نرم‌افزارِ کد باز[۱] و نرم‌افزارِ آزاد[۲]

بهداد اسفهبد (۷۹۱۷۴۹۰۴)

جهتِ ارائه برای درسِ مهندسیِ نرم‌افزار
استاد: دکتر میریان

۱۷ تیر ۱۳۸۲

[۱] Open Source Software
[۲] Free Software

چکیده

نرم‌افزارِ کد باز و نرم‌افزارِ آزاد دو مفهومِ نسبتا جدید در بحثِ مهندسیِ نرم‌افزار می‌باشند، که شاید در نگاهِ اول چندان مرتبط با این مبحث نباشند. پس از نگاهی به مختصر به ایده‌های نگاه‌دارنده‌ی آن‌ها و تحولی که در کلِ روندِ تولیدِ نرم‌افزار به وجود می آورند، از مدلِ تجاریِ متفاوتی که می‌طلبند، تا برنامه‌نویسانِ متفاوت‌شان، همه نشان از آن دارد که برای توسعه‌ی نرم‌افزار در این مدل‌ها نیاز به علمِ مهندسیِ نرم‌افزارِ آشنا با این پدیده‌ها می‌باشد. در این نوشته پس از نگاهی اجمالی به تعاریف و تفاوت‌های این دو مفهوم، سعی می‌کنیم چگونگیِ اجرای هر یک از مراحلِ روندِ مهندسیِ نرم‌افزار را در دنیایِ کد باز معرفی کنیم. به این ترتیب سعی خواهیم کرد به‌جای آن‌که در تئوری با این پدیده برخورد کنیم، با بیانِ مثال‌هایی از پروژه‌های واقعی، به پیاده‌سازیِ این روش‌ها نزدیک گردیم.

۱ مقدمه

نرم‌افزارِ کد باز و نرم‌افزارِ آزاد دیگر دو مفهومِ ناآشنا نیستند. هرکس که با سیستمِ عاملی غیر از مایکروسافت ویندوز[۱] سروکار داشته باشد بدونِ شک از وجود و اهمیتِ این نرم‌افزارها اطلاع دارد. لذا در این نوشته به تعاریفِ اولیه و نحوه‌ی شکل‌گیریِ آن‌ها اشاره نخواهیم کرد، بلکه مروری بر علتِ وجودِ آن‌ها و فوایدشان خواهیم داشت. پس از آن به زمانِ حال برگشته و روندِ تولیدِ نرم‌افزارِ آزاد در جامعه‌ی امروز را بررسی خواهیم کرد. پس از آن نگاهِ مختصری به روندِ مهندسیِ نرم‌افزار در نرم‌افزارهای کد باز خواهیم کرد. سپس نگاهی اجمالی به چند پروژه‌ی موفق و دسته‌بندیِ آن‌ها می‌کنیم. با این کار سعی خواهیم کرد نشان دهیم که از نقطه‌نظرِ تجارت، این مدلِ نرم‌افزار نه‌تنها ضعفی بر انواعِ دیگر ندارد، بلکه اگر درست به کار گرفته شود، بسیار از آن‌ها پیشی خواهد گرفت.

۲ معاهده‌ی کد باز[۲]

در کشورهای مدرن که قانونِ حقِ کپی در آن‌ها رعایت می‌شود، هر کاربرِ عادیِ کامپیوتر نرم‌افزارهای فراوانی دارد که سال‌ها قبل خریده است و دیگر استفاده نمی‌کند. ممکن است کاربر کامپیوترش را ارتقا داده باشد یا از مدلِ دیگری استفاده کند، و برنامه‌هایش دیگر کار نخواهند کرد. نرم‌افزارها ممکن است قدیمی شده باشند. ممکن است برنامه کاری را که کاربر می‌خواهد انجام ندهد. ممکن است دو یا چند کامپیوترِ دیگر خریده باشد و نخواهد بابتِ نسخه‌ی دیگری از همان نرم‌افزار پول بپردازد. به هر دلیل که باشد، نرم‌افزاری که سال‌ها پیش بابتَش پول داده شده است، امروز کارش را انجام نمی‌دهد. آیا لازم است این اتفاق رخ دهد؟

چه می‌شد اگر این حق را داشتید تا هرگاه لازم است نرم‌افزارِ خود را مجانی ارتقا دهید؟ چه می‌شد اگر وقتی از PC به Mac تغییرِ سیستم دادید نسخه‌ی نرم‌افزار را نیز مجانی تغییر می‌دادید؟ چه می‌شد اگر وقتی نرم‌افزار کار نمی‌کرد یا به اندازه‌ی لازم قوی نبود، می‌توانستید بدهید درستش کنند، یا حتی خودتان درستش کنید؟ چه می‌شد اگر نرم‌افزار هم‌چنان پس از آن‌که شرکتِ سازنده‌اش از بازارِ تجارت خارج شد، باز هم پشتیبانی می‌شد؟ چه می‌شد اگر می‌توانستید نرم‌افزارتان را به جای یک کامپوتر، روی کامپیوترِ محلِ کار، منزل، و کامپیوترِ قابلِ‌حمل‌تان نصب کنید؟ در این صورت احتمالا هنوز از نرم‌افزاری که سال‌ها پیش خریده بودید استفاده می‌کردید. پس حقوقی وجود دارند، که کد باز در اختیارتان قرار می‌دهد.

