# 7437 – EDI und E-Business Standards, 4661 – E-Business: Standards und Automatisierung

Praktikumsaufgabe 08:

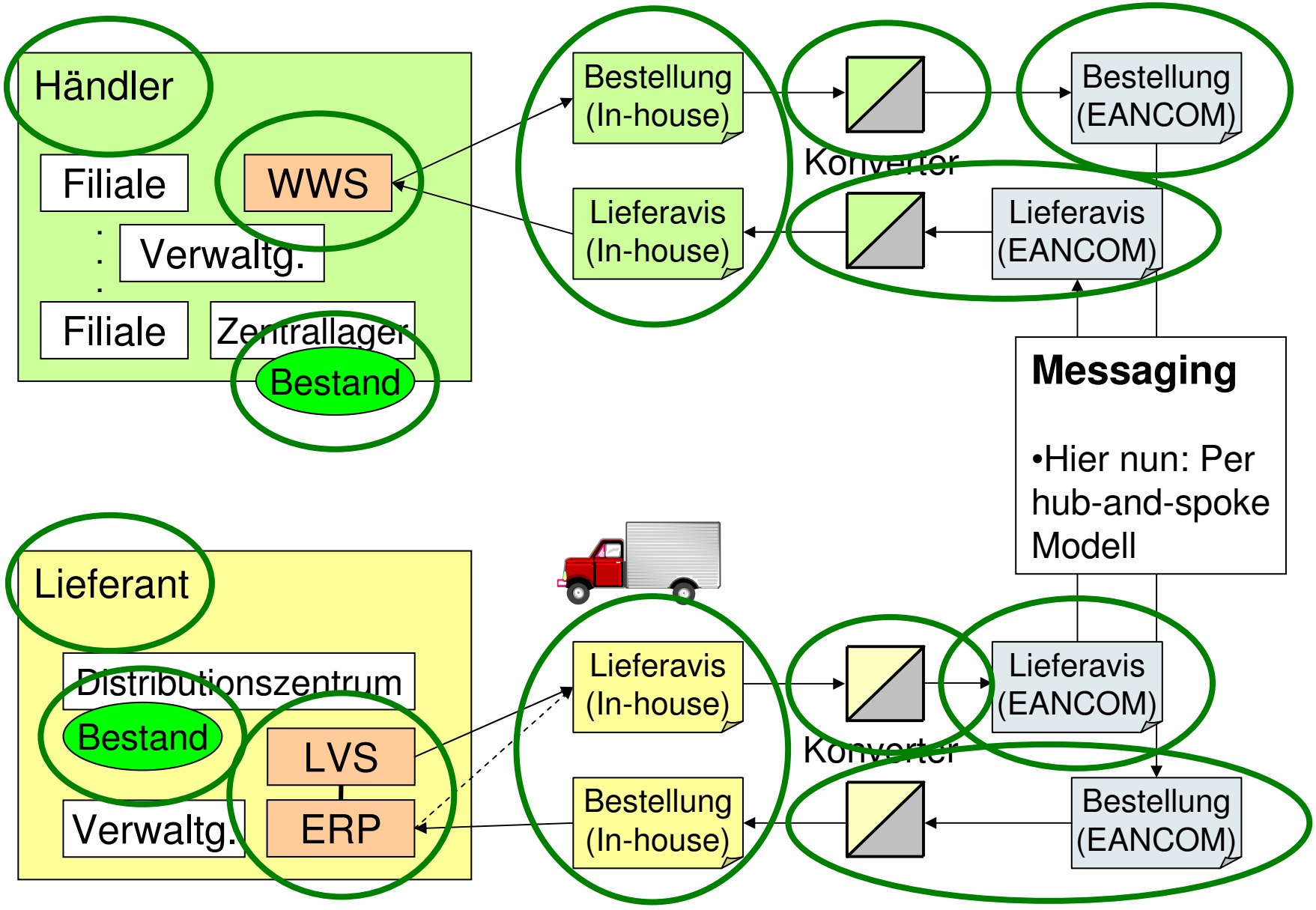
Integration der Abläufe

# Das Szenario

- Szenario
  - Die Praktikumsteilnehmer setzen ihre Handelspartner-Rolle (Lieferanten, Händlern) der Konsumgüterbranche aus Praktikum 01/02 fort.
  - Nachdem inzwischen alle Komponenten für den elektronischen Handel bereit stehen, werden diese nun verbunden und automatisiert.

