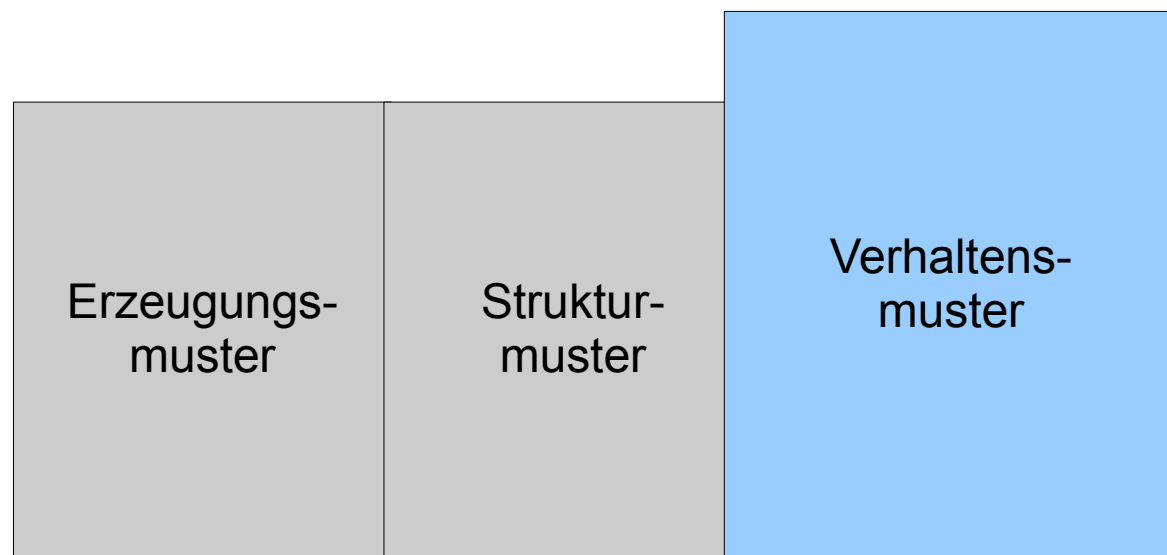

Entwurfsmuster Der Event Bus

Das Social Network für Objekte

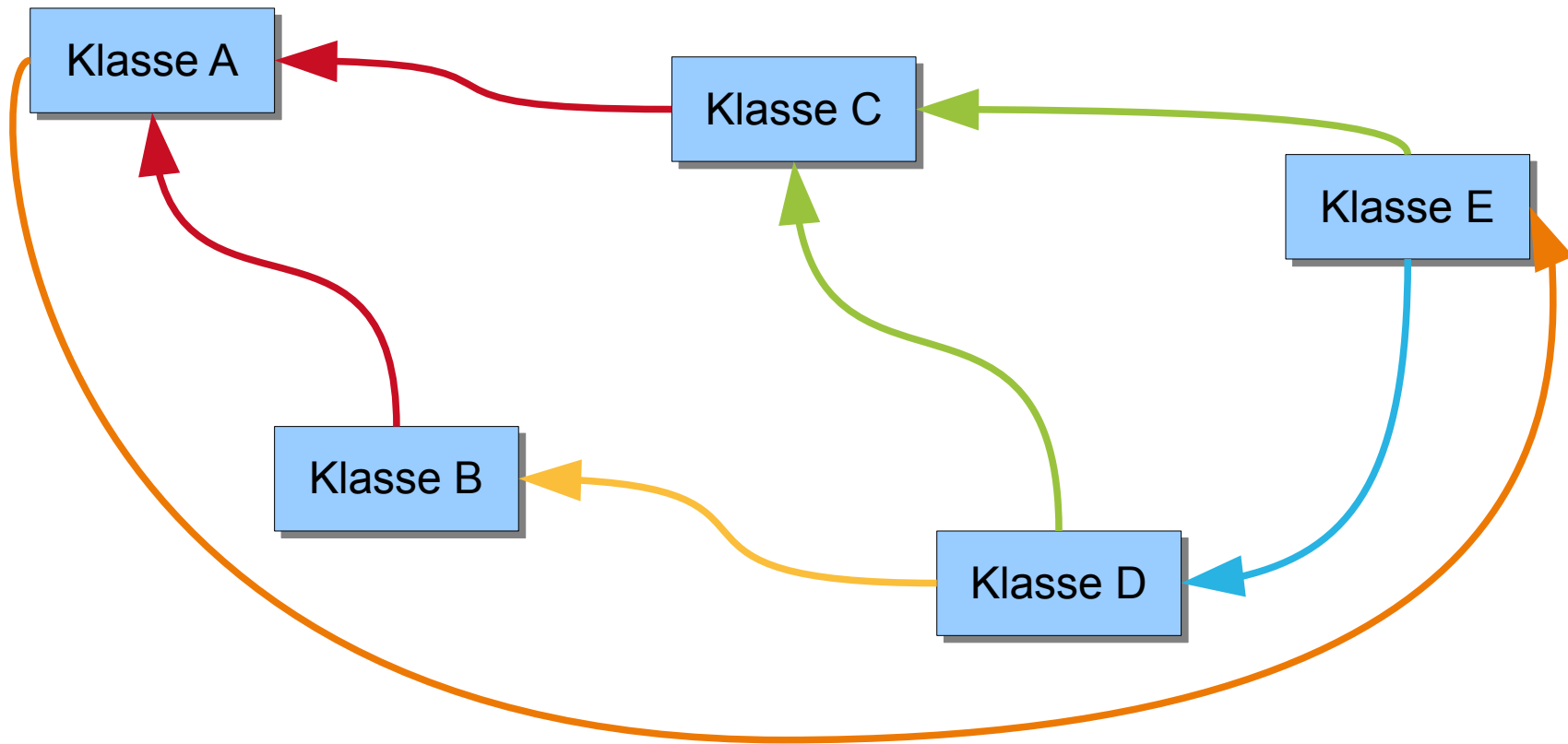
Event Bus

- Klassifikation
 - Kein echtes Entwurfsmuster - abgeleitet
 - Objektbasiertes Verhaltensmuster
 - Langlebig, zentral für Anwendung



