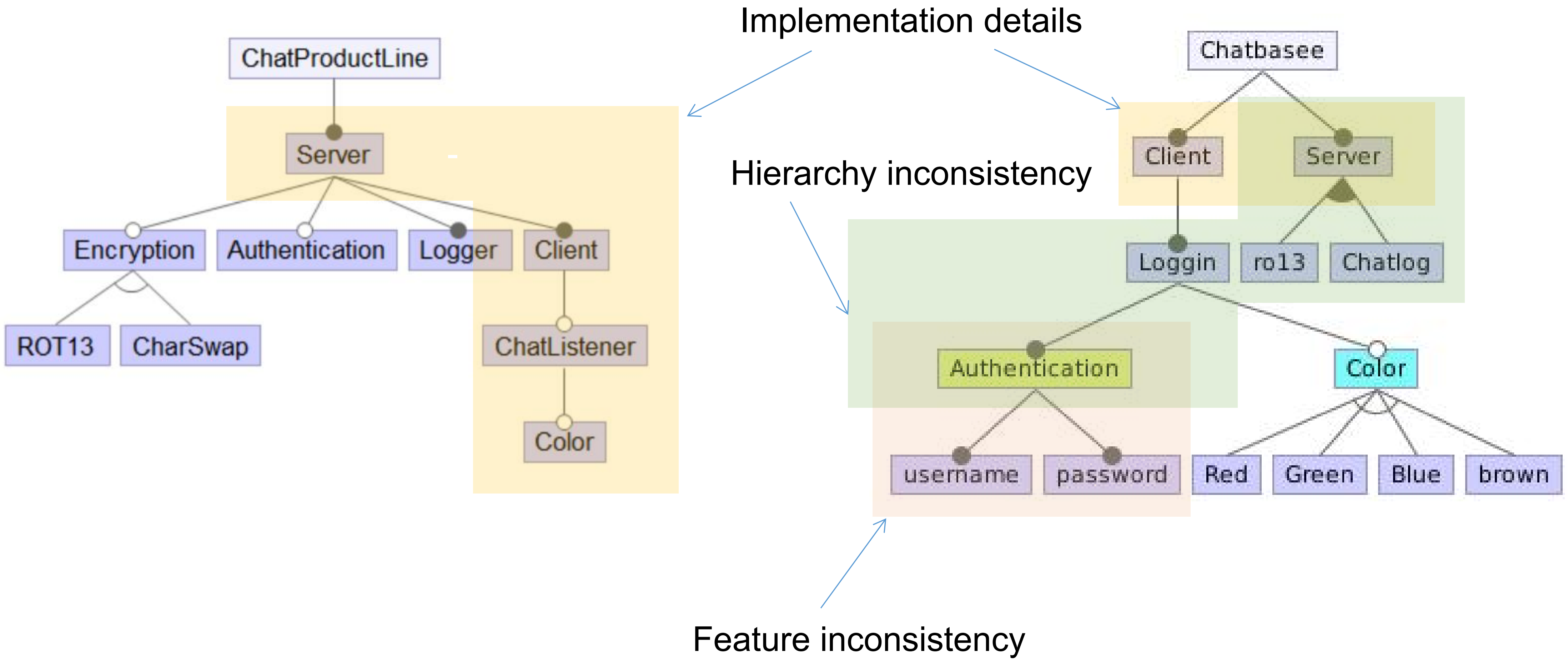
Software Product Line Engineering

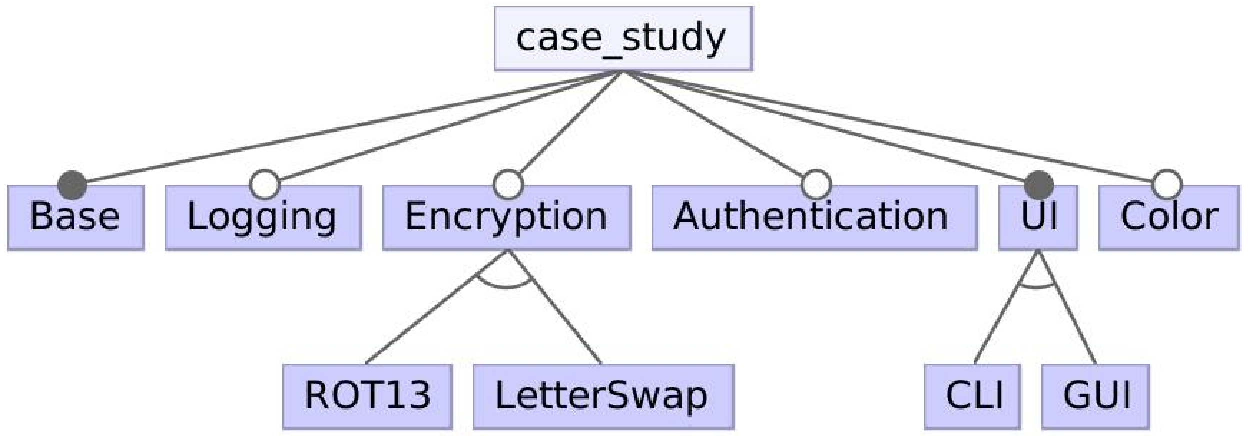
Lab Class 6 / Assignment 2+3

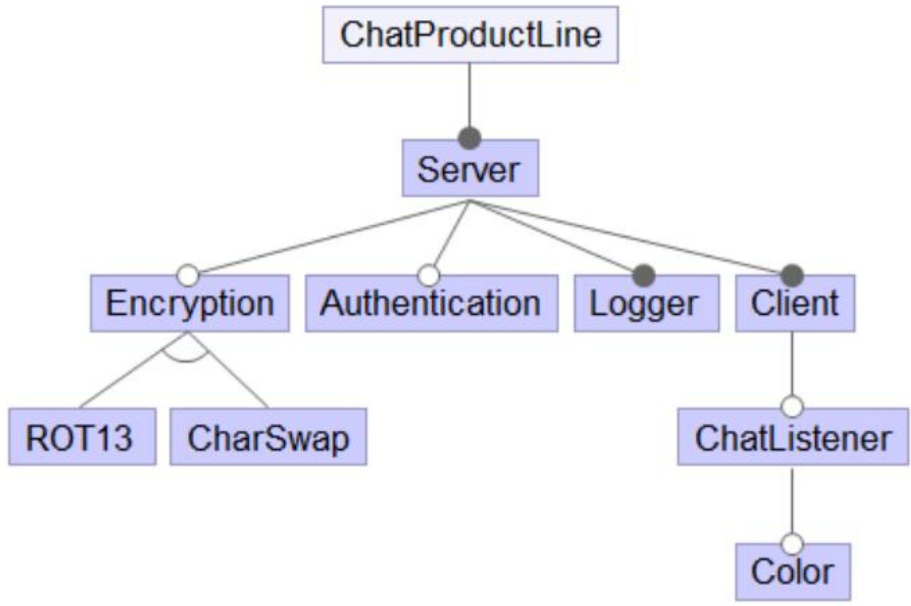
Assignment 2: Feature Modeling

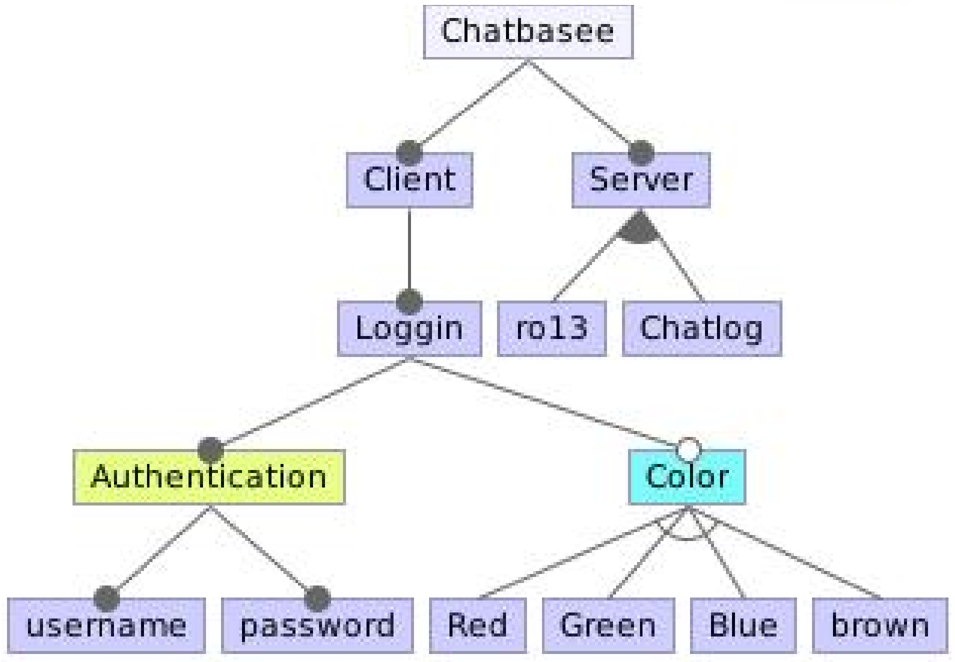


Assignment 2: Problems with Feature Diagrams

- What is a feature and what *not*?
    - Rule of thumb: Each distinct piece of functionality can be a feature
    - Rule of thumb: Only one feature per piece of functionality
- What is *not* a feature?
    - Implementation details, such as architecture (client-server) or single classes/interfaces
- Feature Hierarchy
    - A subfeature implies its parent feature!


case_study
Base
Logging
Encryption
Authentication
UI
Color
ROT13
LetterSwap
CLI
GUI


ChatProductLine
Server
Encryption
Authentication
Logger
Client
ROT13
CharSwap
ChatListener
Color


Chatbasee
Client
Server
Loggin
ro13
Chatlog
Authentication
Color
username
password
Red
Green
Blue
brown

Task 1a) Versions vs variants

- Version: state of a software system/SPL after a sequence of modifications is applied to it (temporal order)

