جلسه سوم مبانی مهندسی نرم افزار

مهندسی نرم افزار

تعریف : بکارگیری یک روش منظم و سیستماتیک و قابل اندازه گیری در جهت تولید، توسعه و نگهداری نرم افزار

فصل ۲ : فرآیند

فرآیند چیست؟ وقتی که کار می کنید تا یک سیستم یا محصول بسازید، حتماً باید یک سری مراحل قابل پیش بینی را چک کنید یک نقشهٔ راه که در ایجاد نتیجه‌ای با کیفیت بالا و به موقع شما را یاری می کند. این نقشه که آن را دنبال می کنید، «فرآیند نرم افزار» نام دارد.

یا به مجموعه فعالیتهایی که هدف آنها توسعه یا تکامل نرم افزار است ، فرآیند نرم افزار گویند

چهار فعالیت اساسی در فرآیند نرم افزار:

۱. تعیین مشخصات نرم افزار : مشخص کردن وظایف نرم افزار و محدودیت های عملیاتی آن

۲. توسعه نرم افزار : تولید نرم افزار با مشخصات تعیین شده در مرحله قبل

۳. اعتبار سنجی نرم افزار: نرم افزار باید اعتبار سنجی شود تا تضمین کند خواسته های کاربر را تامین می کند یا نه..

۴. تکامل نرم افزار: نرم افزار باید تکامل یابد تا نیازهای جدید کاربر را تامین کند

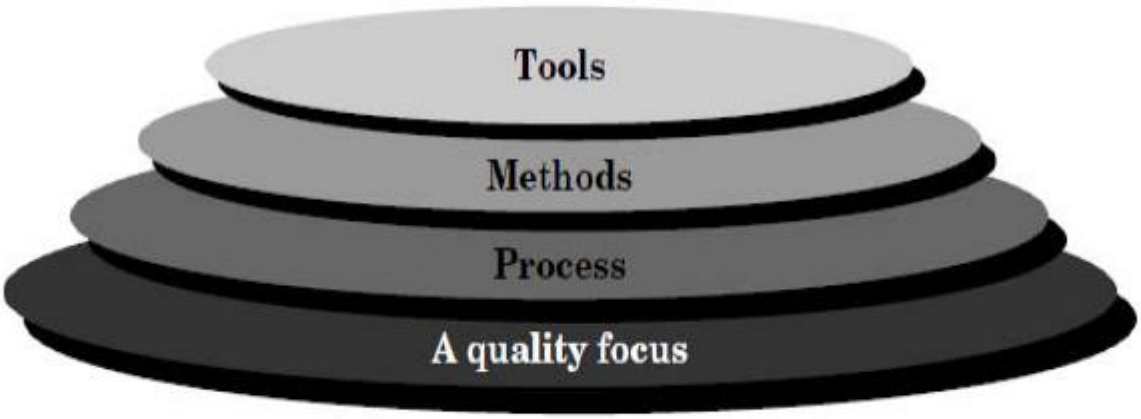
فرآیند، روشها و ابزارها

مهندسی نرم افزار یک فناوری لایه‌ای است. شکل ۱-۲ نشان می دهد که هر روش مهندسی (از جمله مهندسی نرم افزار) باید بر کوشش سازمانی در جهت بهبود کیفیت متکی باشد.

بنیاد مهندسی نرم افزار، لایه فرآیند است. مهندسی نرم افزار به مثابه چسبی عمل می کند که لایه‌های فنّاوری را به هم نگه می دارد و بسط موجه و به موقع نرم افزارهای کامپیوتری را میسر می‌سازد. فرآیند، چارچوبی برای یک مجموعه از زمینه های فرآیند کلیدی (KPA) فراهم می آورد [PAU93] که باید برای تحویل مؤثر فنّاوری مهندسی نرم افزار وضع شوند. زمینه های فرآیند کلیدی، پایه‌ای برای کنترل مدیریتی پروژه های نرم افزاری تشکیل داده بستری برای اِعمال روشهای فنی، تولید محصولات کاری (مدلها، مستندات، داده ها، گزارشها، فرمها و غیره)، تعیین مراحل، حصول اطمینان از کیفیت و مدیریت مناسب تغییرات ایجاد می کنند.