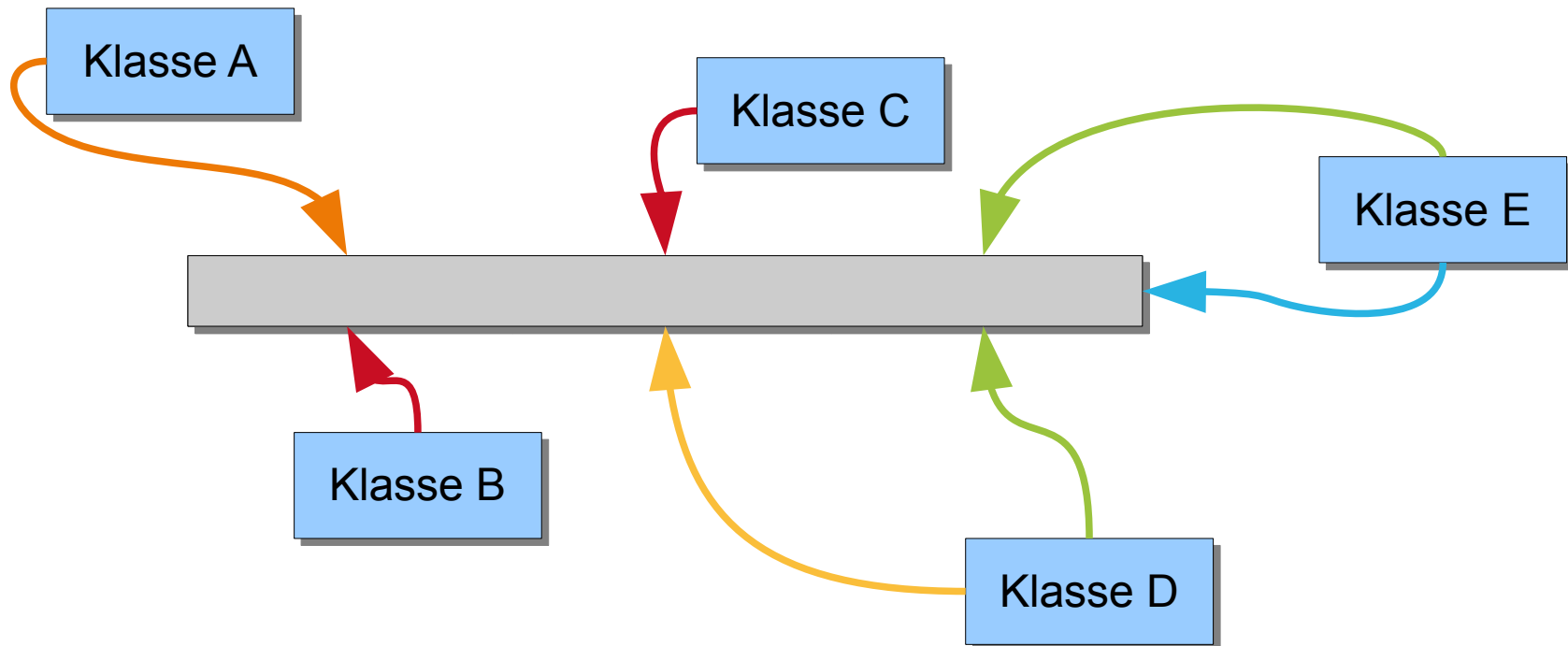
Einleitendes Beispiel

System mit vielen Beobachtern



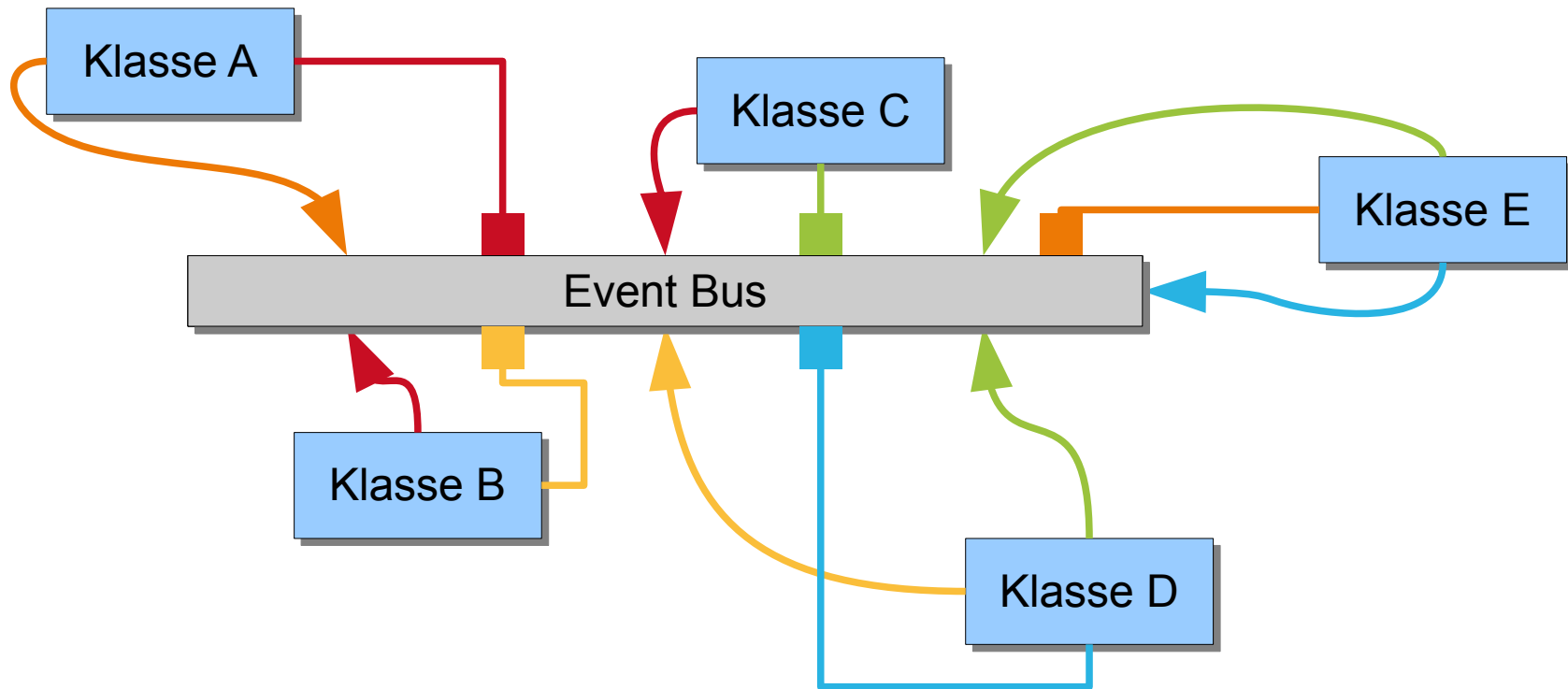
Einleitendes Beispiel

Benachrichtigungsaspekt auslagern



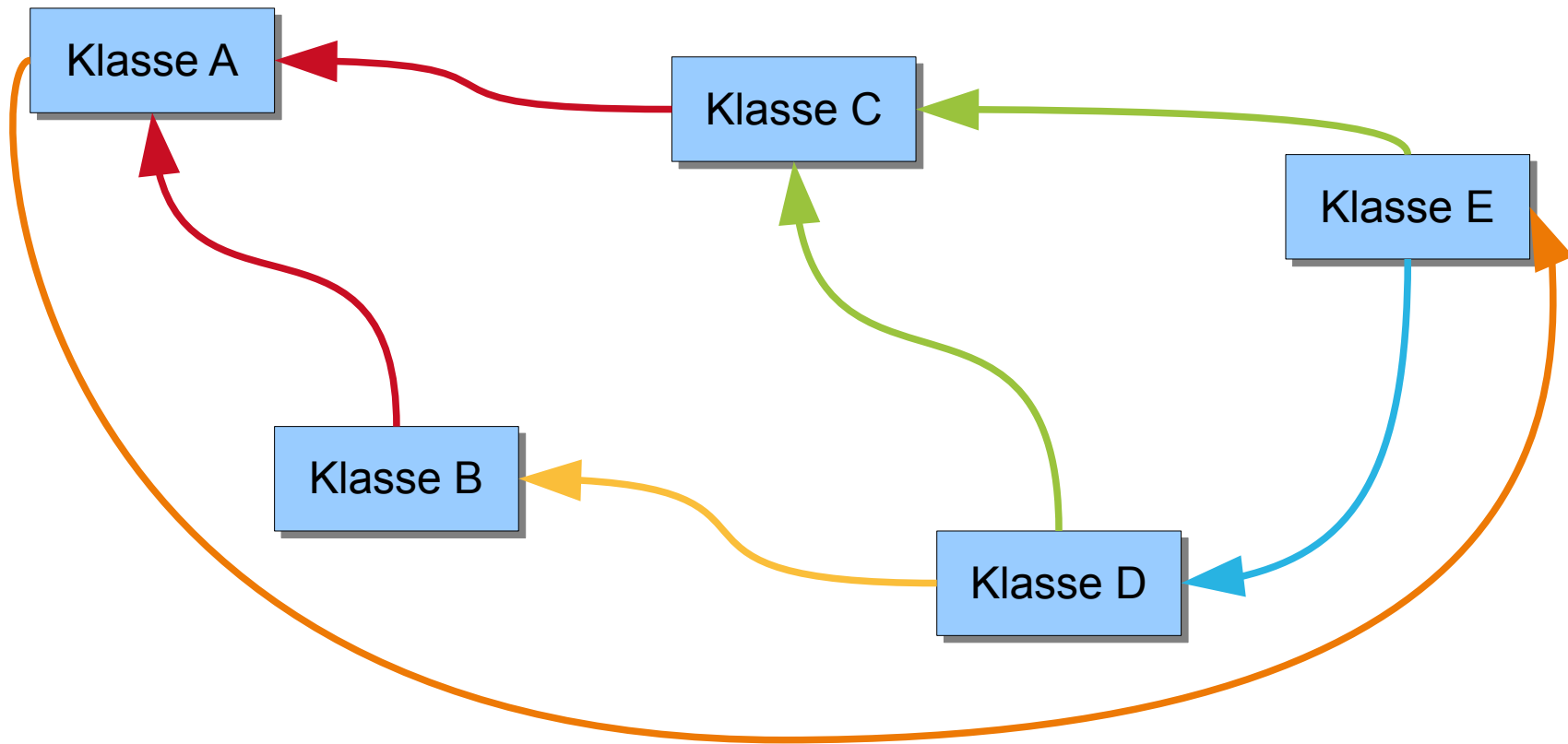