---

[۱] Microsoft Windows

[۲] The Open Source Definition

معاهده‌ی کد باز مجموعه‌ی حقوقی برای کاربرِ کامپیوتر است، که حقوقِ مشخصی را تعریف می‌کند که مجوزِ[۳] نرم‌افزار باید در اختیارتان قرار دهد تا به عنوانِ کد باز شناخته شود. آنان که برنامه‌های خود را کد باز نمی‌سازند رقابت با آنان که می‌سازند را مشکل می‌یابند و کاربران به حقوقِ جدیدی دست می‌یابند که همواره باید می‌داشته‌اند. برنامه‌هایی مانندِ سیستمِ عاملِ لینِکس[۴] و مرورگرِ نت‌سکِیپ[۵] بسیار رایج گشته‌اند و جایگزینِ نرم‌افزارهای دیگری شده‌اند که مجوزِ محدودتری دارند. شرکت‌هایی که از نرم‌افزارهای کد باز استفاده می‌کنند از سرعتِ بالای گسترشِ آن‌ها سود می‌برند، که معمولا توسطِ همکاریِ چند شرکت، و بسیاری از آن توسطِ افرادی که برای رفعِ نیازِ شخصی‌شان این کار را می‌کنند انجام می‌شود.

هم‌چنان که داوطلبانی که محصولاتی چون لینکس را آفریدند به کارِ خود ادامه می‌دهند، شرکت‌ها نیز توسعه‌ی نرم‌افزار به میلِ خودشان می‌پردازند، و این تنها با وجودِ حقوقی که کد باز فراهم می‌آورد محقق گشته است. قشرِ متوسطِ برنامه‌نویسانِ کامپیوتر از این‌که محصولِ ساعت‌ها تلاش‌شان توسطِ صاحبِ نرم‌افزار به فروش برسد بی‌آن‌که در ازایش چیزی عایدشان شده باشد، احساسِ حماقت و حقارت می‌کنند. همین برنامه‌نویسان در کمک به نرم‌افزارهای کد باز احساسِ راحتی می‌کنند زیرا در ازایش به این حقوق دست یافته‌اند:

- حقِ کپی و انتشارِ آن کپی‌ها.
- حقِ دست‌رسی به کدِ منبعِ نرم‌افزار، که پیش‌نیازِ تغییر در آن می‌باشد.
- حقِ گسترش و اصلاحِ برنامه.

این حقوق برای کسی که به گسترشِ نرم‌افزار کمک می‌کند مهم می‌باشند، زیرا همه‌ی گسترش‌دهندگان را نسبت به هم در یک سطح نگاه می‌دارند. هرکس که بخواهد می‌تواند یک نرم‌افزارِ کد باز را بفروشد، در نتیجه قیمت‌ها پایین خواهند بود و گسترش برای دست‌یابی به بازارهای جدید سریع‌تر انجام خواهد شد. هرکس که برای کسبِ اطلاع درباره‌ی یک نرم‌افزارِ کد باز وقت صرف کند می‌تواند آن را پشتیبانی کند، و به کاربران این انتخاب را بدهد که از پشتیبانیِ او بهره‌مند گردند. هر برنامه‌نویسی می‌تواند یک برنامه‌ی کد باز را برای دست‌یابی به بازاری جدید و مشتری‌های جدید تغییر دهد. کسانی که این کارها را انجام می‌دهند مجبور به پرداختِ هزینه‌ای برای اجازه‌ی انجامِ کارشان نیستند.

علتِ موفقیتِ این استراتژیِ کمونیست‌مانند، در حالی که شکستِ کمونیست در سراسرِ جهان مشهود است، این است که اقتصادِ اطلاعات اصولا با اقتصادِ محصولاتِ دیگر متفاوت است.

---

[۳] License
[۴] Linux
[۵] Netscape

هزینه‌ی کپی‌کردنِ تکه‌ای اطلاعات، مانندِ برنامه‌ی کامپیوتر بسیار ناچیز است. هزینه‌ی برق و استهلاکِ تجهیزاتِ آن کاملا قابلِ صرف‌نظر است. در مقابل، نمی‌توان یک قرصِ نان را بدونِ داشتنِ همان‌قدر آرد کپی کرد.