روشهای مهندسی نرم افزار، شیوه های فنی برای ساخت نرم افزار را فراهم می آورند. روشها شامل آرایه وسیعی از وظایف از جمله: تحلیل خواسته ها، طراحی، ساخت برنامه ها، آزمایش و پشتیبانی می شوند. روشهای مهندسی نرم افزار متکی بر یک مجموعه اصول بنیادی است که بر تمام زمینه‌های فنّاوری حاکم بوده شامل فعالیتهای مدلسازی و فنون توصیفی دیگر می شوند.

ابزارهای مهندسی نرم افزار، متضمن پشتیبانی خودکار یا نیمه خودکار برای فرآیند و روشها هستند. هنگامی که ابزارها گرد هم آیند به طوری که اطلاعات ایجاد شده توسط یک ابزار، توسط ابزارهای دیگر قابل استفاده باشند، سیستمی برای پشتیبانی بسط نرم افزار شکل می گیرد که مهندسی نرم افزار به کمک کامپیوتر (CASE) نام دارد.



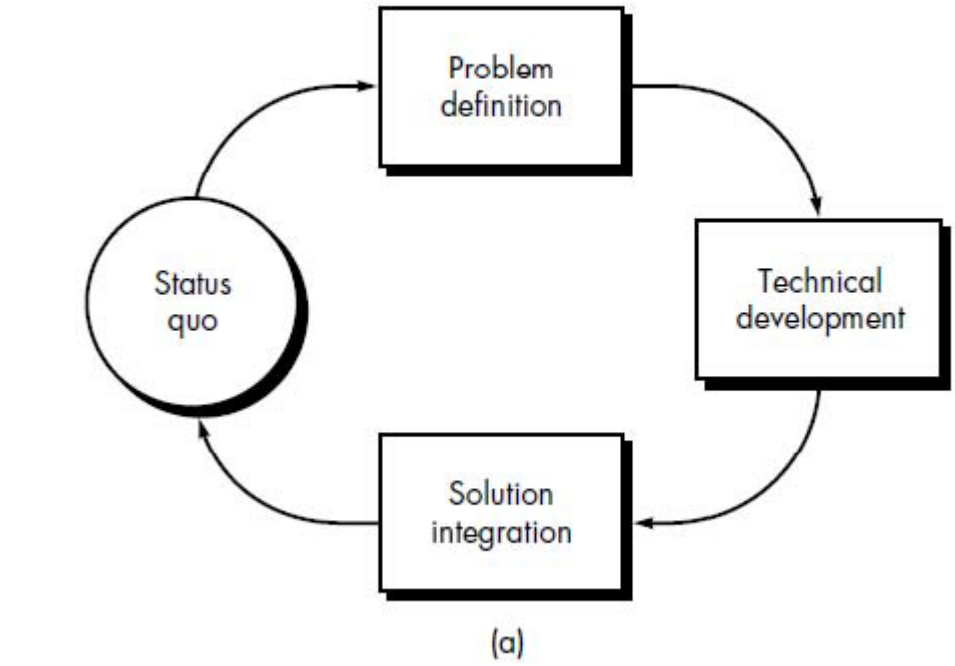
شکل ۱-۲ لایه های مهندسی نرم افزار

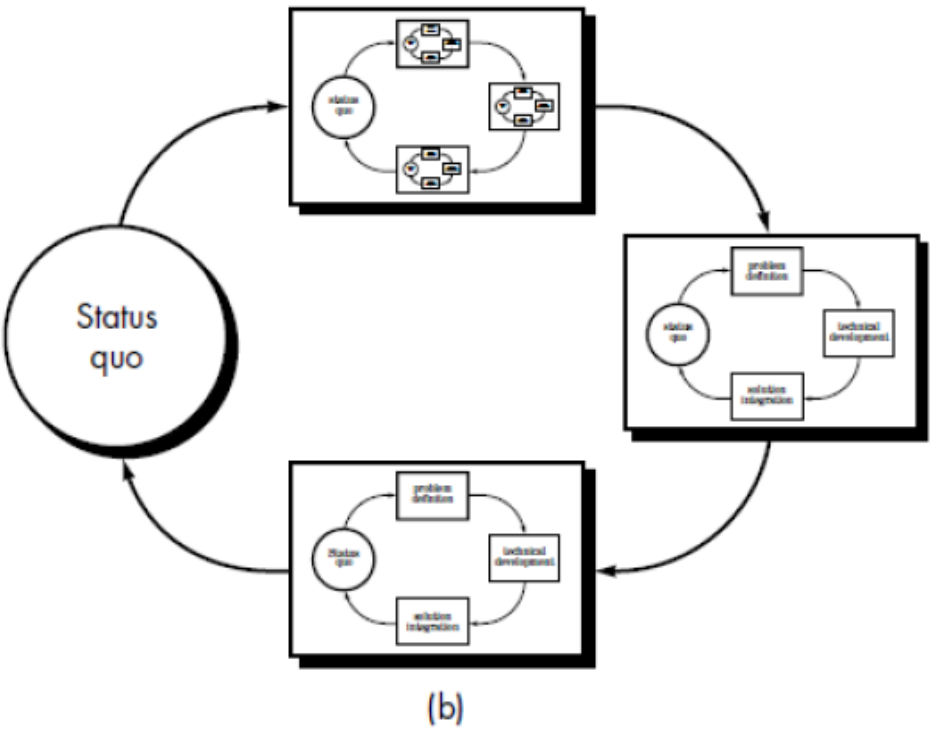
مدل های فرآیند نرم افزار

برای حل مسائل واقعی در یک مجموعه صنعتی، یک مهندس نرم افزار یا تیمی از مهندسان باید یک راهبرد توسعه تعیین کنند که در برگیرنده لایه های فرآیند، روش ها و ابزارها (که در بخش ۱-۱-۲ شرح داده شد) و فازهای کلی شرح داده شده در بخش ۲-۱-۲ باشد این راهبرد را غالبا مدل فرآیند یا الگوی مهندسی نرم افزار می نامند یک مدل فرآیند برای مهندسی نرم افزار براساس ماهیت پروژه و نوع کاربرد، روش ها و ابزارهای مورد استفاده و کنترل ها و قطعات قابل تحویل موردنیاز انتخاب می شود.

