Embedded S1 ۲۴ شهریور

* تعریف Embedded System ←

← پیتر مارودل ← سیستم های امبدد، سیستم هایی هستند که پردازش اطلاعات انجام می دهند و درون یک محصول بزرگتر تعبیه شده اند.

← ادوارد لی ← سیستم های امبدد نرم افزارهایی هستند که با یک فرآیند فیزیکی ترکیب شده اند. شغل اصلی در این جا زمان و همزمانی است

* Cyber-Physic ← ترکیبی از دنیای مجازی و دنیای فیزیکی

← Embedded System + Physical Prcs

← سیستم های فیزیکی و مهندسی شده که توسط یک هسته محاسباتی کنترل و مانیتور شده و کامپوننت ها شبکه شده اند

در محاسبات به صورت نهفته در عناصر فیزیکی انجام می شود. سیستم نهفته یک سیستم Real-time و توزیع شده است و رفتار آن ترکیبی از محاسبات و ویژگی های فیزیکی است

Subject: Date: Day: Time:

* سیستم های نهفته زیر مجموعه ای از سایبر فیزیک ها هستند

* سایبر سیستم ها از سه بخش تشکیل شده اند

Control ← برای تاثیر گذاری روی دنیای فیزیکی

Computation ← پردازش اطلاعات برای تصمیم گیری

Communication ← تبادل اطلاعات برای همکاری (سیستم توزیع شده است)

* سایبر فیزیک ها دارای چنین سیستم های زیر مجموعه است مثل Smarter Plantet, IOT

M2M, Industry4 و غیره است که هر کدام روی جنبه های مختلفی تمرکز دارند

* Real-Time ← یعنی جواب بستگی به لحظه پردازش دارد.

* Smart Grid ← شبکه هوشمند توزیع برق

* در هنگام ساخت سیستم های سایبرفیزیک تناقض هایی بوجود می آید

که باید بین دو مورد مصالحه برقرار کنیم. مثلا

وفقی تکرار شونده - ارتباط بالا امنیت حریم شخصی - کارایی بالا انرژی مصرفی

[illegible] همکاری - مقیاس پذیر قابل اعتماد - فراهم محدودیت منابع

Subject: Date: Day: Time:

* فضای IT ← به سه بخش تقسیم می‌شود 1) زیرساخت (پردازش، ارتباط و ذخیره‌سازی) 2) دیوایس‌ها (موبایل، لپ‌تاپ، شبکه‌های پوشیدنی) 3) سنسورها و اکتورها

* Embedded Software is important and challenging

* جنبه‌های مهم نرم‌افزار امبد ← کیفیت، امنیت، ایمنی

* نرم‌افزار خوب به 1) افراد تراز 2) ابزار مناسب 3) فرایند توسعه درست نیاز دارد.

* باقی ماندن عیب‌ها در نرم‌افزار باعث پایین آمدن کیفیت محصول یا نتیجه

فرایند مناسب است.

* طراحی سیستم امبد ←

Modeling ← فهم چیستی نرم‌افزار what

Design ← طراحی و توسعه How

Analysis ← تحلیل Why

Subject: Date: Day: Time:

Embedded 32 ۲۹ شهریور

* CPS ←

تبدیل دیجیتال به آنالوگ ← پردازش ← تبدیل آنالوگ به دیجیتال

↑ ↓

عملگرها ← محیط ← سنسورها

* CPS ها سیستم‌های Reactive هستند ← سیستم‌های Reactive

سیستم‌هایی هستند که پیوسته با محیط در تعامل است و در نتیجه با سرعت محیط هماهنگ است و با آن سرعت کار می‌کند. مفهوم State ها در این‌ها مهم و شمارشی است

همچنین CPS ها Hybrid هم هستند یعنی ترکیبی از آنالوگ و دیجیتال

* اوایل ES ها روی سخت‌افزار پیاده‌سازی می‌شدند اما امروزه به خاطر هزینه بالا برای تولید یک سخت‌افزار اختصاصی و انعطاف‌پذیری کم به سمت نرم‌افزار حرکت می‌کنیم. البته FPGA ها سعی در حل این مشکل را دارند.

