به نام خدا

تمرین اول متودولوژی‌های شیئ‌گرا
متودولوژی هاجماک

نعیم اصفهانی
۸۴۲۰۱۰۰۳

۱. خلاصه‌ی متودولوژی هاج‌ماک

مقدمه

در زمان ارائه‌ی متودولوژی هاج‌ماک بحث درگیری بین انواع مختلف متودولوژی‌ها و تکنیک‌های مختلف مورد استفاده (تابعی، ساختاری و شیئ‌گرا) هنوز مطرح بوده است. این متودولوژی بر اساس خوبی‌هایی که در شیئ‌گرایی و توانایی‌های آن می‌دیده است آن را به عنوان پایه انتخاب کرده است. در زمان این متودولوژی تعداد زیادی متودولوژی مطرح شده بودند که هیچ کدام به طور کامل و سازگار کل چرخه‌ی تولید نرم‌افزار را نمی‌پوشانده‌اند. این متودولوژی بر پایه‌ی چندین متودولوژی مطرح شده‌ی آن زمان که برخی از آن‌ها شیئ‌گرا هم نبودند مطرح می‌شود و سعی مخلوط کردن آن‌ها و نمادگزاری‌هاشان در یک قالب یکپارچه دارد.

خصوصیات متودولوژی

نتیجه‌ی این فعالیت متودولوژی‌ای است که سیستم را از چندین زاویه‌ی دید ترسیم می‌کند. تمرکز اساسی این متودولوژی بر دو فاز تحلیل و طراحی است. دو ساختار اطلاعاتی و رفتاری سیستم در این متودولوژی استخراج می‌شود که معادل دید ساختاری و رفتاری هستند. در متن این متودولوژی دید عملیاتی نیز مشاهده می‌شود. بعدا به این دیدها می‌پردازیم. برای انعطاف پذیری در این روش تکرار پذیری را قبول می‌کنیم و بازگشت به فازهای قبل برای بهبود و تکامل را می‌پذیریم. در انتهای هر مرحله از تولید نرم‌افزار به ارزیابی آن مرحله می‌پردازد. همان‌طور که در آینده خواهیم دید بسیاری از مستندات و اطلاعات به صورت خودکار قابل تولید هستند بنابراین استفاده از ابزارهای کمکی بسیار مناسب است. این ابزارها همچنین می‌توانند تطابق زوایای دید مختلف مطرح شده را بسنجند.

نمادگزاری‌ها و تکنیک‌های موجود در متودولوژی از جاهای مختلفی آمده‌اند اما این متودولوژی تنها این مجموعه را دور هم جمع‌آوری نکرده است. عملا مجموعه‌ی مولفه‌ها ارزشی افزوده‌تر دارند به گونه‌ای که ما قابلیت ردیابی و پیوستگی را داریم. این دو مزیت به این دلیل به دست می‌آیند که ما از یک مجموعه نمودار اولیه شروع می‌کنیم و در فازهای جلوتر آن‌ها را کامل‌تر می‌نماییم. این کار باعث راحت‌تر و یکنواخت‌تر شدن گذار از یک مرحله به مرحله‌ی دیگر تولید نرم‌افزار می‌شود که عملا در مجموع همان بی‌درز بودن را در بر می‌گیرد. داشتن چندین زاویه‌ی دید، قابلیت فهم سیستم را بالا برده و برای هر نیاز می‌توان به زیر مجموعه‌ای از مستندات که شاید اتوماتیک از روی هم تولید شده باشند مراجعه کرد؛ این زیر مجموعه با توجه به نوع اطلاعات مورد نیاز تعیین می‌شود. مکانیزم‌های اعتبار سنجی معرفی شده در این متودولوژی نیز در طول عمر (تحلیل، طراحی و ...) به صورت تدریجی کامل می‌شوند و می‌توان رگه‌هایی از قابلیت ردیابی، پیوستگی و بی‌درز بودن را در آن‌ها مشاهده کرد.

