No Silver Bullet Essence and Accidents of Software Engineering, summary

Silver Bullet:

در فرهنگ مردم (کشورهای غربی)، گلولهی نقره‌ای یا به لاتین Silver Bullet گلوله‌ای است که از نقره درست شده‌است و تنها اسلحه‌ای است که قابلیت به‌کارگیری در مقابل [گرگینه](https://fa.wikipedia.org/wiki/%DA%AF%D8%B1%DA%AF%DB%8C%D9%86%D9%87) یا سایر [هیولاها](https://fa.wikipedia.org/wiki/%D9%87%DB%8C%D9%88%D9%84%D8%A7) را دارد. این اصطلاح همچنین یک استعاره برای یک راه ساده و به ظاهر جادویی برای یک مشکل دشوار است: برای مثال، پنی سیلین یک گلوله نقره ای بود که بسیاری از عفونت‌های باکتری را درمان کرد.

\*\* در اول مقاله پروژه نرم‌افزاری رو به گرگینه تشبیه میکنه (اول فکر میکنی خوب و خوشگله، یهو هیولا میشه) و میگوید دنبال گلوله تقره ای برای کاهش هزینه های نرم‌اقزار هستیم. (ب 2)

\*\* هیچ گلوله نقره ای پیدا نشده در دهه های قبل، ما سعی میکنیم با بررسی ماهیت مشکل نرم افزار و خصوصیات گلوله‌های پیشنهادی ، دلیل آن را بفهمیم. (ب 3)

\*\* germ theory: پیشرفت با گام به گام و با تلاش فراوان انجام خواهد شد و مراقبتهای مداوم و بی وقفه باید به منظم انجام شود. (امروزه در مورد مهندسی نرم افزار نیز چنین است) (ب 5)

همون اول قسمت دوم حالمون رو میگیره.

\*\* در حال حاضر نه تنها گلوله های نقره ای مشاهده نمی‌شود ، بلکه ماهیت نرم‌افزار نیز وجود چنین چیزی را بعید می‌داند. نمی توان انتظار داشت که هر دو سال یکبار دو برابر سود داشته باشیم. (ب 6)

\*\* اول ، باید مشاهده کرد که ناهنجاری این نیست که پیشرفت نرم افزار بسیار کند است ، بلکه پیشرفت سخت افزار کامپیوتر بسیار سریع است. از زمان آغاز تمدن ، هیچ فناوری دیگری طی 30 سال گذشته 6 مرتبه افزایش قیمت و عملکرد نداشته است. (ب 7)

\*\* برای اینکه بفهمیم انتظار داریم چقدر پیشرفت قراره اتفاق بیفته، باید مشکلات نرم‌افزار رو بررسی کنیم.

که مثل ارسطو اونو دو قسم میکنه:

1- essence difficulties inherent in the nature of the software اصلی

2- accidents, those difficulties which today attend its production but which are not inherent تصادفی (ب 8)

\*\* بررسی مشکلات essence

ماهیت موجودیت نرم افزار ساختاری از مفاهیم بهم پیوسته است: مجموعه داده ها ، روابط بین موارد داده ، الگوریتم ها و فراخوانی توابع. (ب 9)

\*\* نفهمیدم (ب 10)

\*\* اجازه دهید خصوصیات ذاتی این ذات غیرقابل کاهش در سیستم های نرم افزاری مدرن را در نظر بگیریم: پیچیدگی ، انطباق ، تغییر پذیری و نامرئی بودن.

\*\* **پیچیدگی**

موجودیت های نرم افزاری از نظر اندازه خود بیش از هر سازه انسانی پیچیده تری هستند ، زیرا هیچ دو بخشی شبیه یکدیگر نیستند، اگر اینطور نباشه، دو تا قسمت مشابه رو یکی میکنیم. از این نظر ، سیستم های نرم افزاری با رایانه‎‌ها ، ساختمانها یا اتومبیل ها ، که عناصر مکرر در آنها زیاد است ، عمیقاً متفاوت هستند.

\*\* چس ناله درباره پیچیدگی رایانه‌های دیجیتال

\*\* گفته که موجودیت نرم‌افزاری که خطی گنده میشه خطی پیچیده نمیشه بیشتر از خطیه (چی گفتم، میگه پیچیدگی خطی نیست)

\*\* میگه که پیچیدگی تو ذات نرم‎افزارهاست، واسه سایر علوم دانشمندا میان پیچیدگی رو درنظر نمیگیرن و کاراشونو میکنن، ولی این روش وقتی جواب میده که پیچیدگی ویژگی ذاتی نباشه (تصادفی باشه)

\*\* یه سری از این پیچدگی ها و نتیاجشو گفته

\*\* **انطباق**

\*\* به فیزیک حسودی میکنه، میگه برای اثبات و اینا میگن یه اصل و اساسی هست و خدا الکی چیزی نساخته.

\*\* میگه مهندس نرم افزار نمیتونه این کارو کنه، چون بسیاری از پیچیدگی هایی که وی باید به آن تسلط یابد ، پیچیدگی دلخواه است که توسط بسیاری از نهادها و سیستم های انسانی که رابط های او باید مطابقت داشته باشد ، بدون قافیه و دلیل منطقی مجبور می شوند. (کار دست انسانه نه خدا) نمیتونیم انطباق پیدا کنیم با اون طراحی ها

\*\* **تغییرپذیری**

\*\* نرم‎افزار ها به طور مداوم تغییر میکنن، ولی مثلا ساختمون کمتر تغییر میکنه و به جاش با یه مدل دیگه جایگزین میشه.