Einleitendes Beispiel

Benachrichtigung als Objekt modellieren



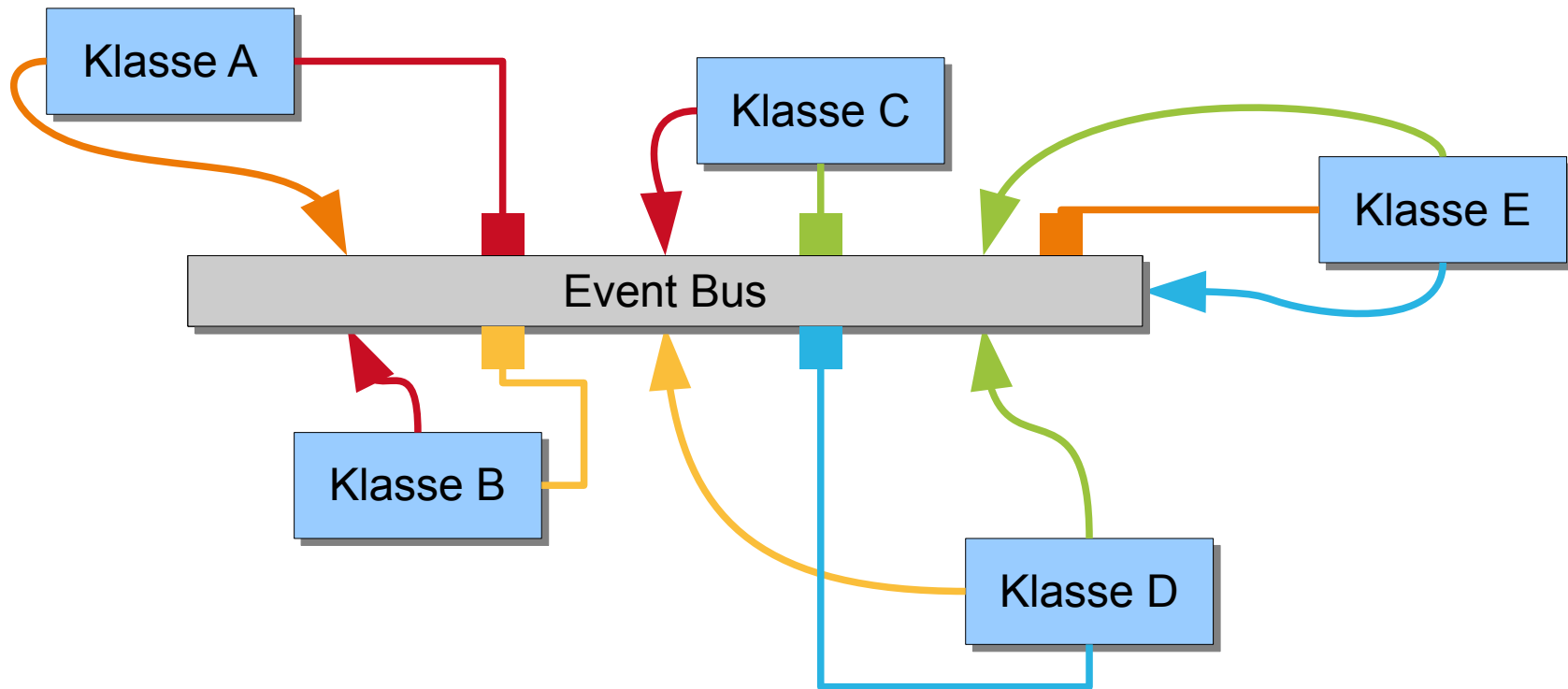
Einleitendes Beispiel

Vergleich beider Architekturen



Einleitendes Beispiel

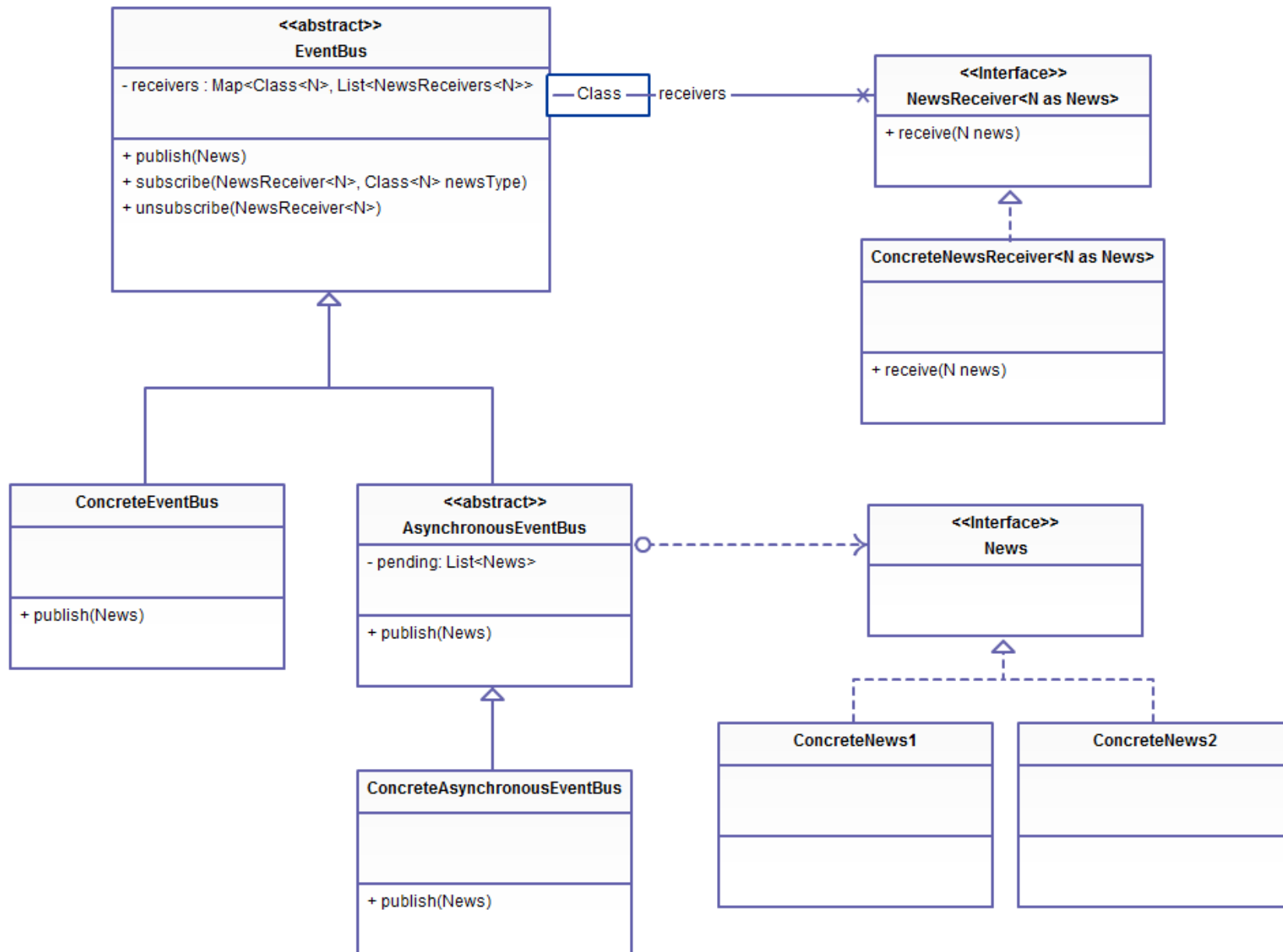
