده و بعد عنصری که در بالاترین وضعیت قرار گرفته است را تقسیم‌بندی می‌کنیم از طریق:

۱) عرضهٔ جزییات رو به افزایش با حرکت عمودی در سلسله مراتب یا ۲) به‌طور کارکردی مسئله را از طریق حرکت افقی در سلسله مراتب متلاشی می‌کنیم. برای نمایش این رهیافتهای تقسیم‌بندی کننده، اجازه دهید تا سیستم ایمنی "خانه امن" توصیف شده در بخش۱۱-۲-۲ را مجدداً در نظر بگیریم. تخصیص نرم‌افزار برای خانه امن (حاصل شده به‌عنوان پیامد مهندسی سیستم و فعالیت‌های **FAST**) را می توان به روش ذیل شرح داد:

نرم‌افزار خانه امن باعث می‌شود تا مالک خانه سیستم ایمنی را زمانی‌که نصب گردید پیکربندی کرده و تمام سنسورهای متصل به سیستم ایمنی را نظارت کند و این دستگاه با مالک خانه از طریق یک صفحه کلید و کلیدهای کارکردی درون صفحه کنترل خانه امن نمایش داده شده در شکل ۱۱-۳، تعامل کند.

در طول نصب دستگاه، صفحه کنترل خانه امن برای "برنامه‌ریزی" و پیکربندی سیستم، مورد استفاده قرار می‌گیرد. هر سنسور دارای یک شماره و یک نوع است. یک اسم رمز اصلی برای فعال و غیر فعال کردن سیستم، برنامه‌ریزی می‌شود. همچنین شماره تلفن‌هایی برای شماره‌گیری، زمانی‌که یک رویداد سنسوری رخ می‌دهد درون‌گذاری می‌شوند.

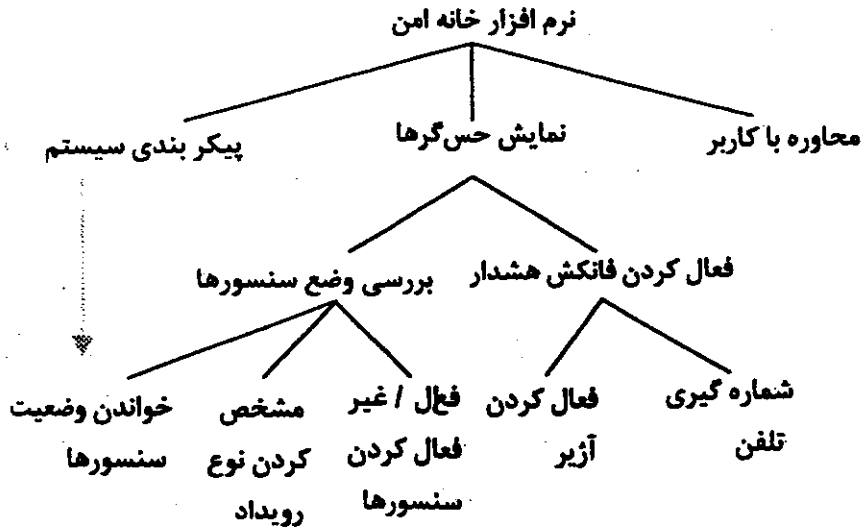
زمانی‌که یک حادثه سنسوری تأیید می‌شود نرم‌افزار یک آژیر قابل شنیدن را به سیستم متصل می‌کند. پس از یک تأخیر زمانی که توسط مالک خانه در طول فعالیت‌های پیکره‌بندی سیستم مشخص می‌شود نرم‌افزار شماره تلفنی را می‌گیرد که به خدمات نظارتی متصل است و اطلاعات در مورد مکان و ماهیت رویداد کشف شده فراهم می‌کند.

تمام تعاملات انجام شده با خانه امن توسط یک سیستم فرعی کاربرد تعامل که ورودی ارائه شده از طریق صفحه کلید و کلیدهای کارکرد، مدیریت می‌شود که شامل نمایش پیام‌های ارسالی روی صفحه نمایش LCD و نمایش اطلاعات وضعیت سیستم روی صفحه نمایش LCD نیز می‌شود. یک صفحه کلید به طریق ذیل تعامل می‌کند ...

نیازمندیهای مربوط به نرم‌افزار خانه امن را می‌توان با تقسیم‌بندی میادین اطلاعاتی، کارکردی و رفتاری محصول، تحلیل کرد. برای نمایش این امر میدان کارکردی شکل، تقسیم‌بندی خواهد شد. شکل ۱۱-۴ تقسیم شدن افقی[2] نرم‌افزار خانه امن را نشان می‌دهد. مسئله، با ارائه کارکردهای اجزاء نرم‌افزار خانه امن تقسیم‌بندی شده است و در سلسله مراتب درختی کارکردی به‌صورت افقی حرکت می‌کند. کارکرد اصلی در اولین سطح از سلسله مراتب بررسی شده‌اند.

1.partitianing

2.horizontal decomposition



شکل ۱۱ـ۵ پارتیشن بندی عمودی کارکرد خانه امن

کارکردهای فرعی همراه با کارکرد اصلی خانه امن را می‌توان از طریق عرضهٔ عمودی جزییات در سلسله مراتب، مانند شکل ۱۱-۵ مورد آزمون قرار داد با حرکت رو به پایین در راستای یک راه منفرد در زیر سنسورهای ناظر کارکرد، تقسیم‌بندی بطور عمودی رخ می‌دهد که سطوح رو به افزایش جزییات کارکردی را نمایش دهد.

