سخنرانی :2توسعه Embedded سیستم های بلادرنگ

سید حسین عطارزاده نیاکی

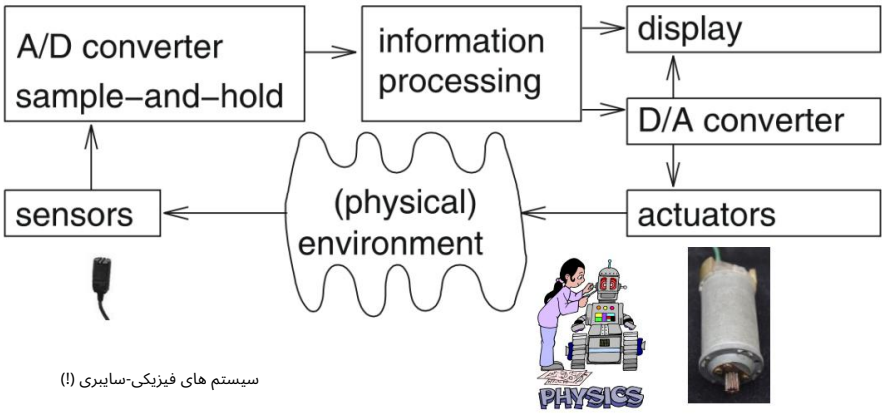
چند اسلاید از پیتر مارودل، ادوارد لی و فیلیپ کوپمن

بررسی کنید

- سیستم جاسازی شده/سایبری-فیزیکی چیست؟
- چرا آنها مهم هستند؟
- چگونه آنها را طراحی کنیم؟

سخت افزار CPSو ES

•سخت افزار CPSو ESاغلب در یک حلقه استفاده می شود ("سخت افزار در یک حلقه"):



سیستم های فیزیکی-سایبری (!)

سیستم های واکنشی و ترکیبی

•به طور معمول، CPSسیستم های واکنشی هستند :

-«سیستم واکنشی سیستمی است که پیوسته باشد تعامل با محیط است و با سرعتی که توسط آن محیط تعیین می شود اجرا می شود.
[برژ، [1995

-رفتار به ورودی و وضعیت فعلی بستگی دارد. مدل خودکار مناسب، مدل توابع قابل محاسبه نامناسب.

•سیستم های هیبریدی (قطعات آنالوگ +دیجیتال).

چالش های پیاده سازی در سخت افزار

- سیستم های تعبیه شده اولیه که اغلب در سخت افزار (بردها) پیاده سازی می شوند
- هزینه ماسک برای مدارهای مجتمع خاص برنامه های تخصصی (ASIC)بسیار گران می شود

(محدوده ،$M وابسته به تکنولوژی)

- عدم انعطاف پذیری (تغییر استانداردها).
- گرایش به پیاده سازی در نرم افزار (یا احتمالاً FPGA)

چالش های پیاده سازی در نرم افزار

اگر CPS/ESبیشتر در نرم افزار پیاده سازی می شود، پس چرا ما فقط از آنچه مهندسان نرم افزار ارائه کرده اند استفاده نمی کنیم؟

چالش های نرم افزار CPS/ES

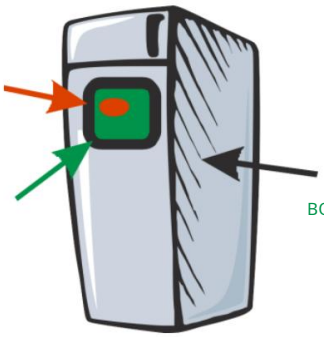
- محیط های پویا
- رفتار مورد نیاز را ثبت کنید!
- تایید مشخصات
- ترجمه کارآمد مشخصات به پیاده سازی!

