Software Engineering

WS 2022/23, Sheet 02

Prof. Dr. Sven Apel
Yannick Lehmen, Maurice Vincon

Handout: 07.11.2021

Task 1

Design a feature diagram for the chat application from exercise sheet 01. The chat application consists of many clients that communicate via a central server. Decide for yourself which features should be *mandatory*, *optional*, *alternatives*, and *or-groups*.

The chat application should offer the following functionalities:

GUI and Terminal The chat app should have a simple graphical interface but can also be used from a terminal.

Color Each message can be colored.

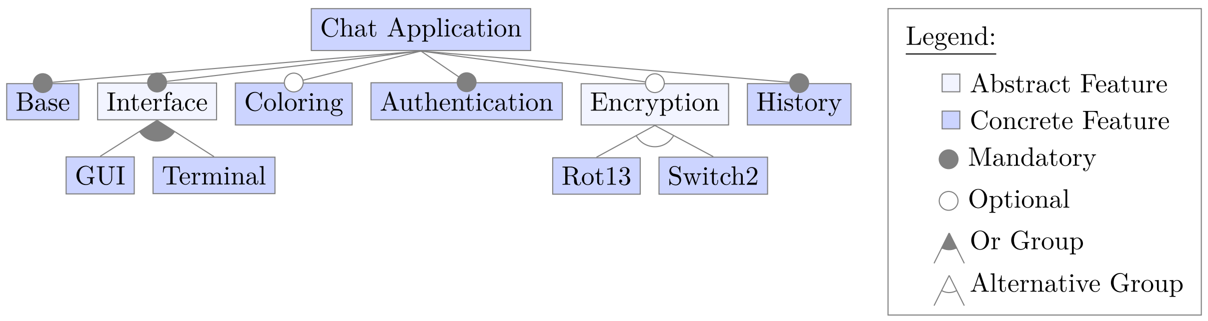
Authentication The client has to authenticate itself by sending a username and password to the server.

Encryption All message exchange is encrypted. Support two simple encryption methods.

History All clients and the server keep a history of received messages.

Solution

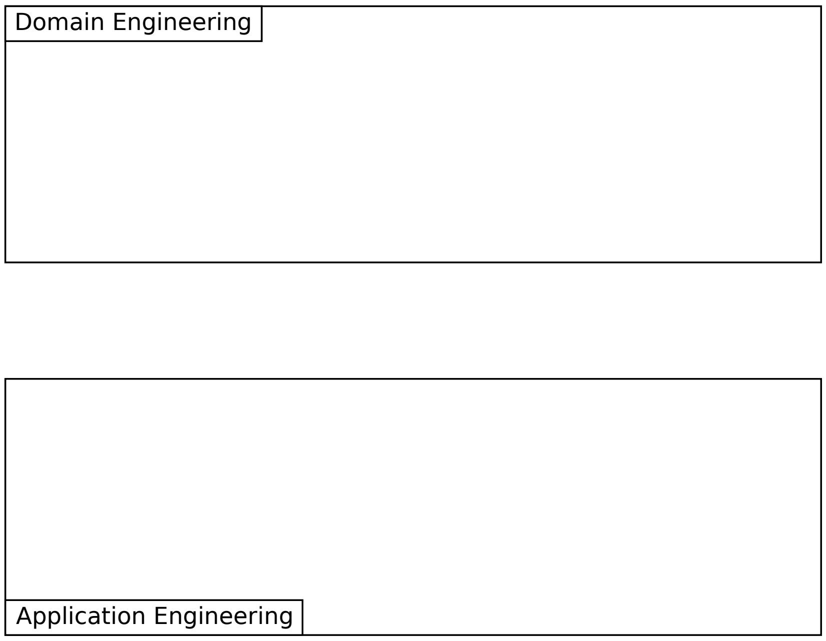
There are multiple possible feature diagrams for this specific configurable software system depending on which cross-tree constraints you introduce. Here is a solution without any cross-tree constraints:



