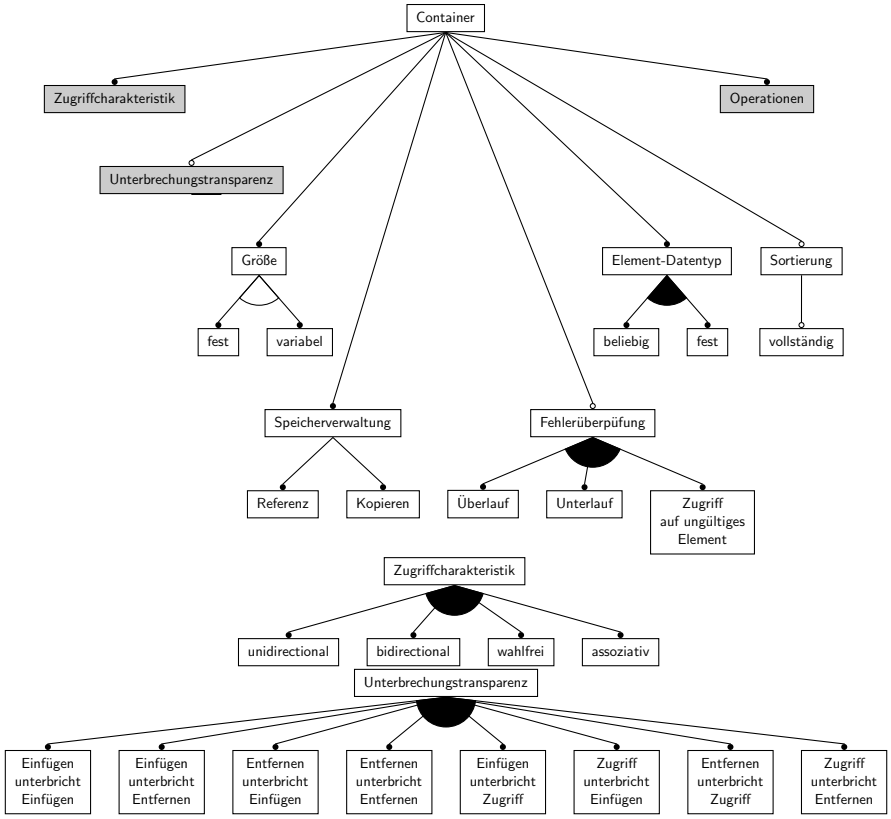
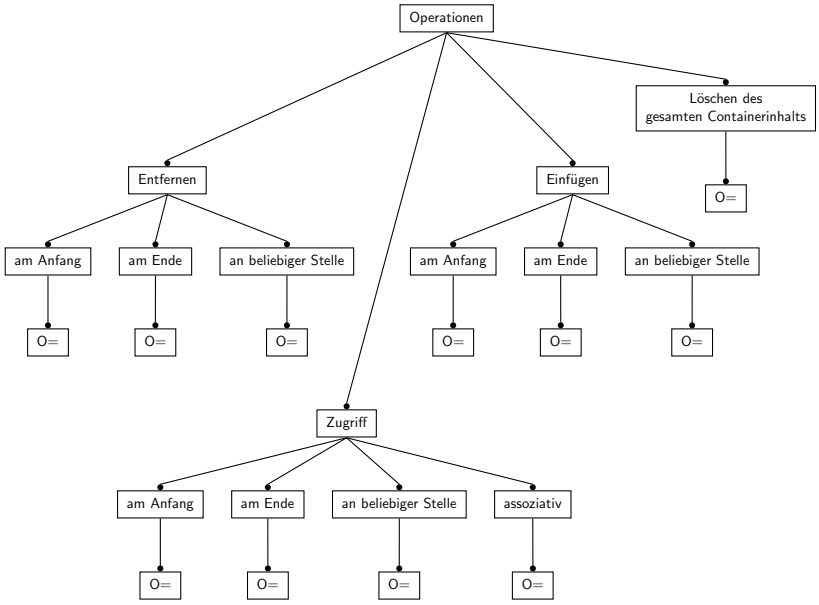
Domänenanalyse – Container

Dieses Dokument beschreibt mit Hilfe von Feature-Diagrammen, das Merkmalsmodell der Domäne Container. In einem Domänenlexikon werden die wichtigsten Begriffe kurz erläutert.