توضیح متودولوژی

این روش پنج فاز تولید شیئ‌گرا را مطرح می‌کند: تحلیل، طراحیِ سیستم، طراحیِ نرم‌افزار، پیاده‌سازی و آزمون که البته به دو فاز آخر که پیاده‌سازی و آزمون است نمی‌پردازد و ادعا دارد که این فازها را می‌توان مطابق روش‌های موجود پیش برد. عملا مزایای این روش در سه فاز اول و باز تعریف کردن آن‌ها نهفته است. در ادامه به توضیح این سه فاز می‌پردازیم.

- **فاز تحلیل**

در این فاز موجودیت‌های پایه‌ای دامنه شناسایی می‌شود. سپس ارتباطات، خصوصیات و رفتارهای پایه‌ای این موجودیت‌ها استخراج می‌شود. در انتهای این فاز سناریوهای آزمون تحلیل آماده می‌شوند. این سناریوها بسیار کلی هستند و تنها می‌آزمایند که آیا تمام نیازمندی‌ها توسط مدل‌های ارائه شده پوشش داده می‌شوند یا نه. این فاز به دنبال راه‌حل در دامنه می‌گردد. در این فاز با استفاده در سه مرحله‌ی تحلیل نیازمندی‌ها، تحلیل اطلاعات و تحلیل وقایع به سه دید عملیاتی، ساختاری و رفتاری می‌پردازیم. این سه مرحله به هم مرتبط هستند و بر روی هم تاثیر می‌گذارند.

  - **تحلیل نیازمندی‌ها**

هدف در این‌جا بیان شفاف حوزه، اهداف و نیازمندی‌های عملیاتی سیستم می‌باشد. این اطلاعات باید از سمت کاربر بیاید که در ذات خود ابهام و ناسازگاری را دارد؛ بنابراین سعی بر پالایش نیازمندی‌ها داریم و ابهام و کمبود نیازمندی‌ها را رفع می‌نماییم.

  - **تحلیل اطلاعات**

با شناسایی سیستم در این مرحله موجودیت‌ها و نیز ارتباطات بین آن‌ها مشخص می‌شوند. اطلاعات به صورت نمودار موجودیت-ارتباط[1] مدل می‌شود. این مدل یک مدل تخت است و معمولا در آن ساختارهای سلسله مراتبی وجود ندارد. مرحله‌ی بعدی این است که موجودیت‌های این مدل را به اشیا تبدیل کنیم. در این تبدیل برخی از موجودیت‌های داده‌ای به شیئ و برخی به خصوصیت یک شیئ تبدیل می‌شوند. مدل به دست آمده تحت نمودار شیئ-ارتباط[2] نمایش داده می‌شود. اشیا در این مدل می‌تواند تحت مباحث[3] دسته‌بندی شوند. هر مبحث شامل مجموعه‌ای از اشیاست که با هم سرویسی از سیستم را به انجام می‌رسانند.

توضیحِ شیئ[4] به عنوان مستندی متنی برای هر شیئ درون نمودرا شیئ-ارتباط، ساخته می‌شود. این مستند به عنوان مستندی درونی در طول چرخه‌ی تولید نرم‌افزار نگهداری و تکمیل می‌شود. مستند دیگری که از جمع شدن توضیحِ شیئ‌ها به دست می‌آید مستند ارجاع متقابل اشیا[5] است. در این مستند جدولی تمام اشیا، سرویس‌هایی که به دیگر اشیا می‌دهند و دریافت می‌کنند نشان داده می‌شود. این

---

1 Entity-Relation Diagram (ERD)
2 Object-Relation Diagram (ORD)
3 Subjects
4 Object Description (OD)
5 Object Cross-Reference (OCR)

مستند می‌تواند به صورت خودکار از مستندات توضیحِ شیئ استخراج شود. ارجاع متقابل اشیا در نگهداری سیستم بسیار کلیدی است. در نهایت اگر رابطه‌ی ارث بری که رابطه‌ی عام-خاص است در بین اشیا دیده شود در نمودار ارث‌بری[1] منعکس می‌شود.