۳ تاریخچه

مفهومِ نرم‌افزارِ آزاد بسیار قدیمی است. ابتدا که کامپیوتر به دانش‌گاه‌ها راه یافت ابزاری تحقیقاتی بود. نرم‌افزار آزادانه در دست‌رس بود و برنامه‌نویس‌ها بابتِ عملِ برنامه‌نویسی حقوق می‌گرفتند، نه برای خودِ برنامه‌ها. تنها بعدها که کامپیوتر به دنیای تجارت راه یافت بود که برنامه‌نویس‌ها شروع به حمایت از خود توسطِ محدودکردنِ حقوقِ کاربران به نرم‌افزارشان، و دریافتِ هزینه‌ای برای هر کپیِ آن کردند. نرم‌افزارِ آزاد به عنوانِ دیدگاهی سیاسی توسط ریچارد استالمن[۶] و از سالِ ۱۹۸۴ آغاز شد. استالمن در این سال بنیادِ نرم‌افزارِ آزاد[۷] را تشکیل داد و پروژه‌ی گنو[۸] را آغاز کرد. استالمن فرض را بر این نهاد که مردم باید آزادیِ بیشتری داشته باشند، و باید از آزادی‌شان بهره ببرند. او سپس مجموعه‌ای از حقوقی را طراحی کرد که معتقد بود همه‌ی کاربران باید داشته باشند، و آن‌ها را در GNU General Public License یا همان GPL مرتب کرد. استالمن خود کارهای باارزشی به عنوانِ نرم‌افزارِ آزاد انجام داد که کامپایلرِ C گنو[۹]، و ویرایش‌گرِ بسیار توان‌مندِ گنو ایمکس[۱۰] از آن دسته‌اند. این آثار در افرادِ بسیارِ دیگری تاثیر نهاده و باعث شد ایشان نیز تحتِ GPL نرم‌افزارِ آزاد تولید کنند. اگرچه هیچ‌جا مستقیما اشاره نشده است، ولی بدونِ شک معاهده‌ی کد باز شاملِ بسیاری از ایده‌های استالمن است و در واقع می‌توان آن را مشتقی از کارهای او دانست.

معاهده‌ی کد باز ابتدا به عنوانِ مرجعِ سیاستِ توزیعی از لینکس به نامِ دبیَن[۱۱] به‌وجود آمد. دبیَن یکی از اولین سیستم‌های لینکس است که امروزه نیز بسیار رایج است. این سیستم کاملا از نرم‌افزارهای آزاد ساخته شده بود، با این وجود، با توجه به تنوعِ مجوزهای مختلفِ نرم‌افزار که در آن زمان یافت می‌شد، دبیَن در تعریفِ این‌که چه نرم‌افزاری آزاد است با مشکل مواجه بود. برای رفعِ این ابهام‌ها، اعضای دبیَن، از جمله بُروس پِرِنس[۱۲] قواعدی را وضع کردند که به راحتی می‌شد با مقایسه‌ی مجوزِ نرم‌افزار با این قواعد تصمیم گرفت که نرم‌افزار آزاد است یا نه. این قواعد بعدها

[۶] Richard Stallman
[۷] Free Software Foundation
[۸] GNU
[۹] GNU C Compiler (GCC)
[۱۰] GNU Emacs
[۱۱] Debian GNU/Linux Distribution
[۱۲] Bruce Perens

در سالِ ۱۹۹۸ با کمکِ فراوانِ اِریک رِیموند[۱۳]، با حذفِ اشاراتی که به دبیَن شده بود، به عنوانِ معاهده‌ی کد باز انتشار یافت.

آن‌چه در این میان گنگ می‌نماید رابطه‌ی دو عبارتِ کد باز (Open Source) و نرم‌افزارِ آزاد (Free Software) است. اگرچه بسیاری معتقد بودند که کلمه‌ی Open در زبانِ انگلیسی به معانیِ بسیار گسترده‌ای استفاده شده است و نباید به این منظور استفاده گردد، اما عده‌ی دیگر آن را با وجودِ تمامِ این معانی، به‌تر از کلمه‌ی Free می‌دانستند که دو معنیِ بسیار متفاوت و گمراه‌کننده در این زمینه دارد. Free در عبارتِ Free Software به معنیِ آزاد استفاده شده است، حال آن‌که اولین معنی که با دیدنِ این عبارت به ذهن متبادر می‌شود معنیِ دیگرِ آن یعنی مجانی می‌باشد، در صورتی که نرم‌افزارِ آزاد هیچ تناقضی با فروشِ نرم‌افزار ندارد. اگرچه بنیادِ نرم‌افزارِ آزاد و جنبشِ کد باز دو موجودیتِ مختلف می‌باشند، ولی به هیچ وجه با هم موازی یا رقیب نبوده و جمعیتِ اطرافِ آن‌ها بسیار با هم اشتراک دارد. این دو هر کدام تعاریف و قوانینِ دقیقِ خود را دارند. آن‌چه در ادامه‌ی این مقاله درباره‌ی آن صحبت خواهیم کرد اختصاصا وابسته به هیچ‌یک از این دو نمی‌باشد، و منظور تمامِ فعالیت‌های نرم‌افزاری است که در قوانینِ خود حقوقِ مشابهی را برای کاربر در نظر می‌گیرند.

