Template

Mai 2022

Über arc42

arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.

Template Version 8.1 DE. (basiert auf AsciiDoc Version), Mai 2022

Siehe https://arc42.org.

Diese Version des Templates enthält Hilfen und Erläuterungen. Sie dient der Einarbeitung in arc42 sowie dem Verständnis der Konzepte. Für die Dokumentation eigener System verwenden Sie besser die *plain* Version.

Einführung und Ziele

Aufgabenstellung

Inhalt

Dieses Dokument beschreibt eine Advanced Client – Server Spielvariante von Tron. Tron ist ein action rennspiel, bei dem Motorräder einen Schatten hinterlassen. Fahren andere Spieler gegen den Schatten, sind sie tot, sollten Spieler „crashen“ oder auch gegen die Wand fahren, sind diese auch tot und haben verloren. Der letzte Überlebende des Spiels, ist der Gewinner.

Das Spiel soll folgende Anforderungen erfüllen:

Use Cases

Nummer: UC-1

Titel: Spieleranzahl festlegen

Akteur: Spieler

Ziel: Die maximale Anzahl der Spieler für die nächste Spielrunde soll festgelegt werden.

Auslöser: Intention eines Spielers, eine vom Standardwert abweichende Anzahl der Mitspieler festzulegen.

Vorbedingung:

- Die Anwendung wurde erfolgreich gestartet.
- Es wird Bildschirm 1 angezeigt.

Nachbedingung:

- Das Eingabefeld enthält einen ganzzahligen Wert von 2 bis 6 für die Spieleranzahl
- Die anwendungsinterne Spieleranzahl wurde auf den eingegebenen Wert geändert, sodass dieser beim nächsten Spielstart verwendet wird.

Erfolgsszenario:

1. System erzeugt GUI Fuer Bildschirm 1 und zeigt diese an
2. Das System zeigt ein Eingabefeld für die Spieleranzahl an, das zunächst den konfigurierten Standardwert enthält.
3. Der Spieler wählt per Mausklick das Eingabefeld an.
4. Der Spieler ersetzt den vorherigen Inhalt des Eingabefeldes mit einem ganzzahligen Wert von 2 bis 6.
5. Beim Mausklick außerhalb des Eingabefeldes wird der gültige Eingabewert für den nächsten Spielstart übernommen.

Fehlerfälle:

3. a) Der Spielers versucht, einen nicht-numerischen Wert einzugeben.
4. a) 1. Die Eingabe des Spielers erscheint nicht im Eingabefeld.
5. a) 2. Weiter bei 3.

3. b) Der Spieler macht eine Eingabe, die keiner Ganzzahl von 2 bis 6 entspricht.

3. b) 1. Das Eingabefeld wird rot umrandet und es wird ein Warnhinweis über die ungültige Eingabe angezeigt. Der Start-Button wird ausgegraut und deaktiviert.
4. b) 2. Weiter bei 3.

Erweiterungsfälle:

4. a) Nach Abschluss des Erfolgsszenarios ist immer noch ein Warnhinweis vorhanden.
5. a) 1. Der Warnhinweis und die rote Umrandung des Eingabefeldes werden ausgeblendet.

Der Startbutton wird wieder freigegeben.

Nummer: UC-2

Titel: Starten eines Spiels

Akteur: Der Spieler

Ziel: Der Spieler möchte ein Spiel spielen

Auslöser: Der Spieler hat das Spiel gestartet, weil er ein Spiel spielen will

Vorbedingungen:

- Spiel(Programm) wurde gestartet

Nachbedingungen:

- Wartebildschirm mit allen Spielern wird angezeigt

Erfolgsszenario

1. Das Spiel zeigt den Startbildschirm an
2. Der Spieler wählt das Feld für die Eingabe der maximalen Spieleranzahl aus
3. Der Spieler gibt eine Spieleranzahl an
4. Das Spiel überprüft die eingegebene Spieleranzahl
5. Der Spieler klickt auf den Button “Spiel starten”
6. Das. System erstellt die GUI fuer das Wartezeit Fenster
7. Das Spiel zeigt das Wartezeit Fenster an

Fehlerfall

5. Das System zeigt an, dass die angegebene Spieleranzahl kein korrekter Wert ist

Zugrundeliegende Anforderungen:

- UC-1 Spieleranzahl festlegen

Nummer: UC-3

Titel: Spiel Beitreten

Aktuer: Spieler

Ziel: Einem Spiel beizutreten.

Auslöser: Spieler hat die intention einem Spiel beizutreten

Vorbedingung:

- Bildschirm 2 wurde angezeigt
- Der Spieler hat die Anzahl der Spieler festgelegt
- Der Spieler hat ein Spiel gestartet

Nachbedingung:

- Der Benutzer ist dem Spiel beigetreten

Erfolgszenario:

1. Das System zeigt den Wartescreen an
2. Das System startet einen Countdown
3. Der jeweilige Spieler klickt auf eine Taste in einer der vordefinierten steuerungsbereichen

4. Das System merkt sich, dass ein neuer Spieler auf dem gerade gedrückten steuerungsbereich beigetreten ist
5. Sobald der Countdown zu ende ist, wird das Spiel gestartet

Erweiterungsfälle:

5.a. Es sind nicht alle Spieler beigetreten:

5.a.1 Das spiel wird nach einem bestimmten Countdown gestartet

Fehlerfälle:

5. b. Es sind nicht genug spieler eingetreten (weniger als 2):

5.b.1 Das Spiel wird abgebrochen und das System kehrt zurück zum start bildschirm

Häufigkeit:

- jedes mal wenn ein Spiel gestartet wird, müssen Spieler beitreten

Zugrundeliegende Anforderungen

- UC-1: Spieleranzahl festlegen
- UC-2: Spiel Starten

Nummer: UC-4

Titel: Bewegung eines spielers

Akteur: Der Spieler

Ziel: Der Spieler möchte, dass sich seine Spielfigur in der nächsten Bewegung in seine gewünschte Richtung bewegt

Auslöser: Spieler drückt eine Richtungstaste Vorbedingung:

Vorbedingungen:

- Der Spieler befindet sich in einem laufendem Spiel
- Der Countdown nach dem Start des Spiels ist abgelaufen Nachbedingung
- Die Spielefigur hat sich in die gewünschte Richtung bewegt