Vergleich beider Architekturen



Motivation

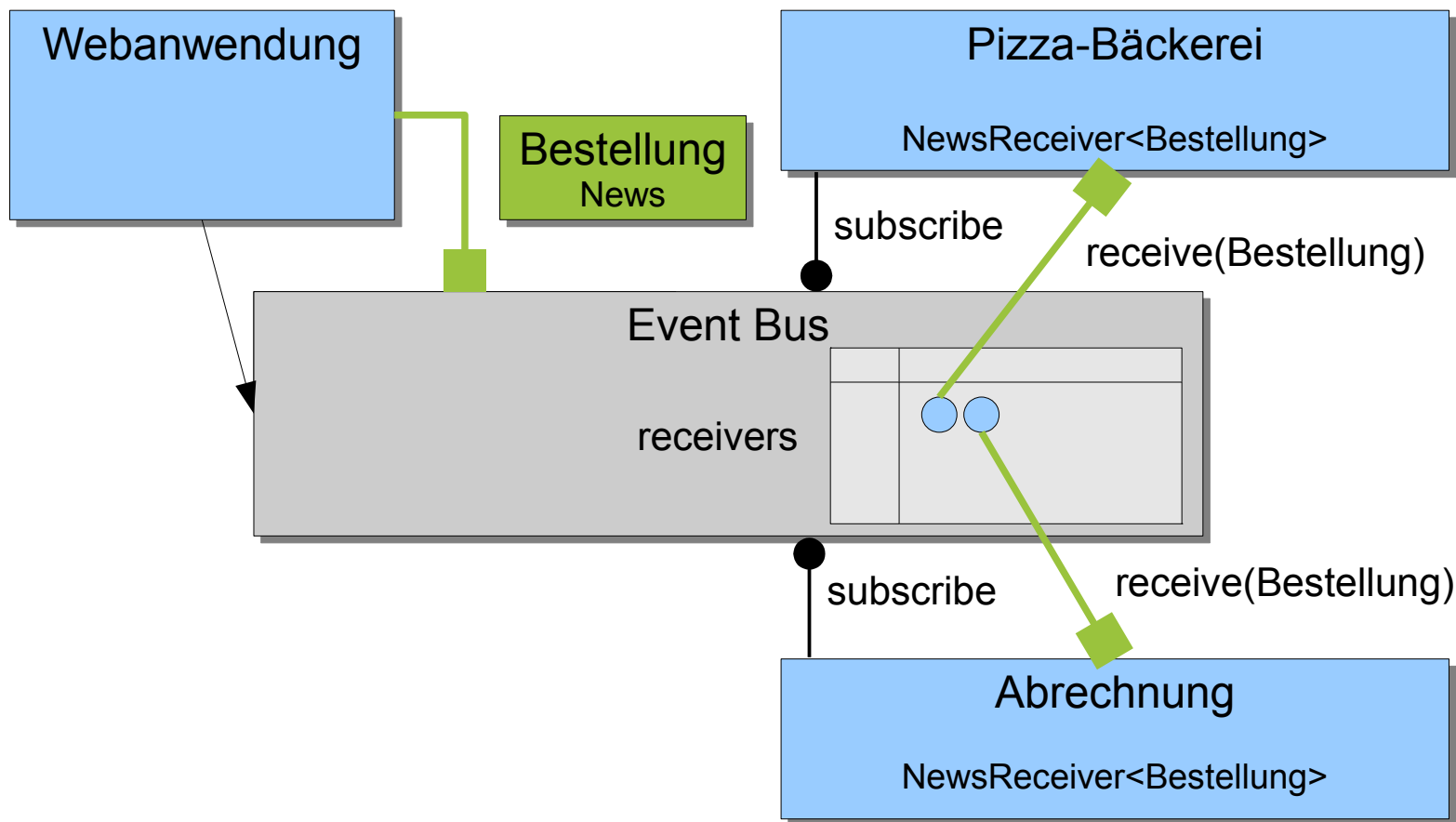
- Entflechtung komplexer Beobachter-Netze
- Architektur-Skalierung
 - Subsystem-Bildung
- Entkopplung der Systemteile
