**نام و نام خانوادگی: فاطمه امیری شماره دانشجویی: 990065419**

**نام درس: مهندسی نرم افزار1 نام دانشکده: فنی و مهندسی**

**گروه: مهندسی کامپیوتر نام استاد: جناب آقای احمد فراهی**

**عنوان: اصول محوری مهندسی نرم افزار**

**نرم افزار**

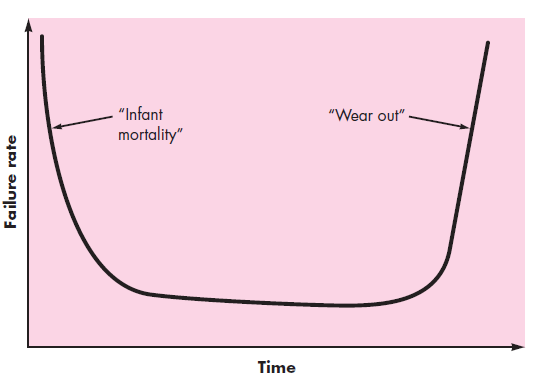
بحث را با تعریف نرم افزار و برشمردن خصوصیات آن آغاز می کنیم. پرسمن در کتاب خود نرم افزار را به گونه زیر تعریف می کند.

نرم افزار عبارت است از: (۱) دستورالعمل هایی که هنگام اجرا، ویژگی، عملکرد و کارایی مطلوب را فراهم می سازند؛ (۲) ساختمان داده هایی که برنامه ها را قادر به پردازش مناسب داده ها کنند و (۳) اطلاعات توصیفی در هر دو قالب کپی سخت و مجازی که راه اندازی و استفاده از برنامه ها را شرح دهند.

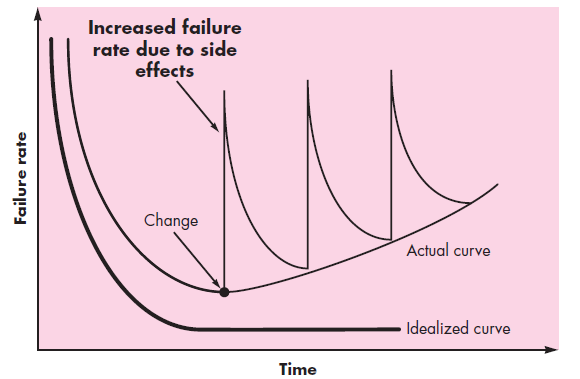
از آنجا که نرم افزار، یک عنصر منطقی است تا یک عنصر فیزیکی، دارای ویژگی هایی است که تفاوت زیادی با سخت افزار دارد:

* نرم افزار، مهندسی و بسط داده می شود و چیزی نیست که به معنای کلاسیک کلمه، ساخته شود.
* نرم افزار فرسوده نمی شود.

شکل نمودار آهنگ شکست را به صورت تابعی از زمان برای سخت افزار نشان می دهد. این رابطه که اغلب منحنی وانی نامیده می شود، نشان می دهد که سخت افزار در ابتدای عمر خود آهنگ شکست نسبتاً شدیدی دارد (این شکست را غالباً می توان به عیوب طراحی و تولید نسبت داد)، این عیوب تصحیح می شوند و آهنگ شکست برای یک دوره زمانی به مقداری ثابت نزول می کند. با گذشت زمان، سخت افزار شروع به فرسایش کرده و دوباره آهنگ شکست شدت می گیرد.



نرم افزار نسبت به ناملایمات محیطی که باعث فرسایش آن می شود، نفوذ پذیر نیست. بنابراین، در تئوری، منحنی شکست برای نرم افزار باید شکل منحنی ایده آل شکل را به خود بگیرد. عیوب کشف نشده باعث آهنگ شکست شدید، در ابتدای عمر برنامه می شود، ولی این عیوب برطرف می شوند و منحنی به طوری که نشان داده شده است، هموار می شود. منحنی ایده آل نسبت به منحنی واقعی مدل های شکست نرم افزار بسیار ساده تر است. ولی معنای آن بسیار واضح است، نرم افزار هرگز دچار فرسایش نمی شود بلکه زوال می یابد.