۴ روندِ تولیدِ نرم‌افزارِ آزاد

نرم‌افزارهای آزاد را می‌توان به دو دسته‌ی کلی تقسیم کرد:

- **سرمایه‌گذاری‌شده**[۱۴]: نرم‌افزارهایی که نسخه‌ی مرکزیِ آن‌ها توسطِ یک شرکت و در راستای اهدافِ آن شرکت تولید می‌شوند. لزومی به ذکر نیست که این شرکت‌ها از کمکِ افرادِ علاقه‌مند در بیرونِ شرکت بهره‌ی فراوان می‌برند. از این دسته می‌توان Apache Web Server و PHP را مثال زد.
- **بدونِ سرمایه‌گذاری**[۱۵]: نرم‌افزارهایی که نسخه‌ی مرکزیِ آن‌ها توسطِ هیچ شرکتی حمایتِ مستقیم نمی‌شود، و به جهتِ علاقه‌ی شخصیِ افرادی که روی آن کار می‌کنند توسعه می‌یابند. مانندِ PostgreSQL.

با کمی دقت در دسته‌بندیِ فوق به نکته‌ی قابلِ توجهی می‌رسیم: نرم‌افزارهای دسته‌ی دوم، بدونِ آن‌که شخص یا شرکتِ مشخصی آن‌ها را سفارش داده باشد، توسعه می‌یابند و اگر به صورتِ یک کل به آن‌ها نگاه شود، روزبه‌روز ما را به هدفِ دست‌یابی به سیستم‌های کامپیوتریِ کاملا آزاد

[۱۳] Eric Raymond
[۱۴] Funded
[۱۵] Unfunded

نزدیک‌تر می‌کنند. بسیاری از طرفدارانِ کد باز و نرم‌افزارِ آزاد انگیزه‌ی افراد برای کار بر روی چنین پروژه‌هایی را علاقه و احساسِ مسئولیت در رسیدن به چنینِ هدفِ نهایی می‌دانند، حال آن‌که شرکت‌های بسیاری نشان داده‌اند که نیازی به چنین دیدگاهی برای توجیه فلسفه‌ی پشتِ این نرم‌افزارها نیست. همان‌طور که جلوتر بیان خواهیم کرد، بعضی معتقدند که امکانِ در دست‌رس بودن، کدِ منبع، برگِ برنده‌ای در رقابت در دنیای تجارت است.

اریک ریموند در کتابِ انقلابیِ خود[۲] این تقسیم‌بندی را از دیدگاهِ دیگری انجام می‌دهد. او محیطِ تولیدِ نرم‌افزارِ آزاد را به دو دسته‌ی زیر تقسیم می‌کند:

- **کلیسای جامع**[۱۶]: در این بستر، نرم‌افزار مرکزی توسطِ یک تیمِ مرکزی و بسته تولید می‌شود، و با کدِ باز به جامعه‌ی کابران ارائه می‌شود. معمولا جزئیاتِ روندِ توسعه‌ی نرم‌افزار چندان باز نیست، و به‌خصوص، خط مشیِ اصلیِ نرم‌افزار توسطِ تیمِ مرکزی تعیین می‌شود. از دیگر نشانه‌های این مدل این است که وصله‌هایی[۱۷] که توسطِ برنامه‌نویسانِ دیگر برای نرم‌افزارِ اصلی نوشته شود، لزوما در زمانِ مناسبی به توزیعِ مرکزی راه نخواهد یافت، و در صورتِ راه‌یابی، تنها پس از وارسیِ دقیقِ تیمِ مرکزی میسر خواهد بود. به طورِ خلاصه، تیمِ مرکزی مانندِ اعضای یک کلیسای جامع عمل می‌کند که وظیفه‌اش صدورِ حکم است (در این مورد نرم‌افزار).

- **بازار**[۱۸]: در این بستر، بر خلافِ بسترِ قبل، نرم‌افزار در محیطی کاملا باز گسترش می‌یابد. هر کس که تواناییِ فنیِ بیشتری داشته باشد، نفوذِ بیش‌تری خواهد داشت، و هر برنامه‌نویس می‌تواند در صورتِ تمایل و تواناییِ فنی، به اندازه‌ی هر برنامه‌نویسِ دیگر در توسعه و جهت‌گیریِ این نرم‌افزارها سهیم باشد. معمولا این روش علاوه بر کارِ گروهی نسبتا مرکزی، که با گذشتِ زمان ممکن است کاملا تغییر کنند، مبتنی بر کمک‌های گسسته و کوچکِ برنامه‌نویسانِ مختلف از سرتاسرِ دنیا است که به سببِ علاقه، یا نیاز، امکانِ جدیدی را به نرم‌افزار اضافه کرده‌اند، و آن را برای نصب در نسخه‌ی مرکزی، در قالبِ یک وصله ارائه می‌کنند. گسترشِ نرم‌افزار در این مدل متکی بر استفاده‌ی سنگین از فهرستِ پستی[۱۹] و ابزارِ CVS[۲۰] می‌باشد. به طورِ مشابه، می‌توان این محیط را به بازاری تشبیه کرد که در آن هر کس محصولِ با کیفیتِ به‌تری ارائه دهد، می‌تواند در رفعِ نیازهای مشتری (در این مورد نرم‌افزار) سهیم باشد. از این دسته می‌توان Mozilla را مثال زد.