توسعه نرم افزار را در کل می توان به عنوان یک حلقه حل مسئله در نظر گرفت(شکل ۳-۲ الف) که در آن چهار مرحله متمایز به چشم می خورد، وضع موجود، تعیین مسئله، توسعه فنی، و جامعیت راه حل. وضع موجود، وضعیت کنونی امور را نشان می دهد؛ تعیین مسئله مشخص می کند که چه مسئله خاصی باید حل شود. توسعه فنی، مسئله را از طریق به کارگیری فناوری حل می کند و با جامعیت راه حل، نتایج(مانند مستندات، برنامه ها، داده ها، وظایف کاری

در جهان واقعی، مرزبندی میان فعالیت ها به صورت ایده آلی که در شکل ۳-۲ ب نشان داده شده است. دشوار است زیرا بین مراحل تداخل رخ می دهد همین دید ساده هم منجر به ایده های بسیار مهمی می شود مدل فرآیند انتخاب شده برای یک محصول نرم افزاری، هر چه باشد همه مراحل فوق وضع موجود، تعیین مسئله، توسعه فنی همزمان از یک درجه اهمیت برخوردار هستند. نظر به ماهیت بازگشتی شکل ۳-۲ ب چهار مرحله شرح داده شده در بالا به یک میزان در تحلیل یک کاربرد کامل و تولید قطعه ای کوچک از کد به اجرا گذاشته می شوند.





شکل ۳-۲

الف) فازهای یک حلقه حل مسئله [RAC95]. **ب) فازهای موجود در داخل فازهای حلقه حل مسئله** [RAC95].