Task 2

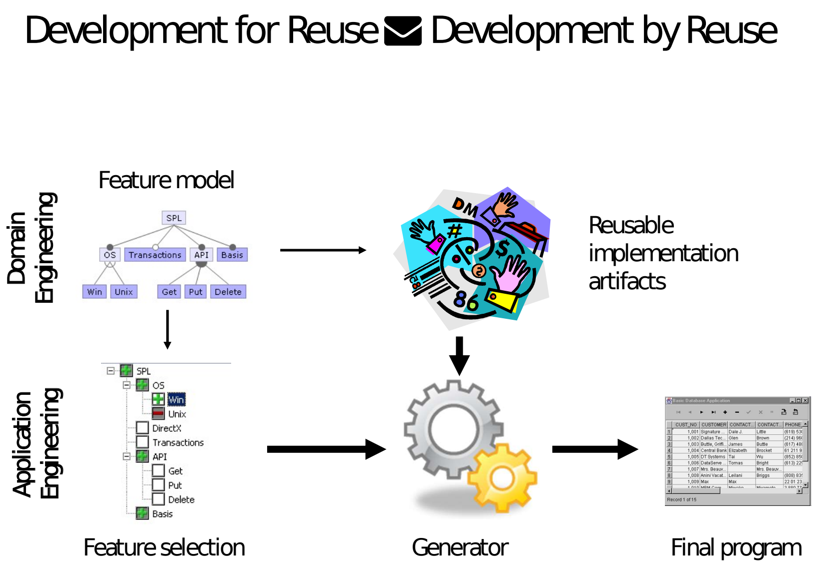
a) Complete the diagram below regarding application and domain engineering.

b) How does the traditional software development of a singles systems differ from software development with application and domain engineering?



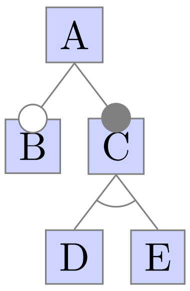
Solution

a)



b) When developing a single system, software is built to fulfill a specific set of requirements. In contrast, when following an approach using domain and application engineering, the software system is built with the whole domain of the system in mind so that new variants with additional requirements can be easily added later on. Developing a single system requires less effort and cost than the initial development of a configurable software system. However, these costs amortize as more variants are integrated into the configurable system until a break-even-point is reached where the overall costs for the configurable software system are lower than the costs for developing each variant as an individual program. This is because the initial effort put into domain and application engineering in the beginning leads to reusable software artifacts that can be employed in new system variants added later on.

Task 3



Given a configurable software system with the feature diagram to the right:

a) Identify all valid and two invalid configurations.

b) Work out a simple algorithm for counting all valid configurations of a configurable software system.

c) Give the propositional formula for the diagram.

d) How does the propositional formula and the set of valid configurations change if an implication $E \Rightarrow B$ is introduced?

e) Redraw the feature diagram including the new implication.

f) What is the relationship between a feature diagram and a propositional formula?

Solution

a) Valid configurations:

- $s_1 = \{A, C, D\}$
- $s_2 = \{A, C, E\}$
- $s_3 = \{A, B, C, D\}$
- $s_4 = \{A, B, C, E\}$

Invalid configurations:

- $s_5 = \{A, B\}$
- $s_6 = \{A, C, D, E\}$

b) For feature diagrams without cross-tree constraints we get the following recursive function $\mathsf{count}(c)$. This function traverses the tree structure of a feature diagram (we use the abstract syntax here). [1]