راهکار تقسیم‌بندی که ما برای بررسی کارکردهای خانه امن به‌کار گرفته‌ایم، همچنین می‌توانند در مورد میدان اطلاعات و میدان رفتاری نیز به‌کار گرفته شوند. در حقیقت تقسیم‌بندی جریان اطلاعات و رفتار سیستم (مورد بررسی قرار گرفته در فصل ۱۲)، شناخت بهتری از نیازهای نرم‌افزار را فراهم می‌آورد. هنگامی‌که مسئله تقسیم‌بندی گردید، تعاملات بین کارکردها استخراج می‌شوند. قلم‌های داده‌ها و کنترل که در پهنهٔ یک تعامل حرکت می‌کنند، باید به ورودی‌های لازم جهت اجرای کارکرد بیان شده، محدود شود و برخروجی‌هایی که از سوی سایر کارکردها یا عناصر سیستم مورد نیاز هستند نیز، این محدودیت اعمال گردد.

۱۱-۳-۴ دیدگاه های اساسی و پیاده سازی[1]

یک دید ضروری از نیازمندیهای نرم‌افزار کارکردهایی را که باید اجرا شوند و اطلاعاتی را که باید پردازش شوند، بدون در نظر گرفتن جزییات پیاده‌سازی ارائه می‌کند. برای مثال، دید ضروری از کارکرد

۱. بسیاری از افراد (واژگان) دید منطقی و فیزیکی را به کار می برند در حالیکه نسبت به این مفاهیم آگاهی ندارند.

"وضعیت خواندن سنسور"[1] در "خانهٔ امن" خود را درگیر شکل فیزیکی داده‌ها و یا نوع سنسور به‌کار رفته، نمی‌کند. در حقیقت، می‌توان چنین استدلال کرد "وضعیت خواندن" یک اسم مناسب‌تر برای این کارکرد است چرا که از جزییات مربوط به مکانیسم ورودی به‌طور کلی صرف‌نظر می‌کند. به‌طور مشابه یک مدل داده‌های ضروری از قلم دادهٔ **شماره تلفن**[2] (القاء شده توسط کارکرد گرفتن شمارهٔ تلفن) می‌تواند در این مرحله ارائه شود بدون توجه به ساختار داده‌های زیر بنای آن (چنانچه وجود داشته باشد) که برای پیاده‌سازی قلم داده‌ها به‌کار رفته است. با متمرکز ساختن توجه خود بر روی ماهیت مشکل در مراحل اولیه مهندسی نیازمندیها، ما گزینه‌های خود را برای مشخص کردن جزییات پیاده‌سازی در طول مراحل بعدی مشخص‌سازی نیازمندیها و طراحی نرم‌افزار در اختیار خود نگاه خواهیم داشت.

دید پیاده‌سازی[3] نیازمندیهای نرم‌افزار، ظاهر شدن کارکردهای پردازش در دنیای واقعی و ساختارهای اطلاعات را میسر می‌کند. در پاره‌ای موارد یک نمایش فیزیکی به‌عنوان اولین مرحله در طراحی نرم‌افزار ارائه می‌شود. در هر حال اکثر سیستم‌های استوار به کامپیوتر به طریقه‌ای مشخص شده‌اند که عرضهٔ پاره‌ای جزییات خاص پیاده‌سازی را بیان کنند.یک دستگاه ورودی خانه امن یک سنسور محیطی (یک سگ نگهبان، نگهبان انسانی یا دام نمی‌باشد) سنسور ورود غیرقانونی را با اختلال در مدار الکترونیکی کشف می‌کند. خصوصیات کلی سنسور باید به‌عنوان قسمتی از مشخصات نیازهای نرم‌افزار مورد توجه واقع شود. تحلیل‌گر باید محدودیت‌های تحمیل شده از سوی عناصر سیستم از پیش تعریف شده (سنسور) متوجه شده و دید پیاده‌سازی کارکرد و اطلاعات را زمانی‌که چنین دیدی مناسب است، در نظر بگیرد.

ما پیش از این توجه داشتیم که مهندسی نیازمندیهای نرم‌افزار باید بر روی آنچه نرم‌افزار قرار است انجام دهد، متمرکز باشد تا این‌که فرآیند چگونه پیاده‌سازی خواهد شد. در هر صورت، دید پیاده‌سازی نباید ضرورتاً به‌عنوان یک بازنمایی چگونگی تفسیر شود. بلکه مدل پیاده‌سازی نمایندگی حالت عملیاتی جاری را به‌عهده دارد، یعنی آن‌که تخصیص فعلی و یا پیشنهاد شدهٔ مربوط به تمام عناصر سیستم را نمایش دهد. مدل اساسی (کارکرد و یا داده‌ها) بر این مفهوم که عملی شدن کارکرد به‌طور صریح اظهار نشده است، ژنریک می‌باشد.

۱۱-۴ نمونه سازی نرم افزار

کار تحلیلی باید بدون توجه به پارادایم مهندسی نرم‌افزار که به‌کار گرفته می‌شود، انجام می‌شود. در هر صورت، شکل تحلیل با یکدیگر متفاوت هستند. در برخی موارد امکان اعمال اصول تحلیل عملیاتی و برگرفتن یک مدل نرم‌افزاری برای کار طراحی امکان‌پذیر است. در دیگر موقعیت‌ها استخراج نیازمندیها (از

1.read sensor status

2.dial phone number

3.implementation view

طریق FAST، QFD، Use-Caseها و دیگر فنون "بررسی آراء و نظرات" [JOR 89][1] انجام می‌شود. اصول تحلیل اعمال می‌شوند و مدلی برای نرم‌افزاری ایجاد می‌شود که نمونهٔ اولیه[2] نامیده شده و برای ارزیابی مشتری و توسعه‌دهنده، ساخته شده است. سرانجام شرایطی وجود دارند که ساخت یک نمونهٔ اولیه برای را آغاز تحلیل ضروری می‌سازد، چرا که مدل تنها وسیلهٔ استخراج مؤثر نیازمندیها بوده است. این مدل سپس به‌صورت نرم‌افزار نهایی به تکامل می‌رسد.