---

[۱۶] Cathedral
[۱۷] Patch
[۱۸] Bazaar
[۱۹] Mailing List
[۲۰] Concurrent Versions System

با توجه به این تعاریف می‌توان فرقِ اصلی بینِ دیدگاهِ بنیادِ نرم‌افزارِ آزاد و جنبشِ کد باز را این‌گونه بیان کرد: بنیادِ نرم‌افزارِ آزاد به دنبالِ حفظِ حقِ کپی، انتشار، و تغییرِ کدِ منبع، برای کاربر می‌باشد، ولی تاکیدِ جنبشِ کد باز بر روی باز بودنِ روندِ تهیه‌ی نرم‌افزار، یعنی همان مدلِ بازار می‌باشد. نیازی به گفتن نیست که حقوقِ کپی، انتشار، و تغییرِ کدِ منبع، پیش‌نیازِ مدلِ بازار نیز هست.

۵ مهندسیِ نرم‌افزار

مهندسیِ نرم‌افزار حوزه‌ای وسیع‌تر از «نوشتنِ برنامه» است. با این حال، در بسیاری از پروژه‌های کد باز، برنامه تنها نوشته می‌شود و بیرون می‌رود. از مثال‌های تاریخی این‌گونه برمی‌آید که لازم نیست نرم‌افزار حتما مهندسی شده باشد تا موردِ استفاده‌ی وسیع قرار گیرد. در زیر به اختصار اجزای اصلیِ مهندسیِ نرم‌افزار را بیان کرده و معادلِ رایجِ آن را در اجتماع‌های کد باز معرفی می‌کنیم. در پایان به نتایجِ این تفاوت‌ها اشاره خواهیم کرد.

اجزای مهندسیِ نرم‌افزار را می‌توان به اختصار این‌گونه برشمرد:

- برآوردِ نیازها[۲۱]
- طراحیِ سطحِ سیستم‌ها[۲۲]
- طراحیِ جزئی[۲۳]
- پیاده‌سازی[۲۴]
- تلفیق و بسته‌بندی[۲۵]
- تستِ حوزه‌ای[۲۶]
- پشتیبانی[۲۷]

در موردِ این اجزا، همان‌طور که در مهندسیِ نرم‌افزار رایج است، هر بخش وابسته به بخش‌های پیش از خود است، و نمی‌تواند پیش از آن‌ها پایان یابد، ولیکن توصیف و گاه پیاده‌سازیِ یک بخش

[۲۱] Marketing Requirements
[۲۲] System-Level Design
[۲۳] Detailed Design
[۲۴] Implementation
[۲۵] Integration
[۲۶] Field Testing
[۲۷] Support

ممکن است پیش از اتمامِ بخش‌های پیش‌نیازش انجام شود. از خواصِ دیگرِ روندِ مهندسیِ نرم‌افزار نیاز به بازبینی[۲۸] در تمامِ مراحل است.

۱.۵ برآوردِ نیازها

اولین قدم از روندِ مهندسیِ نرم‌افزار گردآوری و برآوردِ نیازهای کاربر، امکاناتِ موردِ نیازِ برنامه، و یافتنِ مخاطب است. در این مرحله معمولا حجمِ زیادی نوشته می‌شود، که سندِ برآوردِ نیازها (MRD)[۲۹] از آن دسته است.

تهیه‌ی MRD در پروژه‌های کد باز معمولا از طریقِ فهرستِ پستی یا گروهِ خبری[۳۰] انجام می‌شود، که در آن کاربران و توسعه‌دهندگانِ نرم‌افزار به صورتِ رفت و برگشت مدام پیام می‌فرستند، و سعی می‌کنند به اجماع برسند. از معایبِ این مدل آن است که بسیاری از اوقات به اجماع نرسیدن به معنیِ «تقسیمِ کد[۳۱]» خواهد بود. در بسیاری از موارد حتی سعی نمی‌شود جلوی این کار گرفته شود، یا به کل غیرِ ممکن است.

۲.۵ طراحیِ سطحِ سیستم

در این مرحله توصیفِ سیستم بر اساسِ «پیمانه‌ها» و روابطِ آن‌ها با هم به دست می‌آید. هدف از این مرحله ابتدا حصولِ اطمینان از درستی و قابلِ اجرا بودنِ طرح، و سپس تخمینِ کارِ لازم برای پیاده‌سازیِ آن است.

معمولا در پروژه‌های کد بازبدونِ سرمایه‌گذاری این مرحله وجود ندارد، و سیستم به صورتِ ضمنی طراحی می‌گردد و یا طرح با گذشتِ زمان تغییر می‌کند (مانندِ خودِ نرم‌افزار). در مواقع عدمِ وجودِ MRD یا QA را شاید بتوان با حضورِ برنامه‌نویسانِ خبره تا حدی مخفی کرد، ولی عدمِ وجودِ طراحیِ سیستم، عملا کیفیتِ پروژه محدود خواهد شد.