 - Einheitliche, asynchrone Benachrichtigung
- Benachrichtigungs-Aspekt explizit modellieren
 - Zentraler Punkt für Logging, Lastüberwachung, etc.

UML-Diagramm (Klassen)



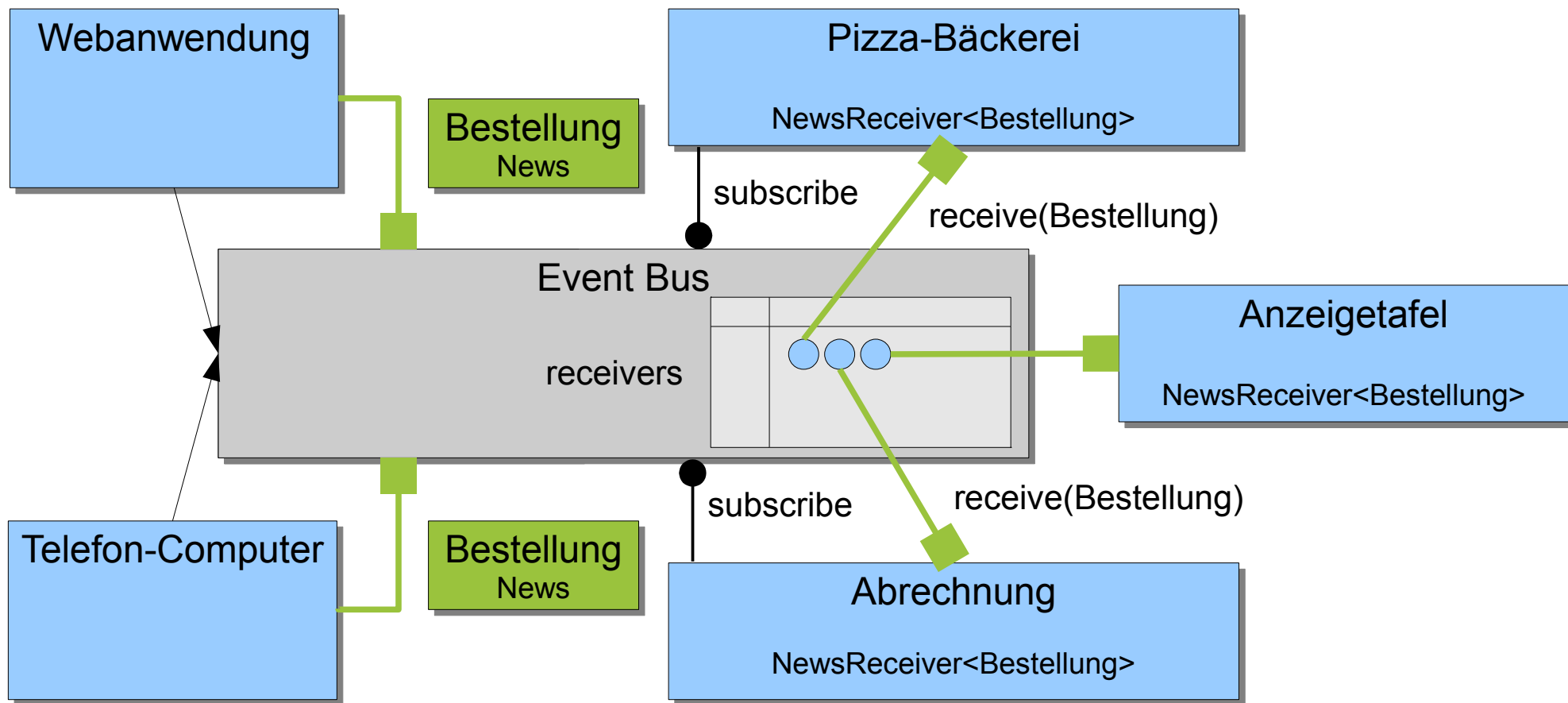
Ablauf (synchroner Bus)