۱۱-۴-۱ انتخاب رهیافت نمونه سازی

پارادایم ساخت نمونه اولیه می‌تواند انتها ـ بسته و یا انتها ـ گشوده باشد. راهکار انتها ـ بسته غالباً نمونه اولیه‌سازی دور انداختنی[3] نامیده می‌شود. با به‌کارگیری این راهکار یک نمونه اولیه فقط به‌عنوان یک نماد کلی از نیازمندیها، به‌کار ما می‌آید. سپس آن را به کناری نهاده و نرم‌افزار با استفاده از پارادایم متفاوت دیگری، مهندسی می‌شود. یک راهکار انتها ـ گشوده، نمونه اولیه‌سازی تکاملی[4] نامیده می‌شود که از نمونه اولیه به‌عنوان اولین قیمت کار تحلیلی که به طراحی و تولید منتهی خواهد شد استفاده می‌کند. نمونه اولیه نرم‌افزار، اولین مرحله تکاملی سیستم به‌اتمام رسیده است.

پرسش

برای تعیین آنکه آیا مدل نمونه، رهی افتی مناسب می باشد یا خیر، به دنبال چه باید باشیم؟

پیش از آن‌که بتوان یکی از این دو راهکار انتها ـ بسته یا انتها ـ گشوده، را انتخاب کرد باید این مسئله که آیا برای سیستم موردنظر می‌توان نمونهٔ اولیه ساخت یا خیر، معین شود. تعدادی عوامل کاندیدا، نمونه اولیه‌سازی [BOA84][5] را می‌توان تعریف کرد :[6] محدودهٔ به‌کارگیری، پیچیدگی به‌کارگیری، خصوصیات مشتری، و خصوصیات پروژه.

به‌طور کلی، هر گونه به‌کارگیری که صفحه نمایش‌های تصویری پویا خلق می‌کند، شدیداً با کاربر تعامل می‌کند و یا به الگوریتم‌ها و فرآیندهای ترکیبی نیاز دارد که باید به طریقی تکامل به‌وجود آیند، را کاندیدایی برای نمونه اولیه‌سازی تلقی می‌کنیم. در هر حال این حوزه‌های به‌کارگیری را باید در برابر پیچیدگی به‌کارگیری بررسی کرد. اگر به‌کارگیری یک کاندیدا (همانی که خصوصاتش قبلاً ذکر شد) نیاز ایجاد دهها هزار خط برنامه داشته باشد، پیش از آن‌که هرگونه کارکرد قابل ارائه را بتوان اجرا کرد، احتمالاً برای امر نمونه اولیه‌سازی بیش از اندازه پیچیده است.[7] چنانچه، بتوان پیچیدگی را تقسیم‌بندی کرد. هم‌چنان امکان ساخت نمونه اولیه برای قسمت‌های مختلف نرم‌افزار وجود دارد.

1.Jordan,P.W.

2.prototype

3.throwaway prototyping

4.evolutionary prototyping

5.Boar,B.

۶. توضیح مفید دیگری در خصوص دیگر عوامل کاندیدا – " چه هنگام مدل نمونه " می توانند در [DAV95b] یافت شوند.

۷. در برخی موارد، نمونه های واقعا" پیچیده، می توانند توسط فنون نسل چهارم یا اجزاء نرم افزاری قابل استفاده محدد به سرعت ساخته شوند.

سؤال	مدل نمونه تکمیل شده	مدل نمونه تکمیل شده	کار اضافی مورد نیاز است
آیا حوزه کاربردی درک شده است ؟	Yes	Yes	No
آیا مسئله مدل شدنی است ؟	Yes	Yes	No
آیا مشتری از نیازمندیهای اولیه سیستم است ؟	Yes / No	Yes / No	No
آیا نیازمندیها تثبیت شده و پایدار است ؟	Yes	No	Yes
آیا هیچ نیازمندی مبهمی وجود دارد ؟	No	Yes	Yes
آیا در نیازمندیها تناقض و تعارض وجود دارد ؟	No	Yes	Yes

شکل ۱۱ـ۶ انتخاب رهیافت مناسب برای مدل نمونه

از آن‌جا مشتری باید با نمونه اولیه در مراحل بعدی تعامل کند، ضرورت دارد که:

۱) منابع مشتری در خدمت ارزیابی و پالایش نمونه اولیه باشند. و ۲) مشتری قادر به اتخاذ تصمیمات مربوط به نیازمندی به‌صورت به هنگام باشد. سرانجام ماهیت پروژهٔ توسعه یک تأثیر بزرگ بر روی کارآیی نمونه اولیه دارد. آیا مدیریت پروژه قادر و مایل به کار با روش نمونه اولیه هست؟ آیا ابزارهای ساخت نمونه اولیه فراهم هستند؟ آندریوله [AND94][1] شش سؤال را پیشنهاد می‌کند [شکل ۱۱-۶] و مجموعه‌های پاسخ معمول و راهکارهای نمونه اولیه‌سازی مربوط به آنها را ارائه داده است.

