### Aufgabe 2 b) Analyse des Use-Case „Ticket Buchung“

Dieser Abschnitt umfasst das Feststellen der NFAs für den Use-Case „Ticket Buchung“, Kommunikationsarten für Systeme, welche Kommunikationsart für das geforderte System am sinnvollsten ist und wie sich die ausgewählte Kommunikationsart auf die NFAs auswirkt. Um die NFAs für diesen Use-Case bestimmen zu können, wird zunächst betrachtet, welche Komponenten benötigt werden und wie überhaupt wie eine „Ticket Buchung“ ablauft.

Durch die Anforderungen an das System, ist das klassische kaufen eines Tickets nicht mehr notwendig, da das System eine bzw. mehrere Fahrten in einem bestimmten Zeitraum zusammenrechnet und somit eine optimale Abrechnung für den Zuggast bereitstellt. Es steht dem Zuggast jedoch frei, eine Reservierung über das Internet oder einen Bahnschalter zu tätigen. Mit diesen beiden Fällen ist der eigentliche Use-Case gemeint, da hier eine vorab Buchung stattfindet.

Die beiden Fälle unterscheiden sich jedoch nicht wirklich. Nur der Einstiegspunkt für den Prozess ist anders. Bei einem Fall nutzt der Zuggast die Möglichkeit über die Internetseite der Bahn, eine Fahrt zu reservieren, was eine Buchung der Strecke automatisch beinhaltet. Dies findet also auf einem externen Gerät statt, welches nicht der Bahn gehört. Da die Internetseite jedoch auf Hardware der Bahn gehostet ist, wird der Prozess jedoch Bahn intern angestoßen. Ähnlich findet der Prozess statt, wenn der Zuggast eine Reservierung über einen Bahnschalter tätigt. Dieser wird auch von der Bahn selbst verwaltet und ist somit intern. Daraus folgt das der Prozess immer auf interner Seite angestoßen wird.

Nachdem der Zuggast also eine Reservierung/Buchung vorgenommen hat, wird das Ticket-Buchungssystem angesteuert. Dort wird dann das Abrechnungssystem angesteuert, damit auch sichergestellt wird, dass das gebuchte Ticket bezahlt wird, da eine weitere Bearbeitung des Prozesses sonst keinen Sinn machen würde. Nachdem sichergestellt worden ist, dass das Ticket bezahlt ist, wird das Sitzplatz-Reservierungssystem darüber informiert, dass es in der Sitzplatzverwaltung eine neue Reservierung hinterlegen muss und somit ein Platz weniger belegt werden kann, durch einen Zuggast, welcher keine Reservierung/Buchung abgeschlossen hat. Zum gleichen Zeitpunkt kann jedoch dem Zuggast bereits ein Feedback gegeben werden, dass die Abbuchung erfolgreich war und sein Ticket im System hinterlegt ist. Natürlich muss das System in der Lage sein, bei einer Ticketbuchung, bei der mehr als ein Ticket gekauft worden ist, die Sitzplätze beim Einsteigen dann so zu verteilen, das alle Personen zumindest in unmittelbarer Nähe sitzen. Es sollte nicht zu dem Fall kommen, das beispielweise eine Mutter in einem Zugabschnitt sitzt und ihre Kinder in einem anderen.

Der nächste Abschnitt beschäftigt sich mit den NFAs die speziell für diesen Use-Case sind.