\*\* در واقع ساختمان ها تغییر می کنند ، اما هزینه های بالای تغییر ، که همه درک می کنند ، باعث می شود هوی و هوس های تعویض ها کاهش یابد. (تا حدی به این دلیل است که نرم افزار می تواند راحت تر تغییر کند اگه به مرحله عملکرد نرسن)

\*\* نرم افزار موفق تغییر میکنند (بیشتر از سمت مشتریانی که عملکرد اصلی را دوست دارند و کاربردهای جدیدی را برای آن اختراع می کنند)

\*\* یه مورد دیگه اینکه نرم افزار ها از رایانه ای که روش اجرا میشن عمرشون بیشتره (مثلا قطعاتش عوض میشه) پس باید آماده تغییر باشن

\*\* به طور خلاصه ، این محصول نرم افزاری در یک ماتریس فرهنگی از برنامه ها ، کاربران ، قوانین و وسایل نقلیه ماشینی جاسازی شده است. همه اینها بطور مداوم تغییر می کنند و تغییرات آنها به طور غیرقابل تحملی مجبور به تغییر روی محصول نرم افزاری می شود.

\*\* **نامرئی بودن**

\*\* نرم افزار نامرئی و غیرقابل مشاهده است. انتزاعات هندسی ابزار قدرتمندی هستند. نقشه طبقه یک ساختمان به معمار و مشتری کمک می کند تا فضاها ، جریان های ترافیکی ، نماها را ارزیابی کنند.

\*\* واقعیت نرم افزار ذاتاً در فضا نهفته نیست. از این رو نمایشی هندسی آماده به روشی ندارد که زمین دارای نقشه است ، تراشه های سیلیکون دارای نمودار هستند ، رایانه ها دارای نمودارهای اتصال هستند. به محض این که بخواهیم ساختار نرم افزار را نمودار کنیم ، متوجه می شویم که این نه یک ، بلکه چندین نمودار کلی کارگردانی است که روی هم قرار گرفته اند. نمودارهای متعدد ممکن است جریان کنترل ، جریان داده ها ، الگوهای وابستگی ، توالی زمانی ، روابط نام و فضا را نشان دهند. اینها معمولاً حتی مسطح هم نیستند ، خیلی کمتر سلسله مراتبی دارند. در واقع ، یکی از راه های ایجاد کنترل مفهومی بر چنین ساختاری اجرای برش پیوند است تا زمانی که یک یا چند نمودار سلسله مراتبی پیدا کنند

3**- مشکلات پیشین که در گذشته حل شده است**

\*\* اگر سه مرحله در فناوری نرم افزار را بررسی کنیم که در گذشته بسیار مثمر ثمر بوده اند ، خواهیم فهمید که هرکدام از آنها به مشکل عمده دیگری در ساخت نرم افزار حمله کرده اند ، اما این مشکلات تصادفی بوده اند و نه اساسی

1. زبانهای سطح بالا

مطمئناً قدرتمندترین ضربه برای بهره وری ، قابلیت اطمینان و سادگی نرم افزار استفاده تدریجی از زبانهای سطح بالا برای برنامه نویسی بوده است. اکثر ناظران این پیشرفت را حداقل با ضریب 5 در بهره وری و همراه با قابلیت اطمینان ، سادگی و قابل فهم بودن اعتبار می دهند.

1. اشتراک زمان

نفهمیدم

1. محیط برنامه نویسی یکپارچه

آنها با تهیه کتابخانه های یکپارچه ، قالب های پرونده یکپارچه و لوله ها و فیلترها ، به مشکلات تصادفی استفاده از برنامه ها با هم حمله می کنند. در نتیجه ، ساختارهای مفهومی که اصولاً همیشه می توانند یکدیگر را فراخوانی ، تغذیه و استفاده کنند ، واقعاً به راحتی می توانند این کار را انجام دهند.

که باعث پیشرفت و توسعه ابزار برای یونیکس و لیپس (اولیا که اینو ایجاد کردند) شده

**4- به امید نقره**

\*\* پیشرفت ها رو میخواد به عنوان گلوله نقره بگیره ولی میپرسه اینا مشکلات ذاتی رو حل کردن یا تصادفی رو ؟

\*\* **زبان آدا**

\*\* **OOP**

**\*\* هوش مصنوعی (معنیشو میگه)**

**امروزه دو تعریف کاملاً متفاوت از هوش مصنوعی مورد استفاده مشترک است. AI-l: استفاده از رایانه برای حل مشکلاتی که قبلاً فقط با استفاده از هوش انسانی قابل حل بودند. AI-2: استفاده از مجموعه خاصی از تکنیک های برنامه نویسی معروف به برنامه نویسی ابتکاری یا مبتنی بر قاعده. در این رویکرد ، متخصصان انسانی مورد بررسی قرار می گیرند تا مشخص کنند که از چه روشهای ابتکاری یا قانونی در حل مسائل استفاده می کنند .... این برنامه برای حل مسئله ای طراحی شده است که به نظر می رسد انسان آن را حل می کند.**

**\*\* سیستم های خبره**