۱۱-۴-۲شیوه ها و ابزارهای نمونه سازی

برای آن‌که نمونه اولیه نرم‌افزار کارآمد باشد یک نمونه اولیه باید با سرعت ایجاد شود تا مشتری نتایج را ارزیابی کرده و تغییرات را توصیه کند. برای ارائه سریع نمونه اولیه، سه کلاس عمومی از روش‌ها و ابزارها (مثلاً [AND94] و [TAN89][2])، فراهم هستند:

فنون نسل چهارم. فنون نسل چهارم (4GT)، طبقه وسیعی از پرسش‌های برپایه داده‌ها، گزارش زبان‌ها، برنامه و مولدهای به‌کارگیری و سایر زبان‌های غیر ـ رویه‌ای سطح بالا را در برمی‌گیرد. از آن‌جا که 4GT مهندس نرم‌افزار را قادر می‌سازد تا برنامه را سریعاً اجرا کند، برای نمونه اولیه‌سازی ایده‌آل است.

1.Andriole,S.

2.Tanik,M.M

اجزاء قابل استفاده مجدد نرم‌افزار. راهکار دیگر برای نمونه اولیه‌سازی سریع، جفت و جور کردن و سولر کردن اجزاء به جای ساختن نمونه اولیه است که با کمک اجزاء نرم‌افزارهای موجود انجام می‌شود. وارد بازی کردن نمونه اولیه‌سازی و به‌کارگیری مجدد اجزاء برنامه فقط زمانی مؤثر است که سیستم کتابخانه موجود باشد تا اجزاء موجود را بتوان کاتالوگ کرده و سپس بدان‌ها دست یافت.

باید خاطرنشان شود که یک محصول نرم‌افزاری موجود را می‌توان به‌عنوان یک نمونه اولیه برای یک محصول رقابتی "جدید و اصلاح شده" به‌کار برد. بهنوعی این کار همان استفاده مجدد از نمونه اولیه نرم‌افزار است.

مشخصات رسمی و محیطهای نمونه اولیه‌سازی. طی دو دهۀ گذشته تعدادی زبان‌های مشخص‌سازی رسمی و ابزارهایی بوجود آمده‌اند که برای فنون مشخص‌سازی زبان جانشین شوند. امروزه، طراحان این زبان‌های رسمی در فرآیند ایجاد محیطهای تعاملی هستند که: ۱) تحلیل‌گر را قادر سازد تا بطور تعاملی یک مشخص‌سازی بر پایه زبان از سیستم یا نرم‌افزار را خلق نماید، ۲) از ابزارهای خودکاری مدد بگیرد که مشخص‌سازی بر پایه زبان را به برنامه‌های قابل اجرا ترجمه کند، ۳) مشتری را قادر سازد تا برنامه قابل اجرای نمونه اولیه را برای پالایش نیازمندیهای رسمی به‌کار گیرد.

۱۱-۵ تعیین مشخصات

تردیدی وجود ندارد که حالت مشخص‌سازی با کیفیت راه‌حل، ارتباط زیادی دارد. مهندسین نرم‌افزار که وادار شده‌اند تا با مشخصات ناقص، ناسازگار یا گمراه‌کننده کارکنند، دلسردی و سردرگمی قطعی ناشی از آن را تجربه کرده‌اند. کیفیت، ارائه بموقع و کامل بودن نرم‌افزار هم از این طریق، صدمه می‌خورد.

۱۱-۵-۱ اصول تعیین مشخصات

مشخص‌سازی، صرفنظر از حالتی که آن را انجام می‌دهیم، می‌تواند به‌عنوان یک فرآیند نماد و نمود تلقی شود. نیازها به طریقه‌ای ارائه شده‌اند که سرانجام به پیاده‌سازی نرم‌افزار مؤفق منتهی می‌شود. تعدادی از اصول مشخص‌سازی، برگرفته از کار تحقیقی بالزر[1] و گودمن[2] [BAL 86][3] به قرار ذیل هستند:

۱) جدا کردن قابلیت کارکردی از پیاده‌سازی.

۲) توسعۀ مدلی از رفتار مطلوب سیستمی که داده‌ها و پاسخ‌های کارکردی سیستم را به انواع محرک‌های محیط در برمی‌گیرد.

1.Balzer

2.Goodman

3.Balzer, R. and N.

۳) ایجاد متنی که نرم‌افزار از طریق مشخص ساختن طریقهٔ تعامل اجزاء سیستم با نرم‌افزار در آن عمل نماید.

۴) تعریف محیطی که سیستم در آن عمل می‌کند و روشن کردن چگونگی واکنش مجموعه‌ای شدیداً درهم تنیده از عوامل با محرک درون محیطی که توسط آن عوامل ایجاد شده است (تغییرات در اشیاء). [BAL 86]

۵) ایجاد یک مدل عقلانی تا یک مدل طراحی یا پیاده‌سازی، مدل عقلانی یک سیستم را آن‌گونه که از سوی جامعه کاربران ادراک می‌شود تشریح می‌کند.

۶) فهم این مسئله که "مشخص‌سازی باید نواقص را تحمل کرده و قابل بهبود باشد." یک مشخص‌سازی همیشه یک مدل است ـ یک انتزاع ـ از چند موقعیت واقعی (یا تحلیلی) که معمولاً کاملاً پیچیده است. بدین خاطر، ناقص خواهد بود و این نقص در سطوح بسیاری از جزییات حضور خواهد داشت.

۷) ایجاد مضمون و ساختار یک مشخص‌سازی به طریقه‌ای که آن را قابل تطبیق با تغییرات نماید.

لیست اصول اساسی مشخص‌سازی ذکر شده در فوق، پایه‌ای را برای نمایش نیازمندیهای نرم‌افزاری فراهم می‌کند. در هر حال، این اصول باید توان به‌عمل درآورده شدن را بدست آورند.

در بخش بعدی، ما مجموعه‌ای از رهنمودها را برای ایجاد یک مشخص‌سازی نیازمندیها مورد بررسی قرار خواهیم داد.