Erfolgsszenario

1. Das Spiel zeigt das Spiel an
2. Das Spiel zeigt den Spieler in der richtigen Richtung an

Nummer: UC-5

Titel: Spieler Sterben / verlieren

Aktuer: Spieler

Ziel: Ein Spieler stirbt im Spiel bzw. verliert das Spiel

Auslöser:

- Ein Spieler fährt gegen eine Wand
- Ein Spieler fährt gegein den Schatten eines anderen Spielers oder gegen seinen eigenen Schatten
- Ein Spieler fährt gegen ein anderes Motorrad
- Beide Spieler sterben wenn sie einen Frontalen zusammenstoß haben

Vorbedingungen:

- Der Spieler besitzt die Tron Applikation
- Der Spieler hat die Spieleranzahl wurde festgelegt
- Der Spieler hat das Spiel gestartet
- Spieler sind dem Spiel beigetreten

Nachbedingungen:

- Der Spieler kann nicht mehr weiter am Spiel teilnehmen
- Der Schatten des Spielers welcher gestorben ist, verschwindet aus dem Spiel

Erfolgszenario:

1. Ein Spieler spielt das Spiel und löst dabei einen der oben genannten Auslöser aus
2. Der Spieler verschwindet aus dem Spielfeld & sein Schatten verschwindet aus dem Spielfeld
3. Das Spiel wird weiter gespielt, ohne den gerade gestorbenen Spieler

Erweiterungsfälle:

1.a Wenn mehrere Spieler gleichzeitig sterben (durch gleiche oder verschiedene Auslöser)

1.a.1 Alle Spieler, welche gleichzeitig sterben, scheiden gleichzeitig aus dem Spiel aus & deren Schatten verschwindet aus dem Spiel

Häufigkeit:

- In jedem Spiel, welches Gespiel wird, werden Spieler sterben / verlieren.

Zugrundeliegende Anforderungen:

- UC-4 Bewegung eines spielers

Nummer: UC-6

Titel: Countdown zum Spielstart wird angezeigt

Akteur: System

Ziel: Die Spieler werden visuell auf den bevorstehenden Start des Spiels hingewiesen.

Auslöser: Abschluss des UC-3. (Spiel wurde gestartet)

Vorbedingung:

- Bildschirm 3 wird angezeigt.

Nachbedingung:

- Spiel ist gestartet.

Erfolgsszenario:

1. Das System erstellt die GUI fuer das Spielfeld
2. Bildschrim 3 (Spielfeld) wird angezeigt.
3. Es erscheinen in großen Buchstaben der Reihenfolge nach die Ausgaben “3”, “2”, “1”, “Go!” im Abstand von jeweils einer Sekunde.
4. Das Spiel startet.

Häufigkeit

- Bei jedem Spielstart.

Zugrundeliegende Anforderungen

- UC-3 Spieler treten dem Spiel bei

Nummer: UC-7

Titel: Startpositionen der Spieler

Akteur: Das System

Ziel: Alle Spieler besitzen Startpositionen, welche ein faires Spiel gewährleisten

Auslöser: Ein Spiel wurde gestartet

Vorbedingungen

- Ein Spiel wurde gestartet
- Das Spiel wurde geladen

Nachbedingungen: Spieler befinden sich auf fairen Startpositionen

Erfolgsszenario

1. Das System zeigt das Spiel an
2. Das System berechnet faire Startpositionen
3. Das System setzt die Spieler bei den Startpositionen ein

Zugrundeliegende Anforderungen:

- UC-2 starten eines Spiels

Nummer: UC-8

Titel: Spieler hinterlassen Schatten

Aktuer: Spieler

Ziel: Der Spieler bzw. das Motorrad des Spielers hinterlässt einen Schatten

Vorbedingung:

- Der Spieler besitzt die Applikation

- Der Spieler hat das Spiel gestartet
- Die Spieler sind dem Spiel beigetreten
- Die Spieler können sich auf dem Spielfeld bewegen

Nachbedingungen:

- Nach jeder Bewegung tuacht an der vorherigen Position des Spielers ein Schatten auf

Erfolgszenario:

1. Der Spieler bewegt sich auf dem Spielfeld
2. Bei jeder Bewegung des Spielers wird ein stück schatten mehr generiert und taucht hinter dem Spieler auf in der Farbe des Spielers

Erweiterungsfälle:

Fehlerfälle:

Häufigkeit:

- In jedem Spiel, welches gespielt wird, wird bei jedem Spieler ein Schatten hinterlassen

Zugrundeliegende Anforderungen

- UC-4 Bewegung eines spielers

Nummer: UC-9

Titel: Game over Screen

Aktuer: System

Ziel: Der Game over screen wird angezeigt mit den jeweiligen Gewinnern oder Unentschieden

Auslöser: Ein Spiel wurde beendet. bzw. in einem Spiel verbleiben weniger als 2 Spieler

Vorbedingung:

- Ein Spiel wurde gestartet und von mindestens 2 Spielern gespielt
- Das Spiel wurde beendet indem im Spiel weniger als 2 Spieler verblieben sind.

Nachbedingungen:

- Der Startbildschirm wird nach dem “Game over” screen angezeigt

Erfolgszenario:

1. Das Spiel wird beendet
2. Das System erstellt die GUI fuer den Game Over screen
3. Das System wechselt vom “Spielfeld” screen zum “Game over” screen
4. Das System startet einen 10 sek. countdown.

5. Auf dem Game Over Screen wird der Spieler angezeigt, welcher gewonnen hat. Dies wird durch “Spieler X” angezeigt, in der jeweiligen Spieler farbe
6. Sobald der 10 sek. countdown abgelaufen ist, wird auf den Startbildschirm der Applikation gewechsel.

Erweiterungsfälle:

4.a Wenn das Spiel mit einem Unentschieden beendet wurden ist

4.a.1: Anstatt “Spieler X” wird der Schriftzug “Unentschieden” angezeigt

Fehlerfälle:

Häufigkeit:

- Jedes mal, wenn ein Spiel gespielt wurden ist, wird dieser Screen angezeigt.