○ تحلیل وقایع

وقایع، رفتارهایی هستند که موجودیت‌ها در سیستم از خود نشان می‌دهند. در تحلیل وقایع سیستم را به صورتی ماشینی که به مجموعه‌ای از محرک‌های بیرونی پاسخ می‌دهد می‌بینیم. هدف به دست آوردن رفتار سیستم از دیدگاه خارجی است. تحلیل وقایع و اطلاعات به دلیل اشتراکاتی که با هم دارند می‌توانند در تکمیل یکدیگر و آزمودن یکدیگر استفاده شوند. از بررسی و تحلیل عملیات می‌توان به اطلاعات مورد نیاز برای انجام آن رسید. عکس این مسیر نیز می‌تواند صادق باشد.

فعالیت‌هایی که باید در جواب محرک‌های خارجی انجام شود به دو قسمت فعالیت‌های پایه‌ای و جنبی تقسیم می‌شوند. مجموعه‌ی محرک‌ها و پاسخ‌ها در مستند رفتار سیستم[2] لیست می‌شوند. از آن‌جا که رفتار کلی سیستم از دید ناظر خارجی مهم است از نمودار رفتار سیستم[3] استفاده می‌شود. این نمودار تغییر حالت‌ها و حالت‌های سیستم که از دید ناظر خارجی مشهود است را نشان می‌دهد. محرک‌هایی که در این نمودار وجود دارند در مستند رفتار سیستم هم آمده‌اند. بعد از مشخص شدن محرک‌ها به مشخص کردن اطلاعاتی که در قالب آن‌ها رد و بدل می‌شوند می‌پردازیم. بسیاری از این اطلاعات در مرحله‌ی تحلیل اطلاعات مشخص شده‌اند و بر انتخاب قبلی صحه می‌گذارند. اما برخی اطلاعات جدید که شناخته می‌شوند باید به نمودار موجودیت-ارتباط اضافه شوند و سپس تاثیر خود را در نمودار شیئ-ارتباط بگذارند. در نهایت نیز مستندات توضیحِ شیئ با اضافه شدن محرک‌ها و پاسخ‌ها تکمیل می‌شوند.

○ گذار به طراحی سیستم

محصول نهایی فاز تحلیل نمودار مشتری-کارگزار[4] است. این نمودار که در آن جهت مشتری به کارگزار، چندی ارتباطات، پیام‌های ردو بدل شده و گروه‌بندی مباحث در آن مشخص شده است، گامی است برای گذار از تحلیل به طراحی سیستم.

• فاز طراحی سیستم

در این فاز جزئیات رفتار موجودیت‌های پایه‌ای شناسایی شده و چگونگی ارتباط آن‌ها به صورت کامل و شفاف مشخص می‌شود. عملا موجودیت‌های فاز تحلیل پالایش شده و موجودیت‌های این فاز نتیجه می‌شوند. سپس برخی موجودیت‌ها که پایه‌ای نیستند ولی برای انجام کار لازم هستند به سیستم اضافه می‌شوند. در انتهای این فاز سناریوهای آزمون تحلیل پالوده شده و سناریوهای آرمون طراحی سیستم را نتیجه می‌دهند. این سناریوها بررسی می‌کنند که آیا تمام مکانیزم‌های لازم برای تحقق نیازمندی‌های سیستم حاضر هستند یا نه. این فاز به چگونگی کارکرد سیستم می‌پردازد.