- چگونه می توانیم بررسی کنیم که محدودیت های زمان واقعی را رعایت کرده ایم؟

- چگونه می توانیم در زمان واقعی تعبیه شده را اعتبارسنجی کنیم

نرم افزار؟ (حجم زیاد داده، آزمایش ممکن است برای ایمنی حیاتی باشد)

طوری رفتار کنید که انگار محصولات شما زندگی می کنند یا می میرند نرم افزار آنها



BOM = صورتحساب مواد

نرم افزار
0%از هزینه BOM
90درصد محصول
تمایز
سیستم مکانیکی 90%هزینه BOM

بیشتر کالایی
کنترل کننده الکترونیکی 10%هزینه BOM

بیشتر کالایی

پیچیدگی نرم افزار یک چالش است

- افزایش تصاعدی پیچیدگی نرم افزار

- 70 <درصد از هزینه توسعه سیستم های پیچیده مانند ... الکترونیک خودرو و سیستم های ارتباطی به دلیل توسعه نرم افزار است

[A. Sangiovanni-Vincentelli، 1999]

نرم افزار در یک تلویزیون

- منبع :*1

سال	سایز 0
1979	
1965	1کیلوبایت
1990	64کیلوبایت
2000	2مگابایت

- منبع 10 :°2برابر در هر 6-7سال

اندازه	
1986	1مگابایت
1992	2مگابایت
1998	4مگابایت
2008	8مگابایت

*به نقل از: Gerrit Rob van Ommering، COPA Tutorial،
مولر: فرصت ها و چالش ها در سیستم های تعبیه شده،
موسسه سیستم های جاسازی شده آیندهوون، 2004

R. Kommeren, P. Parviainen: تجارب فیلیپس در سطح جهانی
توسعه نرم افزار توزیع شده، Empir Software Eng. (2007) 12:647-660

تست محصول همه اشکالات را پیدا نمی کند

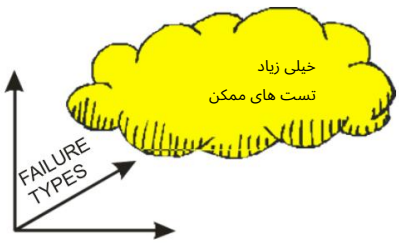
- یک سوم خطاها را می گیرد
بیش از 5000سال برای ظهور

آدامز، "NE، بهینه سازی خدمات پیشگیرانه محصول نرم افزاری"،
مجله تحقیق و توسعه، IBM، 28(1)، 1984. ص. 2-14(جدول
2، صفحه 9، ستون 60کیلومتری)

- تست نرم افزار بد
به سادگی آن را کمتر بد می کند
-آزمایش نمی تواند نرم افزار خوبی را در تمام مراحل تولید کند
خود

- مشتریان شما مرتباً با اشکالاتی مواجه می شوند که در طول آزمایش آنها را نخواهید دید