۳.۵ طراحیِ جزئی

در این مرحله جزئیاتِ هر پیمانه، از جلمه واسطِ آن، و رابطه‌های وابستگیِ پیمانه‌ها به یکدیگر به دست می‌آید. به عنوانِ خروجیِ این مرحله معمولا نمودارهای PERT و GANT نیز ارائه خواهد

---

[۲۸] Review
[۲۹] Marketing Requirements Document
[۳۰] Newsgroup
[۳۱] Code Splits

شد، که زمان و ترتیبِ انجامِ کارها را نشان می‌دهند. خروجیِ این فاز را سندِ طراحیِ جزئی (DDD)[۳۲] می‌نامند.

از دیگر نتایجِ نداشتنِ سرمایه‌گذار و نیاز به تفریح، طراحیِ جزئی است. بسیاری از مردم این مرحله را دوست دارند، اما معمولا در عمل طراحی از پیاده‌سازی تاثیر می‌گیرد: «می‌دانم که به یک پارسر نیاز دارم، پس یکی می‌نویسم.» مستندسازیِ واسط‌ها نیز اجباری نیست، که البته باید از آن خجالت کشید. حتی اگر پیمانه‌ای فقط برای استفاده‌ی داخلی استفاده شده باشد و هیچ ارتباطی با دنیای خارج نداشته باشد، باز هم باید واسط‌های آن مستندسازی شوند، چرا که در هر صورت کدِ منبع در اختیارِ مردم است و هر چه مستندات بیشتر باشد، احتمالِ استفاده‌ی مجدد توسطِ دیگران بیش‌تر می‌شود.

۴.۵ پیاده‌سازی

پیمانه‌های مطرح شده در DDD در این بخش پیاده‌سازی می‌شوند. این همان عملِ کوچکِ برنامه‌نویسیِ هر پیمانه می‌باشد، که قلب و روحِ مهندسیِ نرم‌افزار است. متاسفانه گاه دیده می‌شود که این بخش تنها بخشی است که تدریس می‌شود (یاد گرفته می‌شود)، در حالی که در عینِ حال تنها بخشی از مهندسیِ نرم‌افزار است که شخص می‌تواند به راحتی به خود بیاموزد.

این قسمتِ لذت‌بخشِ کار است. پیاده‌سازی قسمتی است که برنامه‌نویسان بیش از همه دوست دارند. این است آن‌چه آن‌ها را بیدار نگاه می‌دارد، هنگامی که می‌توانند خواب باشند. امکانِ نوشتنِ کد یکی از انگیزه‌های اولیه برای تمامِ تلاش‌های گسترشِ نرم‌افزارِ کد باز بوده است. معمولا شخص با نوشتنِ کد در پروژه‌ی کد باز می‌تواند جدیدترین دست‌آوردهای خود را در این زمینه به معرضِ دیدِ دیگران بگذارد.

۵.۵ تلفیق و بسته‌بندی

در این مرحله پیمانه‌های پیاده‌سازی‌شده جمع گشته و به یک درختِ کدِ واحد اضافه می‌شوند و با هم کامپایل و لینک می‌گردند. در صورتِ لزوم نرم‌افزارهای نصب برای هر پیمانه نوشته می‌شوند، و توسطِ سیستمِ تستِ اتوماتیک، تمامِ نقاطِ ورود به پیمانه‌ها و رفتارهای موردِ انتظار تست می‌شوند.

در یک پروژه‌ی کد باز این مرحله معمولا شاملِ نوشتنِ چند صفحه‌ی راهنما، حصولِ اطمینان از کامپایلِ صحیح روی انواعِ مختلفِ دست‌گاه‌های در دست‌رس، تمیزکاریِ Makefile، نوشتنِ یک

[۳۲] Detailed Design Document

README، ساختنِ بسته‌ی فشرده‌ی Tarball[۳۳]، قرار دادنِ آن روی یک سایتِ FTP، و نهایتا اعلانِ عمومی در فهرست‌های پستی و گروه‌های خبری می‌باشد. همان‌گونه که متوجه شدید تستِ سیستم در کار نیست. این بخش معمولا در پروژه‌های کد باز بسیار سبک است (استثنا نیز وجود دارد، مانندِ Perl و PostgreSQL). البته به دلایلِ دیگری که بیان خواهند شد، این مسئله چندان ضعفِ بزرگی به حساب نمی‌آید.

۶.۵ تستِ حوزه‌ای

در این مرحله ابتدا کارمندان خود شروع به استفاده از محصول می‌کنند. پس از مدتی لازم است که افرادی خارج از تیمِ پروژه نیز این کار را انجام دهند، زیرا آن‌ها دیدِ متفاوتی نسبت به سیستمِ پیاده‌سازی‌شده دارند و اشکالاتی را کشف خواهند کرد که توسطِ تیم کشف نخواهند شد.