- Variant: product (one of many possible ones) derived from a SPL with behavior specified by its configuration

- Configuration space and temporal dimension are orthogonal.

Task 1b) How can we implement SPLs?

Version Control Systems (VCS)	Build Tools	Pre-processors
Each *variant* can be developed on a *separate branch*. Variants are developed by *merging* feature branches into each variant branch.	For each *variant*, a *build target* or routine is maintained explicitely. Variants are derived by simply *building* a target.	For each *feature*, lines are guarded with pre-processor directives. Variants are derived by setting pre-processor flags, *pre-compilation*, and final compilation.

1c) Advantages / Disadvantages

	Version Control Systems (VCS)	Build Tools	Pre-processors
+	• Well established tool • Minimal preplanning	• Orchestration of (pre-) processors and runtime options • File-level granularity	• Easy to use • Line-level granularity • no run-time overhead
-	• Development of variants, not features • No structured reuse • Propagation of bugfixes • Mixed features/variants	• Coarse-grained (files only) • hard to maintain for large/complex build scripts	• Feature Scattering/Tangling • Error-prone if complex • Hard to maintain if excessively used
Scenario	e.g., customer-specific software that is developed rapidly (agile)	e.g., development of a Linux distribution, builds for different, but few platforms	e.g., software where features are fine-grained.

1d) Use cases

	Version Control Systems (VCS)	Build Tools	Pre-processors
Performance	Code of deselected features is not present in any variant.	Code of deselected features is not present in any variant.	Code of deselected features is not present in any variant.
Collaboration	Main purpose of VCSs	Problematic. Different developers can modify the same build script.	Problematic. Different developers modify the same directives.
Third-Party Software	Problematic. Features are not modularized.	Problematic. Features are not modularized.	Problematic. Features are not modularized.
Granularity	variant-level (very coarse-grained)	file-level (coarse-grained)	line-level (fine-grained)