۱۱-۵-۲ بازنمایی

ما پیش از این دیده‌ایم که نیازمندیهای نرم‌افزار می‌تواند به طرق مختلف مشخص شود. در هر صورت اگر نیازمندیها روی کاغذ آمده و یا از وسیله‌ای الکترونیک (که غالباً باید چنین باشد) برای نمایش دادن آنها استفاده شود، یکسری رهنمودها ارزش به‌کارگیری خواهند داشت.

نمایش و ارائه چارچوب و مضمون باید با مسئله مرتبط باشد. یک طرح عمومی برای مضامین مشخص‌سازی نیازمندیهای نرم‌افزار[1] را می‌توان ترسیم کرد. به هر حال، شکلهای بازنمایی مسئله در برگرفته شده توسط مشخص‌سازی احتمالاً با توجه به حوزهٔ به‌کارگیری تغییر می‌کنند. مثلاً، مشخص‌سازی یک سیستم اتوماسیون ساخت از نمادهای متفاوت، دیاگرام‌ها و زبان متفاوت از آنچه برای کامپایلر زبان برنامه‌نویسی مشخص شده است، استفاده می‌کند.

پرسش
رهنمودهای اولیه برای بازنمایی نیازمندیها کدام است؟

اطلاعات لحاظ شده در مشخص‌سازی باید تو در تو باشند. بازنمایی مسایل باید لایه‌های اطلاعات را به‌نحوی که یک خواننده به سطح جزییاتی که مورد نیاز حرکت کند، آشکار نماید. شمای شمارش پاراگراف و دیاگرام‌ها باید بیانگر سطح جزییاتی باشد که در حال ارائه شده می‌باشد. گاهی اوقات ارائه اطلاعات یکسان در سطوح مختلف برای کمک به فهم مسایل حائز ارزش است.

1. Software Requirements Specification

دیاگرام‌ها و دیگر علائم، باید از نظر تعداد محدود شده و از نظر به‌کارگیری سازگاری داشته باشد. علامت‌گذاری گیج‌کننده یا ناسازگار، گرافیکی یا نمادین تفاوتی ندارد، ادراک را کاهش داده و اشتباهات را زیاد می‌کند.

بازنمایی باید قابل تجدیدنظر باشد. مضمون مشخص‌سازی تغییر خواهد کرد. به‌طور ایده‌آل، ابزارهای CASE باید برای روزآمد کردن تمام موارد بازنمایی که تحت‌تأثیر هر تغییری قرار می‌گیرند، فراهم باشند.

تحقیقات زیادی انجام شده است (مثلاً [HOL95][1] و [CUR85][2]) تا شناخت بهتری از فاکتورهای همراه با مشخص‌سازی، بدست آید. تردیدی کمی وجود دارد که نماد ـ شناختی و ترتیب بر روی فهم امور، تأثیر می‌گذارد. در هر حال، مهندسین نرم‌افزار ظاهراً علایق شخصی خود را نسبت به انواع نمادها و دیاگرام‌ها دارند.

آشنایی داشتن با این علایم و نمادها، پایه و اساس ایجاد گرایش در به‌کارگیری آنها بوده ولی دیگر فاکتورهای ملموس مانند نظم فضایی، الگوها و طرح‌های به‌راحتی قابل ادراک و مقداری رسمی بودن این علایم، اغلب در انتخاب هر فرد تأثیر دارد.

۱۱-۵-۳ تعیین مشخصات نیازمندیهای نرم افزار

مشخص‌سازی نیازهای نرم‌افزار[3]، حاصل‌نهایی کار تحلیلی است. کارکرد و عملکرد نرم‌افزار، به‌عنوان قسمتی از مهندسی سیستم از طریق ایجاد یک شرح اطلاعات کامل، شرح تفصیلی کارکرد، ارائه رفتار سیستم، بیان نیازمندیهای عملکردی و محدودیت‌های آن، معیارهای اعتبارسنجی مناسب و دیگر اطلاعات مربوط به نیازمندیها، پالایش می‌شود.

بازبینی مشخصات نیازمندیهای نرم افزار

معرفی[4] مشخص‌سازی نیازهای نرم‌افزار بیانگر اهداف و مقاصد نرم‌افزار و شرح آن در بافت سیستم استوار بر کامپیوتر است. در واقع این معرفی شاید چیزی به‌جز میدان عمل نرم‌افزار در مورد تنظیم و طراحی سند نباشد.

شرح اطلاعات[5] یک توصیف تفصیلی از مسئله‌ای را که نرم‌افزار باید حل کند، فراهم می‌کند. مضمون اطلاعات، جریان و ساختار آن مستند می‌شوند. سخت‌افزار و نرم‌افزار و تعاملات انسانی برای عناصر سیستم خارجی و کارکردهای درونی نرم‌افزار شرح داده شده‌اند.

1.Holtzblatt,K. and E.

2.Curtis,B.

3.Software Requirements Specification

4.Introduction

5.Information Description

شرح هر کارکرد[1] که برای حل مسئله مورد نیاز است در شرح کارکردی ارائه می‌شود. یک شرح روایتی پردازشگر برای هر کارکرد ارائه می‌شود: محدودیت‌های طراحی، بیان و توجیه می‌شوند، خصوصیات عملکردی آن اظهار شده و یک یا چند دیاگرام برای نمایش گرافیکی ساختار کلی نرم‌افزار لحاظ می‌شوند. بخش شرح رفتاری[2] مشخص‌سازی عملیات نرم‌افزار را به‌عنوان خصوصیات کنترل سلسله‌ای از حوادث بیرونی و درونی، مورد آزمون قرار می‌دهد.