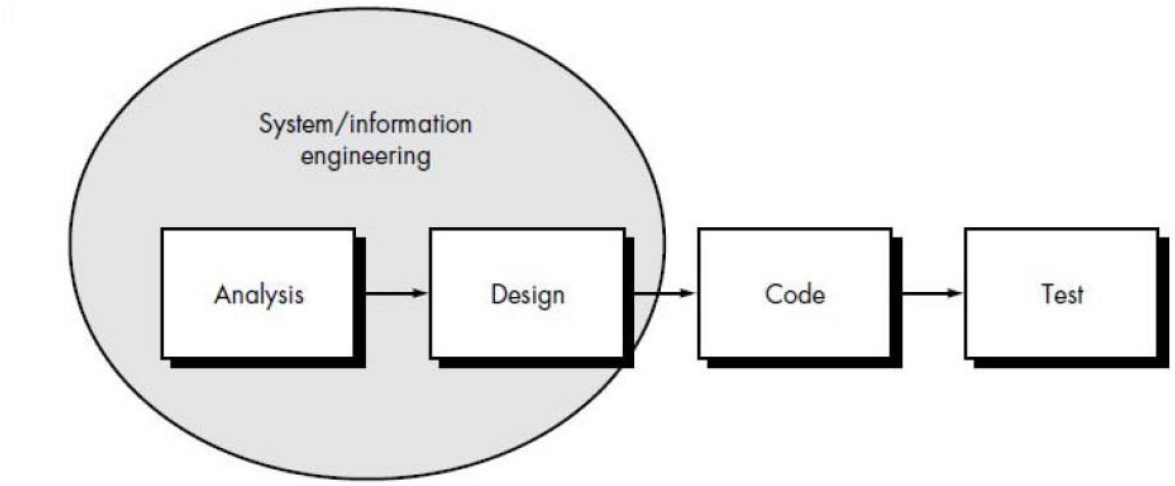
راکون [RAC95] یک مدل آشوب پیشنهاد می کند که «بسط نرم افزار» را به عنوان طیفی گسترده از کاربر به سازنده و از سازنده به فناوری توصیف می کند با پیشرفت کار به سمت یک سیستم کامل مراحل شرح داده شده در بالا به طور بازگشتی در مورد نیازهای کاربر و مشخصات فنی سازنده نرم افزار به اجرا گذاشته می شود.

در بخش های بعدی، چند مدل فرآیند متفاوت برای مهندسی نرم افزار مورد بحث قرار خواهد گرفت هر یک از این مدل ها کوششی برای به نظم درآوردن یک فعالیت ذاتا آشوبی را به تصویر می کشد ضمنا به خاطر داشته باشید که هر یک از این مدل ها به شیوه ای تعریف شده است که (انشاالله) به کنترل و هماهنگ سازی یک پروژه نرم افزاری واقعی کمک کند با این همه، این مدل ها ویژگی های مدل آشوب را از خود نشان می دهند.

مدل ترتیبی خطی

این مدل که گاه «مدل آبشار» یا «چرخه حیات کلاسیک» نیز خوانده می شود، یک روش سیستماتیک و ترتیبی برای بسط نرم افزار پیشنهاد می کند که در سطح سیستمی آغاز می شود و به تحلیل، طراحی، کدنویسی، آزمایش و پشتیبانی پیشروی می کند شکل ٤-٢ مدل ترتیبی خطی برای مهندسی نرم افزار را نشان می دهد مدل ترتیبی خطی که از روی چرخه مهندسی سنتی گرفته شده است، شامل فعالیت های زیر می شود.

مهندسی سیستم/ اطلاعات و مدل سازی: از آنجا که نرم افزار همواره بخشی از یک سیستم (یا یک مقوله کاری) بزرگتر است، کار با تعیین خواسته های مربوط به همه عناصر سیسستم و سپس اختصاص دادن زیر مجموعه ای از این خواسته ها به نرم افزار آغاز می شود این دید سیستمی هنگامی اهمیت می یابد که قرار است نرم افزار با عناصر دیگری از قبیل سخت افزار، افراد و بانک های اطلاعاتی تعامل داشته باشد تحلیل و مهندسی سیستم شامل جمع آوری خواسته ها در سطح سیسستم، با مقدار اندکی تحلیل و طراحی سطح بالا است، مهندسی اطلاعات شامل جمع آوری خواسته ها در سطح تجارت راهبردی و در سطح زمینه کاری است.



شکل ٤-٢ مدل ترتیبی خطی

تحلیل خواسته های نرم افزار. فرآیند جمع آوری خواسته ها به طرزی خاص بر روی نرم افزار تشدید و متمرکز می شود برای درک ماهیت برنامه (هایی) که باید ساخته شوند، مهندس یا تحلیلگر نرم افزار باید دامنه اطلاعات را برای نرم افزار و همچنین عمل مورد نیاز، رفتار، کارآیی و ایجاد واسط، درک کند.