- Ziele der Übung
  - *Messaging:* Simulation per *mailboxing und hub-and-spoke* Modell
  - Automatisierungstechniken:
    - Hindergrundprozess, *Logging*, Fehlerbenachrichtigung per *E-Mail*
  - Vorteile und Grenzen der Standardisierung
    - Einheitlichen Datenaustausch mit mehreren Geschäftspartnern trotz unterschiedlicher *inhouse*-Formate erleben
    - Verbleibende Lücken und Interpretationsspielräume in den EANCOM-Daten entdecken.

# Das Praktikumsprojekt



# Die Aufgabe

- Die Aufgabe
  - Schreiben Sie ein (vereinfachtes) „*daemon*“-Programm, das Ihre Geschäftsabläufe automatisiert
    - Es läuft permanent als Hintergrundprozess
    - Es prüft einmal pro Minute, ob etwas zu tun ist
    - Ggf. leitet es alle notwendigen Schritte ein.
    - Dabei führt es Protokoll und benachrichtigt bei Abbruch per E-Mail
  - Testen Sie die Abläufe (a) zunächst per *loopback*-Test, (b) gemeinsam mit Ihren Geschäftspartnern (Woche 2)!
    - Erzeugen Sie dann Daten für Ihre bzw. beantworten Sie Daten von Ihren Geschäftspartnern.
  - Programm-Name:

    `controld.rb`
  - Gewünschter Aufruf: `controld.rb &`

Als Hintergrundprozess betreiben.
Einfacher Ersatz für echte Daemons
bzw. für crontab

# Händler

- Händlersicht
  - Rufen Sie **`bestellen.rb`** aus Übung 04 jede Minute auf
    - Dieses Programm prüft Ihre Bestände und generiert ggf. Bestellungen in Ihrem *inhouse*-Format
  - Falls Bestellung(en) generiert:
    - Rufen Sie **`mapper06.rb`** für jede erzeugte, nach Geschäftspartnern vorsortierte Datei auf, um sie nach EANCOM zu wandeln.
    - Kopieren Sie die EANCOM-Dateien in das **`out`**-Verzeichnis Ihrer Mailbox (s.u.), löschen Sie sie quellseitig bei Erfolg.
  - Prüfen Sie jede Minute das **`in`**-Verzeichnis Ihrer Mailbox
  - Falls neue (EANCOM-) Daten eingetroffen: Abholen!
    - Rufen Sie **`mapper07.rb`** auf und wandeln Sie diese Daten in Ihr *inhouse*-Format für Lieferavise um, löschen Sie sie bei Erfolg.
    - Rufen Sie schließlich **`ware_annehmen.rb`** aus Übung 04 für die *inhouse*-Daten auf, um den Bestellprozess abzuschließen.

# Lieferanten

- Lieferantensicht
  - Prüfen Sie jede Minute das **`in`**-Verzeichnis Ihrer Mailbox
  - Falls neue (EANCOM-) Daten eingetroffen sind:
    - Holen Sie diese Daten ab (quellseitig bei Erfolg löschen)
    - Rufen Sie **`mapper07.rb`** auf und wandeln Sie diese Daten in Ihr *inhouse*-Format für Bestellungen um.
    - Entfernen Sie bei Erfolg die verarbeiteten Daten aus der „*inbox*“.
    - Rufen Sie nun für jede *inhouse*-Datei **`liefern.rb`** aus Übung 04 auf, und erzeugen Sie so entsprechende Lieferavise in Ihrem *inhouse*-Format.
    - Rufen Sie **`mapper06.rb`** für jede erzeugte (nach Geschäftspartnern vorsortierte) Datei auf, um diese Daten nach EANCOM zu wandeln.
    - Kopieren Sie die EANCOM-Dateien in das **`out`**-Verzeichnis Ihrer Mailbox (s.u.).

# Generelles zu Verarbeitungsketten

- Erledigtes löschen
  - Eine abgeholte Datei wird quellseitig sofort gelöscht, damit sie nicht mehrfach verarbeitet werden kann.

- Prinzip „Transaktionsschutz incl. Rollback“
  - Zu jedem Zeitpunkt sollte nur eine gültige Dateikopie existieren
  - Dies erreicht man z.B. durch Umwege über temporäre Dateien:
    1. Kopiere `remote_host:src.edi` → `localhost:src.tmp`
    2. Bei Erfolg: Lösche `remote_host:src.edi`
    3. Umbenennen, „atomar“ auf `localhost: mv src.tmp src.edi`
    4. Nun lokale Weiterverarbeitung von `src.edi` …

- Hindernisse aus dem Weg räumen
  - Scheitert ein Schritt (Konvertierung, Validierung, Kopieren), sollte die Ursache protokolliert und der Auslöser (Datei) aus dem Weg geräumt werden
    - Verarbeitung weiterer Dateien wird so ermöglicht
    - Fehler treten nur einmal auf. Ggf. Alarmierung per E-Mail!