$$\begin{aligned}
\mathsf{count}(\bigstar\langle\!\langle\, n \,\rangle\!\rangle) &= \mathsf{count}\ n \\
\mathsf{count}(\bullet\langle\!\langle\, f, n \,\rangle\!\rangle) &= \mathsf{count}\ n \\
\mathsf{count}(\circ\langle\!\langle\, f, n \,\rangle\!\rangle) &= \mathsf{count}\ n + 1 \\
\mathsf{count}(\wedge\langle\!\langle\, f, d_1 \dots d_k \,\rangle\!\rangle) &= \mathsf{count}\ d_1 \times \dots \times \mathsf{count}\ d_k \\
\mathsf{count}(\triangle\langle\!\langle\, f, d_1 \dots d_k \,\rangle\!\rangle) &= \mathsf{count}\ n_1 + \dots + \mathsf{count}\ n_k \\
\mathsf{count}(\blacktriangle\langle\!\langle\, f, d_1 \dots d_k \,\rangle\!\rangle) &= (\mathsf{count}\ n_1 + 1) \times \dots \times (\mathsf{count}\ n_k + 1) - 1 \\
\mathsf{count}(f) &= 1
\end{aligned}$$

Example:

$$\begin{aligned}
\mathsf{count}(\bigstar\langle\!\langle \wedge\langle\!\langle A, \circ\langle\!\langle A, B \rangle\!\rangle, \bullet\langle\!\langle A,\ \triangle\langle\!\langle C, D, E \rangle\!\rangle \rangle\!\rangle \rangle\!\rangle \rangle\!\rangle) &= \mathsf{count}(\wedge\langle\!\langle A, \circ\langle\!\langle A, B \rangle\!\rangle, \bullet\langle\!\langle A,\ \triangle\langle\!\langle C, D, E \rangle\!\rangle \rangle\!\rangle \rangle\!\rangle) \\
&= \mathsf{count}(\circ\langle\!\langle A, B \rangle\!\rangle) \ \times\ \mathsf{count}(\bullet\langle\!\langle A,\ \triangle\langle\!\langle C, D, E \rangle\!\rangle \rangle\!\rangle) \\
&= (\mathsf{count}(B) + 1) \ \times\ \mathsf{count}(\triangle\langle\!\langle C, D, E \rangle\!\rangle) \\
&= (1 + 1) \ \times\ (\mathsf{count}(D) + \mathsf{count}(E)) \\
&= (1 + 1) \ \times\ (1 + 1) = 4
\end{aligned}$$

c)

$$\begin{aligned}
F &= \{A, B, C, D, E\} \\
\Phi &= A \\
&\quad \wedge (B \Rightarrow A) \\
&\quad \wedge (A \Leftrightarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E)
\end{aligned}$$

d)

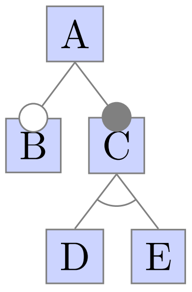
$$\begin{aligned}
F &= \{A, B, C, D, E\} \\
\Phi &= A \\
&\quad \wedge (B \Rightarrow A) \\
&\quad \wedge (A \Leftrightarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E) \\
&\quad \wedge (E \Rightarrow B)
\end{aligned}$$

Valid configurations:

[1] Feature-Oriented Software Product Lines: Concepts and Implementation. S. Apel, et al., Springer, 2013

- $s_1 = \{A, C, D\}$
- $s_2 = \{A, C, E\} \leftarrow$ no longer valid
- $s_3 = \{A, B, C, D\}$
- $s_4 = \{A, B, C, E\}$

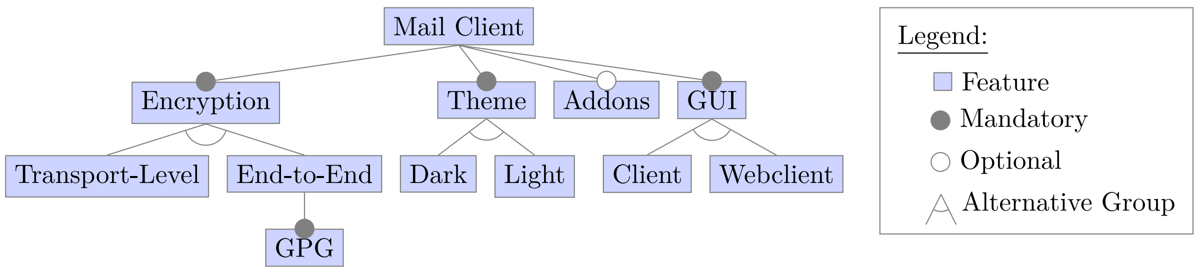
e) The easiest way to add this implication is by adding a cross-tree constraint:



$$E \Rightarrow B$$

f) Feature diagrams are a graphical representation of propositional formulas. A feature diagram can always be represented as a propositional formula. However, without cross-tree constraints, not every propositional formula can be represented as a feature diagram.