طراحی. طراحی نرم افزار فرآیندی چند مرحله ای است که بر چهار صفت متمایز از یک برنامه تمرکز دارد، ساختمان داده ها، معماری نرم افزار، نمایش واسط ها و جزئیات رویه ای (الگوریتم ها). فرآیند طراحی، خواسته ها را به نمودی از نرم افزار ترجمه می کند که قبل از کدنویسی می توان از آن برای ارزیابی کیفی استفاده کرد.

تولید کد. طراحی باید به یک شکل قابل فهم برای ماشین تبدیل شود در مرحله تولید کد این وظیفه انجام می شود اگر طراحی به تفضیل صورت پذیرد، تولید کد را می توان به طور خودکار انجام داد.

آزمایش. هنگامی که کدها تولید شدند آزمایش برنامه آغاز می شود فرآیند آزمایش، بر منطق داخلی نرم افزار متمرکز می شود تا اطمینان حاصل شود که همه دستورها مورد آزمایش قرار گرفته اند علاوه بر این، بر اجزای خارجی عملیاتی نیز متمرکز می شود، یعنی آزمون هایی برای کشف خطاها و حصول اطمینان از این که ورودی تعیین شده نتایجی واقعی تولید می کند که با نتایج مورد نیاز همساز است، انجام می گیرد.

پشتیبانی. بدون شک پس از آنکه نرم افزار تحویل مشتری شد، دستخوش تغییر می شود (البته نرم افزارهای تعبیه شده از این قاعده مستثنی هستند). تغییرات از آن رو رخ می دهند که خطاهایی به چشم می خورد، زیرا نرم افزار باید برای سازش با تغییرات محیط خارجی (از قبیل تغییرات لازم به سبب وارد شدن یک سیستم عامل جدید و یا یک دستگاه جانبی جدید) خود را مطابقت دهد، یا این که مشتری نیاز به بهتر کردن عملیات و کارایی دارد پشتیبانی و نگهداری نرم افزار، هر یک از فازهای ذکر شده در بالا را در مورد یک برنامه موجود دوباره به کار می گیرد نه در مورد یک برنامه جدید.

مدل ترتیبی خطی، قدیمی ترین و پرکاربرد ترین الگو برای مهندسی نرم افزار است، ولی نقد این الگو باعث شده که حتی هواداران فعال آن نیز بازدهی آن را مورد تردید قرار دهند. [HAN95]. از جمله مشکلاتی که به هنگام اجرای مدل ترتیبی خطی پیش می آید، می توان به موارد زیر اشاره کرد:

۱. پروژه های واقعی به ندرت جریان ترتیبی پیشنهاد شده توسط این مدل را دنبال می کنند. اگر چه مدل خطی می تواند پذیرای تکرار باشد، این عمل را به طور غیر مستقیم انجام می دهد. در نتیجه، با پیش رفتن تیم پروژه ممکن است تغییرات باعث ایجاد سر در گمی شوند.

۲. غالبا برای مشتری دشوار است که همه نیازهای خود را به وضوح بیان کند، مدل ترتیبی خطی، به بیان واضح نیاز دارد و به خوبی از پس موارد غیر قطعی که در آغاز اکثر پروژه ها وجود دارند بر نمی آید.

۳. مشتری باید حوصله داشته باشد یک نسخه کاری از برنامه ها تا آخرین روز های پروژه در دسترس او قرار نخواهد گرفت یک اشتباه عمده که تا زمان بازبینی برنامه کاری از دید، پنهان بماند می تواند بسیار دردسر آفرین باشد.