Zugrundeliegende Anforderungen:

- UC-5 Spieler sterben / verlieren

Funktionale Anforderungen

UC-1 Spieleranzahl festlegen

- void displayStartScreen()
  - Zeigt Bildschirm 1 an:
    - Ruft displayInputBox() auf
    - Zeigt den Start-Button an
  - Initialisiert UserInput-Handling
- void displayInputBox()
  - Zeigt das Eingabefeld für die Anzahl der Spieler an.
  - Das Eingabefeld wird so konfiguriert, dass nur numerische Eingaben übernommen werden.
- boolean checkInput(String input)
  - Überprüft die Nutzereingabe auf Einhaltung der minimalen und maximalen Spielerzahl und darauf, ob überhaupt eine Zahl eingegeben wurde
  - Bei fehlerhafter Eingabe wird displayError() aufgerufen, sonst setNumOfPlayers()
- void displayError(String msg)
  - Umrandet das Eingabefeld rot
  - Zeigt eine Fehlernachricht mit Hinweis auf die ungültige Eingabe an
  - Sperrt den Start-Button
- void setNumOfPlayers(int numOfPlayers)
  - Übernimmt numOfPlayers als maximale Spieleranzahl für den nächsten Spielstart

---

UC-2 Spiel Starten

- boolean startWaitingLobby()
    - Startet den Warte Screen mit countdown

UC-3: Spiel Beitreten

- void startCountdown(int)
    - Startet einen Countdownd fuer int sekunden
    - Nachdem der Countdown zu ende ist, wird das Spielfeld gerendert → renderGame()
- boolean countPlayers()
    - Zaehlt wie viele Spieler dem Spiel bisher beigetreten sind
    - setter der inkrementiert
    - Bei weniger als 2 Spielern wird quitGame() aufgerufen
- void checkKeys()
    - Checkt ob welche der vordefinierten keys aus der config datei gedrueckt werden, wenn ja, wird auf diesen keys createPlayerObject() aufgerufen mit der jeweiligen Tasten kombi als Param.
    - countPlayers() aufgerufen
- void quitGame()
    - Das Spiel wird beendet und es wird zum screen 1 gegangen
- void displayWaitingScreen()
    - Das System zeigt den Wartescreen an, bis alle spieler da sind oder der countdown vorbei ist.
- void renderGame()
    - holt sich die Spielfeld groeße etc. und baut das Spielfeld auf.
    - Im anschluss werden die spieler erzeugt → createPlayer() und die Spieler angezeigt
    - danach werden diese Spieler auf dem Spielfeld Positioniert → setStartPositions(Player)
- Player createPlayerObject(Array<String>)
    - in jeweils zufaelligen Farben
    - Erzeugt ein Spielerobjekt
    - Array<String> die Tasten belegung

UC-4 Spieler koennen ihre Figuren Steuern

- void changePlayerDirection(String)
    - wenn pfeiltaste etc. geklickt wird, wird die jeweilige Richtung eingeschlagen beim naechsten “spielzug”
    - Richtungen koennte man als ENUM (Attribut vom Spieler) modellieren, und jeweils in dieser Methode setzen
- void movePlayer(Player)
    - setzt den jeweiligen spieler immer ein kaestchen nach vorne pro zeitienheit
    - in die richtung, in der das ENUM gesetzt ist.

UC-5: Spieler Sterben / verlieren

- void removePlayer(Player)

    - entfernt den Spieler vom Spiel
    - entfernt auch seinen Schatten
    - wenn checkCollision true, wird removePlayer aufgerufen
- boolean checkPlayerCount()
    - Prueft waehrend des Spiels, wie viele Spieler sich noch im Spiel befinden.
    - Wenn weniger als 2, dann wird gameOverWinner(String) aufgerufen und damit spiel beendet
    - Wenn weniger als 1, dann wird gameOverTie() aufgerufen und damit spiel beendet
- boolean checkCollision(Player)
    - Prueft die positionen der Hindernisse (in einer liste gespeichert) und prueft, ob der Player gerade da gegen faehrt.
    - wenn true , wird removePlayer(Player) aufgerufen

UC-6 Countdown zum Spielstart wird angezeigt

- void displayGameStartCountdown()
    - Startet einen Countdown bis zum Spielstart
    - Ruft im Sekundentakt clearMessage() und displayMessage() mit den Parametern „3“,“2“,“1“,“GO!“ auf.
    - Ruft nach dem letzten Aufruf von displayMessage() startGame() auf
- void displayMessage(String msg)
    - Zeigt eine große Nachricht mit msg als Inhalt auf dem Bildschirm an.
- void clearMessage()
    - Entfernt die aktuell angezeigte Nachricht vom Bildschirm
- void startGame()
    - Startet das Spiel

UC-7 Startposition der Spieler

- void setStartPositions(List<Player>)
    - rechnet je nach spieleranzahl die faire start position der spieler auf dem Spielfeld aus. und setzt die Spieler objekte auf die position

---

UC-8: Spieler hinterlassen Schatten

- void createShadow(Player)
    - Erzeugt einen Schatten , also ein kaestchen mit schatten, an der letzten position des Spielers
    - Der Schatten wird in einer “obsticles” liste gespeichert oder so aehnlich
    - Diese Methode wird jedesmal aufgerufen, wenn sich der Spieler 1 nach vorne bewegt.

UC-9: Game Over Screen

- gameOverTie()
    - ruft den gameOver screen auf mit einem Schriftzug „unentschieden“

- gameOverWinner(Player)
    - Parameter = gewinner (player1. etc…)

Motivation

Die wesentliche Motivation für uns, dieses Spiel zu implementieren ist es, die PVL zu erhalten. Weitere Motivationspunkte wären aber auch, neues zu lernen und unser bisheriges Wissen zu vertiefen.

Qualitätsziele

Qualitätsziele:
Gut definierte Schnittstellen
Kompatibilität zu einer anderen Gruppe (Mindestens zwei Teams müssen miteinander Spielen können)
Fehlertoleranz (Wenn ein Spieler abstürzt, egal welcher Spieler, dann geht das Spiel trotzdem weiter) -> Stabilität
Das spiel soll gleich schnell laufen für alle (keine Jitter-abhängigkeit)
Ein Spiel mit 6 Lueten, soll einmal komplett ohne Fehler durchlaufen.

Stakeholder

Inhalt

Unsere Stakeholder sind die Entwickler (Studenten), der Kunde (Professor) und die Spieler (Studenten).