Pizza-Lieferdienst mit Webanwendung



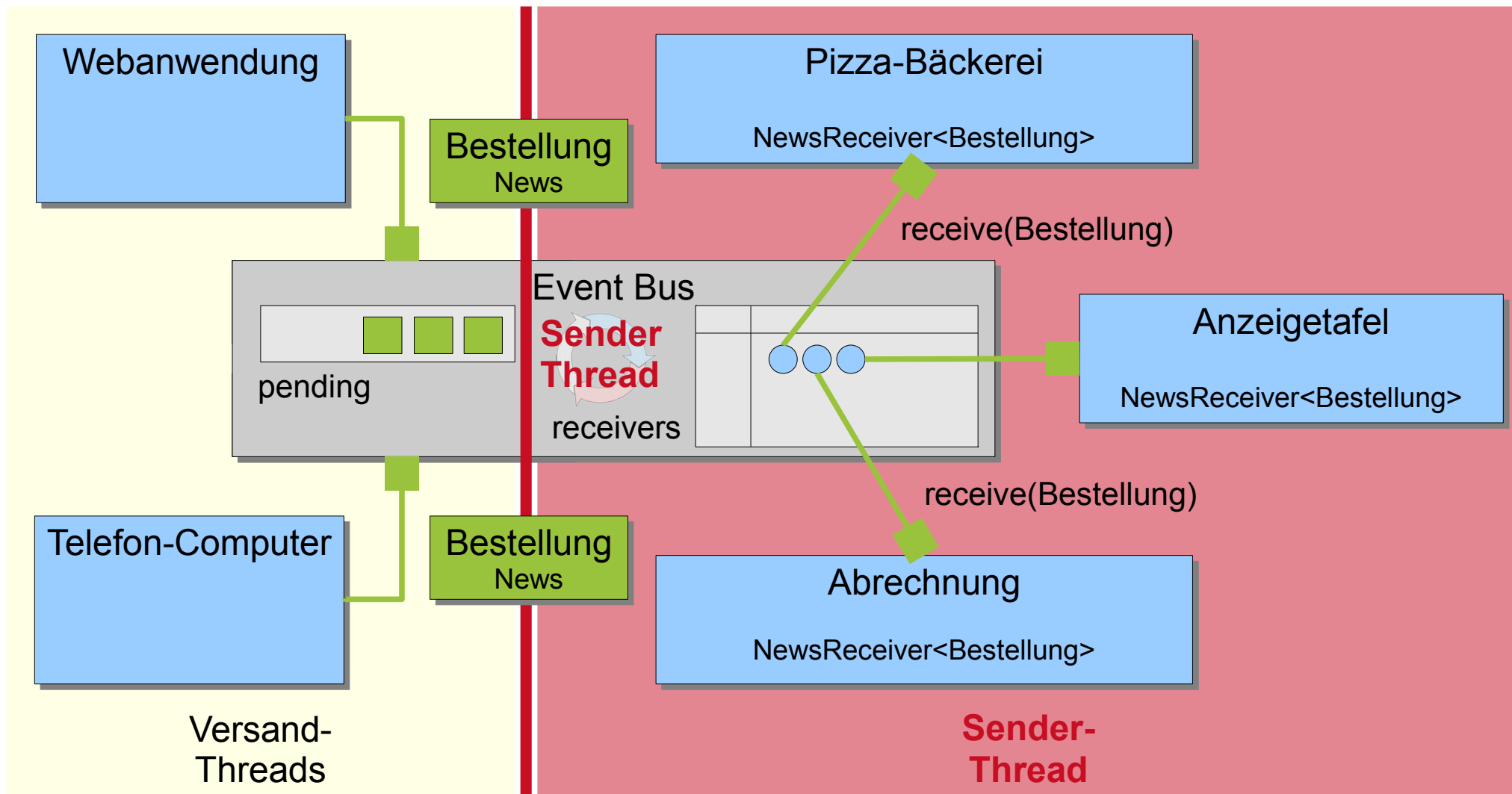
Ablauf (synchroner Bus)

Erweiterungsmöglichkeiten



Ablauf (asynchroner Bus)

