اعمال قوانین طراحی نرمافزار

قوانین طراحی نرمافزار، دارایی های ارزشمند جامعهی مهندسی نرمافزار، قواعد تأییدشدهای است که استفادهی صحیح آنها در طراحی نرمافزار، میتواند کیفیت نرمافزار را به شدت افزایش دهد. استفاده و اعمال قوانین طراحی نرمافزار به منظور غلبه بر مشکالت مربوط به طراحی که در عمل رایج می باشند، ضروری بوده و در ادامه تعریف میشوند.

طراحی برای تغییر

معماری برنامه‌ی اصلی و زیر برنامه ها شکل درختی دارد. این ساختار، ازیک برنامه ی اصلی و تعدادی زیربرنامه تشکیل شده است. برنامه ی اصلی زیربرنامه های فرزند خود را فراخوانی می کند و این زیربرنامه ها نیز به نوبه ی خود زیربرنامه های سطوح پایین تر را فراخوانی می نمایند. معمولا زیربرنامه های فرزند، از چپ به راست فراخوانی می گردند. اگر دو گره ی والد، یک زیربرنامه ی فرزند یکسان را فراخوانی کنند، معماری برنامه‌ی اصلی و زیربرنامه ها شکل شبکه ای خواهد داشت.

از این رو با استفاده از این سبک معماری می‌توان به طور مستقل و بدون تغییر در سایر نودها و زیر برنامه‌های درخت اصلی دست به تغییر یک نود و اعمال نیازمندی‌های جدید زد.

جداسازی دغدغه ها

قانون جداسازی دغدغه ها، مسئله ی طراحی نرم افزار را در دو سطح در نظر می‌گیرد. در سطح بالاتر، مسئله، چگونگی انجام فرایند کلی طراحی است. در سطح پایین تر، مسئله، چگونگی طراحی اجزا و مولفه های سیستم نرم افزاری است. به عبارت دیگر، طراحی نرم افزار با هر دو دغدغه ی فرایند طراحی و محصول طراحی در ارتباط است.

بنابر این استفاده از قانون جداسازی دغدغه ها برای طراحی معماری به این معناست که مسئولیت های مربوط به دغدغه های مختلف، به زیرسیستم های مختلف اختصاص داده شود، این کار، به چسبندگی عملیاتی بالا منجر خواهد شد و فهم و استفاده ی مجدد از زیرسیستم ها را آسان تر خواهد کرد.

بنابر این می‌توان هر زیرسیستم را به عنوان یک زیربرنامه (زیر درخت) در نظر گرفت و هر دغدغه را به یک زیربرنامه‌ی مستقل اختصاص داد.

پنهان سازی اطلاعات

قانون پنهان سازی اطلاعات در طراحی معماری، به معنای طراحی سیستم نرم افزاری برای محافظت کردن از جزییات پیاده سازی بخش هایی از سیستم از دید بقیه ی سیستم است.

مطابق این قانون، جزییات پیادهسازی یک بدنهی نرمافزاری، برای کاهش اثرات تغییر آن بر سایر قسمتهای سیستم نرمافزاری، محافظت میشود. به دلیل وجود معماری چند لایه و پنهانسازی برخی اجزای لایه­ها از لایه­های دیگر، رعایت کپسولهسازی و شیءگرایی در سامانه سکو، اثرات تغییرات این گونه اجزا بر بخشهای دیگر سیستم به حداقل رسیده است و این اصل نیز به خوبی در سیستم به کار برده شده است.

چسبندگی زیاد

ایده ی قانون چسبندگی زیاد، از طراحی پیمانه ای در تحلیل و طراحی ساختمند سنتی گرفته شده است. در طراحی ساختمند، سیستم نرم افزاری به یک سلسله مراتب درختی از پیمانه ها تقسیم می شود که در آن، پیمانه های سطوح بالاتر، پیمانه های سطوح پایین تر را فراخوانی می کنند و نتایج بازگردانده شده از آن ها را ترکیب می نمایند. هر پیمانه ، یک مولفه یا زیر سیستم را پیاده سازی می کند و از یک مجموعه داده تشکیل می شود.

در سیستم سکو به کمک معماری n -لایه، نیازمندیهای در نظر گرفته شده برای هر بخش فقط توسط زیر سیستم مربوط به آن بخش پیادهسازی شده است که این موضوع مستقل بودن زیر سیستمهای مختلف از یکدیگر را نشان می­دهد و منجر به آن میشود که هر بخش تنها وظایف مربوط به خود را انجام دهد .

جفت شدگی کم

استفاده از قانون جفت شدگی کم در طراحی معماری، به معنای کاهش اثرات زمان اجرا و تاثیر تغییر در هر زیرسیستم بر زیرسیستم های دیگر است. به عبارت دیگر، این قانون تعیین می نماید که وقتی پیاده سازی یک پیمانه‌ی مشخص تغییر می کند، چند پیمانه‌ی دیگر باید تغییر داده شوند

همانگونه که در قانون طراحی برای تغییر و قانون جداسازی دغدغه‌ها ذکر شد، زیرسیستم‌های سیستم اصلی ما به گونه‌ای انتخاب و طراحی شده‌اند که کمترین ارتباط را با یکدیگر داشته باشند. همچنین ارتباط کم بین این زیرسیستم‌ها باعث کاهش اثرات زمان اجرا نیز میگردد.

ساده و احمقانه فرض کن

قانون ساده و احمقانه فرض کن، طراحیهای ساده، سرراست و قابلفهم را توصیه میکند. بنابراین در این سیستم سعی شده است کالسها و اجزا تا حد امکان تک کاره و برای مسئولیتهای مشخص و کوچک طراحی شوند.

برای مثال با استفاده از این سبک معماری، می‌توان به کمک زیربرنامه های مختلف یک سیستم را به زیرسیستم‌های ساده تر و کوچک تر تقسیم نمود.

ملیکا اقاجانیان صباغ

نرگس جهرمی‌زاده

زهرا امیری‌نژاد