Die Stakeholder mit deren Kontakt werden in der unteren Tabelle aufgelistet:

Rolle	Kontakt	Erwartungshaltung
Entwickler	Dominik.martin@haw-hamburg.de	*Ein gutes Spiel zu programmieren & die PVL zu erhalten*

Rolle	Kontakt	Erwartungshaltung
Entwickler	*Can.heintze@haw-hamburg.de*	*Ein gutes Spiel zu programmieren & die PVL zu erhalten*
Entwickler	*Dominik.mueller@haw-hamburg.de*	*Ein gutes Spiel zu programmieren & die PVL zu erhalten*
Kunde	*Martin.becke@haw-hamburg.de*	*Ein Lauffaehiges Spiel, bei dem 6 Spieler gleichzeitig ein komplettes Spiel ohne fehler durchspielen koennen.*
Spieler	*n/a*	*Ein funktionierendes Spiel spielen und dabei Spaß haben*

Randbedingungen

Inhalt

Randbedingungen und Vorgaben, die ihre Freiheiten bezüglich Entwurf, Implementierung oder Ihres Entwicklungsprozesses einschränken. Diese Randbedingungen gelten manchmal organisations- oder firmenweit über die Grenzen einzelner Systeme hinweg.

Motivation

Für eine tragfähige Architektur sollten Sie genau wissen, wo Ihre Freiheitsgrade bezüglich der Entwurfsentscheidungen liegen und wo Sie Randbedingungen beachten müssen. Sie können Randbedingungen vielleicht noch verhandeln, zunächst sind sie aber da.

Form

Einfache Tabellen der Randbedingungen mit Erläuterungen. Bei Bedarf unterscheiden Sie technische, organisatorische und politische Randbedingungen oder übergreifende Konventionen (beispielsweise Programmier- oder Versionierungsrichtlinien, Dokumentations- oder Namenskonvention).

Siehe Randbedingungen in der online-Dokumentation (auf Englisch!).

Kontextabgrenzung

Inhalt

Die Kontextabgrenzung grenzt das System gegen alle Kommunikationspartner (Nachbarsysteme und Benutzerrollen) ab. Sie legt damit die externen Schnittstellen fest

und zeigt damit auch die Verantwortlichkeit (scope) Ihres Systems: Welche Verantwortung trägt das System und welche Verantwortung übernehmen die Nachbarsysteme?

Differenzieren Sie fachlichen (Ein- und Ausgaben) und technischen Kontext (Kanäle, Protokolle, Hardware), falls nötig.

Motivation

Die fachlichen und technischen Schnittstellen zur Kommunikation gehören zu den kritischsten Aspekten eines Systems. Stellen Sie sicher, dass Sie diese komplett verstanden haben.

Form

Verschiedene Optionen:

- Diverse Kontextdiagramme
- Listen von Kommunikationsbeziehungen mit deren Schnittstellen

Siehe Kontextabgrenzung in der online-Dokumentation (auf Englisch!).

Fachlicher Kontext

Inhalt

Festlegung **aller** Kommunikationsbeziehungen (Nutzer, IT-Systeme, …) mit Erklärung der fachlichen Ein- und Ausgabedaten oder Schnittstellen. Zusätzlich (bei Bedarf) fachliche Datenformate oder Protokolle der Kommunikation mit den Nachbarsystemen.

Motivation

Alle Beteiligten müssen verstehen, welche fachlichen Informationen mit der Umwelt ausgetauscht werden.

Form

Alle Diagrammarten, die das System als Blackbox darstellen und die fachlichen Schnittstellen zu den Nachbarsystemen beschreiben.

Alternativ oder ergänzend können Sie eine Tabelle verwenden. Der Titel gibt den Namen Ihres Systems wieder; die drei Spalten sind: Kommunikationsbeziehung, Eingabe, Ausgabe.

<Diagramm und/oder Tabelle>

<optional: Erläuterung der externen fachlichen Schnittstellen>

Technischer Kontext

Inhalt

Technische Schnittstellen (Kanäle, Übertragungsmedien) zwischen dem System und seiner Umwelt. Zusätzlich eine Erklärung (*mapping*), welche fachlichen Ein- und Ausgaben über welche technischen Kanäle fließen.

Motivation

Viele Stakeholder treffen Architekturentscheidungen auf Basis der technischen Schnittstellen des Systems zu seinem Kontext.

Insbesondere bei der Entwicklung von Infrastruktur oder Hardware sind diese technischen Schnittstellen durchaus entscheidend.

Form

Beispielsweise UML Deployment-Diagramme mit den Kanälen zu Nachbarsystemen, begleitet von einer Tabelle, die Kanäle auf Ein-/Ausgaben abbildet.

<Diagramm oder Tabelle>

<optional: Erläuterung der externen technischen Schnittstellen>

<Mapping fachliche auf technische Schnittstellen>

Lösungsstrategie

Inhalt

Kurzer Überblick über die grundlegenden Entscheidungen und Lösungsansätze, die Entwurf und Implementierung des Systems prägen. Hierzu gehören:

- Technologieentscheidungen
- Entscheidungen über die Top-Level-Zerlegung des Systems, beispielsweise die Verwendung gesamthaft prägender Entwurfs- oder Architekturmuster,
- Entscheidungen zur Erreichung der wichtigsten Qualitätsanforderungen sowie
- relevante organisatorische Entscheidungen, beispielsweise für bestimmte Entwicklungsprozesse oder Delegation bestimmter Aufgaben an andere Stakeholder.

Motivation

Diese wichtigen Entscheidungen bilden wesentliche „Eckpfeiler“ der Architektur. Von ihnen hängen viele weitere Entscheidungen oder Implementierungsregeln ab.

Form

Fassen Sie die zentralen Entwurfsentscheidungen **kurz** zusammen. Motivieren Sie, ausgehend von Aufgabenstellung, Qualitätszielen und Randbedingungen, was Sie

entschieden haben und warum Sie so entschieden haben. Vermeiden Sie redundante Beschreibungen und verweisen Sie eher auf weitere Ausführungen in Folgeabschnitten.

Siehe Lösungsstrategie in der online-Dokumentation (auf Englisch!).

Bausteinsicht

Inhalt

Die Bausteinsicht zeigt die statische Zerlegung des Systems in Bausteine (Module, Komponenten, Subsysteme, Klassen, Schnittstellen, Pakete, Bibliotheken, Frameworks, Schichten, Partitionen, Tiers, Funktionen, Makros, Operationen, Datenstrukturen, …) sowie deren Abhängigkeiten (Beziehungen, Assoziationen, …)

Diese Sicht sollte in jeder Architekturdokumentation vorhanden sein. In der Analogie zum Hausbau bildet die Bausteinsicht den *Grundrissplan*.

