

Offset	Topic
00:17	<ul style="list-style-type: none"> • Intro <ul style="list-style-type: none"> • ADDCast #47 <ul style="list-style-type: none"> • http://addcast.net • This was the reason my voice was rough
02:43	<ul style="list-style-type: none"> • Word of the Week: considered harmful <ul style="list-style-type: none"> • http://www.catb.org/jargon/html/C/considered-harmful.html
04:22	<ul style="list-style-type: none"> • Inner Chapter: Over-Engineering <ul style="list-style-type: none"> • Few things get to me like over engineered code <ul style="list-style-type: none"> • Difficult to use <ul style="list-style-type: none"> • Involves way too many parameters • Typically exposes too much about the internal workings • Exposure requires implicit knowledge • Difficult to maintain <ul style="list-style-type: none"> • Often brittle • Slight changes cause unexpected side effects • Even tracking down a piece code responsible for some feature is an effort • What is over-engineered? <ul style="list-style-type: none"> • Typically tries too hard to be general • Violates single responsibility in the sense of trying to solve too many problems • How do you recognize it? <ul style="list-style-type: none"> • Too many parameters in methods, functions <ul style="list-style-type: none"> • Re-factoring to introduce a parameter object can help • Often exposes a concept that was over-looked • Some shared state <ul style="list-style-type: none"> • Is that state unique to the code in question? • If not, can probably be abstracted away into another package or library • If so, how much does a caller really need to know versus internal concern? • Often calls have to be "just so" <ul style="list-style-type: none"> • Argument values seem arbitrary • Certain combinations don't work • This often begs for some defaults • Pick what will serve for 80% of the time and remove those arguments or pieces of state • Violates locality of reference <ul style="list-style-type: none"> • You have to travel more than one or two frames to find a declaration

- Argument, state is initialized by some unrelated, remote code
- Address this by copying existing state into local type
- Copy construction can enforce assumptions, raise local errors
- Again think about that shared state. is it hap hazard or intentional?
- Nothing wrong with promoting state objects to their own concern
- Think about various calling context's, like requests and sessions
- Too many interfaces
 - An interface is very helpful for stipulating a contract with no constraint on implementation
 - One or two is often enough
 - More can be confusing as interface don't speak as clearly to cooperations
 - Provide default implementations of interfaces with reasonable default behavior
 - Just because some aspect of your code can be extended doesn't mean it should
 - What is truly an external concern?
 - What is part of the encapsulated logic?
- Things to keep in mind
 - What is the most simple design, implementation that does the job?
 - The more specialized your code, the harder it is to re-use
 - Simple interfaces, logic, lead to surprising re-use
 - The less code you have, the less room for bugs
 - Also, the easier it is to harden, defend code
 - The fewer relationships, responsibilities, the easier it is to describe
 - The easier it is to describe, the less likely someone will misuse
- My favorite example of just enough engineering, Java Collections
 - Very few interfaces
 - Very consistent behavior across all interfaces
 - Strong use of convention
 - Once you know how to do something with one type, that's how it works with all
 - Advanced uses are possible, and not difficult
 - More sophisticated features don't make simple use any harder
 - Good defaults
- Learning how not to over-engineer can be one of the most difficult challenges
- Know when to say enough and stop coding

- Contact me
 - Email to feedback@thecommandline.net
 - Web site at <http://thecommandline.net/>

Offset**Topic**

- IM to command.line@skype
- Listener comment line is 360-252-7284
- del.icio.us tag is "for:cmdln"
- <http://twitter.com/cmdln>
- I'd like to thank libsyn.com for AAC hosting and Wouter de Bie for MP3 hosting
- These notes and the show audio and music are covered by a Creative Commons license
 - <http://creativecommons.org/licenses/by-nc-sa/3.0/us/>
 - Attribution, non-commercial, share alike