Domänen Featurediagramm





Domänen Lexikon

- **Algorithmen:**
  Operationen auf Containern, die über die Grundfunktionen(Einfügen, Entfernen, Zugriff) hinausgehen, wie zum Beispiel eine Sortierfunktion. Um Algorithmen auf jede Art von Containern anzuwenden, setzen diese auf der Einheitlichen Iterator-Schnittstelle auf.

- **Allokatoren:**
  Schnittstelle zwischen den Domänen *Speicherverwaltung* und *Container*. Ein Allokator belegt und gibt Speicher frei, indem er die z.B. Funktionen der globalen Speicherverwaltung verwendet. Ausserdem ist es möglich einen Allokator zu implementieren, der eine eigene lokale Speicherverwaltung besitz.

- **Container:**
  Behälter, die Objekte aufnehmen und verwalten können.

- **Containeradapter:**
  Stellt ein Containerschnittstelle bereit, die auf der Basis anderer Container implementiert ist.

- **Element-Datentyp:**
  Der Datentyp den ein Container verwalten kann. (z.B. `char` in einem *Character-Array*).

- **Iteratoren:**
  Schnittstelle zum einheitlichen Zugriff auf Containerinhalte.

- **Sortierung:**
  Bei einem sortiertem Container werden die Elemente beim geordnet eingefügt. Es ist auch eine nicht-vollständige denkbar, wie zum Beispiel eine Heap-Struktur.

- **Speicherverwaltung:**
  Um Objekte zu verwalten, muss im Fall **variabler Container-Größe** dynamisch Speicher für diese Objekte angefordert werden. Dieser Speicher wird entweder vom Container selbst oder vom Benutzer des Containers angefordert und gelöscht.

- **STL:**
  Standard **T**emplate **L**ibrary (C++); beinhaltet Implementierungen von Containern und Algorithmen. Siehe auch [Str00] oder [KS00].

- **Überlauf:**
  Versuch mehr Elemente als möglich in einem Container **fester Größe** zu abzulegen.

- **Unterlauf:**
  Versuch mehr als die vorhandenen Elemente aus einem Container zu entnehmen.

- **Unterbrechungstransparenz:**
  Ermöglicht bestimmten Kobinationen von Operationen, sich gegenseitig zu unterbrechen, ohne dass daraus ein inkonsistenter Zustand des Containerinhalt ergibt.

- **Zugriff:**
  Im Kontext dieser Domainanalyse ist darunter eine Leseoperation zu verstehen.

- **Zugriffscharakteristik:**
  Legt die Verwaltungsstrukturen fest, auf denen der Container arbeitet.

  - **unidirectional:**
    Die einzelnen Containerelemente können nur in einer Richtung *effizient* durchlaufen werden. Beispiel: einfach verkettete Liste
  - **bidirectional:**
    Die einzelnen Containerelemente können nur in zwei Richtung *effizient* durchlaufen werden. Beispiel: doppelt verkettete Liste
  - **wahlfrei:**
    Es ist ein *effizienter* Zugriff auf jedes einzelne Containerelement möglich. Beispiel: Feld (eng.: array)

– **assoziativ:**
Es ist ein *effizienter* Zugriff auf Containerelemente mittels eines Schlüssel möglich. Beispiel: Map der STL.