Subject: Date: Day: Time:

* مشکلات پیاده سازی نرم افزاری CPS ← محیط متغیر ، نمی توانیم دقیق بیان کنیم سیستم باید چیکار کند ، اعتبارسنجی توصیف سخت است ، پیاده سازی توصیف ها نقض شدن از این که Real time است. چگونه نرم افزار را بسنجیم

* تست کردن نمی تواند تمام bug های سیستم را آشکار کند و bug ها در طول زمان و هنگام استفاده خودشون رو نشون می دهند. تست کردن یک نرم افزار بد فقط اون رو عقب تر می کند و باعث بهتر شدن نمی شود.

* پیاده سازی Waterfall ←

توصیف محصول ← توصیف نرم افزار ← ساخت معماری نرم افزار ← طراحی ماژول ها ← پیاده سازی ← تست و اعتبارسنجی ← تحویل

مشکل این روش این است که خیلی دیر تست می شود و اگر باگی وجود داشته پیدا کردن و رفعش آن مشکل است و بیشتر برای پروژه های مناسب است که خیلی بزرگ نیستند و محدوده پروژه مشخص است

Subject: Date: Day: Time:

فرآیند توسعه را به چند مرحله تقسیم ← Swiss Cheese *

می‌توانیم مرحله‌های مختلف را ست کنیم.

Design Peer Review _ Code Peer Review _ Software Testing

Product Testing

* ساخت هم در توسعه نرم‌افزار ←

بخش‌های بزرگ و پیچیده را به زیربخش‌های کوچک‌تر تقسیم کنیم

فرآیند توسعه نرم‌افزار باید سازوکار خوب و پایه‌ای داشته باشد

نیازمندی‌ها تغییر می‌کند. سعی کنیم باگ‌ها را هرچه زودتر پیدا کنیم.

* مدل V ←

توصیف محصول ← → Product

توصیف نرم‌افزار ← → Software

معماری نرم‌افزار ← → Integration

طراحی ماژول ← → Unit test

پیاده‌سازی

AVANGE

Subject: Date: Day: Time:

* Agile + Embedded ←

مشکلات ← بدون قاعده و قانون کردن، نبودن مستندها برای پروژه‌های طولانی مدت، تست پذیرش را حذف می‌کند. SQA نداریم

* Agile ← برای تیم‌ها و پروژه‌های کوچک برای دقت کیفیت روزانه نرم افزار خوبه برای وقت‌هایی که نیازمندی‌ها مدام در حال تغییر است. برای افراد حرفه‌ای

AVANGE

Subject: Date: Day: Time:

Embedded 33) ۳۱ شهریور

* Model ← یک یا مدل سازی (حذف جزئیات و نگه داشتن ویژگی های لازم برای انجام یک کار خاص) از یک موجودیت دیگر می تواند یک موجودیت فیزیکی یا یک مدل دیگر باشد. ← مثل سیمولیشن شده‌ی یک کار خاص

* انسان نمی تواند همزمان به بیش از 5 مورد فکر کند در نتیجه استفاده از Hierarchy مهم است

← Hierarchy ← Structural ← سلسله مراتبی که در یک ساختار اتفاق می افتد مثلا پردازنده ← هسته CPU ← ترانزیستورها

← Behavioral ← سلسله مراتبی که در حالت‌های فرآیند ها یا در Function ها اتفاق می افتد مثلا توابع مختلف در main یا برنامه

* Component-Based ← تجزیه یک مدل به چندین کامپوننت که به صورت همروند اجرا می شوند و باهم همگام سازی انجام می دهند و اطلاعات را بین خودشان انتقال می دهند

Subject: Date: Day: Time:

* مدل ها باید Reactive باشند یعنی پیوسته ورودی بگیرند و نتیجه بدهند

State-Oriented Behavior

Event handling

Exception Oriented Behavior

* ویژگی های مدل

قابل اجرا باشند. سیستم های بزرگ و پیچیدگی کنند. طراحی مختص حوزه باشد

خاص، عوامل، I/O های غیر استاندارد را پشتیبانی کنند، ویژگی های Non-Functional داشته باشد، پشتیبانی از مدل محاسباتی ما