* Funktionalität:
  + Angemessenheit: Das System muss die Funktionen bereitstellen, damit dieser Prozess/Use-Case durchgeführt werden kann.
  + Richtigkeit: Es muss gewährleistest werden, dass die Daten, die genutzt werden, bei allen Komponenten identisch sind. Das Abbuchen bei einer anderen Person muss zu 100% ausgeschlossen sein.
  + Interoperabilität: Für diesen Prozess muss die Interoperabilität nicht ausgeprägt sein, da das System sich stark nach diesem Prozess richten muss und nicht anders rum.
  + Konformität: Es müssen für den Ablauf des Prozesses ausschließlich gesetzliche Bedingungen eingehalten werden. Konventionen gibt es an dieser Stelle keine.
  + Ordnungsmäßigkeit: Der Prozess muss sich an die gesetzlichen Bedingungen halten. Weitere anwendungsspezifischen Regelungen gibt es nicht.
  + Sicherheit: Es muss sichergestellt werden, das während der ganzen Transaktion keine Daten abgefangen werden können. Dafür sollte eine geeignete Verschlüsselung genutzt werden, weil besonders bei dieser Transaktion Bankdaten des jeweiligen Zuggasts involviert sind. Die Daten müssen zu 100% geschützt sein.
* Zuverlässigkeit:
  + Reife: Es muss möglich sein, auch bei teilweise auftretenden Fehlern ein abschließen des Prozesses zu garantieren. Es muss ein hoher Grad an Reife vorhanden sein.
  + Konformität: Die Konformität bei diesem Prozess muss hoch sein, denn dieser ist ein häufiger Prozess, sobald das System in der Produktion eingesetzt wird.
  + Fehlertoleranz: Der Prozess hat eine geringe Fehlertoleranz, daher müssen die für den Prozess wichtigen Komponenten zu jedem Zeitpunkt funktionieren.
  + Wiederherstellbarkeit:
* Benutzbarkeit:
  + Verständlichkeit: Muss gering sein. Bezieht sich auf das TBs und nicht das Bahn 2.0 System.
  + Erlernbarkeit: Die Erlernbarkeit muss leicht sein. Diese hängt von anderen Faktoren ab. Dieser Faktor ist jedoch nicht messbar für unser System, sondern das Fremdsystem TBs.
  + Konformität: Der Grad der Konformität bezüglich der Benutzbarkeit muss nicht groß sein, da im System nur der Prozess abgewickelt wird und die UI auf dem TBs.
  + Attraktivität: Nicht relevant für das Bahn 2.0 System. Wichtiger für NFAs des TBs.
  + Bedienbarkeit: Siehe „Attraktivität“
* Effizienz:
  + Zeitverhalten: Das System muss in der Lage sein, dem Fremdsystem eine Rückmeldung in unter fünf Sekunden zu geben.
  + Konformität: Die Konformität im Bereich Effizienz muss sich an die Vereinbarungen mit dem Kunden orientieren. Beispielsweise unter fünf Sekunden Antwortzeit.
  + Verbrauchsverhalten: Die Hardware muss in der Lage sein, auch bei höherer Auslastung einen schnellen Lese-/Schreibzugriff zu gewährleisten. Der Prozess muss auch ressourcensparend umgesetzt sein, da hier keine größeren Änderungen von Nöten ist.
* Änderbarkeit:
  + Analysierbarkeit: Es muss eine Dokumentation über die gesamte Software angelegt und verwaltet werden, damit eine gute Analysierbarkeit gewährleistet kann. Es handelt sich um eine größere Software, die über Generationen gewartet wird. Wird ein Fehler in der Dokumentation gemacht, hat dies starke Auswirkungen auf die Analysierbarkeit. Die Analysierbarkeit des Systems muss groß sein, was bedeutet, dass ein geringer Aufwand, für das durchsuchen des Quellcodes, von Nöten ist.
  + Konformität: Die Software muss einen mittleren Grad an Konformität im Bereich der Änderbarkeit aufweisen. Eine Änderung des Systems sollte nicht oft vorkommen.
  + Modifizierbarkeit: Eine gute Modifizierbarkeit im Bereich der Optimierung muss nicht gewährleistet werden, jedoch im Bereich der Fehlerbeseitigung. Grundsätzlich sollte für so etwas ein ausführlicher Testplan geschrieben werden.
  + Stabilität: Das System muss über einen hohen Grad an Stabilität verfügen, da es sich hier um eine Software handelt, ohne die der Konzern nicht arbeiten kann.
  + Testbarkeit: Der Aufwand für das Testen einer Änderung muss gering sein.
* Übertragbarkeit:
  + Anpassbarkeit: Die Software muss in der Lage sein, auch bei einer Umsiedelung oder einer Datenbankmigration, schnellst möglich wieder einsatzbereit zu sein. Ein maximaler Ausfall von acht Stunden muss gewährleistet sein (Umzug über Nacht).
  + Installierbarkeit: Der Aufwand zur Installation der Software muss einfach und unkompliziert sein.
  + Austauschbarkeit: Das System muss austauschbar sein.
  + Konformität: Das System muss eine hohe Konformität im Bereich der Übertragbarkeit bereitstellen.
  + Koexistenz: Das System muss in der Lage sein, auch in einem Dualbetrieb einwandfrei zu Laufen.

Nachdem wir im oberen Abschnitt eine Analyse durchgeführt haben, welche NFAs wichtig für unseren Use-Case sind, schauen wir uns nun an, wie sich das System unter den jeweiligen Kommunikationsarten verhalten würde. Zum besseren Verständnis, folgt zunächst aber eine Erläuterung zu den drei Kommunikationsarten synchron, asynchron und caching.