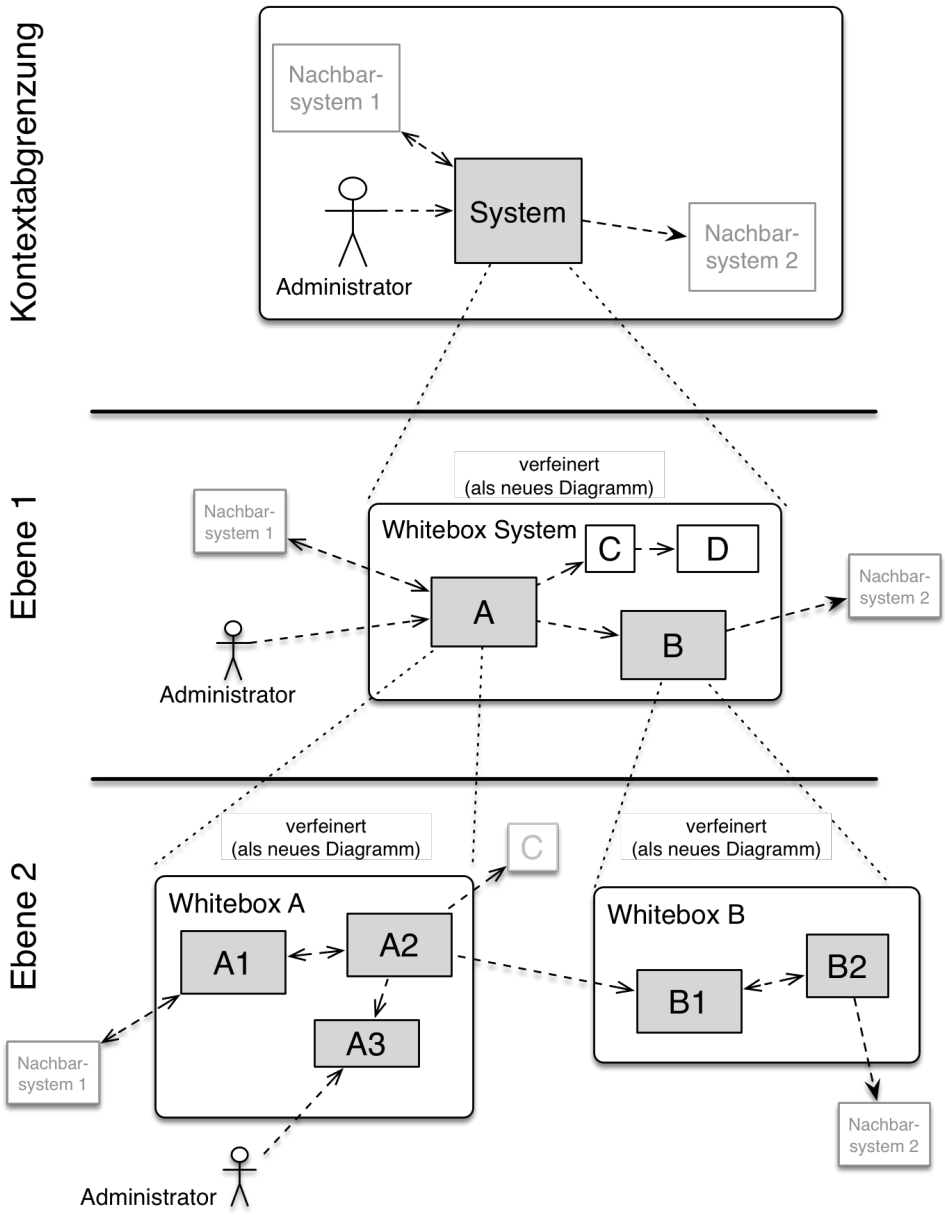
Motivation

Behalten Sie den Überblick über den Quellcode, indem Sie die statische Struktur des Systems durch Abstraktion verständlich machen.

Damit ermöglichen Sie Kommunikation auf abstrakterer Ebene, ohne zu viele Implementierungsdetails offenlegen zu müssen.

Form

Die Bausteinsicht ist eine hierarchische Sammlung von Blackboxen und Whiteboxen (siehe Abbildung unten) und deren Beschreibungen.



Ebene 1 ist die Whitebox-Beschreibung des Gesamtsystems, zusammen mit Blackbox-Beschreibungen der darin enthaltenen Bausteine.

Ebene 2 zoomt in einige Bausteine der Ebene 1 hinein. Sie enthält somit die Whitebox-Beschreibungen ausgewählter Bausteine der Ebene 1, jeweils zusammen mit Blackbox-Beschreibungen darin enthaltener Bausteine.

Ebene 3 zoomt in einige Bausteine der Ebene 2 hinein, usw.

Siehe Bausteinsicht in der online-Dokumentation (auf Englisch!).

Whitebox Gesamtsystem

An dieser Stelle beschreiben Sie die Zerlegung des Gesamtsystems anhand des nachfolgenden Whitebox-Templates. Dieses enthält:

- Ein Übersichtsdiagramm
- die Begründung dieser Zerlegung
- Blackbox-Beschreibungen der hier enthaltenen Bausteine. Dafür haben Sie verschiedene Optionen:
  - in *einer* Tabelle, gibt einen kurzen und pragmatischen Überblick über die enthaltenen Bausteine sowie deren Schnittstellen.
  - als Liste von Blackbox-Beschreibungen der Bausteine, gemäß dem Blackbox-Template (siehe unten). Diese Liste können Sie, je nach Werkzeug, etwa in Form von Unterkapiteln (Text), Unter-Seiten (Wiki) oder geschachtelten Elementen (Modellierungswerkzeug) darstellen.
- (optional:) wichtige Schnittstellen, die nicht bereits im Blackbox-Template eines der Bausteine erläutert werden, aber für das Verständnis der Whitebox von zentraler Bedeutung sind. Aufgrund der vielfältigen Möglichkeiten oder Ausprägungen von Schnittstellen geben wir hierzu kein weiteres Template vor. Im schlimmsten Fall müssen Sie Syntax, Semantik, Protokolle, Fehlerverhalten, Restriktionen, Versionen, Qualitätseigenschaften, notwendige Kompatibilitäten und vieles mehr spezifizieren oder beschreiben. Im besten Fall kommen Sie mit Beispielen oder einfachen Signaturen zurecht.