نرم‌افزارهای کد بازبدونِ سرمایه‌گذاری بهترین سیستمِ تستِ حوزه‌ای را در دنیا دارند. دلیلِ بسیار ساده‌ی آن این است که کاربران وقتی برای نرم‌افزار پول نداده باشند بسیار دوستانه‌تر در موردِ مشکلاتِ نرم‌افزار برخورد می‌کنند، و حتی تا حدِ امکان سعی می‌کنند خود عیب را برطرف سازند، و وصله‌ی مربوطه را بفرستند.

از فوایدِ دیگرِ پروژه‌های کد باز بازبینیِ کد توسطِ صدها جفت چشمِ برنامه‌نویسانِ دیگر است، که از راهِ خواندنِ کدِ برنامه اشکالاتِ آن را می‌یابند، نه از طریقِ اجرای آن. بعضی از این افراد پول می‌گیرند تا نرم‌افزاری را از نظرِ امنیت، با خواندنِ کد، بررسی کنند. در اغلبِ موارد اشکالاتِ پیداشده از این طریق به تیمِ نویسنده گزارش می‌شوند.

۷.۵ پشتیبانی

اشکالاتی که پس در مراحلِ تستِ حوزه‌ای و پس از آن یافت می‌شوند باید در یک پایگاهِ داده‌ی مرکزی ضبط و نگه‌داری شوند. پس از آن باید هر اشکال به یکی از مهندسینِ مربوطه واگذار شود، و مهندسِ مربوطه راهِ حلی برای رفعِ مشکل ارائه کند. این راهِ حل و مشکل در پایگاهِ داده ثبت شده و نگه‌داری می‌شوند، همچنین هرگونه داده‌ی آزمونِ[۳۴] جدید که در شناساییِ اشکالِ مربوطه به کار می‌رود، به مجموعه‌ی داده‌های آزمون اضافه می‌گردد تا از بروزِ مجددِ این مشکل در آینده جلوگیری شود.

امروزه بسیاری از پروژه‌های کد بازبدونِ سرمایه‌گذاری نیز آن‌چنان حرفه‌ای عمل می‌کنند که

---

[۳۳] .tar.gz

[۳۴] Test Data

هیچ نقصی در پشتیبانی ندارند، مانندِ PostgreSQL، ولی در اغلبِ پروژه‌های کوچک‌تر، جوابی که در مقابلِ گزارشِ یک اشکال خواهید گرفت، در حد: «اوه، ببخشید.» است. از طرفِ دیگر این نداشتنِ پشتیبانی جای را برای تیم‌های پشتیبانیِ نرم‌افزارِ مستقل باز می‌کند.

۶ نگاهی بر چند نرم‌افزارِ کد باز

در این بخش به چند پرونده‌ی جالب و فراموش نشدنی در تاریخِ جنبشِ کد باز اشاره خواهیم کرد. با نگاهی دقیق به این واقعیت‌ها، می‌توان به مطالبِ فراوانی در موردِ جنبه‌های تجاریِ کد باز دست یافت.

۱.۶ Linux

هسته‌ی لینِکس را می‌توان بزرگ‌ترین موفقیتِ جنبشِ کد باز دانست. این هسته که طراحی و پیاده‌سازیِ آن توسط لینِس تُروالدز[۳۵] شروع شد، پس از مدتِ کمی توانست با جذبِ برنامه‌نویسانِ قدرت‌مندی از سراسرِ دنیا به یک هسته‌ی سیستمِ عاملِ بسیار قدرت‌مند تبدیل گردد. لینِکس در مدلِ بازار تولید شده است ولی برای حفظ یک‌نواختی و معماریِ درستِ آن، تاییدِ نهاییِ هر تغییری به عهده‌ی شخصِ تُروالدز است.

۲.۶ QT و KDE

سیستمِ KDE به عنوان اولین واسطِ گرافیکیِ لینکس مطرح شد، اما مشکلِ آن این بود که بر پایه‌ی کتاب‌خانه‌ای به نامِ QT نوشته شده بود که مجوزِ آن به طورِ کامل کد باز نبود. در این‌جا جنبشِ کد باز با فشاری که به شرکتِ تولید کننده‌ی QT یعنی TrollTech وارد آورد آن شرکت را وادار به تغییرِ مجوزِ QT به یک مجوزِ کاملا کد باز کرد. این فشارها چیزی نبود جز شروعِ دو پروژه‌ی GNOME و Harmony. سیستمِ GNOME یک واسطِ گرافیکی معادلِ KDE بود که از ابتدا با کد باز نوشته می‌شد، و Harmony یک پیاده‌سازیِ مستقل و آزاد از واسطِ کتاب‌خانه‌ی QT بود، که در صورتِ اتمام می‌توانست وابستگیِ KDE را به QT حذف کند. شرکتِ TrollTech بسیار قبل از این‌که این دو پروژه به مرحله‌ی قابلِ استفاده برسند تسلیم شد.