Abbildung Unterschied zwischen Synchroner und Asynchroner Kommunikation

Anhand von Abbildung 1 kann man gut den unterschied von synchroner und asynchroner Kommunikation erklären. Synchrone und asynchrone Kommunikation funktionieren so, dass ein Prozess (Prozess A) beispielweise einen anderen Prozess (Prozess B) anstößt. Bis hier hin unterscheiden sich synchron und asynchron noch nicht. Der entscheide unterschied folgt nun, denn synchrone Kommunikation unterbricht Prozess A und wartet bis Prozess B abgeschlossen ist und gegebenenfalls einen Rückgabewert liefert. Da bedeutet also, sollte Prozess B einen längeren Zeitraum benötigen, so wartet Prozess A auch so lange. Daraus ergibt sich dann folgende Laufzeit:

Bei der asynchronen Kommunikation hingegen, wird Prozess A nicht unterbrochen und läuft weiter durch. Also:

Wann sollte man nun eine synchrone und wann eine asynchrone Kommunikation verwenden? Grundsätzlich dient als grober Leitfaden, ob sich beide Prozesse im selben System befinden. Ist dies nicht der Fall, sollte man asynchrone Kommunikation wählen, weil man nicht abschätzen kann, wie viel Zeit der Prozess des Fremdsystem benötigt und somit das eigene System unterbrochen wird. Ist im Gegenzug der andere Prozess im selben System (schließt Subsysteme mit ein), kann man im Normalfall abschätzen wie lange der Prozess dauert. Meist wird hier dann auf eine synchrone Kommunikation gesetzt, wenn man weiß, dass der Prozess keine längere Zeit benötigt und der aufrufende Prozess diese Zeit warten kann.

Als weitere Kommunikationsart gibt es noch Caching, welches so funktioniert, dass die Daten beim ersten Aufruf initial in einen temporären Speicher (Cache) geladen werden. Das hat zur Folge, dass die Daten nicht immer neu geladen werden müssen, sondern aus dem Cache geladen werden können. Nach einer gewissen Zeit verfallen dann diese Daten, da man sonst nicht garantieren kann, dass diese immer auf dem aktuellen Stand sind. Natürlich kann man diese Art der Kommunikation nur nutzen, wenn auf Daten zugegriffen werden, die sich selten ändern, da man sonst die Daten über keinen längeren Zeitraum (>= 30 Min.) zwischen speichern könnte, ohne dass das andere System die Änderungen mit bekommt. Nutzbare Daten wären beispielsweise die Profildaten, da sich diese zwar ändern können, dieser Fall jedoch eher selten vorkommt bzw. der Zeitraum zwischen zwei Änderungen relativ groß ist (>=1 Tag).

Nachdem wir nun geklärt haben, wie die drei Kommunikationsarten funktionieren, schauen wir nun, wie die Umsetzung in dem Bahn 2.0 System aussehen würde. Vorab aber nochmal eine Kurzfassung des Use-Cases.

Der Kunde interagiert mit einer Webseite/Bahnschalter, welche/r auf Seiten der Bahn gehostet wird. Diese Systeme steuern das Ticket-Buchungssystem (TBs) an. Von dort wird eine Nachricht an das Abrechnungssystem (As) versendet. Nach Rückmeldung des Systems, wird dem Sitzplatz-Reservierungssystem (S-Rs) gesagt, dass ein Platz geblockt werden soll. Dieses System lässt dieses dann in der Sitzplatzverwaltung (Sv) des jeweiligen Zuges eintragen.

(BILD?)

Zunächst wird die Kommunikation zwischen dem Einstiegspunkt und dem TBs betrachtet. Würde man sich an dieser Stelle für eine caching Kommunikation entscheiden, könnte man nicht garantieren, dass die Daten auf beiden Seiten Konsistent sind, da das TBs in einem bestimmten Zeitpunkt nur die Daten mit der Webseite/Bahnschalter synchronisiert. Daher macht Caching an dieser Stelle keinen Sinn. Als nächstes gäbe es die asynchrone Kommunikation. Dies würde bedeuten, dass das System weiterlaufen würde, auch wenn keine Rückmeldung des TBs kommen würde. Prinzipiell wäre dies möglich, wäre aber an dieser Stelle sehr gefährlich und fahrlässig, da zu diesem Zeitpunkt nicht garantiert werden kann, ob die Abbuchung erfolgreich war. Daher muss an dieser Stelle eine synchrone Kommunikation genutzt werden. Dadurch das es sich bei beiden Systemen um ein internes System handelt, kann man an dieser Stelle auch von einer schnellen Antwortzeit ausgehen bzw. von einer geeigneten Fehlerbehandlung bei einer Zeitüberschreitung (>5 Sek.)