<Übersichtsdiagramm>

Begründung
<Erläuternder Text>

Enthaltene Bausteine
<Beschreibung der enthaltenen Bausteine (Blackboxen)>

Wichtige Schnittstellen
<Beschreibung wichtiger Schnittstellen>

Hier folgen jetzt Erläuterungen zu Blackboxen der Ebene 1.

Falls Sie die tabellarische Beschreibung wählen, so werden Blackboxen darin nur mit Name und Verantwortung nach folgendem Muster beschrieben:

Name	Verantwortung
<Blackbox 1>	*<Text>*
<Blackbox 2>	*<Text>*

Falls Sie die ausführliche Liste von Blackbox-Beschreibungen wählen, beschreiben Sie jede wichtige Blackbox in einem eigenen Blackbox-Template. Dessen Überschrift ist jeweils der Namen dieser Blackbox.

<Name Blackbox 1>

Beschreiben Sie die <Blackbox 1> anhand des folgenden Blackbox-Templates:

- Zweck/Verantwortung
- Schnittstelle(n), sofern diese nicht als eigenständige Beschreibungen herausgezogen sind. Hierzu gehören eventuell auch Qualitäts- und Leistungsmerkmale dieser Schnittstelle.
- (Optional) Qualitäts-/Leistungsmerkmale der Blackbox, beispielsweise Verfügbarkeit, Laufzeitverhalten o. Ä.
- (Optional) Ablageort/Datei(en)
- (Optional) Erfüllte Anforderungen, falls Sie Traceability zu Anforderungen benötigen.
- (Optional) Offene Punkte/Probleme/Risiken

<Zweck/Verantwortung>

<Schnittstelle(n)>

<(Optional) Qualitäts-/Leistungsmerkmale>

<(Optional) Ablageort/Datei(en)>

<(Optional) Erfüllte Anforderungen>

<(optional) Offene Punkte/Probleme/Risiken>

<Name Blackbox 2>

<Blackbox-Template>

<Name Blackbox n>

<Blackbox-Template>

<Name Schnittstelle 1>

…

<Name Schnittstelle m>

Ebene 2

Beschreiben Sie den inneren Aufbau (einiger) Bausteine aus Ebene 1 als Whitebox.

Welche Bausteine Ihres Systems Sie hier beschreiben, müssen Sie selbst entscheiden. Bitte stellen Sie dabei Relevanz vor Vollständigkeit. Skizzieren Sie wichtige, überraschende, riskante, komplexe oder besonders volatile Bausteine. Normale, einfache oder standardisierte Teile sollten Sie weglassen.

Whitebox *<Baustein 1>*

…zeigt das Innenleben von *Baustein 1*.

<Whitebox-Template>

Whitebox *<Baustein 2>*

<Whitebox-Template>

…

Whitebox *<Baustein m>*

<Whitebox-Template>

Ebene 3

Beschreiben Sie den inneren Aufbau (einiger) Bausteine aus Ebene 2 als Whitebox.

Bei tieferen Gliederungen der Architektur kopieren Sie diesen Teil von arc42 für die weiteren Ebenen.

Whitebox <_Baustein x.1_>

…zeigt das Innenleben von *Baustein x.1*.

<Whitebox-Template>

Whitebox <_Baustein x.2_>

<Whitebox-Template>

Whitebox <_Baustein y.1_>

<Whitebox-Template>

Laufzeitsicht

Inhalt

Diese Sicht erklärt konkrete Abläufe und Beziehungen zwischen Bausteinen in Form von Szenarien aus den folgenden Bereichen:

- Wichtige Abläufe oder *Features*: Wie führen die Bausteine der Architektur die wichtigsten Abläufe durch?

- Interaktionen an kritischen externen Schnittstellen: Wie arbeiten Bausteine mit Nutzern und Nachbarsystemen zusammen?
- Betrieb und Administration: Inbetriebnahme, Start, Stop.
- Fehler- und Ausnahmeszenarien

Anmerkung: Das Kriterium für die Auswahl der möglichen Szenarien (d.h. Abläufe) des Systems ist deren Architekturrelevanz. Es geht nicht darum, möglichst viele Abläufe darzustellen, sondern eine angemessene Auswahl zu dokumentieren.

Motivation

Sie sollten verstehen, wie (Instanzen von) Bausteine(n) Ihres Systems ihre jeweiligen Aufgaben erfüllen und zur Laufzeit miteinander kommunizieren.

Nutzen Sie diese Szenarien in der Dokumentation hauptsächlich für eine verständlichere Kommunikation mit denjenigen Stakeholdern, die die statischen Modelle (z.B. Bausteinsicht, Verteilungssicht) weniger verständlich finden.

Form

Für die Beschreibung von Szenarien gibt es zahlreiche Ausdrucksmöglichkeiten. Nutzen Sie beispielsweise:

- Nummerierte Schrittfolgen oder Aufzählungen in Umgangssprache
- Aktivitäts- oder Flussdiagramme
- Sequenzdiagramme
- BPMN (Geschäftsprozessmodell und -notation) oder EPKs (Ereignis-Prozessketten)
- Zustandsautomaten
- …

Siehe Laufzeitsicht in der online-Dokumentation (auf Englisch!).

<Bezeichnung Laufzeitszenario 1>

- <hier Laufzeitdiagramm oder Ablaufbeschreibung einfügen>
- <hier Besonderheiten bei dem Zusammenspiel der Bausteine in diesem Szenario erläutern>

<Bezeichnung Laufzeitszenario 2>

…

<Bezeichnung Laufzeitszenario n>

…

Verteilungssicht

Inhalt

Die Verteilungssicht beschreibt:

1. die technische Infrastruktur, auf der Ihr System ausgeführt wird, mit Infrastrukturelementen wie Standorten, Umgebungen, Rechnern, Prozessoren, Kanälen und Netztopologien sowie sonstigen Bestandteilen, und

2. die Abbildung von (Software-)Bausteinen auf diese Infrastruktur.

Häufig laufen Systeme in unterschiedlichen Umgebungen, beispielsweise Entwicklung-/Test- oder Produktionsumgebungen. In solchen Fällen sollten Sie alle relevanten Umgebungen aufzeigen.

Nutzen Sie die Verteilungssicht insbesondere dann, wenn Ihre Software auf mehr als einem Rechner, Prozessor, Server oder Container abläuft oder Sie Ihre Hardware sogar selbst konstruieren.