# Technische Hinweise

- Logging:

```
#!/usr/bin/env ruby
require 'logger'

# Log levels: DEBUG < INFO < WARN < FATAL < UNKNOWN
log = Logger.new("mein_edi_log.txt")
log.level = Logger::WARN # Optionale Filterung: Log ab WARN
log.info("Programmstart")# Unsichtbar
log.warn("Eine Warnung") # Sichtbar!
```

- Benachrichtigung im Fehlerfall (Unix/Linux)

```
begin
  # … Ihr Code
rescue => e # Alles abfangen
    IO.popen("mail -s 'controld-Abbruch!' werntges", "w") do |io|
      io.puts e, e.backtrace.join("\n")
    end
end
```

*Mail subject*

Eigene Empfängeradresse (bzw. *user account name)* verwenden!

# Technische Hinweise

- Aufruf externer Programme (Beispiele):
  - Mit „Backquotes“:

```
srcname = 'meineQuelldatei.inh'
`ruby mapper06.rb #{srcname} > meinErgebnis.edi`

# Rückgabewert = stdout des aufgerufenen Kommandos,
# Fehlercode (process status) in $?
```

  - Mit Kommando „system“:

```
src = 'meineQuelldatei.inh'
system("ruby mapper06.rb #{src} > meinErgebnis.edi")

# Fehlercode (process status) in $?
# SystemCallError bei Problemen.
```

# Technische Hinweise

- Was protokollieren?:
  - Jeden Start von „controld.rb“ Level: **info**
  - Jedes reguläre Ende von „controld.rb“ Level: **info**
  - Jeden Abbruch von „controld.rb“ Level: **fatal**
    - Gleichzeitig Benachrichtigung per E-Mail!
  - Eingeleitete Aktionen Level: **info**
    - Z.B. Namen erzeugter oder angetroffener Dateien
    - Aufrufe von mapper06 & mapper07, von „liefern“ und „ware_annehmen“
  - Fehlercodes in Aufrufen Ihrer Programme, Level: **error**
    - mapper06, mapper07 und alle „Anwendungen“ aus Übung 04

# Technische Hinweise

- ftp mit Ruby „fernsteuern“
  - Nutzen Sie einfach die Standardbibliothek Net::ftp
  - Einzelheiten:
    http://www.ruby-doc.org/stdlib/libdoc/net/ftp/rdoc/index.html

- Code-Beispiel:

```
Net::FTP.open('your.ftp-server.xx') do |ftp|
  ftp.login( user, passwd ) # user & passwd vorher anlegen
  ftp.chdir('in')           # Wechsel in eigene Inbox
  puts ftp.list.join("\n")  # Optionale Anzeige
  ftp.nlst('*.edi').each do |fname|
    ftp.getbinaryfile(fname) # Oder: ftp.putbinaryfile(...)
    ftp.delete(fname)        # Achtung - Fehlerbehandlung ergänzen!
  end
end
```

# Übergangslösung

- Vor Einsatz des EDI-Hubs
  - „Anonymous ftp“-Server auf „lx2-beam“
    - Loggen Sie sich dort ein
    - Legen Sie im Ordner „incoming“ einen Unterordner an. Er lautet wie die GLN Ihrer Firma.
    - Nutzen Sie diesen für erste Tests, indem Sie die EANCOM-Testdaten dorthin übertragen bzw. von dort abholen.
  - Tipp:

```
Net::FTP.new('lx2-beam').open do |ftp|
  ftp.login
  ftp.chdir('incoming') # Später: 'incoming/1234567890123'
  ftp.mkdir('1234567890123') # Später auslassen
  # etc.
end
```

- Ausblick auf Betrieb mit dem EDI-Hub
  - Server und Unterverzeichnisse werden ausgetauscht
  - Anmeldung mit user/passwd

# Abgaben

- Abzugeben
  - `controld.rb  # Ihr Programm-Code`
- Abgabeordner
  - Wie üblich, unter `~werntges/lv/edi/abgaben/a/<matrnr>`
- Abgabetermin?
  - Dies ist der erste (und größte) Teil einer Doppel-Übung!
  - Diese stellt den Abschluss des Praktikums-Projekts dar.
  - Abgabe ist erst nach Integration des EDI-Hubs vorgesehen, d.h. in zwei Wochen.
- Bemerkungen
  - Nutzen Sie diese erste Woche, um möglichst viel Codierarbeit zu erledigen. Umso mehr Zeit bleibt Ihnen in der kommenden Woche für Integrationstests mit dem EDI-Hub und Ihren „Geschäftspartnern“!