---

[1] Inheritance Diagram (ID)
2 System-Behavior Script (SBS)
[3] System-Behavior Diagram (SBD)
[4] Client-Server Diagram (CSD)

برای نمایش ارتباطات بین اشیا از دید یک شیئ نمودار واسط شیئ[1] استفاده می‌شود. این نمودار معمولا از روی نمودار مشتری-کارگزار استخراج می‌شود که ارتباطات اشیا در سطح سیستم را نمایش می‌دهد. برای رفتار نیز می‌توان از خاص کردن مستندات در سطح سیستم به مستندات در سطح یک شیئ رسید. به عبارتی از نمودار رفتار سیستم به نمودار رفتار شیئ[2] و از مستند رفتار سیستم به مستند رفتار شیئ[3] می‌رسیم. البته چون تمام اشیا تغییر حالات پیچیده‌ای ندارند ممکن است نمودار رفتار شیئ مورد نیاز نباشد. مستندات توضیحِ شیئ و مجموع آن‌ها در ارجاع متقابل اشیا نیز با اضافه شدن اطلاعات سرویس‌ها کامل‌تر می‌شوند.

- **فاز طراحی نرم‌افزار**

در این مرحله به اضافه کردن اطلاعات وابسته به پیاده‌سازی به موجودیت‌های تعریف شده در فازهای قبل می‌پردازیم. مانند فازهای قبل، سناریوهای آزمون آن فازها را تکمیل‌تر کرده و مجموعه‌ای از داده‌های آزمون سیستم را به آن اضافه می‌کنیم. این فاز به چگونگی استفاده از یک زبان برنامه‌نویسی خاص، سخت افزار خاص و یا نرم‌افزار خاص برای پیاده سازی می‌پردازد.

مستندات تولید شده در فازهای قبل ممکن است در اثر بررسی‌های این فاز تغییر کنند و کامل‌تر شوند. به طور مثال توضیحِ شیئ اطلاعاتی در مورد سرویس‌های فراخوانی شده از سیستم عامل به ما بدهد. نمودار شیئ-پردازش[4] در این‌جا مطرح می‌شود تا ارتباط بین عملیات‌ها و خصوصیات یک شیئ را نشان دهد. در این فاز سطح دسترسی (عمومی، سطح شیئ و ...) عملیات نیز مشخص می‌شود. در این فاز برای توصیف الگوریتم‌های پیچیده ممکن است از شبه کد[5] استفاده شود. بعد از این فاز پیاده‌سازی سرراست خواهد بود.

۲. تحلیل متودولوژی

نقاط قوت

تعاریف شفاف و منطقی است.
فرآیندی بدون درز است.
عملی و قابل استفاده است.
زبان مدل کردن آن سه دیدگاه ساختاری، رفتاری و عمل‌کردی را دارد.
در مرحله‌ی طراحی نرم‌افزار به گونه‌ای تبدیل از منطقی به فیزیکی را انجام می‌دهد.
مدل‌ها هم شامل نمودارهای غیر رسمی و هم شامل متون با ساختار رسمی و مشخص است.
قابلیت ردیابی یک فعالیت وجود دارد. اما این ردیابی به نیازمندی منتهی نمی‌شود.

---

[1] Object-Interface Diagram (OID)
[2] Object-Behavior Diagram (OBD)
[3] Object-Behavior Script (OBS)
[4] Object-Processing Diagram
[5] Pseudo-Code

نقاط ضعف

برخلاف ادعای خود کل چرخه‌ی نرم‌افزار را نمی‌پوشاند.

- درست است که فاز تحلیل بزرگی دارد ولی عملا بیش‌تر در آن طراحی می‌کند و برخی فعالیت‌های مربوط به تحلیل انجام نمی‌گیرد. به پیاده سازی و نگهداری هم نمی‌پردازد.

فرآیندهای چتری را پشتیبانی نمی‌کند.

بر پایه‌ی نیازمندی‌ها نیست. اصولا نیازمندی‌ها به صورت مستقیم دیده نمی‌شوند.

- تاثیر نیازمندی‌ها در تحلیل و سناریوهای آزمون دیده می‌شود.

مستندات خیلی آزمون پذیر نیستند.

کاربر را در فرآیندها در نظر نمی‌گیرد.

مکانیزمی کارا برای حل کردن پیچیدگی‌ها چه در سطح مدل سازی و چه در سطح فرآیند ندارد.