Entkopplung von Versand und Empfang

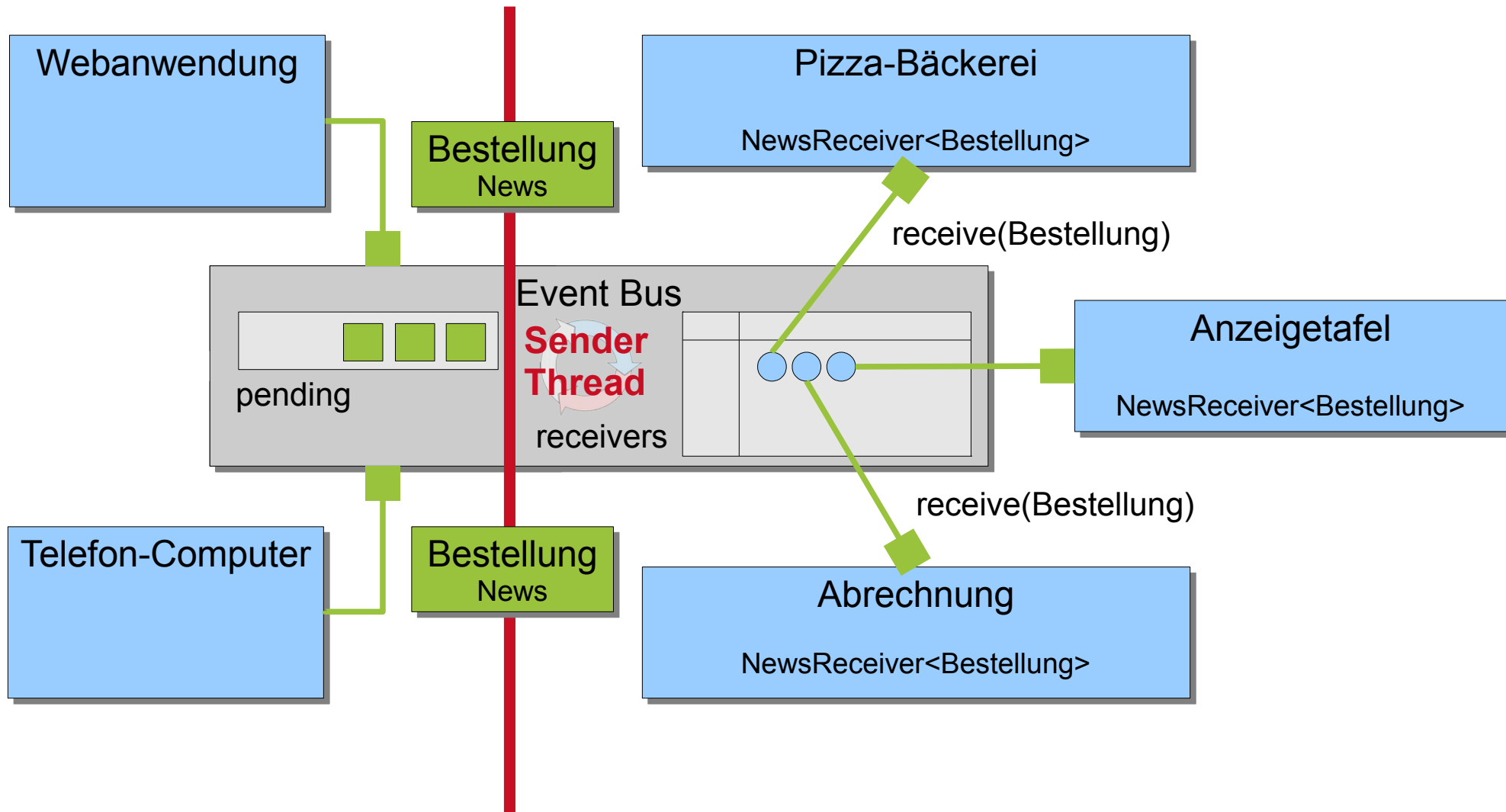


Vorteile

- Trennung von Sender und Empfänger
- Sender muss Empfänger nicht kennen
 - Ist beim Beobachter auch so
- Empfänger passt sich nur an Botschaft an
 - Beim Beobachter muss Empfänger-Typ stimmen
- Universale Implementierung
 - Anknüpfungspunkt für Aspekte (Cross-Cutting Concerns)
 - Hoher Wiederverwendungseffekt
- Einfache Verwendung bei Programmierung
 - Systemweite, anonyme Kommunikation

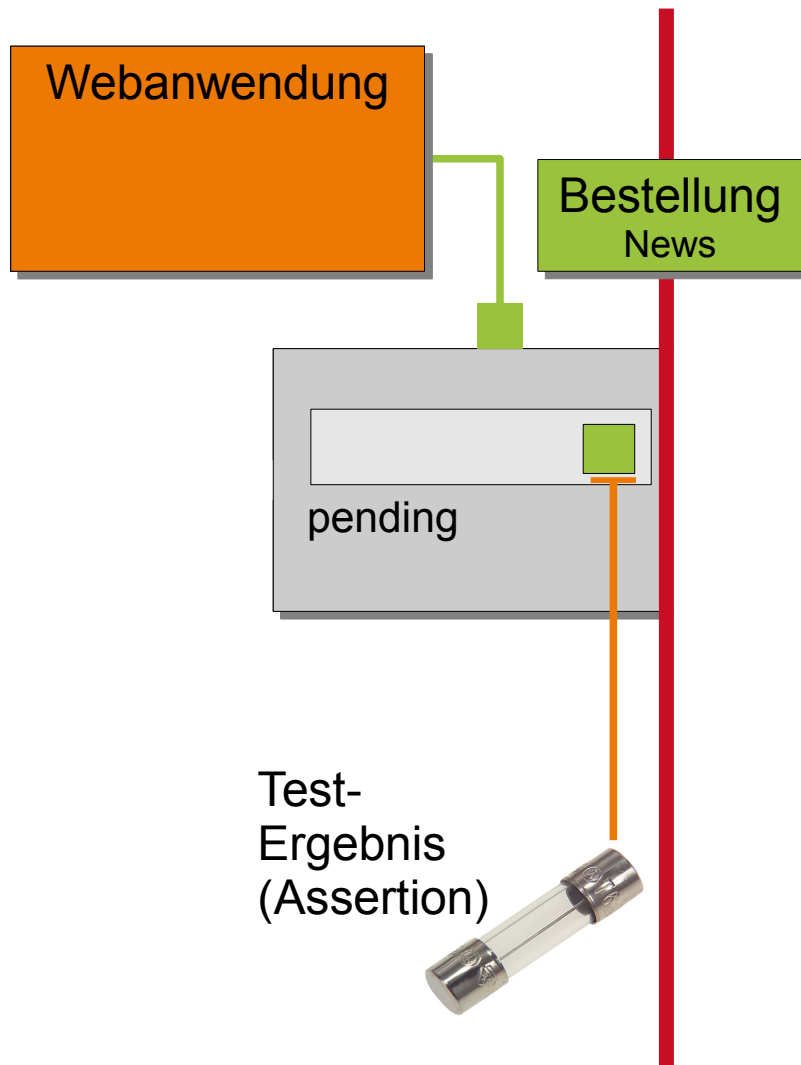
Event Bus und Tests

Der Event Bus unterstützt Komponenten-Tests



Event Bus und Tests

Der Event Bus unterstützt Komponenten-Tests



- Komponenten können anhand ihrer Ausgaben auf dem Event Bus überprüft werden
- Das Verhalten anderer Komponenten kann durch künstlich erzeugte Nachrichten auf dem Event Bus simuliert werden (Mocking)

Event Bus als Aspekt-Saum

- Der Event Bus ist die zentrale Kommunikationsstrecke für die Subsysteme
- Damit perfekter Anknüpfungspunkt für Cross-Cutting Concerns (Aspekte) auf Komponenten-Ebene
 - Protokollierung wichtiger Systemereignisse
 - Überwachung der Systemauslastung
 - Statistiken über Systemnutzung