- برای اکثر محصولات، شما
حتی نمی توان 5000سال را آزمایش کرد



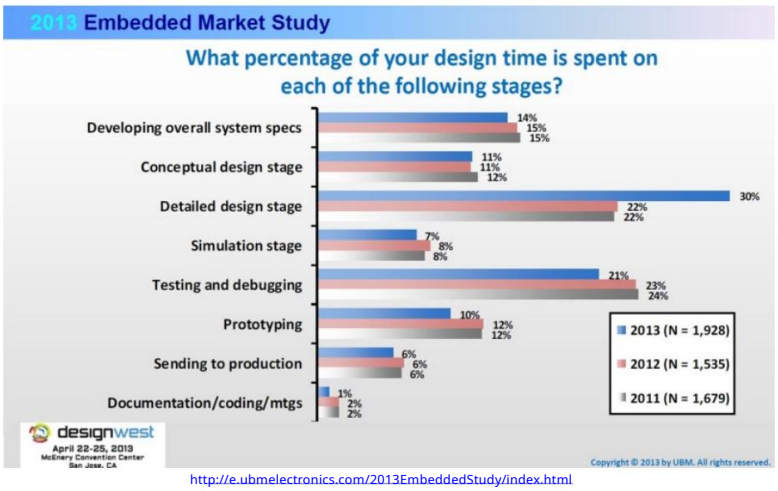
OPE R ATIONAL

توسعه نرم افزار

فرآیند

کد نویسی اساساً 0درصد است

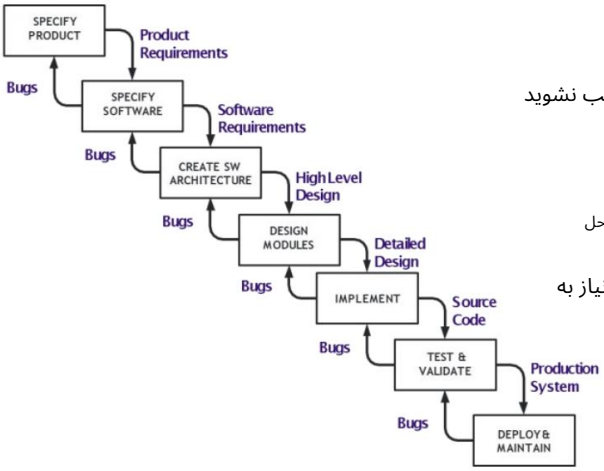
ایجاد نرم افزار



http://e.ubmelectronics.com/2013EmbeddedStudy/index.html

قدیمی مدرسه

چرخه توسعه آبشار



•موثر برای چاه
دامنه های قابل درک

-اگر اشتباهات بزرگ زیادی مرتکب نشوید
بهترین کار را دارد

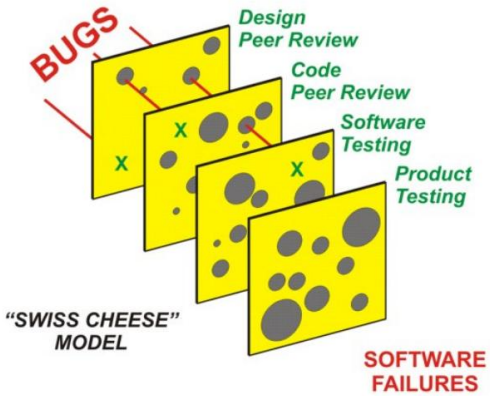
•تغییرات در سیستم های موجود

•گران قیمت برای تعمیر چیزهایی که به مراحل
آزمایشی فرار می کنند

-هر مشکلی که با آن مواجه شد نیاز به
عقب نشینی دارد

•توجه: آبشار اصلی
کاغذ این فلش های عقب را داشت! این
هرگز فقط یک فرآیند یک طرفه نبود