- هر چند در برخی نمودارها سطح اضافه کرده است ولی مکانیزم کارایی نیست.

قابل گسترش و تنظیم نیست. انعطاف پذیر و مقیاس پذیر هم نیست.

مکانیزمی برای منطبق کردن آن با حوزه‌ی مساله نداریم.

سیستم را در چند سطح (سازمان، سیستم، زیر سیستم و ...) نمی‌بیند.

۳. بی‌درز بودن و ردیابی تا نیازمندی‌ها

این فرآیند بی‌درز است و تا مراحل اولیه می‌توان ردیابی را انجام داد. اما این مراحل اولیه لزوما نیازمندی‌ها نیستند.

در این فرآیند مستندات و مدل‌ها به تدریج کامل می‌شوند. مدلی که در مرحله‌ی قبل ایجاد شده و اطلاعات اولیه‌ای در آن وجود دارد در مراحل بعدی غنی‌تر شده و اگر هم مدلی در مرحله‌ای اضافه می‌شود قابل تولید از روی مدل‌های دیگر و با اندکی تغییر است. این یکنواختی در تکامل سناریوهای آزمون در فازهای مختلف تولید سیستم نیز مشهود است.

از آن‌جا که در طول عمر تولید سیستم ما تبدیل پیچیده‌ای از مدل‌های یک مرحله به مرحله‌ی بعد نداریم می‌توان رد یک عملیات را در مراحل مختلف پیدا کرد. اما همان‌طوری که مشخص است این ردیابی به نیازمندی‌ها ختم نمی‌شود چون اصولا در ابتدای این فرآیند حرفی از نیازمندی زده نمی‌شود. احتمالا این فرآیند نیز مانند فیوژن فرض بر داشتن یک مجموعه نیازمندی اولیه داشته است.

۴. مقایسه با فیوژن

فیوژن به صورت Adhoc ساخته شده است در حالی که هاجماک خود یک روش کامل را مطرح کرده است. چرخه‌ی عمر هر دو ناقص است و از تحلیل و نگهداری و بعضا پیاده‌سازی کم دارند. فاصله‌ی بین فازها بین هردو مشخص است، البته هاجماک تا حد کمی تکرار شونده است. در هاجماک سیستم مدل می‌شود در حالی که در فیوژن کل دامنه مدل می‌شود. شناسایی عملیات به نوعی در هردو مشابه است ولی در هاجماک در حد فیوژن طراحی رفتاری نداریم؛ به طور مثال ترتیب مجاز عملیات در آن

استخراج نمی‌شود. در هاجماک ارث بری در تحلیل مطرح می‌شود ولی فیوژن سعی می‌کند این کار را تا مراحل بعدی به تعویق بیاندازد؛ البته باید در نظر داشت فاز تحلیل در هاجماک آن‌قدر بزرگ است که عملا تشخیص ارث‌بری نزدیک به روش فیوژن شبیه می‌شود. هردو در مراحل نهایی بحث کارگزار/مشتری را مطرح می‌کنند.

هردو تمام جنبه‌های ساختاری، رفتاری و عملکردی را می‌پوشانند. هردو گذار تدریجی دارند و بنابراین بی‌درز و قابل ردیابی هستند. هردو توانایی مدل‌سازی نسبتا قوی‌ای دارند. هردو فرض می‌کنند که یک مجموعه نیازمندی اولیه وجود دارد و تمام مراحل ساخت نرم‌افزار را در نظر نمی‌گیرند. نمودارها و مستندات در هردو بسیار زیاد است.

۵. مدل مشابه با مدل‌های متودولوژی‌های پیشین

نمودار شیئ-ارتباط[1] این متودولوژی شباهت نسبتا زیادی با مدل شیئ[2] در متودولوژی فیوژن و نمودار شیئ و کلاس[3] در متودولوژی کود-یوردان دارد.

[1] Object-Relation Diagram (ORD)
[2] Object Model
[3] Class-&-Object Diagram