* Determinstic یکی از ویژگی های مدل های خوب قطعی بودن است

یعنی اگر در یک حالت مشخص بارها و بارها یک ورودی بیان بدهیم مدل

یک خروجی بیان بدهد

* پیاده سازی باید رفتار مدل ها را حفظ کند

AVANGE

Subject: Date: Day: Time:

Requirement *

← Functional ←

1) Data Collection → جذب اطلاعات از محیط پیرامون

2) Direct digital Control → اعمال خودکار روی محیط پیرامون

3) man_machine interaction → برقراری ارتباط با انسان ها

← Non-Functional ←

← Real-time ← خروجی درست را در بازه درست برگرداند

معمولا برای deal line در نظر گرفته می شود

تقسیم می شوند ←

← Soft ← از دست دادن داده

قابل قبول است وقتی سیستم اضطراری نیست

AVANGE

Subject: Date: Day: Time:

Firm ← از دست دادن ددلاین ←

قابل قبول است و سیستم را نابود نمی‌کند

تبدیل می‌شود

hard ← از دست دادن ددلاین نابود

فاجعه به بار می‌آورد.

(Time Req.)

* سیگنال‌ها باید time را در نظر بگیرند نه معمولی و هیچ سیگنال مختلف اتفاق می‌افتد

Measure ← فاصله بین دو اتفاق را می‌سنجد

Delaying ← برای تأخیر انداختن

Timeout ← یک مدت زمانی را منتظر یک اتفاق بشو و بعد از آن

واکنش را انجام بده

Deadline ← Soft, Firm, hard

AVANGE

Subject: Date: Day: Time:

← Efficiency Requirement *

Code-Size - Run-time (Performance) - Weight

Cost - Energy

← Dependability Req *

← Reliability ← R(t) ← احتمال این که سیستم در لحظه t در دسترس باشد

← Maintainability ← M(d) ← احتمال درست کار کردن سیستم در بازه زمان d بعد از وقوع یک ارور (Recovery)

← Availability ← یکی از زمان‌ها که سیستم در دسترس است

← Safety ← آسیبی به بدنی وارد نکند (ایرادات داخلی)

← Security ← جلوگیری از حملات خارجی

AVANGE

Subject: Date: Day: Time:

* نیازمندی ها را همچنین می توان به دو دسته زیر تقسیم کرد ←

← Emergent ← مربوط به کل سیستم مثل Functional ها Non-functional ها

کارهایی که سیستم باید انجام بدهد

← Design Constrains ← استانداردها و پروتکل هایی که باید در پروژه

رعایت کرد. مثلا زبان و بودجه پروژه

* CPS Req ←

Marketing	Product	Software
↓	↓	↓
مربوط به مشتری	توصیف دقیق تر	چگونه به اهداف برسیم

* نیازمندی خوب چه ویژگی هایی دارد؟ ← خلاصه و کوتاه باشد، کمترین قید ممکن روی آن اعمال شده باشد (فقط باید شرح این باشد چه کاری باید انجام دهد نه توضیح دقیق این که چگونه باید این کار را انجام داد)، دقیق باشد همراه با یک تلورانس، نیازمندی ها را باید شماره گذاری کرد که قابل ردیابی باشند، قابلیت تست کردن باید داشته باشند، نیازمندی ها باید منطقی باشند در صورت لزوم دلیل آورده شود،

Subject: Date: Day: Time:

بدون ابهام باشد و نیازمندی ها نباید با هم دیگر تناقض داشته باشند

* روش های نوشتن نیازمندی ها ← فایل متنی یکی از نیازمندی ها (1) (2)

Agile user Stories (3) UML use case (2)

Funetional Decomposition (5) Prototyping (4)

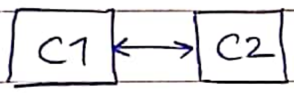
* نیازمندی خوب Six-C ← شفاف و واضح و مختصر و