نحوه دریافت نرم افزار با کیفیت بالا



•تست محصول

-دیر و گران

-بسیاری از میدان ها فرار می کنند

•تست نرم افزار

–آزمون واحد و ادغام

•بررسی همتایان کد

-زودتر و ارزان تر

•بررسی همتایان طراحی

-زودتر و ارزان تر

آنچه در آن آموخته ایم

بیش از 50سال نرم افزار

•تقسیم به زیرسیستم ها حیاتی است

-معماری بد یک پروژه را نابود می کند

•رسمی بودن فرآیند سرمایه گذاری خوبی است

-قابلیت ردیابی، بررسی های رسمی و غیره

-پرش از مراحل در نهایت هزینه بیشتری دارد

•الزامات تغییر می کند

-استفاده از یک رویکرد تکراری را پیشنهاد می کند

•یافتن زودهنگام اشکالات مهم است

–قابلیت ردیابی از سطوح بالا به پایین

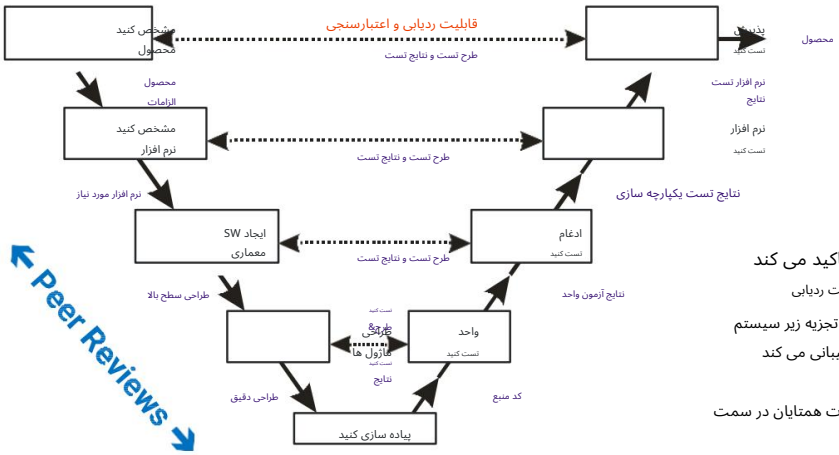
-تست لایه ای

-نظرات همتایان، به خصوص در سمت چپ V

اگر نیمه دوم پروژه "اشکال زدایی" است، باید به این معنی باشد که نیمه اول "اشکال" است.

•جک گانسل
http://www.ganssle.com/rants/ontesting.htm(پیش عبارت)

چرخه توسعه V(یا "Vee").



تاکید می کند
قابلیت ردیابی
از تجزیه زیر سیستم
پشتیبانی می کند

نظرات همتایان در سمت چپ V

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

http://agilemanifesto.org/principles.html

روش های چابک

Agile به طور کلی ارزش های زیر را دارد:

افراد و تعاملات بر روی فرآیندها و ابزارها نرم افزار کار بر روی اسناد جامع

همکاری مشتری بر سر مذاکره قرارداد پاسخ به تغییر در پیروی از یک برنامه

مثال: اسکرام

جلسات روزانه "استاند آپ" ("اسکرام") برای همکاری رو در رو

۲ تا ۴هفته دوی سرعت برای افزودن تدریجی عملکرد

-هر اسپرینت آیتم هایی را از یک بک لاگ اجرا می کند

-نسخه ی نمایشی در پایان اسپرینت؛ از نظر تئوری یک محصول قابل حمل است

داستان های کاربر به عنوان الزامات عمل می کنند

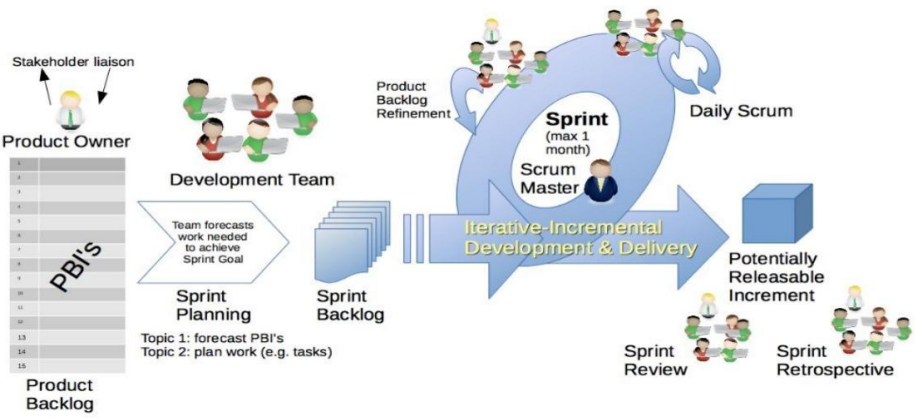
چالش های اسکرام

-تقسیم جغرافیایی تیم ها با ارتباطات غیر رسمی -وابستگی های خارجی (به عنوان مثال، سایر بخش های تغییر سیستم)

-زمانی برای آزمایش گسترده، به خصوص سخت افزار تعبیه شده وجود ندارد

مثال فرآیند اسکرام

•دانش ضمنی بسیار زیاد –آیا می توانید موارد مورد نیاز، طراحی، طرح آزمون، آزمون پذیرش را بیابید؟



روش های چابک +جاسازی شده (؟)

•مزیت قابل توجه این است که توسعه دهندگان (خوب) را خوشحال تر می کند
-اگر به خوبی انجام شود می تواند به نیازهای در حال تحول کمک کند
-اما، شما باید خطرات را مدیریت و تعدیل کنید

