Cloning in Software: Why, When, and How?

Minhaz F. Zibran (minhaz.zibran@usask.ca) and Chanchal K. Roy (croy@cs.usask.ca)
Software Research Lab, Department of Computer Science, University of Saskatchewan, Canada

Introduction

- Code Clones
  - exact clones: code fragments that are exact duplicates
  - near-miss clones: similar code fragments
- We address:
  - reasons: why are clones created?
  - consequences: how code clones affect development and maintenance?
  - detection: how can we detect clones?
  - management: how can we efficiently manage and refactor clones?

Code Clones in Software Systems

- 9%-17% clones exist in software systems, sometimes even 50%

Reasons for cloning

- Intentional clones: deliberately created clones
  - code reuse by copy-paste-modification
  - implementation of same/similar feature
  - minimization of coupling, risks
- Unintentional/Accidental clones: eventually similar code fragments
  - use of design patterns, frameworks
  - restrictions from programming language/paradigm
  - development and implementation of same/similar functionality

Effects of Cloning

- Development benefits:
  - speed up development process
  - risk minimization by confident reuse of working code
- Increased maintenance effort:
  - bug propagation
  - code inflation
  - consistent modification of clones

Clone Detection

Clone Analysis: Existence and Evolution

- An Empirical Study
  - 18 software systems
  - total 1,636 releases
  - forecasting using regression analysis

Clone Management and Refactoring

- Formulated a refactoring effort model
- Implemented code clone refactoring schedule.
- applied constraint programming (CP) techniques
- implemented using OPL
- used IBM ILOG CPLEX Optimization Studio 12.2

Cloning in Software: Why, When, and How?


This work is supported in part by the Natural Science and Engineering Research Council of Canada (NSERC)