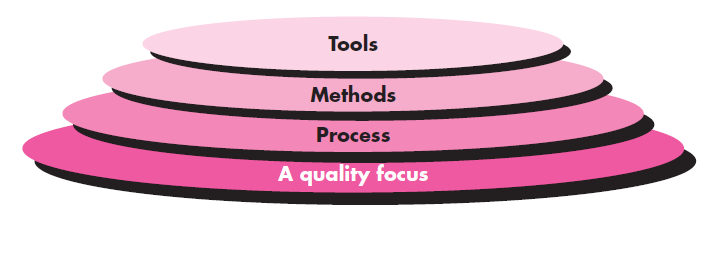
این تناقض ظاهری را می توان با در نظر گرفتن (منحنی واقعی) به بهترین وجه توضیح داد. نرم افزار در دوران حیات خود دستخوش تغییر می شود(نگهداری). با اعمال این تغییرات، احتمال دارد که برخی عیوب جدید وارد شوند و باعث خیز منحنی آهنگ شکست شوند. پیش از آنکه منحنی بتواند به آهنگ شکست منظم اولیه خود برسد، تغییر دیگری درخواست می شود که باعث خیز دوباره منحنی می شود. حداقل میزان شکست به آهستگی افزایش می یابد- نرم افزار در اثر تغییر فاسد می شود.

گر چه صنعت در حال حرکت به سوی مونتاژ قطعات است، اکثر نرم افزارها همچنان به صورت سفارشی ساخته می شوند. در جهان سخت افزار، استفاده مجدد از قطعات، بخشی از فرایند مهندسی است. در مهندسی نرم افزار این امر به تازگی مورد توجه قرار گرفته است. در واقع یک مولفه نرم افزاری باید چنان طراحی و پیاده سازی شود که بتوان در برنامه های متفاوت از آن بهره برد.

**مهندسی نرم افزار**

برای مهندسی نرم افزار تعارف متفاوتی ارائه شده است، IEEE مهندسی نرم افزار را اینگونه شرح می دهد: کاربرد یک روش سیستماتیک، علمی و کمیت پذیر در بسط، راه اندازی و نگهداری نرم افزار، یعنی استفاده از مهندسی نرم افزار.

مهندسی نرم افزار یک فن آوری لایه ای است. با توجه به شکل ، هر روش مهندسی (از جمله مهندسی نرم افزار) باید متکی به تعهد سازمانی به کیفیت باشد. در واقع سنگ بنای نگهدارنده مهندسی نرم افزار، توجه به کیفیت است.



بنیاد مهندسی نرم افزار، لایه فرایند است. فرایند چارچوبی را تعریف می کند که باید برای تحویل موثر فناوری مهندسی نرم افزار وضع شود. فرایند نرم افزار، پایه ای برای کنترل مدیریتی پروژه های نرم افزاری تشکیل داده، بستری برای اعمال روش های فنی، تولید محصولات کاری (مدل ها، مستندات، داده ها، گزارشات، فرم ها و غیره)، تعیین مراحل، حصول اطمینان از کیفیت و مدیریت مناسب تغییرات ایجاد می کند.

روش های مهندسی نرم افزار، شیوه های فنی برای ساخت نرم افزار را فراهم می آورند. این روش ها شامل آرایه وسیعی از وظایف از جمله: تحلیل خواسته ها، طراحی، ساخت برنامه ها، آزمایش و پشتیبانی می شود.

ابزارهای مهندسی نرم افزار، متضمن پشتیبانی خودکار یا نیمه خودکار برای فرایند و روش هایی 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 نرم افزار شکل می گیرد که مهندسی نرم افزار به کمک کامپیوتر نام دارد.

**فرایند مهندسی نرم افزار**

چارچوب فرایند با تعیین تعداد کوچکی از فعالیت های چارچوبی که برای کلیه پروژه های نرم افزاری قابل استفاده باشند، صرف نظر از اندازه و پیچیدگی آنها، شالوده ای برای یک فرایند مهندسی نرم افزار کامل پی ریزی می کند. یک چارچوب فرایند کلی برای مهندسی نرم افزار شامل پنج فعالیت می شود.