Task 4



$$Encryption \Rightarrow Client$$

a) Give the propositional formula for the feature diagram.

b) Give the sets of mortal, immortal, and dead features for the feature model. Use the notation introduced in the lecture.

c) Give two valid configurations of the mail client using the function representation.

Solution

a)

$$\begin{aligned}
F &= \{Mail\ Client, Encryption, Theme, Addons, GUI, Transport\text{-}Level, End\text{-}to\text{-}End, Dark, Light, Client, Webclient, GP \\
\Phi &= Mail \\
&\quad \land (Mail\ Client \Leftrightarrow Encryption) \land ((Transport\text{-}Level \lor End\text{-}to\text{-}End) \Leftrightarrow Encryption) \land \neg(Transport\text{-}Level \land End\text{-}to \\
&\quad \land (End\text{-}to\text{-}End \Leftrightarrow GPG) \\
&\quad \land (Mail\ Client \Leftrightarrow Theme) \land ((Dark \lor Light) \Leftrightarrow Theme) \land \neg(Dark \land Light) \\
&\quad \land (Addons \Rightarrow Mail\ Client) \\
&\quad \land (Mail\ Client \Leftrightarrow GUI) \land ((Client \lor Webclient) \Leftrightarrow GUI) \land \neg(Client \land Webclient) \\
&\quad \land (Encryption \Rightarrow Client)
\end{aligned}$$

b)

$$\begin{aligned}
immortal(\Phi) &= \{MailClient, Encryption, Theme, GUI, Client\} \\
mortal(\Phi) &= \{Transport\text{-}Level, End\text{-}to\text{-}End, GPG, Dark, Light, Addons\} \\
dead(\Phi) &= \{Webclient\}
\end{aligned}$$

c)

$$\begin{aligned}
v_1 = \{&(Mail\ Client, 1), (Encryption, 1), (Transport\text{-}Level, 1), (End\text{-}to\text{-}End, 0), (GPG, 0), (Theme, 1), \\
&(Dark, 1), (Light, 0), (Addons, 0), (GUI, 1), (Client, 1), (Webclient, 0)\} \\
v_2 = \{&(Mail\ Client, 1), (Encryption, 1), (Transport\text{-}Level, 0), (End\text{-}to\text{-}End, 1), (GPG, 1), (Theme, 1), \\
&(Dark, 1), (Light, 0), (Addons, 1), (GUI, 1), (Client, 1), (Webclient, 0)\}
\end{aligned}$$

Task 5

a) Describe the terms *Refactoring*, *Generalization*, and *Specialization* in the context of feature modeling in your own words.

b) Given the following propositional formula of a feature model, give a new formula for each of the three change methods mentioned in a).

$$\begin{aligned} F =&\{A, B, C, D, E, F, G, H\} \\ \Phi =&\ A \\ &\wedge (A \Rightarrow B) \wedge (B \Rightarrow A) \\ &\wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \\ &\wedge (A \Leftrightarrow F) \wedge ((G \vee H) \Leftrightarrow F) \wedge \neg(G \wedge H) \end{aligned}$$

c) Given the following propositional formula of a feature model, decide if the formulas in (i) – (iii) represent generalizations, refactorings, or specializations of the formula. Why?

$$\begin{aligned} F =&\ \{A, B, C, D, E, F, G, H\} \\ \Phi =&\ A \\ &\wedge (A \Leftrightarrow B) \\ &\wedge (A \Rightarrow F) \\ &\wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E) \\ &\wedge (A \Leftarrow F) \wedge ((G \vee H) \Leftrightarrow F) \end{aligned}$$