[۳۵] Linus Torvalds

۳.۶ Netscape

این شرکت با مطالعه‌ی مقالاتِ اریک ریموند[۲]، به باز کردنِ کدِ خود علاقه نشان داد، که نتیجه‌ی آن امروز مرورگرِ بسیار توانای Mozilla است. این مرورگر حاصلِ هم‌کاریِ هزاران برنامه‌نویس است که کاملا در مدلِ بازار تولید شده است.

۴.۶ RedHat

این شرکت یکی از موفق‌ترین شرکت‌هایی بوده که با طرح تجاریِ مبتنی بر کد باز توانسته است به بازارِ بورسِ آمریکا برسد. باب یانگ[۳۶]، موسس و مدیرعاملِ RedHat مبنای شرکتِ خود را بر اساسِ ارائه‌ی یک مارکِ خوش‌نام و جاافتاده در زمینه‌ی لینکس نهاد. او ایده‌ی خود را از این‌جا گرفت که با آن‌که آبِ آشامیدنی بسیار ارزان و در دست‌رس است، ولی شرکت‌هایی هستند که سودِ هنگفتی از بسته‌بندی و ارائه‌ی آبِ آشامیدنی به دست می‌آورند. همین‌طور در موردِ شرکت‌های اتومبیل‌سازی، که خود قطعات‌شان را نمی‌سازند، بلکه قطعات را به هم بسته، و از راهِ مارک و اعتبارشان پول در می‌آورند. پس RedHat نیز شروع به جمع‌آوری و بسته‌بندیِ بیش از چهارصد نرم‌افزارِ کد باز و ارائه‌ی آن‌ها تحتِ نامی واحد کرد و از این راه توانست اعتبار و سرمایه‌ی زیادی کسب کند.

۵.۶ Apache Foundation

این گروه که به تولیدِ نرم‌افزارهای آزاد می‌پردازد، پشتیبانیِ مالیِ خود را از شرکتِ Sun Microsystems می‌گیرد و در ازای آن نرم‌افزارهای بسیار قدرت‌مندی به زبانِ جاوا تولید می‌کند. شرکتِ Sun نیز سودِ خود را از افزایشِ علاقه‌ی برنامه‌نویسان به زبانِ جاوا به دست می‌آورد.

۶.۶ MySQL و PostgreSQL

این دو سیستم هر دو پایگاه‌های داده‌ی رابطه‌ایِ کد باز می‌باشند، با این تفاوت که MySQL در مدلِ کلیسا و تحتِ نظارتِ دقیقِ یک تیمِ متمرکز تولید می‌شود، ولی PostgreSQL به صورتِ کاملا باز و توسطِ برنامه‌نویسانِ بسیار. جالب است که با آن‌که MySQL حاصلِ مدلِ توسعه‌ی خود را کیفیتِ کدِ بالاتر می‌داند، ولی در عمل هنوز امکاناتِ پایه‌ی یک پایگاهِ داده، از جمله Transaction را پشتیبانی

[۳۶] Bob Young

نمی‌کند. از سوی دیگر، PostgreSQL با سرعتِ بسیار زیادی رو به گشترش است، دارای قابلیت‌های بسیاری می‌باشد، و هم‌اکنون قابلِ مقایسه با سیستم‌های گران‌قیمتی مانندِ Oracle می‌باشد.

نتیجه

امروزه شواهدِ کافی وجود دارد که توسعه‌ی نرم‌افزار در مدلِ کد باز تنها جهتِ تفریح و در اوقاتِ فراغت انجام نخواهد شد. این روشِ تولیدِ نرم‌افزار هم‌اکنون تحدیدی جدی برای بسیاری از شرکت‌ها می‌باشد که با کاهش از حجمِ ایجادِ اندیشه‌های بدیع، هنوز از روی اعتبار و مارک‌شان پول در می‌آورند. جنبشِ کد باز هنوز بسیار جوان است، و در دنیای فردا که این جنبش زمانِ کافی را برای تولیدِ سیستم‌های کلیدیِ مختلف یافته است و محصولاتِ بی‌نقصِ خود را به بازار عرضه کند، راهی جز روی آوردن به این مدل برای مقابله با ورشکستگی برای غالبِ شرکت‌های نرم‌افزاری نمی‌ماند. لذا مطالعه در زمینه‌ی روش‌های مهندسیِ نرم‌افزار در این مدل بسیار ضروری می‌نماید.

منابع

[1] DiBona, Chris *et al*, *Open Sources, Voices from the Open Source Revolution*, 1st ed, 1999, O'Reilly.

[2] Raymond, Eric S., *The Cathedral & The Bazaar*, Revised, 2001, O'Reilly.

[3] Stallman, Richard, *Free as in Freedom*, 1st ed, 2002, Sam Williams.

[4] Torvalds, Linus and David Diamond, *Just for Fun*, 1st ed, 2001, HarperBusiness.