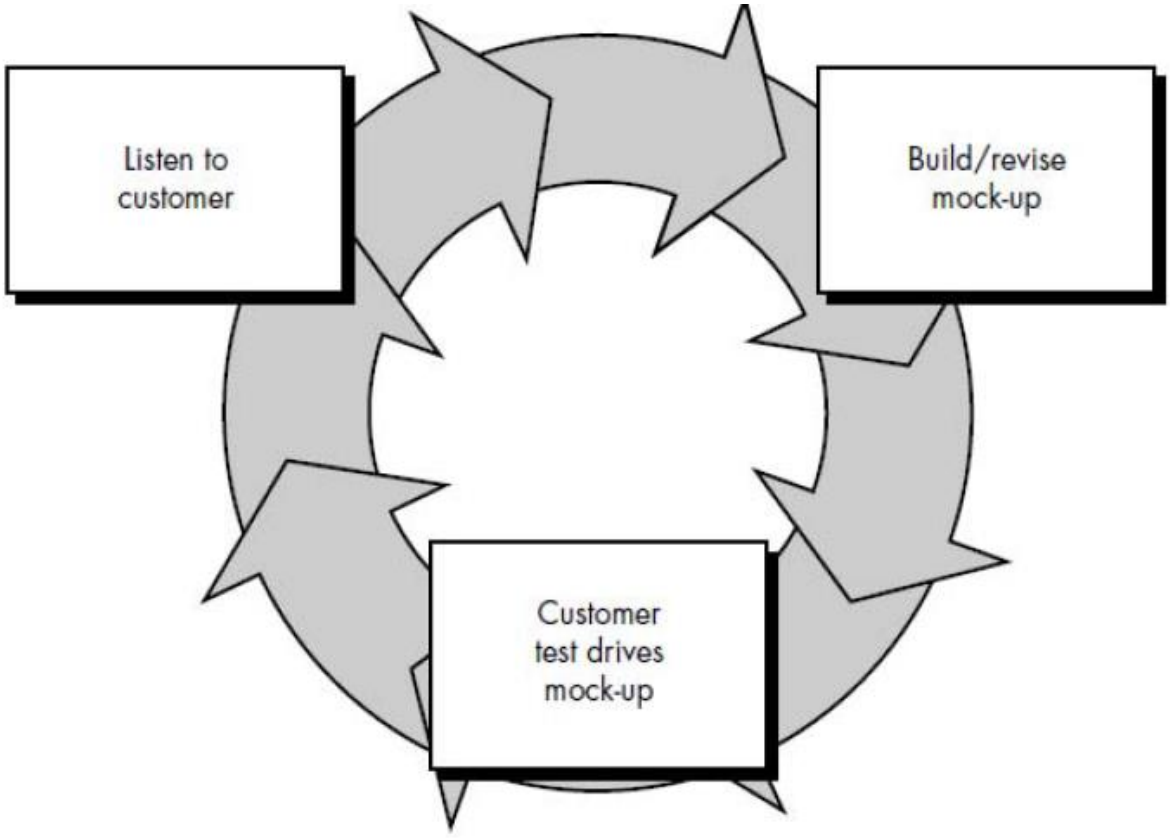
همه این مشکلات جدی هستند، ولی الگوی چرخه حیات کلاسیک جایگاهی مشخص و مهم در کار مهندسی نرم افزار دارد الگویی برای قرار دادن روش های تحلیل، طراحی، کدنویسی، آزمایش و پشتیبانی ارائه می کند چرخه حیات کلاسیک همچنان پر کاربردترین مدل رویه ای برای مهندسی نرم افزار باقی می ماند با وجود نقاط ضعف، به مراتب بهتر از یک روش اتفاقی برای ساخت نرم افزار است.

مدل ساخت نمونه اولیه

غالبا مشتری، یک مجموعه اهداف برای نرم افزار تعیین می کند، ولی جزئیات ورودی ها، پردازش و یا خواسته های خروجی را مشخص نمی کند در موارد دیگر، ممکن است سازنده از بازدهی یک الگوریتم، قابلیت تطابق

الگوی ساخت نمونه اولیه، (شکل ۵-۲) با جمع آوری خواسته ها آغاز می شود مشتری و سازنده با هم ملاقات می کنند و اهداف کلی نرم افزار را تعیین می کنند همه خواسته های معلوم را شناسایی می کنند و زمینه هایی را مطرح می کنند که تعریف بیشتر در آنها الزامی است سپس یک «طراحی سریع» صورت می پذیرد در طراحی سریع، هدف اصلی، ارائه آن دسته از ویژگی های نرم افزار است که به چشم مشتری، کاربرد می آیند (مثل روش های وارد کردن اطلاعات و فرمت های خروجی) طراحی سریع منجر به ساخت یک نمونه اولیه می شود، نمونه اولیه مورد ارزیابی مشتری/ کاربر قرار گرفته و از آن برای پالایش خواسته های نرم افزاری استفاده می شود که قرار است ساخته شود. با تنظیم نمونه اولیه برای برآوردن نیازهای مشتری، تکرار رخ می دهد و در عین حال، سازنده بهتر می فهمد که چه نیازهایی باید برآورده شود.