•مسئله: "چابک" فقط کد نویسی کابوی نیست
-فرآیندهای تعریف نشده و بی انضباط خبر بدی هستند
-بله، تیم های چابک باید یک فرآیند دقیق تعریف شده را دنبال کنند

•موضوع: چابک "بدون کاغذ" برای سیستم های با عمر طولانی نامناسب است
-دانش ضمنی کارآمد است، اما با تیم تبخیر می شود
–سیستم‌های قدیمی غیرقانونی بیش از 10سال یک کابوس هستند

•مسئله: Agileآزمون پذیرش 100%خودکار را فرض می کند
-تست سیستم 100%خودکار اغلب برای رابط های فیزیکی غیرعملی است
-اغلب به طور ضمنی فرض می کند که فرار از نقص هزینه کم است زیرا جدید است نسخه 2-4هفته دیگر است

•مشکل: Agileمعمولاً نظارت بر فرآیند مستقل (SQA)ندارد .
-تضمین کیفیت نرم افزار (SQA)به شما می گوید که آیا فرآیند شما کار می کند یا خیر
-تیم های چابک ممکن است ناکارآمد باشند و تصوری از این اتفاق نداشته باشند
•یا ممکن است حالشان خوب باشد –اما چه کسی می داند که آیا واقعاً سالم هستند یا نه؟

چه زمانی Agileمناسب است؟

منبع: Boehm & Turner 2004, Balancing Agility and Discipline

برنامه محور (سنتی)	چابک
•تیم های بزرگ. محصولات بزرگ	•تیم های کوچک. محصولات کوچک
•محصولات ماموریت حیاتی	•کیفیت نرم افزار " Everyday "
•الزامات پایدار	•نیازهای سریع تغییر می کند
•مهارت بالا در درجه اول در مرحله طراحی	•کارشناسان با مهارت بالا در طول پروژه
-نسخه های اصلی نیاز به طراحی متخصص دارند	-از جمله نگهداری چرخه عمر
•اکثر توسعه دهندگان اینطور نیستند توانمند معمولا جوان	•توسعه دهندگان می توانند رسیدگی کنند توانمند شدن؛ معمولا ارشد

بهترین روش ها برای فرآیند نرم افزار

•یک فرآیند تعریف شده را دنبال کنید -باید شامل تمام جنبه های نشان داده شده در Veeباشد •و ،SQAبررسی های همتا -
تغییر نام و سازماندهی مجدد مراحل مشکلی ندارد •تمام مراحل باید انجام شوند •برای دیدن "AgileFall"و غیره معمول است.
فرآیندی که با جدیدترین کلمات کلیدی پوشیده شده است