معیار اعتبار سنجی[3] شاید مهم‌ترین و اغلب مورد بی‌اعتنایی قرار گرفته‌ترین بخش مشخص‌سازی نیازهای نرم‌افزار است. چگونه می‌توان یک پیاده‌سازی مؤفق را شناخت؟ چه گروهی از آزمون‌ها باید برای اعتباردهی به کارکرد، عملکرد و محدودیت‌ها انجام شوند؟

ما از این بخش صرفنظر می‌کنیم، چرا که تکمیل آن نیاز به فهم کامل نیازمندیهای نرم‌افزاری دارد- چیزی که غالباً در این مرحله فاقد آن هستیم. با این وصف، مشخص‌سازی معیارهای اعتبارسنجی به‌عنوان یک بازبینی تلویحی کلیه نیازمندیها عمل می‌کند. صرف وقت و توجه به این بخش ضرورت دارد. سرانجام مشخص‌سازی شامل یک کتابشناسی و پیوست[4] می‌شود.

کتابشناسی ارجاع به تمام مستندات مربوط به نرم‌افزار دارد. مشتمل بر مستندات مهندسی نرم‌افزار، ارجاعات فنی، استانداردهای تولیدکننده، پیوست مشتمل بر اطلاعاتی است که تولیدکنندگان مشخص می‌سازند:داده‌های جدولی، توصیف مفصل الگوریتمها، چارتها، گرافها و دیگر مواد. در بسیاری موارد، مشخص‌سازی نیازهای نرم‌افزار می‌تواند با یک نمونه اولیه قابل اجرا (که در برخی مواقع می‌تواند جانشین مشخص‌سازی شود) یک نمونه اولیه کاغذی یا راهنمای کاربر مبتدی[1]، همراه شود. راهنمای کاربر مبتدی نرم‌افزار را به‌عنوان یک جعبه سیاه، نمایش می‌دهد. یعنی آن‌که تأکید را روی ورودی کاربر و خروجی بدست آمده، قرار می‌دهد. این راهنما می‌تواند یک ابزار با ارزش برای آشکار کردن مشکلات در تعامل انسان - ماشین باشد.

۱۱-۶ بازبینی مشخصات

مروری بر مشخص‌سازی نیازهای نرم‌افزار (و/ یا نمونه اولیه) توسط مهندس نرم‌افزار کاربر هر دو انجام می‌شود. چرا که مشخص‌سازی، زیربنای مرحلهٔ توسعه را تشکیل داده و باید در مورد این بررسی و مرور دقت زیادی صورت بگیرد. این مرور در آغاز در سطح ماکروسکوپیک (کلان) انجام می‌شود. یعنی مرورکنندگان در تلاش هستند تا بررسی کنند آیا کامل بودن، سازگار بودن و دقیق بودن مشخص‌سازی

بازبینی مشخصات نیازمندیهای نرم افزار

1.Functional Description
2.Behavioral Description
3. Validation Criteria
4.Bibiography and Appendix

هنگامی‌که اطلاعات کلی، کارکردی و حوزه رفتاری در نظر گرفته می‌شوند، یا خیر و از این نظر موارد را تأیید کنند. در هر حال، برای بررسی کامل هر یک از این حوزه‌ها، بررسی دارای جزییات بیشتری می‌شود و نه فقط شرح‌های وسیع را امتحان می‌کند بلکه طریقهٔ ارائه شدن نیازمندیها، بررسی و کنترل می‌شود. مثلاً، زمانی‌که یک مشخص‌سازی دارای "واژه‌های مبهم" است. (مثلاً بعضی، گاهی، اغلب، معمولاً، به‌طور معمول، اکثراً)[2] بررسی‌کننده باید موضوع را مورد بررسی بیشتر قرار دهد تا نکته روشن شود.

زمانی‌که بررسی کامل شد، مشخص‌سازی نیازهای نرم‌افزار از سوی تحلیل‌گر و مشتری هر دو به کناری نهاده می‌شود. مشخص‌سازی برای توسعهٔ نرم‌افزار مانند یک "قرارداد" شده است. تقاضا در مورد اعمال تغییرات در نیازها پس از نهایی شده مشخص‌سازی حذف نخواهند شد. اما مشتری باید بداند که این‌گونه تغییرات پس از نهایی شدن کار به‌معنای صرف هزینهٔ بیشتر و طول مدت بیشتر قرارداد خواهد بود. حتی با وجود به‌کارگیری بهترین رویه‌های بررسی، تعدادی مشکلات مربوط به مشخص‌سازی هم‌چنان وجود دارند. آزمودن مشخص‌سازی به هر روش معناداری، مشکل و دشوار است و بنابراین عدم سازگاری‌ها یا موارد حذف می‌توانند بدون آن‌که تشخیص داده شوند، وارد کار شوند. در طول بررسی، تغییرات اعمال شده در مشخص‌سازی توصیه می‌شود. بسیار مشکل است که تأثیر عمومی یک تغییر را ارزیابی کرد. یعنی این‌که چگونه تغییری در یک کارکرد، نیازمندیهای کارکردهای دیگر را تحت‌تأثیر قرار می‌دهد؟ محیط‌های مهندسی نرم‌افزار مدرن (فصل ۳۱) از ابزارهای CASE برای حل مشکلات استفاده می‌کنند.

۱۱-۷ خلاصه

تحلیل نیازمندیها اولین مرحلهٔ فنی در فرآیند نرم‌افزار است. در این نقطه است که یک اظهاریه کلی در مورد میدان عمل نرم‌افزار تبدیل به مشخص‌سازی روشن و پالایش یافته می‌شود تا تمام مهندسین نرم‌افزار از آن استفاده نمایند.