در حالت ایده آل، نمونه اولیه به عنوان راهکاری برای تشخیص خواسته های نرم افزار عمل می کند، اگر یک نمونه اولیه کاری ساخته شود سازنده می کوشد تا از قطعات برنامه موجود استفاده کند یا از ابزارهایی (مانند مولد گزارش، مدیریت پنجره و غیره) استفاده کند تا برنامه های کاری به سرعت تولید شود.



شکل ۵-۲ الگوی ساخت نمونه اولیه

ولی هنگامی که نمونه اولیه، اهداف ذکر شده در بالا را بر آورده ساخت، با آن چه می کنیم؟ بروکز [BRO75] چنین پاسخ می دهد:

در اکثر پروژه ها، نخستین سیستمی که ساخته می شود چندان قابل استفاده نیست، ممکن است بیش از حد آهسته باشد بیش از حد بزرگ باشد، استفاده از آن دشوار باشد یا اینکه هر سه عیب را با هم داشته باشد چاره ای جز شروع دوباره وجود ندارد باید نسخه دیگری ساخت که این مشکلات در آن حل شده باشد.. هنگامی که یک فناوری جدید یا یک مفهوم سیستمی جدید به کار می رود، باید سیستمی برای دور انداختن درست کرد، زیرا حتی بهترین طراحی ها

نمونه اولیه می تواند به عنوان نخستین سیستم عمل کند، یعنی همان طور که بروکز توصیه می کند، دور انداخته شود، ولی این ممکن است یک دید ایده آل باشد این درست است که هم مشتریان و هم سازندگان، الگوی ساخت نمونه اولیه را دوست دارند کاربران احساس می کنند که یک سیستم واقعی را آزمایش می کنند و سازندگان چیزی را بلافاصله ساخته اند با این همه، ساخت نمونه اولیه نیز می تواند به دلایل زیر مشکل آفرین باشد:

۱. مشتری چیزی را می بیند که ظاهرا یک نسخه کاری از نرم افزار است ولی نمی داند که این نمونه اولیه با موم سر هم بندی شده است نمی داند که به لحاظ شتابی که به کارگیری داشته ایم، کیفیت کلی نرم افزار و قابلیت نگهداری دراز مدت مدنظر نبوده است هنگامی که مطلع می شود محصول باید بازسازی شود تا به سطوح بالای کیفیت برسد از کوره در می رود و تقاضا می کند با چند ترمیم جزئی این نمونه اولیه به یک محصول کاری تبدیل شود، اکثر اوقات هم مدیریتی ساخت نرم افزارکوتاه می آید.

۲. سازندگان غالبا برای به کارگیری هر چه سریعتر نمونه اولیه، در پیاده سازی آن کوتاه می آیند، ممکن است از یک سیستم عامل یا زبان برنامه نویسی نامناسب استفاده شود صرفا به خاطر این که در دسترس و شناخته شده است ممکن است یک الگوریتم ناکارآمد پیاده سازی شود صرفا برای آنکه قابلیت برنامه نشان داده شود، پس از مدتی برنامه نویس ممکن است با این انتخاب ها مانوس شود و کلا فراموش کند که چرا مناسب بوده اند انتخاب « کمتر از ایده آل» اکنون به بخشی از سیستم تبدیل شده است.

ممکن است مشکلاتی رخ دهد ولی ساخت نمونه اولیه می تواند الگوی موثری برای مهندسی نرم افزار باشد کیلوکار، تعیین قواعد بازی در همان آغاز است، یعنی مشتری و سازنده هر دو باید بپذیرند که نمونه اولیه بدین منظور ساخته می شود تا به عنوان راهکارهای برای تعیین خواسته ها عمل کند.