* ا**رتباطات:**
* پیش از اینکه هرگونه کار فنی آغاز شود، برقراری ارتباط و همکاری با مشتری بسیار مهم است.

هدف، درک اهداف طرف های ذی نفع برای پروژه و جمع آوری خواسته هایی است که می توانند ویژگی ها و قابلیت های عملیاتی نرم افزار را تعیین کنند.

* **برنامه ریزی:**
* یک پروژه نرم افزاری، سفری پیچیده است و فعالیت برنامه ریزی، نقشه ای ایجاد می کند که به راهنمایی تیم در انجام این سفر کمک می کند. این نقشه با توصیف وظایف فنی که قرار است اجرا شوند، خطرات احتمالی، منابعی که مورد نیاز خواهند بود، محصولات کاری ای که باید تولید شوند و زمانبندی کاری، مهندسی نرم افزار را مشخص می کند.
* **مدل سازی:**
* یک معمار، هر روز با مدل ها کار می کند، اِتودی می زند تا تصویر بزرگ را درک کند، اینکه از نظر معماری چه ظاهری دارد، بخش های سازنده اش چگونه با هم جور در خواهند آمد، و بسیاری خصوصیات دیگر. مهندسی نرم افزار با ایجاد مدل هایی جهت درک بهتر خواسته ها و طراحی که به این خواسته ها برسد، همین کار را می کند.
* **ساخت:**
* این فعالیت، تولید کدها و آزمون لازم برای آشکار کردن خطاهای موجود در کدها را با هم تلفیق می کند.
* **استقرار:**
* نرم افزار به مشتری تحویل داده می شود تا محصول تحویل داده شده را ارزیابی کرده و بر اساس این ارزیابی، بازخوردی ارائه دهد. برای بسیاری از پروژه های نرم افزاری، فعالیت های چارچوبی به موازات پیشرفت پروژه به صورت تکراری به کار برده می شوند. در هر دور از تکرار پروژه، یک نسخه از نرم افزار ایجاد می شود که زیر مجموعه ای از قابلیت های عملیاتی و ویژگی های نرم افزار کامل را در اختیار افراد ذی نفع قرار می دهد. با تولید هر نمو، نرم افزار کامل و کامل تر می شود.

فعالیت های چارچوبی فرایند مهندسی نرم افزار توسط تعدادی از فعالیت های چتری تکمیل می شوند که عبارتند از:

* **کنترل و پیگیری پروژه های نرم افزاری:**
* به تیم نرم افزاری امکان می دهد تا پیشرفت را در مقایسه با نقشه پروژه بسنجد و هر گونه کنش لازم را برای حفظ زمان بندی به عمل آورد.
* **مدیریت ریسک:**
* خطراتی را ارزیابی می کند که ممکن است بر نتیجه پروژه یا کیفیت محصول تاثیر بگذارند.
* **تضمین کیفیت نرم افزار:**
* فعالیت های لازم برای حصول اطمینان از کیفیت نرم افزار را معین می کند.
* **بازبینی فنی:**
* محصولات کاری مهندسی نرم افزار را در تلاش برای آشکار کردن خطاها قبل از انتشار آنها در فعالیت بعدی و برطرف کردن آنها ارزیابی می کند.
* **اندازه گیری:**
* موازینی از فرایند، پروژه و محصول را تعریف می کند که نیازهای طرف های ذی نفع را برطرف می سازند.
* **مدیریت پیکربندی نرم افزار:**
* اثرات تغییرات را در سراسر فرایند نرم افزار مدیریت می کند.
* **مدیریت قابلیت استفاده مجدد:**
* ملاک های مربوط به استفاده مجدد (از جمله قطعات نرم افزاری) را تعریف می کند و سازوکارهایی برای دستیابی به قطعات قابل استفاده مجدد برقرار می سازد.
* **تهیه و تولید محصول کاری:**
* شامل فعالیت های لازم برای ایجاد محصولات کاری از قبیل مدل ها، مستندات، وقایع نگارها(کارنامه ها)، فرم ها و فهرست ها می شود.

