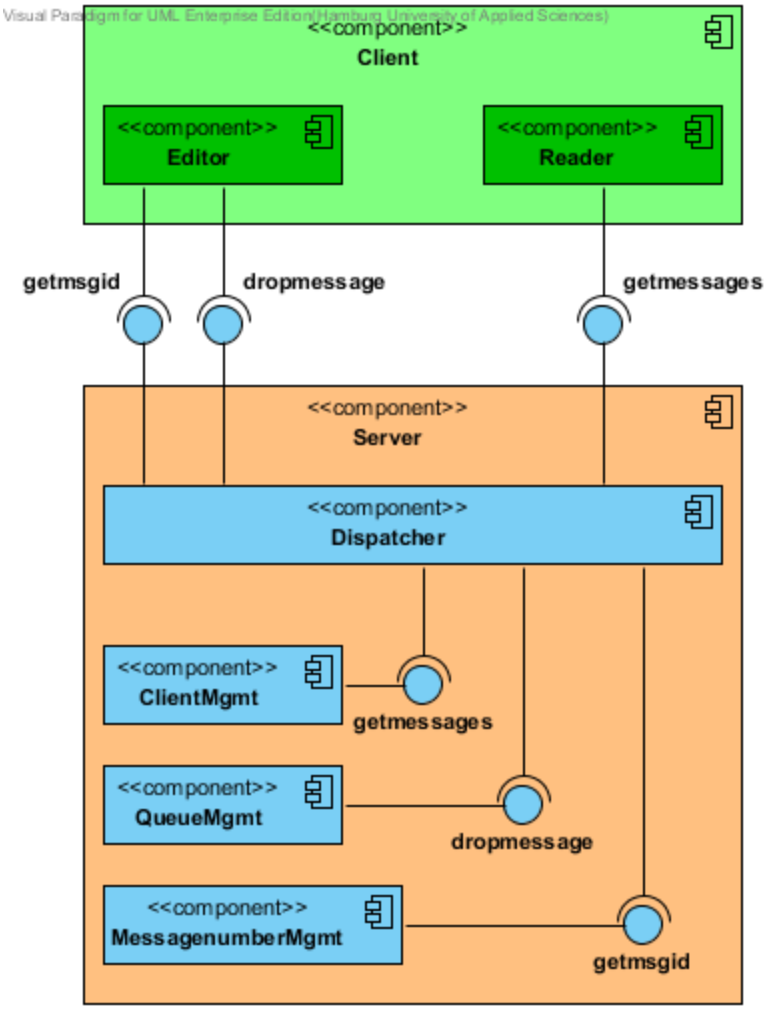
# Entwurf

## Komponentendiagramm

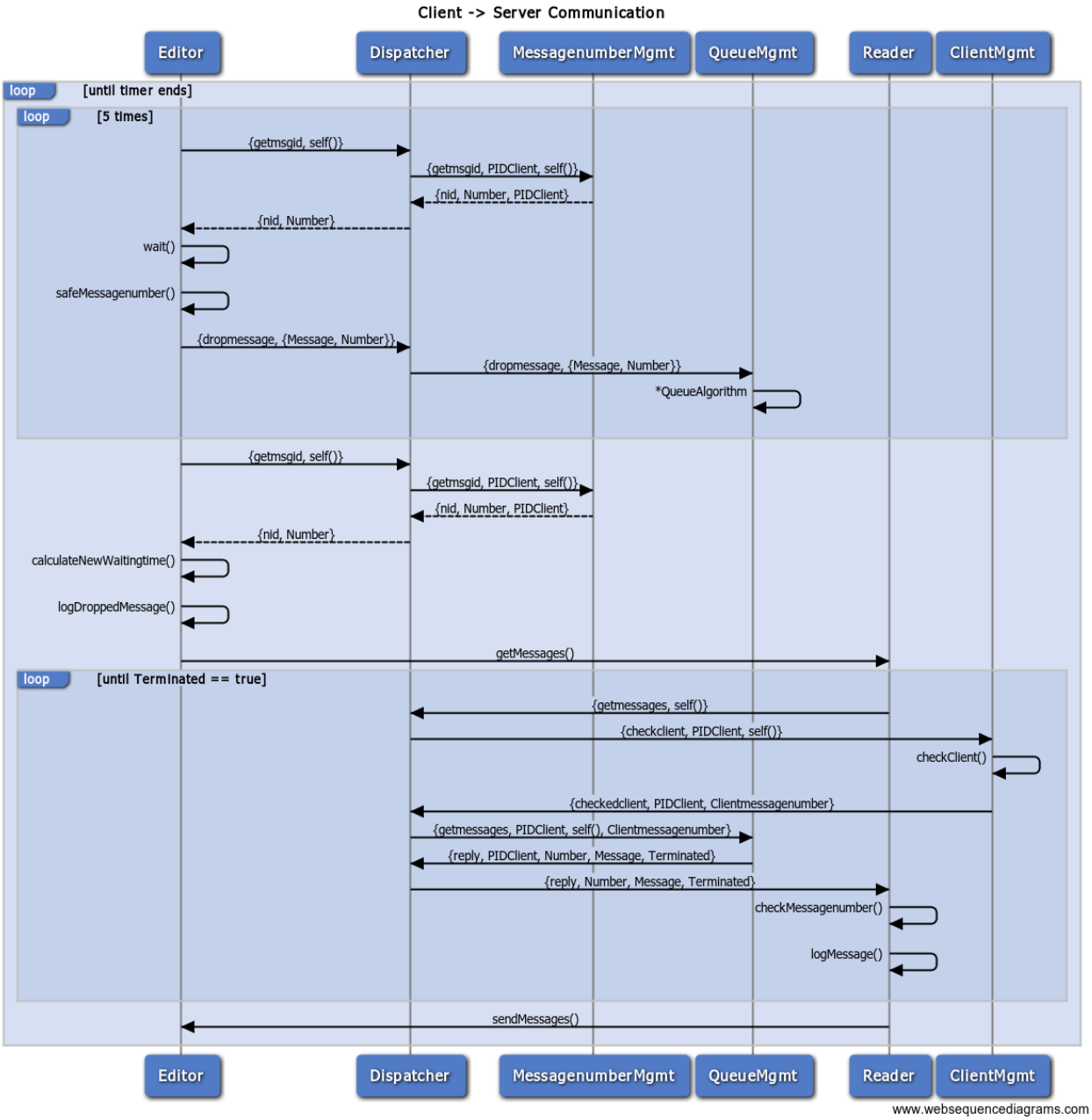



### Erläuterungen

- Der *Editor* und der *Reader* werden als ein gemeinsamer Clientprozess gestartet
    - die gesendeten Nachrichtennummern werden intern gespeichert und bei dem Wechsel zwischen Editor-Modus und Reader-Modus dem Reader übergeben
- *Editor* und *Reader* werden sequentiell ausgeführt
- Alle Anfragen des Clients werden an den *Dispatcher* des Servers geschickt
- Der *Dispatcher* verteilt die Nachrichten an die entsprechenden Serverdienste
- Der Server wird mit 2 Prozessen abgebildet
    - Prozess 1: *Dispatcher, ClientMgmt, MessagenumberMgmt* - dient als allgemeine Verwaltung
    - Prozess 2: *QueueMgmt* - ist für das verwalten und versenden der Nachrichten zuständig

- Das *QueueMgmt* verwaltet die *Holdbackqueue (HBQ)* und die *Deliveryqueue (DLQ)*
- Das *ClientMgmt* verwaltet die Clientinformationen (IDs, aktuelle Nachrichtennummer, Timeout des Clienten)
- Das *MessagenumberMgmt* verwaltet die Nummerierung der Nachrichten
- Die interne Kommunikation des Servers verläuft über den *Dispatcher*

# Sequenzdiagramm



(beide Bilder sind separat als .png beigefügt)

### Erläuterungen

- Jede Nachricht von einem Clienten wird von dem Dispatcher um die eigene PID ergänzt. Somit können die Serverprozesse die Antworten über den Dispatcher an die Clienten zurück schicken.
- ***QueueAlgorithm**: bei jeder erhaltenen Message wird in dem QueueMgmt die Bedingung zur Übertragung der Messages aus der HBQ in die DLQ überprüft. [AnzahlMessages(HBQ) > maxGröße(DLQ) / 2 ]