Aus Softwaresicht genügt es, auf die Aspekte zu achten, die für die Softwareverteilung relevant sind. Insbesondere bei der Hardwareentwicklung kann es notwendig sein, die Infrastruktur mit beliebigen Details zu beschreiben.

Motivation

Software läuft nicht ohne Infrastruktur. Diese zugrundeliegende Infrastruktur beeinflusst Ihr System und/oder querschnittliche Lösungskonzepte, daher müssen Sie diese Infrastruktur kennen.

Form

Das oberste Verteilungsdiagramm könnte bereits in Ihrem technischen Kontext enthalten sein, mit Ihrer Infrastruktur als EINE Blackbox. Jetzt zoomen Sie in diese Infrastruktur mit weiteren Verteilungsdiagrammen hinein:

- Die UML stellt mit Verteilungsdiagrammen (Deployment diagrams) eine Diagrammart zur Verfügung, um diese Sicht auszudrücken. Nutzen Sie diese, evtl. auch geschachtelt, wenn Ihre Verteilungsstruktur es verlangt.

- Falls Ihre Infrastruktur-Stakeholder andere Diagrammarten bevorzugen, die beispielsweise Prozessoren und Kanäle zeigen, sind diese hier ebenfalls einsetzbar.

Siehe Verteilungssicht in der online-Dokumentation (auf Englisch!).

Infrastruktur Ebene 1

An dieser Stelle beschreiben Sie (als Kombination von Diagrammen mit Tabellen oder Texten):

- die Verteilung des Gesamtsystems auf mehrere Standorte, Umgebungen, Rechner, Prozessoren o. Ä., sowie die physischen Verbindungskanäle zwischen diesen,
- wichtige Begründungen für diese Verteilungsstruktur,
- Qualitäts- und/oder Leistungsmerkmale dieser Infrastruktur,
- Zuordnung von Softwareartefakten zu Bestandteilen der Infrastruktur

Für mehrere Umgebungen oder alternative Deployments kopieren Sie diesen Teil von arc42 für alle wichtigen Umgebungen/Varianten.

<Übersichtsdiagramm>

Begründung
<Erläuternder Text>

Qualitäts- und/oder Leistungsmerkmale
<Erläuternder Text>

Zuordnung von Bausteinen zu Infrastruktur
<Beschreibung der Zuordnung>

Infrastruktur Ebene 2

An dieser Stelle können Sie den inneren Aufbau (einiger) Infrastrukturelemente aus Ebene 1 beschreiben.

Für jedes Infrastrukturelement kopieren Sie die Struktur aus Ebene 1.

<Infrastrukturelement 1>

<Diagramm + Erläuterungen>

<Infrastrukturelement 2>

<Diagramm + Erläuterungen>

...

<Infrastrukturelement n>

<Diagramm + Erläuterungen>

Querschnittliche Konzepte

Inhalt

Dieser Abschnitt beschreibt übergreifende, prinzipielle Regelungen und Lösungsansätze, die an mehreren Stellen (=*querschnittlich*) relevant sind.

Solche Konzepte betreffen oft mehrere Bausteine. Dazu können vielerlei Themen gehören, beispielsweise:

- Modelle, insbesondere fachliche Modelle
- Architektur- oder Entwurfsmuster
- Regeln für den konkreten Einsatz von Technologien
- prinzipielle — meist technische — Festlegungen übergreifender Art
- Implementierungsregeln

Motivation

Konzepte bilden die Grundlage für *konzeptionelle Integrität* (Konsistenz, Homogenität) der Architektur und damit eine wesentliche Grundlage für die innere Qualität Ihrer Systeme.

Manche dieser Themen lassen sich nur schwer als Baustein in der Architektur unterbringen (z.B. das Thema „Sicherheit“).

Form

Kann vielfältig sein:

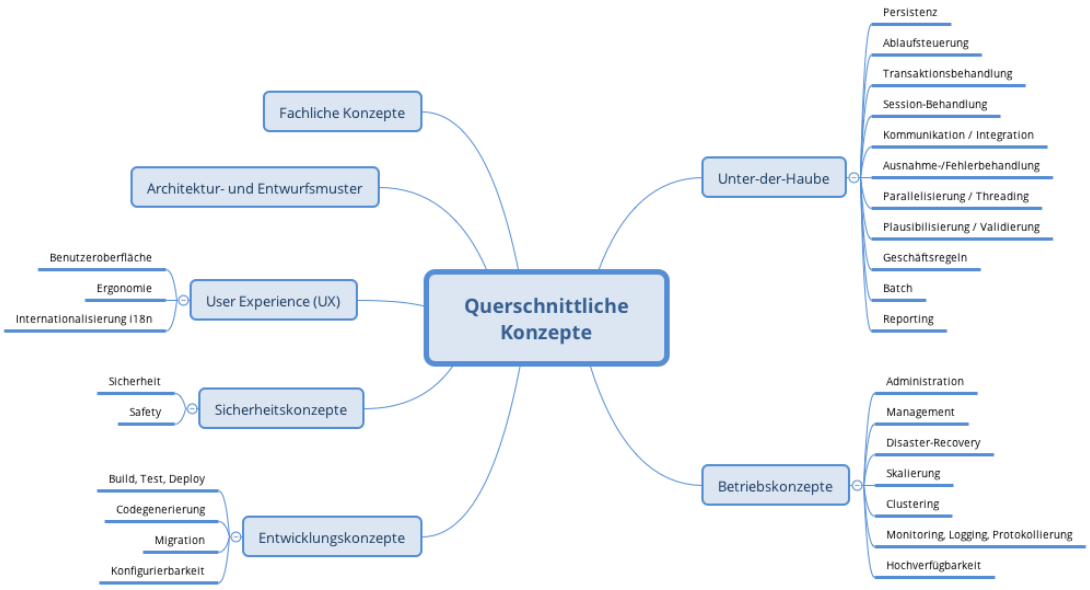
- Konzeptpapiere mit beliebiger Gliederung,
- übergreifende Modelle/Szenarien mit Notationen, die Sie auch in den Architektursichten nutzen,
- beispielhafte Implementierung speziell für technische Konzepte,
- Verweise auf „übliche“ Nutzung von Standard-Frameworks (beispielsweise die Nutzung von Hibernate als Object/Relational Mapper).