Abdeckungsüberprüfung

Hier werden mit Hilfe einer Tabelle exemplarisch einige Containerausprägungen auf eine Merkmalskonfiguration abgebildet.

verwendete Container:

- **(Wosch) Queue:**
Entnommen aus [SP00].
- **STL Vector:**
Ein STL Container (siehe [Str00]).
- **STL Deque:**
Ein STL Container (siehe [Str00]).
- **STL List:**
Ein STL Container (siehe [Str00]).
- **STL map:**
Ein STL Container (siehe [Str00]).
- `char` **Array:**
Ein Feld vom Typ `char`.

	Feature	Wosch Queue	STL Vector	STL Deque	STL List	char Array	STL map
1	Container						
1.1	Operationen						
1.1.1	Löschen des gesamten Containerinhalts	x	x	x	x	x	x
1.1.1.1	O=	1	n	n	n	1	n
1.1.2	Zugriff						
1.1.2.1	am Anfang	x	x	x	x	x	x
1.1.2.1.1	O=	1	1	1	1	1	1
1.1.2.2	am Ende	x	x	x	x	x	x
1.1.2.2.1	O=	1	1	1	1	1	1
1.1.2.3	an beliebiger Stelle	x	x	x	x	x	
1.1.2.3.1	O=	n	1	1	n	1	
1.1.2.4	assoziativ						x
1.1.2.4.1	O=						log(n)
1.1.3	Entfernen						
1.1.3.1	am Anfang	x	x	x	x	x	
1.1.3.1.1	O=	1	n	1	1	n	
1.1.3.2	am Ende	x	x	x	x	x	
1.1.3.2.1	O=	1	1	1	1	1	
1.1.3.3	an beliebiger Stelle	x	x	x	x	x	x
1.1.3.3.1	O=	1	n	n	1	n	log(n)
1.1.4	Einfügen						
1.1.4.1	am Anfang	x	x	x	x	x	
1.1.4.1.1	O=	1	n	1	1	n	
1.1.4.2	am Ende	x	x	x	x	x	

	Feature	Wosch Queue	STL Vector	STL Deque	STL List	char Array	STL map
1.1.4.2.1	O=	1	1	1	1	1	
1.1.4.3	an beliebiger Stelle		x	x	x	x	x
1.1.4.3.1	O=		n	n	1	n	log(n)
1.2	Zugriffcharakteristik						
1.2.1	unidirectional	x					
1.2.2	bidirectional				x		
1.2.3	wahlfrei		x	x		x	
1.2.4	assoziativ						x
1.3	Unterbrechungstransparenz						
1.3.1	Einfügen unterbricht Einfügen	x					
1.3.2	Einfügen unterbricht Entfernen	x					
1.3.3	Entfernen unterbricht Einfügen						
1.3.4	Entfernen unterbricht Entfernen						
1.3.5	Entfernen unterbricht Zugriff						
1.3.6	Einfügen unterbricht Zugriff						
1.3.7	Zugriff unterbricht Entfernen						
1.3.8	Zugriff unterbricht Einfügen						
1.4	Größe						
1.4.1	fest					x	
1.4.2	variabel	x	x	x	x		x
1.5	Speicherreservierung und -freigabe						
1.5.1	Referenz	x					
1.5.2	Kopie		x	x	x	x	x
1.6	Fehlerüberpüfung						

	Feature	Wosch Queue	STL Vector	STL Deque	STL List	char Array	STL map
1.6.1	Überlauf						
1.6.2	Unterlauf						
1.6.3	Zugriff auf gültiges Element		x	x			
1.7	Sortierung						x
1.7.1	vollständig						x
1.8	Element-Datentyp						x
1.8.1	fest		x	x	x	x	x
1.8.2	variabel	x					x

Literatur

[KS00] Stefan Kuhlins and Martin Schader. *Die C++-Standardbibliothek*. Springer, Heidelberg, 2000.

[SP00] Wolfgang Schröder-Preikschat. Betriebssysteme. http://www4.informatik.uni-erlangen.de, 2000. Vorlesungsskript.

[Str00] Bjarne Stroustrup. *Die C++-Programmiersprache*. Addison-Wesley, München, 2000. 4. aktualisierte und erweiterte Auflage.