- Alle Schleifen beziehen sich auf einen einzelnen Clienten, da dieses Sequenzdiagramm den Ablauf des Systems aus der Sicht der Clienten darstellt
- Nach der Ausführung der ersten inneren Schleife (*5 times*) wechselt der Clientprozess von der Ausführung im Editor-Modus in den Reader-Modus [Funktion: getMessages()]
- Ist die zweite innere Schleife ausgeführt und hat der Client alle Nachrichten empfangen, wechselt der Clientprozess wieder in den Editor-Modus [Funktion: sendMessages()]

## Allgemeine Implementierungsdetails

### Fehlertextzeilen

Fehlende Nachrichten werden gebündelt als eine Fehlernachricht in die DLQ eingefügt.
Falls eine als fehlend markierte Nachricht nachträglich doch noch eintreffen sollte, wird diese verworfen und ist für unser System nicht mehr relevant.
Beispiel:
DLQ: max. Größe 30; enthaltene Nachrichten 1-10
HBQ: 15 Nachrichten im Speicher; kleinste Nachrichtennummer 14

In diesem Fall wird eine Fehlernachricht für die DLQ erzeugt, welche als Nachricht auf die fehlenden Elemente verweist, z.B. ‘Nachrichten 11-13 fehlen’. Diese erzeugte Nachricht wird dann mit der höchsten fehlenden Nummer (13) in die DLQ geschrieben. Anschließend werden die nächsten gültigen Nachrichten aus der HBQ in die DLQ geschoben.

### Nachrichtenverlust

Bei einem Nachrichtenverlust während der Übertragung von dem Server zum Reader hat dieser keine Chance eine Nachricht als fehlend dem Server zu melden. Somit können Lücken bei dem Leser auftreten.
Dieses Problem könnte man beheben, indem man jede erhaltene Nachricht als Empfangen bestätigt. (Request-Response-Prinzip)
Genau wie in dem genannten Fall könnte der Server beim Empfangen von Nachrichten eines Redakteur jede Nachricht bestätigen.