تحلیل باید روی میادین اطلاعات، کارکردی و رفتاری یک مسئله، متمرکز شود. برای فهم بهتر آنچه که مورد نیاز است. مدل‌هایی خلق می‌شوند، مسئله تقسیم‌بندی می‌شود و عوامل نمایشی و ارائه مطالب ماهیت نیازمندیها را روشن کرده و بعداً جزییات پیاده‌سازی مطرح می‌شوند.

در بسیاری موارد امکان مشخص کردن کامل یک مسئله در مرحلهٔ اولیه نیست. نمونهٔ اولیه رهیافت‌های جانشین را ارائه می‌دهد که حاصل آن یک مدل قابل اجرای نرم‌افزاری که نیازمندیها را می‌توان از طریق آن پالایش کرده، می‌باشد. برای انجام مناسب ساخت نمونه اولیه ابزارها و فنون ویژه مورد نیاز است.

1.Preliminary User's Manual

2.some, sometimes, often, usually, ordinarily, most, or mostly

مشخص‌سازی نیازمندیهای نرم‌افزار به‌عنوان پیامد تحلیل به‌وجود آمده است. مرور و بررسی برای حصول اطمینان از این امرکه برداشت و ادراک مشتری و تحلیل‌گر از سیستم یکسان است یا خیر، ضروری است. متأسفانه حتی با بهترین شیوه‌ها، مسئله آن است که (هر) مسئله مدام تغییر می‌کند.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۱-۱۱ تحلیل نیازهای نرم‌افزار بدون شک مرحله‌ای است که شدیداً در پروسهٔ نرم‌افزار الزام به برقراری ارتباطات دیده می شود. چرا مسیر ارتباطات مرتباً دچار مشکل می‌شود؟

۲-۱۱ غالباً تبعات سیاسی سنگینی هنگام شروع تحلیل نیازهای نرم‌افزار (و/ یا تحلیل سیستم) وجود دارد. مثلاً کارگران ممکن است احساس کنند که امنیت شغلی آنان از سوی یک سیستم خودکار جدید، به‌خطر می‌افتد. این مشکلات را چه عواملی سبب می‌شوند؟ آیا امر تحلیل می‌تواند به طریقی انجام شود که نقش سیاست به حداقل برسد؟

۳-۱۱ استنباط خود را دربارهٔ آموزش و پیش‌زمینهٔ یک تحلیل‌گر سیستم‌ها، بحث کنید.

۴-۱۱ طی این فصل ما در مورد "مشتری" سخن گفتیم. "مشتری" را برای تحلیل‌گران سیستم‌های اطلاعات، سازندگان محصولات کامپیوتری یا مرتبط با کامپیوتر و سازندگان سیستم شرح دهید. در این مورد دقت کنید شاید این مشکل دارای جنبه‌هایی باشد که در نظر اول متوجه آنها نشده باشید.

۵-۱۱ یک "کیت" کاربردهای تسهیل یافتهٔ به‌کارگیری مشخص‌سازی (FAST) بسازید. این کیت باید دارای مجموعه‌ای از خطوط راهنمایی‌کننده جهت ادارهٔ جلسات FAST، موادی که برای سهولت تهیه لیستها می‌توانند به‌کار بردند و هر موردی که در شناسایی نیازها به ما کمک کند باشد.

۶-۱۱ مربی شما کلاس را به گروه‌های ۴ یا ۶ دانش‌آموزی خواهد کرد. نیمی از کلاس نقش دپارتمان بازاریابیو نیم دیگر نقش مهندس نرم‌افزار را بازی خواهد کرد. کار شما تعیین نیازها برای سیستم ایمنی خانه امن توصیف شده در این فصل است. یک جلسه ملاقات FAST را با کمک خطوط راهنمایی‌کنندهٔ این فصل برگزار کنید.

۷-۱۱ آیا شایسته است که بگوییم یک کتابچهٔ راهنمای کاربر مبتدی، نوعی از نمونه اولیه است؟ پاسخ خود را شرح دهید.

۸-۱۱ حوزه اطلاعات برای خانه امن را تحلیل کنید. جریان اطلاعات را در سیستم، مضمون اطلاعات و هر نوع ساختار اطلاعات که با آن مرتبط است. ارائه کنید (از هر نوع روش علایم‌گذاری که به‌نظر مناسب برآید، استفاده کنید)

۹-۱۱ حوزهٔ کارکردی خانه امن را تقسیم‌بندی کنید. اول تقسیم‌بندی افقی را انجام دهید و بعد تقسیم‌بعدی عمودی را انجام دهید.

۱۰-۱۱ نمایشگرهای اساسی و پیاده‌سازی سیستم خانه امن را ایجاد کنید.

۱۱-۱۱ یک نمونه اولیه کاغذی (یا نمونه اولیه واقعی) از خانه امن ایجاد کنید. از نمایش تعامل مالک با کارکرد کل سیستم، اطمینان حاصل کنید.

۱۱-۱۲ تلاش کنید تا اجزاء نرم‌افزار خانه امن را که ممکن است در دیگر محصولات یا سیستم‌ها قابل استفاده مجدد باشد، تعیین کنید. سعی کنید تا این اجزاء را گروه‌بندی کنید.

۱۱-۱۳ یک مشخص‌سازی مکتوب برای خانه امن را با کمک طرح کلی ارائه شده در وب‌سایت SEPA، تهیه نمایید. (توجه: مربی شما پیشنهاد خواهد کرد که کدام بخش‌ها در این زمان تکمیل می‌شوند.) اطمینان حاصل کنید که سؤالات به‌کار رفته برای تشریح بررسی و مرور مشخصات را به‌کار خواهید گرفت.