توجه به این نکته ضروری است که فرایند مهندسی نرم افزار یک دستورالعمل نهایی و غیر قابل تغییر نیست که تیم نرم افزاری باید با تعصب از آن پیروی کند بلکه باید سریع الانتقال و انطباق پذیر باشد(برای مساله، برای پروژه، برای تیم و برای فرهنگ سازمانی). بنابراین فرایندی که برای یک پروژه پذیرفته می شود، ممکن است با فرایند پذیرفته شده برای پروژه های دیگر تفاوتی چشمگیر داشته باشد. در مقالات بعدی مدل های مختلف فرایندها را شرح خواهیم داد.

**مهندسی نرم افزار در عمل**

جورج پولیا در یک کتاب کلاسیک با عنوان (چگونگی حل مساله) که قبل از وجود کامپیوترهای مدرن نوشته شده است، جوهر حل مساله و در نتیجه (جوهر عمل) در مهندسی نرم افزار را چنین مطرح می کند:

* شناخت مساله (برقراری ارتباط و تحلیل)
* طرح ریزی برای یک حل (مدل سازی و طراحی نرم افزار)
* اجرای برنامه ریزی (ایجاد کد)
* بررسی نتیجه برای صحت (آزمایش و تضمیین کیفیت)

**اصول کلی**

دیوید هوکر هفت اصل را مطرح نموده است که توجه به آنها در مهندسی نرم افزار بسیار ضروری به نظر می رسد:

* **اصل یکم) دلیل وجود سیستم:**
* هر سیستم به یک وجود نیاز دارد: این که برای کاربرانش ارزش فراهم سازد. همه تصمیم گیری ها باید با مد نظر داشتن این نکته انجام شود.
* **اصل دوم) ساده نگه داشتن:**
* همه طراحی ها باید تا حد امکان ساده باشند. این باعث می شود که یک سیستم قابل فهم تر با قابلیت نگهداری بالاتر را داشته باشید.
* **اصل سوم) حفظ چشم انداز:**
* برای موفقیت یک پروژه نرم افزاری، چشم اندازی روشن، ضروری است و بدون آن پروژه تقریبا همواره به جایی می رسد که دو یا چند ایده بر آن حاکم خواهد شد. یک سیستم بدون یکپارچگی مفهومی، به مجموعه ی ناجوری از طراحی های ناسازگار تبدیل می شود که به یکدیگر وصله-پینه شده اند. مسامحه در مورد خصوص چشم انداز معماری یک سیستم نرم افزاری باعث تضعیف سیستمی با طراحی خوب و سرانجام از کار افتادن آن می شود.
* **اصل چهارم) آنچه که شما تولید می کنید، دیگران مصرف می کنند:**
* همواره تعیین مشخصات، طراحی و پیاده سازی را طوری انجام دهید که دیگران نیز قادر به درک کار شما باشند.
* **اصل پنجم) آینده نگری:**
* سیستمی با طول عمر بالا از ارزش بیشتری برخوردار است. سیستم ها باید آمادگی انطباق بر تغییرات را داشته باشند. سیستم هایی که این ویژگی ها را با موفقیت ارائه می دهند، از ابتدا با این ویژگی ها طراحی می شوند.
* **اصل ششم) برنامه ریزی پیشاپیش برای استفاده مجدد:**
* استفاده مجدد باعث صرفه جویی در زمان و کار می شود. استفاده مجدد از کد ها و طراحی ها به عنوان مزیت اصلی فن آوری های شیئ گرا مطرح شده است ولی این امکان در برنامه نویسی شیئ گرا نیازمند برنامه ریزی قبلی است.
* **اصل هفتم) تفکر:**
* این آخرین اصل احتمالاً بیش از بقیه مورد بی مهری قرار می گیرد. تعقل و تفکر کامل و روشن قبل از اقدام به عمل، همواره نتایج بهتری به بار می آورد. با تفکر روشن درباره سیستم، ارزش آن بالا می رود. به کارگیری شش اصل نخست نیاز به تفکر عمیق دارد و در این صورت، فایده بسیاری از آن عاید خواهد شد.