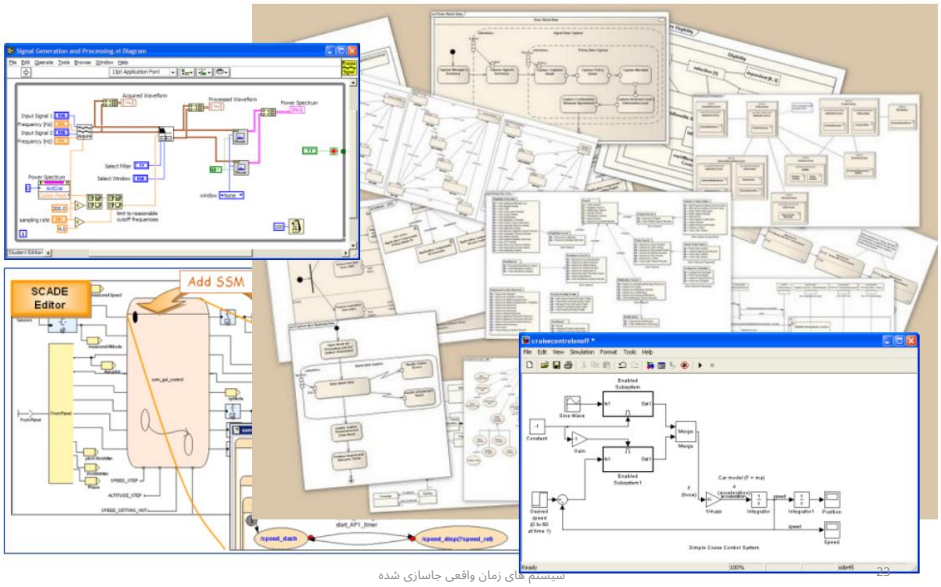
•مشکلات فرآیند نرم افزار

-پرش از مراحل برای رسیدن به تست سریعتر به معنی اشکالات بیشتر در تست است
•یافتن اشکالات در آزمایش هزینه بیشتری دارد

-استفاده از فرآیند اشتباه برای هدف نادرست
•عمر محصول 3هفته و عمر محصول 30سال شرایط متفاوتی هستند

طراحی مبتنی بر مدل

روی مدل ها تمرکز کنید



انگیزه برای

در نظر گرفتن مشخصات و مدل ها

- چرا مشخصات و مدل ها را با جزئیات در نظر می گیریم؟

-اگر مشکلی در مشخصات وجود دارد،
در این صورت درست کردن طرح مشکل خواهد بود و احتمالاً زمان زیادی را هدر می دهد.

- به طور معمول، ما با مدل های سیستم در حال طراحی (SUD)کار می کنیم.
- به هر حال مدل چیست ؟

مدل ها

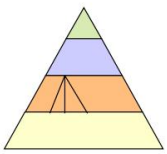
تعریف: مدل ساده سازی موجودیت دیگری است که می تواند یک چیز فیزیکی یا مدل دیگری باشد. مدل دقیقاً شامل آن ویژگی ها و ویژگی های موجودیت مدل شده است که برای یک کار معین مرتبط است. یک مدل

در صورتی که دارای ویژگی های دیگری غیر از ویژگی های مربوط به کار نباشد، نسبت به یک کار حداقلی است .

[یانچ، 2004]

چه الزاماتی برای مدل های خود داریم؟

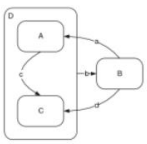
سلسله مراتب مدل



•انسان ها قادر به درک سیستم های حاوی بیش از 5شیء نیستند.

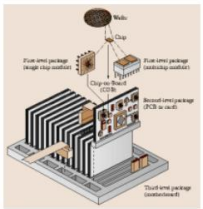
اکثر سیستم های واقعی به اشیاء بیشتری نیاز دارند

سلسله مراتب + انتزاع)



-سلسله مراتب رفتاری
مثال ها: حالت ها، فرآیندها، رویه ها.

-سلسله مراتب ساختاری
به عنوان مثال: پردازنده ها، قفسه ها، برد مدار چاپی



طراحی مبتنی بر کامپوننت

- سیستم ها باید از اجزاء طراحی شوند
  - مخصوصا برای طراحی معماری
- باید «آسان» رفتار کرد
  رفتار زیرسیستم ها
- همزمانی
- همگام سازی و ارتباط

زمان در مدل ها

- رفتار زمان بندی
  برای سیستم های جاسازی شده و cy-phy ضروری است!

-از اطلاعات اضافی (دوره ها، وابستگی ها، سناریوها، موارد استفاده) استقبال می شود

-همچنین سرعت سکوی زیرین باید مشخص باشد

-پیامدهای گسترده برای فرآیندهای طراحی!

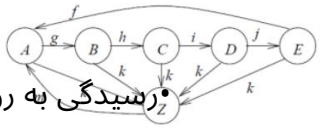
«مفهوم زمان بدون ابهام (که در علوم کامپیوتر) یک نقص است، از منظر علم نرم افزار» [لی، 2005]

«مفهوم زمان بدون ابهام (که در علوم کامپیوتر) یک نقص است، از منظر علم تعبیه شده) یک نقص است، از دیدگاه نرم افزارهای جاسازی شده" [لی، 2005]

