Att skriva robust kod

Mikael Kalms, EA DICE

Om mig själv

32 år gammal

Programmerat sedan 1990:
68k/x86/sh4/vu asm
C/C++
Och en aning python & java

Senaste releaser: Battlefield Bad Company 2 PC,
Medal Of Honor 2010 (multiplayer)

Innehåll

1. Teori

2. Teori + praktik

3. Oklart just nu

1. Torrsim, även känt som ”teori”

Skriver du ny kod? Tänk på...

Livslängd – är det testkod... eller kommer den att leva länge?

Omfattning – är du den enda användaren av koden?

Isolation - är det intern kod... eller är det ett API?

Risk (sannolikhet x effekt)
- hur stor är risken att koden krånglar?
- vad är det värsta som kan hända om koden inte fungerar som tänkt?

När du ändrar i existerande kod

Anpassa dig efter mönstren i existerande kod

Dina ändringar bör inte försämra kodkvaliteten överlag

När är kod "live"?

Så fort andra använder den.

Varför är det dyrare / svårare att ändra på kod efter att den är live?

Restrain thyself

Det är bra att tänka framåt... men skriv inte kod som inte behövs.

Vad är "utifall-att"-kod? Det är svårt att svara på!

2. Teori + praktik

Att skriva kod som andra också förstår

- kod skrivs 1 gång men läses 10 ggr (*)
- vilken är din målpublik?
- vad kan du förvänta dig av dem?

(*) 73% av all statistik är påhittad

Principer för koddesign

- Don't Repeat Yourself
- Skriv kod som *inte kan* gå fel
- Förenkla din minneshantering
- Lös ägandeskapsfrågan
- Använd ett lagom subset av språket
- Portabel kod är bättre än oportabel (även om du inte ska porta)

Dokumentation?

Kan du göra koden uppenbar så behöver du inte dokumentera lika mycket

Dokumentera syfte, inte implementation

”självdokumenterande kod”

Kodreviews

Förklara för en kompis vad din kod gör och hur den funkar.

Ha en dialog.

Du är inte färdig med koden innan din kompis också är nöjd med den.

Är standardiserade containers bra?

… ja, vad tycker ni om dem i C++? i C#? i Java?

Är design patterns bra?

Observer, Visitor, Singleton, etc…

vilket syfte fyller de egentligen?

Låt datorn hjälpa dig

Programmerare + kompilator: enhetstester
Kompilator: bygg med så mycket varningar på som möjligt
Kompilator: bygg med flera olika kompilatorer
Byggsystem: automatiska byggen (Continuous integration)
Byggsystem: smoketest, soaktest

Programmering för inbyggda system

Koden måste anpassa sig efter hårdvaran.
Du har ingen swapfil... slut på minne => game over
Du kan inte byta till en maskin med mer minne.
Du kan inte byta till en maskin med snabbare CPU.
Kompilatorn är gammal. Eller ny.
Debuggern funkar ... ibland.

Men det är ganska kul.

3. Oklart

Tack!

mikael@kalms.org