درست و سازگار (بدون تناقض) و قابل تعامل و قابل تست

Subject: Date: Day: Time:

Embedded S4) 5 سر

* Models of Computation (مدل های محاسباتی) ← تشکیل شده از یک سری کامپوننت که مشخص می کند هر کدام موقع اجرا چه کاری انجام می دهند که به وسیله یک سری از فلش ها به هم متصل شده اند که این فلش ها مشخص می کنند ارتباطات چگونه انجام می شود و چه اطلاعاتی بین کامپوننت ها رد و بدل می شود

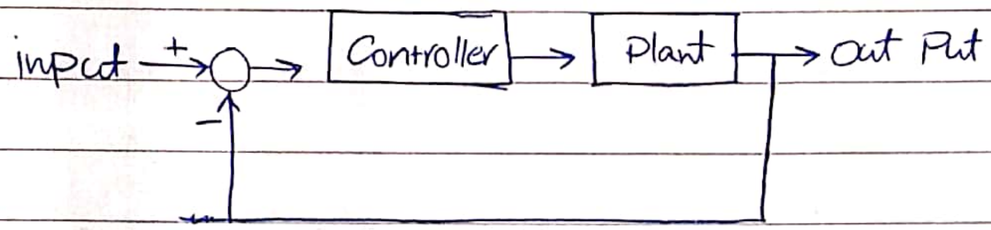


* مدل ها برای بررسی سیستم های داینامیک استفاده می شوند و به دسته بندی
1) داینامیک پیوسته 2) داینامیک گسسته 3) هیبرید تقسیم کرد
4) همروند

* مثال هلیکوپتر در اسلایدها

Subject: Date: Day: Time:

Feed back Control *



Plant ← محیطی که قصد کنترل آن را داریم (مثلا یک اتاق که می‌خواهیم دمای آن را تنظیم کنیم)

Controller ← وظیفه اصلاح کردن میزان خطا را دارد

میزان خطا ← و مقدار OutPut (در این مثال دمای فعلی محیط)

مقدار input یا مقدار مورد نظر که به صورت ورودی به سیستم داده می‌شود

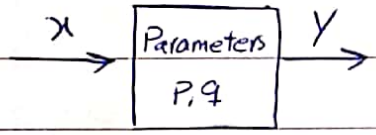
اختلاف مقدار OutPut و input برابر است با میزان خطا

* پدیده های فیزیکی توسط متخصص های حوزه مورد نظر مدل شده و به ما داده می‌شود که بعضی اوقات به خاطر پیچیدگی ها نیاز به ساده سازی دارد به این ساده سازی Model order reduction می‌گویند

Subject: Date: Day: Time:

← Actor Model of System *

سیستم یا تابع است که یک سیگنال ورودی می‌گیرد و یک سیگنال خروجی تولید می‌کند



← Composition of Actor Model *

حتی اوقات یک سیگنال پیچیده می‌تواند ترکیبی از سیگنال‌های ساده‌تر باشد

* Actor Model ها می‌تواند چند ورودی چند خروجی نیز داشته باشند

* ویژگی‌های سیستم ←

← Casual ← یعنی محاسبه خروجی به مقدار ورودی لحظه فعلی و لحظات قبلی نیاز داریم.

← Memory less ← مقدار خروجی فقط وابسته به مقدار ورودی در همان لحظه است نه لحظات قبلی

AVANGE

Subject: Date: Day: Time:

← linear and time-invariant ←

linear → $S(ax_1 + bx_2) = aS(x_1) + bS(x_2)$

Time-invariant → $S(D(x)) = D(S(x))$

$D()$ ← تابع delay

Stable ← ورودی کرانه‌دار به خروجی نیز کرانه‌دار باشد ←

← Proportional Controller *

در این جا کنترلر (فیدبک کنترل) فقط میزان خطا (input - output) را در یک ضریب ثابت ضرب می‌کند

Subject: Date: Day: Time:

Embedded 85) جلسه 7

High Level Design ←

توصیفی است از معماری (اسم‌ها) (ساختارهایی که وظیفه دارند نیازمندی‌ها را برآورده کنند) و نیازمندی (فعل‌ها) (چه کارهایی باید انجام دهد)