(i)

$$\begin{aligned} F' =&\ \{A, B, C, D, E, F, G, H\} \\ \Phi' =&\ A \\ &\wedge (A \Leftrightarrow B) \\ &\wedge (A \Rightarrow F) \\ &\wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \\ &\wedge (A \Leftarrow F) \wedge ((G \vee H) \Leftrightarrow F) \end{aligned}$$

(ii)

$$\begin{aligned} F'' =&\ \{A, B, C, D, E, F, G, H\} \\ \Phi'' =&\ A \\ &\wedge (A \Leftrightarrow B) \\ &\wedge (A \Rightarrow F) \\ &\wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E) \\ &\wedge (A \Leftarrow F) \wedge ((G \vee H) \Leftrightarrow F) \\ &\wedge (D \Rightarrow H) \end{aligned}$$

(iii)

$$\begin{aligned} F''' =&\ \{A, B, C, D, E, F, G, H\} \\ \Phi''' =&\ A \\ &\wedge (A \Leftrightarrow B) \\ &\wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E) \\ &\wedge (A \Leftrightarrow F) \wedge ((G \vee H) \Leftrightarrow F) \end{aligned}$$

Solution

a) **Refactoring** is a reformulation of the formula while the sets of valid and invalid configurations stay the same.

Generalization of a feature model makes additional configurations valid while preserving all previously valid configurations and, therefore, making it more generally applicable.

Specialization takes away valid configurations of a feature model making it more restrictive.

Note: Adding or removing *features* always leads to the feature models being *incomparable*!

b) (i) Refactoring:

$$\begin{aligned}
F' &= \{A, B, C, D, E, F, G, H\}\\
\Phi' &= A\\
&\quad \wedge (A \Leftrightarrow B)\\
&\quad \wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C)\\
&\quad \wedge (A \Leftrightarrow F) \wedge ((G \vee H) \Leftrightarrow F) \wedge \neg(G \wedge H)
\end{aligned}$$

(ii) Generalization:

$$\begin{aligned}
F'' &= \{A, B, C, D, E, F, G, H\}\\
\Phi'' &= A\\
&\quad \wedge (A \Rightarrow B) \wedge (B \Rightarrow A)\\
&\quad \wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C)\\
&\quad \wedge (A \Leftrightarrow F) \wedge ((G \vee H) \Leftrightarrow F)
\end{aligned}$$

(iii) Specialization:

$$\begin{aligned}
F''' &= \{A, B, C, D, E, F, G, H\}\\
\Phi''' &= A\\
&\quad \wedge (A \Rightarrow B) \wedge (B \Rightarrow A)\\
&\quad \wedge (A \Leftarrow C) \wedge ((D \vee E) \Leftrightarrow C) \wedge \neg(D \wedge E)\\
&\quad \wedge (A \Leftrightarrow F) \wedge ((G \vee H) \Leftrightarrow F) \wedge \neg(G \wedge H)
\end{aligned}$$

c) (i) *Generalization*, because we get more configurations transforming the alternative-group to an or-group without losing any of the previous configurations.

(ii) *Specialization*, because we limit the number of configurations by adding the implication of D to H.

(iii) *Refactoring*, because we simply rewrite the formula without losing or adding any configurations.

Task 6

Given the set of features $F = \{a, b, c, d\}$:

a) Give the set S of all possible feature selections.

b) Give the set C of valid feature selections with the following constraints: $\Phi = (b \Rightarrow \neg c) \wedge (a \Rightarrow b)$

c) Convert the set $\chi(v) = \{a, c\}$ to its corresponding function representation v. Then, convert this representation to the propositional formula $\eta(v)$

