

Deeper 🚫

- Why does it matter if our architecture “anticipates” future changes?

If the architecture anticipates a future change, it may be possible to add that feature without having to make many changes to the already written software.

If the architecture does not reasonably anticipate a future change, we may find that we have to rewrite a great deal of old code in order to add a new feature.

10/1/2009

System Architecture (how)

34

Deeper 🚫

- Give examples of poorly implemented features, and how we could abstract around them.

Fixed size tables ... have a method to enumerate them or return their size.

Inefficient list implementation ... hide the internal structure and provide well general operations for insertion, deletion, traversal.

Poor scheduling algorithm ... ensure you have enough information to enable better scheduling, but implement only the simpler algorithm.

10/1/2009

System Architecture (how)

35

Deeper 🚫

- Do all unsupported features leave design scars?

No. Some features are relatively orthogonal to all other functionality, and can be added later with no significant design changes.

- Suggest unsupported features that we might want to architect for from the start.

If we expected to support additional audio formats, we should design around plug-in audio format decoders ... even if we will only start with one static choice (e.g. mp3).

10/1/2009

System Architecture (how)

36

Deeper 😊

- How might changing OS affect an application program?

A different OS might support slightly different semantics for file I/O or inter-process communication.

- How would we abstract around this?

Design a wrapper layer that provides functionality that we can easily implement atop any of the target OSs, and write the rest of our application on top of that layer.

10/1/2009

System Architecture (how)

37

Deeper 😊

- How might changing our ISA affect applications programs (written in a portable higher level language)?

A different ISA might use a different word length and/or byte order, which would affect on-disk/over-wire data representations.

- How would we abstract around this?

Design a standard external data representation (XDR), and provide routines to convert between the internal and canonical representations.

10/1/2009

System Architecture (how)

38

Deeper 😊

- How might supporting other languages affect an applications program?

All displayed character strings must be translated into other languages.

- How would we abstract around this?

Extract all displayable strings into a run-time loadable module.

Create a translation layer that finds a desired string, based on a key.

Change all displayable-string handling code to use these abstractions, and not to assume that it can construct (or deconstruct) sentences from/to pieces.

10/1/2009

System Architecture (how)

39

Deeper

- What else might have to change for a program that had to run in other countries?
Time, date, number, and currency formats.
Character set, and width (7, 8, or 16 bit)
Non-ASCII collating sequences for letters.
Input editing widgets (left-right vs. right-left)

10/1/2009

System Architecture (how)

40

Deeper

- Give a examples of policy decisions
Who is allowed to perform which operations
Default file protections
Mail delivery options
Program to use as a text editor
Time-out intervals
Error response (abort, retry, fail this operation)
Scheduling/priority algorithms

10/1/2009

System Architecture (how)

41

Deeper

- Give examples of policy configuration mechanisms
Preference menus and files
Environment variables
User provided rule files (e.g. Makefile)
File protections and Access Control Lists
Classing engines
Strategy selection APIs
System registry
Application specific configuration files

10/1/2009

System Architecture (how)

42

Deeper

- What does XP say about pursuing general solutions?
They do advocate good design, but ...
The XP value of simplicity argues you should write code for today, rather than tomorrow.
They recommend refactoring to improve code after you better understand the way it will be used.

10/1/2009

System Architecture (how)

43

Deeper

- Architecting for the future and building for today would seem to be antithetical. Is this really the case?
XP has no problem with designing a general solution, and then implementing (today) only the subset that you need (today). What it cautions against is building a lot of software that you don't yet need.
XP does warn us about putting too much work into hypothetical features. Weigh the expected costs and benefits ... and avoid speculative investments.

10/1/2009

System Architecture (how)

44

Deeper

- What do I mean by “constructive decomposition”
It does not already exist in parts that we can remove and study (like a dissection or reverse engineering).
We must imagine a sequence of possible decompositions.

10/1/2009

System Architecture (how)

45

Deeper

- Why is this process usually iterative

Because our initial proposals often fail, but in studying why and how they failed, we learn more about the problem and various approaches to it.

They are not merely a succession of guesses, but each should be informed by the failures of the previous attempts.

This learning iteration is often a key part of the process.

10/1/2009

System Architecture (how)

46

Deeper

- What do I mean by information, potency, and slack

Information – enough information is available when the component is invoked to permit intelligent decisions.

Potency – the component occupies a position from which it is possible to effect the desired actions.

Slack – the specified interfaces do not over-constrain the component's implementation.

10/1/2009

System Architecture (how)

47