فهرست منابع و مراجع

[AKA90] Akao, Y., ed., *Quality Function Deployment: Integrating Customer Requirements in Product Design* (translated by G. Mazur), Productivity Press, 1990.

[AND92] Andriole, S., *Rapid Application Prototyping*, QED, 1992.

[BAL86] Balzer, R. and N. Goodman, "Principles of Good Specification and Their Implications for Specification Languages," in *Software Specification Techniques* (Gehani, N. and A. McGetrick, eds.), Addison-Wesley, 1986, pp. 25-39.

[BOA84] Boar, B., *Application Prototyping*, Wiley-Interscience, 1984.

[BOS91] Bossert, J.L., *Quality Function Deployment: A Practitioner's Approach*, ASQC Press,1991.

[CUR85] Curtis, B., *Human Factors in Software Development*, IEEE Computer Society Press, 1985.

[DAV93] Davis, A., *Software Requirements: Objects, Functions and States*, Prentice-Hall, 1993.

[DAV95a] Davis, A., *201 Principles of Software Development*, McGraw-Hill, 1995.

[DAV95b] Davis, A., "Software Prototyping," in *Advances in Computers*, volume 40, Academic Press, 1995.

[GAU89] Gause, D.C. and G.M. Weinberg, *Exploring Requirements: Quality Before Design*, Dorset House, 1989.

[HOL95] Holtzblatt, K. and E. Carmel (eds.), "Requirements Gathering: The Human Factor," special issue of *CACM*, vol. 38, no. 5, May 1995.

[JAC92] Jacobson, I., *Object-Oriented Software Engineering*, Addison-Wesley, 1992.

[JOR89] Jordan, P.W., et al., "Software Storming: Combining Rapid Prototyping and Knowledge Engineering," *IEEE Computer*, vol. 22, no. 5, May 1989, pp. 39-50.

[RE194] Reifer, D.J., "Requirements Engineering," in *Encyclopedia of Software Engineering (J.J.* Marciniak, ed.), Wiley, 1994, pp. 1043-1054.

[TAN89] Tanik, M.M. and R.T. Yeh (eds.), "Rapid Prototyping in Software Development," special issue of *IEEE Computer*, vol. 22, no. 5, May 1989.

[WYD96] Wyder, T, "Capturing Requirements with Use-Cases," *Software Development*, February 1996, pp. 37-40.

[ZAH90] Zahniser, R.A., "Building Software in Groups," *American Programmer*, vol. 3, nos. 7-8, July-August 1990.

[ZUL92] Zultner, R., "Quality Function Deployment for Software: Satisfying Customers," *American Programmer*, February 1992, pp. 28-4 I.

خواندنیهای دیگر و منابع اطلاعاتی

Books that address requirements engineering provide a good foundation for the study of basic analysis concepts and principles. Thayer and Dorfman *(Software Requirements Engineering*, 2nd ed., IEEE Computer Society Press, 1997) present a worthwhile anthology on the subject. Graham and Graham *(Requirements Engineering and Rapid*

Development, Addison-Wesley, 1998) emphasize rapid development and the use of object-oriented methods in their discussion of requirements engineering, while Mac-Cauley *(Requirements Engineering*, Springer-Verlag, 1996) presents a brief academic treatment of the subject.

In years past, the literature emphasized requirements modeling and specification methods, but today, equal emphasis has been given to effective methods for software requirements elicitation. Wood and Silver *(Joint Application Development*, 2nd ed., Wiley, 1995) have written the definitive treatment of joint application development. Cohen and Cohen *(Quality Function Deployment*, Addison-Wesley, 1995), Terninko *(Step-by-Step QFD: Customer-Driven Product Design*, Saint Lucie Press, 1997), Gause and Weinberg [GAU89], and Zahniser [ZAH90] discuss the mechanics of effective meetings, methods for brainstorming, and elicitation approaches that can be used to clarify results and a variety of other useful issues. Use-cases have become an important part of object-oriented requirements analysis, but they can be used regardless of the implementation technology selected. Rosenburg and Scott *(Use-Case Driven Object Modeling with UML: A Practical Approach*, Addison-Wesley, 1999), Schneider et al. *(Applying Use-Cases: A Practical Guide*, Addison-Wesley, 1998). and Texel and Williams *(Use-Cases Combined With Booch/OMT/UML*, Prentice-Hall, 1997) provide detailed guidance and many useful examples.

Information domain analysis is a fundamental principle of requirements analysis. Books by Mattison *(The Object-Oriented Enterprise*, McGraw-Hill, 1994), Tillman (*A Practical Guide to Logical Data Modeling*, McGraw-Hill, 1993), and Modell *(Data Analysis, Data Modeling and Classification*, McGraw-Hill, 1992) cover various aspects of this important subject.

A recent book by Harrison *(prototyping and Software Development*, Springer-Verlag, 1999) provides a modern perspective on software prototyping. Two books by Connell and Shafer *(Structured Rapid Prototyping*, Prentice-Hall, 1989) and *(Object-Oriented Rapid Prototyping*, Yourdon Press, 1994) show how this important analysis technique can be used in both conventional and object -oriented environments. Other books by Pomberger et al. *(Object Orientation and Prototyping in Software Engineering*, Prentice-Hall, 1996) and Krief et al. *(Prototyping with Objects*, Prentice-Hall, 1996) examine prototyping from the object-oriented perspective. The *IEEE Proceedings of the International Workshop on Rapid System Prototyping* (published yearly) presents current research in the area.

A wide variety of information sources on requirements analysis and related subjects is available on the Internet. An up-to-date list of World Wide Web references that are relevant to analysis concepts and methods can be found at the SEPA Web site:

http://www.mhhe.com/engcs/compsci/pressman/resources/reqm.mhtml