Implementierungen

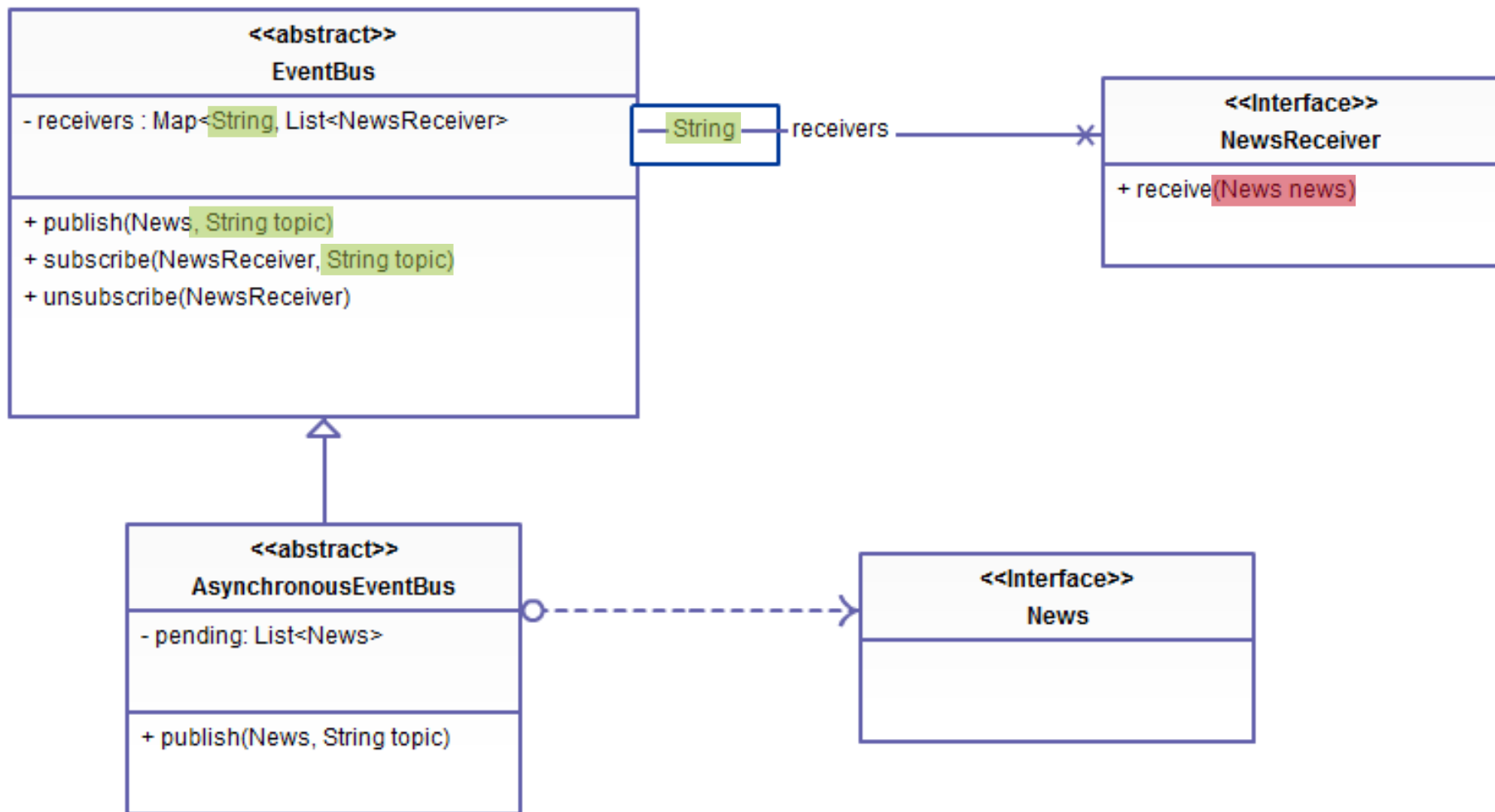
- Es gibt bereits jede Menge EventBus-Bibliotheken
- Für Java
 - Google Guava Event Bus
 - abas-eB Event Bus
- Für Android
 - Greenrobot Event Bus
- Tipp: Vorhandene Bibliothek verwenden
 - Delegiert Problemstellen-Lösung an andere(TM)
 - Thread-Locking, Exception Handling, References

Varianten

- Synchroner und asynchroner Versand
- Asynchroner Versand mit mehreren Threads (Threadpool)
- Versand-Warteschlange mit Prioritäten
- Empfänger-Auswahl
 - Über News-Typ (Klasse)
 - Nachteil (in Java): Nur ein News-Typ pro NewsReceiver
 - Ursache: Type Erasure für Java Generics
 - Über zusätzliches Topic (z.B. String)
 - Über Reflection und Annotations (siehe Guava)

Versand über Topic

- Nicht der News-Typ entscheidet, sondern ein zusätzliches „Thema“



Versand mit Topic

- Verschiedene News-Typen können an die gleiche Empfänger-Gruppe geschickt werden
 - Diese müssen damit umgehen können (Downcast)
- Empfänger-Gruppe kann vom Versender mit beeinflusst werden
 - Wildcard-Topics
 - „*“ für „Alle Empfänger“
 - „*-debug“ für „Alle Empfänger, die sich für ein Thema angemeldet haben, das auf -debug endet“
- Nachteil: Verlust der Typsicherheit

Nachteile

- Der Event Bus ist ein Architekturmuster
 - Beeinflusst die gesamte Anwendung
- Ein viel genutzter Event Bus wird zum Flaschenhals für die Systemleistung
 - Dient auch als zentraler Messpunkt für Auslastung
- Die Verbindung zwischen Sender und Empfänger ist nicht direkt ersichtlich
 - Empfänger muss evtl. nicht spezifischer Typ sein
 - Sender enthält keine Referenzen auf den Empfänger

Zusammenfassung

- Event Bus
- Objektbasiertes Verhaltensmuster
- Architekturmuster – Systemstruktur wird beeinflusst
- Trennt Sender und Empfänger einer Botschaft
- Viele Varianten
 - Meistens asynchroner Bus
- Ideales Werkzeug für Integrationstests
- Nachteil: Implizite Beziehungen

Evolution von Mustern

- Muster können aus „kleineren“ Mustern entwickelt werden
- Beispiel:
 - Direkte Benachrichtigung
 - Beobachter
 - Event Bus