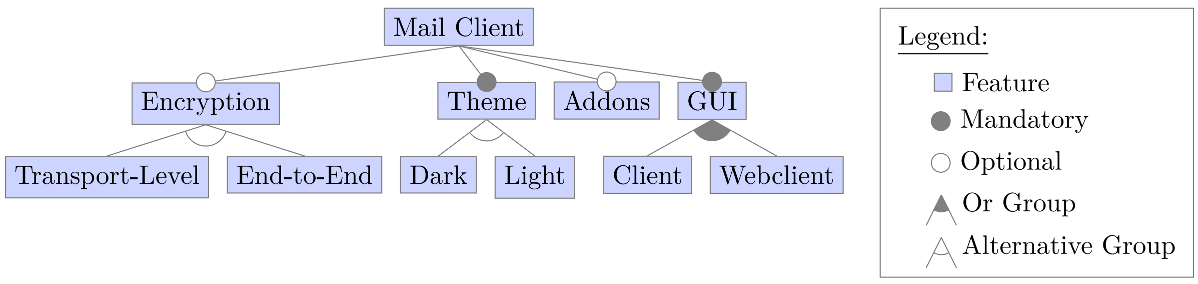
Solution

a) $S = \{\{\}, \{a\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}, \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \{a, b, c, d\}\}$

b) $C = \{\{\}, \{b\}, \{c\}, \{d\}, \{a, b\}, \{b, d\}, \{c, d\}, \{a, b, d\}\}$

c) $v = \{(a, 1), (b, 0), (c, 1), (d, 0)\}$ and $\eta(v) = a \wedge \neg b \wedge c \wedge \neg d$

Task 7



a) Convert the feature diagram to its abstract syntax.

b) Convert the abstract syntax representation fn to its propositional formula $[\![fn]\!]$.

Solution

a)

$$
\begin{aligned}
fn = &\bigstar\langle\!\langle \wedge\langle\!\langle \mathit{Mail\ Client},\\
&\quad \bigcirc\langle\!\langle \mathit{Mail\ Client},\ \vartriangle\langle\!\langle \mathit{Encryption}, \mathit{Transport\text{-}Level}, \mathit{End\text{-}to\text{-}End} \rangle\!\rangle \rangle\!\rangle,\\
&\quad \bullet\langle\!\langle \mathit{Mail\ Client},\ \vartriangle\langle\!\langle \mathit{Theme}, \mathit{Dark}, \mathit{Light} \rangle\!\rangle \rangle\!\rangle,\\
&\quad \bigcirc\langle\!\langle \mathit{Mail\ Client}, \mathit{Addons} \rangle\!\rangle,\\
&\quad \bullet\langle\!\langle \mathit{Mail\ Client}, \blacktriangle\langle\!\langle \mathit{GUI}, \mathit{Client}, \mathit{Webclient} \rangle\!\rangle \rangle\!\rangle\\
&\quad \rangle\!\rangle \rangle\!\rangle
\end{aligned}
$$

b)

$$
\begin{aligned}
[\![fn]\!] = &\mathit{Mail\ Client}\\
&\quad \wedge (\mathit{Mail\ Client} \Leftarrow \mathit{Encryption})\\
&\qquad \wedge ((\mathit{Transport\text{-}Level} \vee \mathit{End\text{-}to\text{-}End}) \Leftrightarrow \mathit{Encryption}) \wedge \neg(\mathit{Transport\text{-}Level} \wedge \mathit{End\text{-}to\text{-}End})\\
&\quad \wedge (\mathit{Mail\ Client} \Leftrightarrow \mathit{Theme})\\
&\qquad \wedge ((\mathit{Dark} \vee \mathit{Light}) \Leftrightarrow \mathit{Theme}) \wedge \neg(\mathit{Dark} \wedge \mathit{Light})\\
&\quad \wedge (\mathit{Mail\ Client} \Leftarrow \mathit{Addons})\\
&\quad \wedge (\mathit{Mail\ Client} \Leftrightarrow \mathit{GUI})\\
&\qquad \wedge ((\mathit{Client} \vee \mathit{Webclient}) \Leftrightarrow \mathit{GUI})
\end{aligned}
$$