Danach muss zwischen dem TBs und dem As kommuniziert werden. Auch an dieser Stelle macht Caching wenig Sinn, da ein dauerhafter Abgleich von Daten notwendig ist und das TBs ein schnelles (< 5 Sek.) Feedback braucht, welches es an den Kunden weitergeben kann. Wie bei der Kommunikation zwischen dem Einstiegspunkt und dem TBs, ist es an dieser Stelle ratsam, eine synchrone Kommunikation zu verwenden, da beide Systeme intern sind und mithilfe der Rückmeldung erst entschieden werden kann, ob eine weitere Bearbeitung des Prozesses überhaupt möglich ist.

Nachdem das TBs nun das Feedback von dem As bekommen hat, gibt es dies an die Internetseite bzw. den Bahnschalter weiter, damit der Kunde weiß, ob seine Fahrt abgebucht worden ist oder nicht. Der Prozess ist an dieser Stelle jedoch noch nicht zu ende und das TBs interagiert mit dem S-Rs. Hier kann sowohl eine asynchrone als eine synchrone Kommunikation verwendet werden, weil es sich um interne Systeme handelt, setzt man im Normallfall auch auf die synchrone Kommunikation. Sollte das S-Rs nämlich einen Fehler zurückgeben, muss das TBs über eine geeignete Fehlerbehandlung Verfügen wie beispielweise einen Neuversuch oder das Versenden einer E-Mail, welche an den Zuggast raus geht, dass dessen Ticket im System hinterlegt ist, jedoch kein Platz geblockt wurde und er sich bei einem Zugmitarbeiter melden soll. Diese Fehlerbehandlung kann jedoch auch bei einer asynchronen Kommunikation eingebaut werden. Caching macht auch bei dieser Kommunikation zwischen den Backends wenig Sinn, da eine Inkonsistenz fatale Folgen haben könnte.

Zu guter Letzt wäre noch die Kommunikation zwischen dem S-Rs und der Sv zu analysieren. Zur Erinnerung, im Abschnitt der TI-Architektur wurde entschieden, dass in jedem Zug eine eigene Sv vorhanden ist. Dieser Punkt ist ausschlaggebend für die Eignung der Kommunikationswahl zwischen den Systemen. Würde man sich für eine synchrone Kommunikation entscheiden, würde das S-Rs in der Zeit wo es auf die Sv wartet, in einen Wartemodus sein. Das bedeutet das in dieser Zeit keine anderen Prozesse, wie beispielweise weitere Buchungen, angestoßen werden könnten. Aus den Eigenschaften der Bandbreite im Zug, kann man entnehmen, dass diese außerhalb eines Bahnhofs, nicht als ausreichend garantiert werden kann. Die in diesem Fall beste Variante wäre die Kommunikationsart Caching. Diese Kommunikation würde stattfinden, sobald der Zug in einen Bahnhof einfährt. Dann würde die Sv, des jeweiligen Zugs, seine Daten mit der S-Rs abgleichen, damit die Daten wieder konsistent sind. Ausschließlich zu diesem Zeitpunkt ist es nämlich notwendig, dass der Zug weiß, wie viele Sitzplätze (nicht jedoch welche) er für Reservierungen blocken muss. Während einer Fahrt muss die Sv des jeweiligen Zugs dies noch nicht zu wissen, da während der Fahrt keine Zuggäste ein- oder aussteigen. Alternativ könnte man sich auch für eine asynchron Kommunikation entscheiden, welche jedoch nicht so effektiv wie Caching wäre, da die Sv bereits während der Fahrt mit Daten versorgt wird, welche sie aber noch nicht braucht, woraus eine höhere Auslastung des Speichers im Zug entsteht.

Abschließen kommt man also zu dem Ergebnis, da man bei den internen Systemen auf eine synchrone Kommunikation setzt, weil man sowohl Feedback von den Systemen braucht und eine kurze Antwortzeit (> 5 Sek.) garantieren kann und alle Systeme sich intern befinden. Ausschließlich die Kommunikation zwischen dem S-Rs und dem Sv des jeweiligen Zugs wird auf Caching gesetzt, da hier eine dauerhafte Synchronisation nicht von Nöten ist.