Struktur

Eine mögliche (nicht aber notwendige!) Untergliederung dieses Abschnittes könnte wie folgt aussehen (wobei die Zuordnung von Themen zu den Gruppen nicht immer eindeutig ist):

- Fachliche Konzepte
- User Experience (UX)
- Sicherheitskonzepte (Safety und Security)
- Architektur- und Entwurfsmuster
- Unter-der-Haube
- Entwicklungskonzepte

- Betriebskonzepte



Siehe Querschnittliche Konzepte in der online-Dokumentation (auf Englisch).

<Konzept 1>

<Erklärung>

<Konzept 2>

<Erklärung>

...

<Konzept n>

<Erklärung>

Architekturentscheidungen

Inhalt

Wichtige, teure, große oder riskante Architektur- oder Entwurfsentscheidungen inklusive der jeweiligen Begründungen. Mit "Entscheidungen" meinen wir hier die Auswahl einer von mehreren Alternativen unter vorgegebenen Kriterien.

Wägen Sie ab, inwiefern Sie Entscheidungen hier zentral beschreiben, oder wo eine lokale Beschreibung (z.B. in der Whitebox-Sicht von Bausteinen) sinnvoller ist. Vermeiden Sie Redundanz. Verweisen Sie evtl. auf Abschnitt 4, wo schon grundlegende strategische Entscheidungen beschrieben wurden.

Motivation

Stakeholder des Systems sollten wichtige Entscheidungen verstehen und nachvollziehen können.

Form

Verschiedene Möglichkeiten:

- ADR (Documenting Architecture Decisions) für jede wichtige Entscheidung
- Liste oder Tabelle, nach Wichtigkeit und Tragweite der Entscheidungen geordnet
- ausführlicher in Form einzelner Unterkapitel je Entscheidung

Siehe Architekturentscheidungen in der arc42 Dokumentation (auf Englisch!). Dort finden Sie Links und Beispiele zum Thema ADR.

Qualitätsanforderungen

Inhalt

Dieser Abschnitt enthält möglichst alle Qualitätsanforderungen als Qualitätsbaum mit Szenarien. Die wichtigsten davon haben Sie bereits in Abschnitt 1.2 (Qualitätsziele) hervorgehoben.

Nehmen Sie hier auch Qualitätsanforderungen geringerer Priorität auf, deren Nichteinhaltung oder -erreichung geringe Risiken birgt.

Motivation

Weil Qualitätsanforderungen die Architekturentscheidungen oft maßgeblich beeinflussen, sollten Sie die für Ihre Stakeholder relevanten Qualitätsanforderungen kennen, möglichst konkret und operationalisiert.

Weiterführende Informationen

Siehe Qualitätsanforderungen in der online-Dokumentation (auf Englisch!).

Qualitätsbaum

Inhalt

Der Qualitätsbaum (à la ATAM) mit Qualitätsszenarien an den Blättern.

Motivation

Die mit Prioritäten versehene Baumstruktur gibt Überblick über die — oftmals zahlreichen — Qualitätsanforderungen.

- Baumartige Verfeinerung des Begriffes „Qualität“, mit „Qualität“ oder „Nützlichkeit“ als Wurzel.
- Mindmap mit Qualitätsoberbegriffen als Hauptzweige

In jedem Fall sollten Sie hier Verweise auf die Qualitätsszenarien des folgenden Abschnittes aufnehmen.

Qualitätsszenarien

Inhalt

Konkretisierung der (in der Praxis oftmals vagen oder impliziten) Qualitätsanforderungen durch (Qualitäts-)Szenarien.

Diese Szenarien beschreiben, was beim Eintreffen eines Stimulus auf ein System in bestimmten Situationen geschieht.

Wesentlich sind zwei Arten von Szenarien:

- Nutzungsszenarien (auch bekannt als Anwendungs- oder Anwendungsfallszenarien) beschreiben, wie das System zur Laufzeit auf einen bestimmten Auslöser reagieren soll. Hierunter fallen auch Szenarien zur Beschreibung von Effizienz oder Performance. Beispiel: Das System beantwortet eine Benutzeranfrage innerhalb einer Sekunde.
- Änderungsszenarien beschreiben eine Modifikation des Systems oder seiner unmittelbaren Umgebung. Beispiel: Eine zusätzliche Funktionalität wird implementiert oder die Anforderung an ein Qualitätsmerkmal ändert sich.

Motivation

Szenarien operationalisieren Qualitätsanforderungen und machen deren Erfüllung mess- oder entscheidbar.

Insbesondere wenn Sie die Qualität Ihrer Architektur mit Methoden wie ATAM überprüfen wollen, bedürfen die in Abschnitt 1.2 genannten Qualitätsziele einer weiteren Präzisierung bis auf die Ebene von diskutierbaren und nachprüfbaren Szenarien.

Form

Entweder tabellarisch oder als Freitext.

Risiken und technische Schulden

Inhalt

Eine nach Prioritäten geordnete Liste der erkannten Architekturrisiken und/oder technischen Schulden.

> Risikomanagement ist Projektmanagement für Erwachsene.
>
> — Tim Lister Atlantic Systems Guild

Unter diesem Motto sollten Sie Architekturrisiken und/oder technische Schulden gezielt ermitteln, bewerten und Ihren Management-Stakeholdern (z.B. Projektleitung, Product-Owner) transparent machen.

Form

Liste oder Tabelle von Risiken und/oder technischen Schulden, eventuell mit vorgeschlagenen Maßnahmen zur Risikovermeidung, Risikominimierung oder dem Abbau der technischen Schulden.

Siehe Risiken und technische Schulden in der online-Dokumentation (auf Englisch!).

Glossar

Inhalt

Die wesentlichen fachlichen und technischen Begriffe, die Stakeholder im Zusammenhang mit dem System verwenden.

Nutzen Sie das Glossar ebenfalls als Übersetzungsreferenz, falls Sie in mehrsprachigen Teams arbeiten.

Motivation

Sie sollten relevante Begriffe klar definieren, so dass alle Beteiligten

- diese Begriffe identisch verstehen, und
- vermeiden, mehrere Begriffe für die gleiche Sache zu haben.
- Zweispaltige Tabelle mit <Begriff> und <Definition>
- Eventuell weitere Spalten mit Übersetzungen, falls notwendig.

Siehe Glossar in der online-Dokumentation (auf Englisch!).

Begriff	Definition
<Begriff-1>	*<Definition-1>*
<Begriff-2	*<Definition-2>*