High Level Design = Architecture + Requirement

ADL ← Architecture Description language

برای توصیف معماری به یک زبان نیاز داریم که 3 دسته‌بندی زیر تقسیم می‌شوند:

1) box and line ← یک روش غیررسمی برای توصیف سطح بالا

که مستقل‌ها نشان‌دهنده objectها و مفاهیم نشان‌دهنده روابط هستند. که در کاربردهای مختلف معنی این مستقل‌ها و مفاهیم فرق می‌کند و باید از قبل تعیین شود

Subject: Date: Day: Time:

2) زبان های رسمی مثل AADL، EAST-ADL، EADL که برای آکادمیک بیشتر استفاده می‌شوند و تروط صنعت خیلی پذیرفته شده نیستند.

3) UML-Based ← استاندارد سازی شده عوده های مختلف است ولی توصیف دقیق از معماری دارد

* عوده box and line می‌تواند برای عوده معماری های مختلف استفاده شود مثل نرم افزار، سخت افزار، شبکه، فلوچارت، Call graph و ...

* مثال Vending Machine ، GPS در اسلایدها

* UML use case ← روش برای نوشتن نیازمندی ها

چند بازیگر یا actor دارد به این actor ها می‌تواند کارهای مختلفی با سیستم انجام دهند این کارها را use case می‌نویسند.

AVANGE

Subject: Date: Day: Time:

* روش دیگر نوشتن نیازمندی‌ها نوشتن متنی دقیق است. مناسب است در نوشتن متنی باید توجه کرد که نقل‌ها تفکیک شده و معنای مشخصی دارند. نیازمندی‌ها را شماره‌گذاری می‌کنند تا Traceability افزایش پیدا کند بعد با استفاده از ماتریس این نیازمندی‌های شماره‌گذاری شده را به Use case های مختلف مرتبط می‌کنیم.

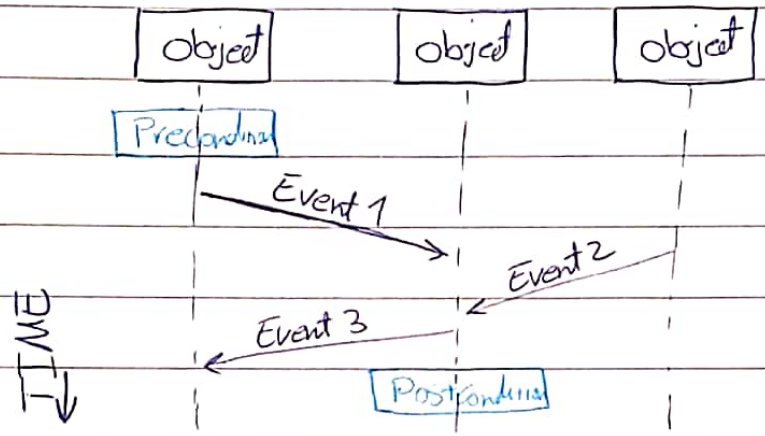
* نمودار class Diagram را نیز می‌توان برای توصیف معماری مورد استفاده قرار داد.

* Sequence Diagram ←

نمودارها و ابزارهایی که تا الان برای توصیف معماری (HLD) معرفی شدند شکل نداشتن زمان و ترتیب زمانی را دارند برای همین از این نمودار استفاده می‌کنیم.

AVANGE

Subject: Date: Day: Time:



پیش شرط‌هایی که باید برقرار شود تا یک ← PreCondition ←

شروع به کار کند object

شرایطی که بعد از انجام شدن در یک object ← PostCondition ←

به وجود می‌آید

* هر UseCase تبدیل به یک سناریو می‌شود و با چند SD

می‌توان یک سناریو را نمایش داد

* مقصود توصیف HLD باید توجه داشته باشیم که تعامل‌ها و ارتباطات بین هندسه‌ها

را تعیین کرده باشیم و این طراحی سطح بالا انجام می‌دهیم به دور از جزئیات و

نمی‌باشد.

AVANGE