پشتیبانی مدلسازی برای طراحی سیستم های راکتیو

- رفتار دولت محور

مورد نیاز برای سیستم های واکنشی؛ اتوماتای کلاسیک کافی نیست



- رفتار • رسیدگی به رویداد (رویدادهای خارجی یا داخلی)
استثنا محور

توصیف استثناها برای هر ایالت قابل قبول نیست

سایر الزامات مدلسازی

•قابلیت اجرا (نه تنها مشخصات جبری) •پشتیبانی از طراحی سیستم های بزرگ • (?OO)پشتیبانی از دامنه خاص •خوانایی •قابلیت حمل و انعطاف پذیری •پشتیبانی از دستگاه های I/Oغیر استاندارد •ویژگی های فوق العاده •پشتیبانی از طراحی سیستم های قابل اعتماد •عدم وجود مانع برای پیاده سازی کارآمد •مدل های محاسباتی کافی به چه معناست؟

مشکلات با نظریه CSکلاسیک و محاسبات فون نویمان (رشته).

- حتی مفهوم اصلی ... "قابل محاسبه" با الزامات نرم افزار تعبیه شده در تضاد است.
- در این مفهوم، محاسبات مفید خاتمه می یابد، اما خاتمه می یابد غیر قابل تصمیم گیری است
- در نرم افزارهای تعبیه شده، خاتمه یک شکست است، و هنوز برای بدست آوردن زمان قابل پیش بینی، محاسبات فرعی باید به طور قطعی خاتمه دادن

آنچه مورد نیاز است تقریباً اختراع مجدد علوم کامپیوتر است.

Positively on Time، IEEE Computer، Edward A. Lee: Absolutely ژوئیه، 2005

موضوع غیر فون نویمان غیر رشته‌ای، غیرمبتنی بر MoCs فون نویمان

قاطعیت

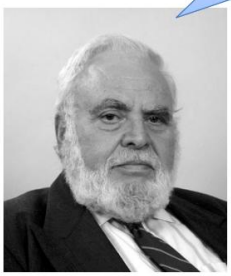
برخی از با ارزش ترین مدل ها هستند قطعی

یک مدل قطعی است اگر با توجه به حالت اولیه و ورودی ها، مدل دقیقا یک رفتار را تعریف کند.

مدل‌های قطعی در گذشته بسیار ارزشمند بوده‌اند.

مهندسان اغلب مدل را با هدفش اشتباه می گیرند

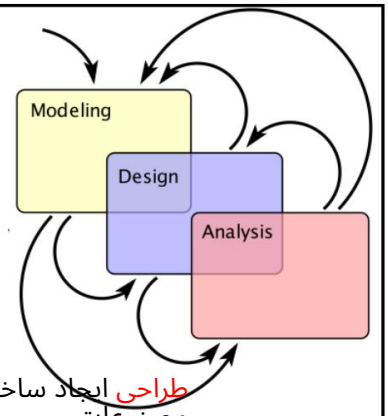
شما هرگز با حفاری از طریق نقشه
به نفت ضربه نخواهید زد!



Solomon Wolf Golomb

اما این به هیچ وجه از ارزش یک نقشه کم نمی کند!

مدلسازی، طراحی، تحلیل
یک فرآیند تکرار شونده



مدل سازی فرآیند به دست آوردن درک عمیق تر از یک سیستم از طریق تقلید است.

مدل ها بیانگر چه سیستمی هستند
انجام می دهد یا باید انجام دهد.

طراحی ایجاد ساختار یافته از مصنوعات
مشخص می کند که یک سیستم چگونه کاری را که انجام می دهد انجام می دهد.

تجزیه و تحلیل فرآیند به دست آوردن درک عمیق تر از یک سیستم از طریق تشریح است.
مشخص می‌کند که چرا یک سیستم کاری را که انجام می‌دهد انجام می‌دهد (یا آنچه را که یک مدل می‌گوید انجام نمی‌دهد) انجام نمی‌دهد.

سخنرانی بعدی

- الزامات CPS و ES
  - الزامات عملکردی
  - الزامات فوق عملکردی
- تجزیه و تحلیل و مشخصات مورد نیاز