Function follows design...

Live-Dokumentation dynamischer Systeme

Piotr Orlowski und Christoph Oelmüller

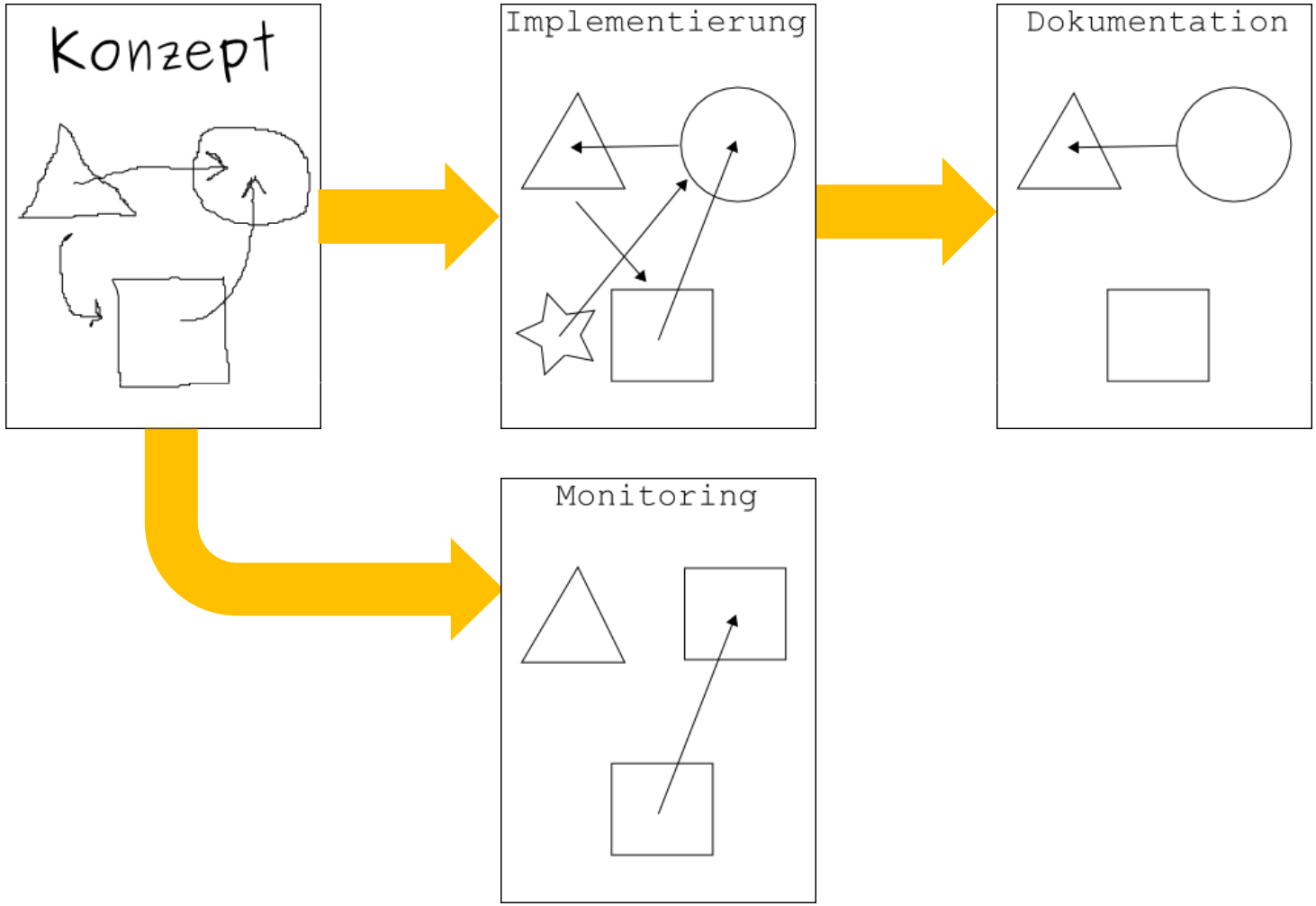
SLAC 2. Dezember 2011

logica

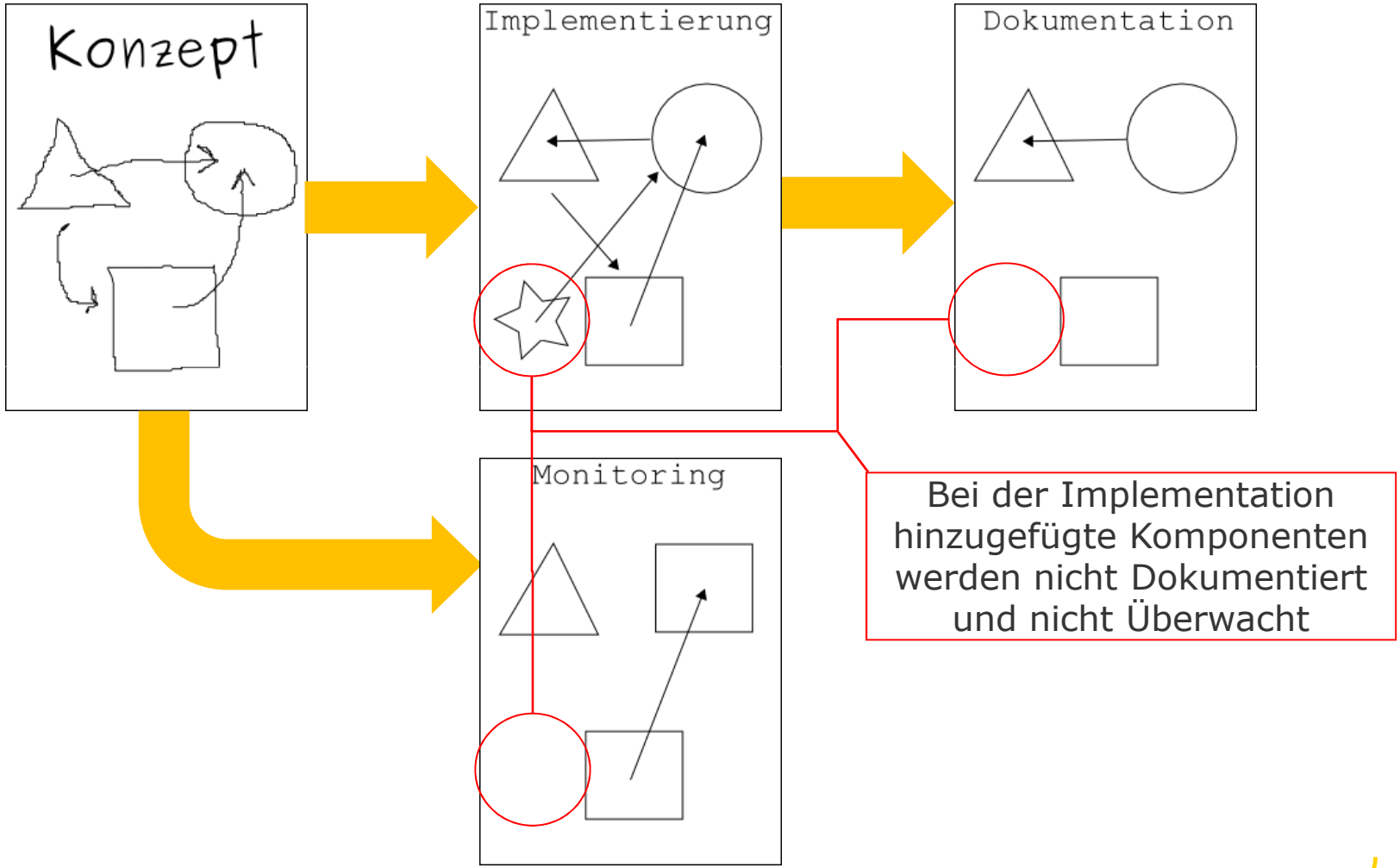
be brilliant together

Das Problem

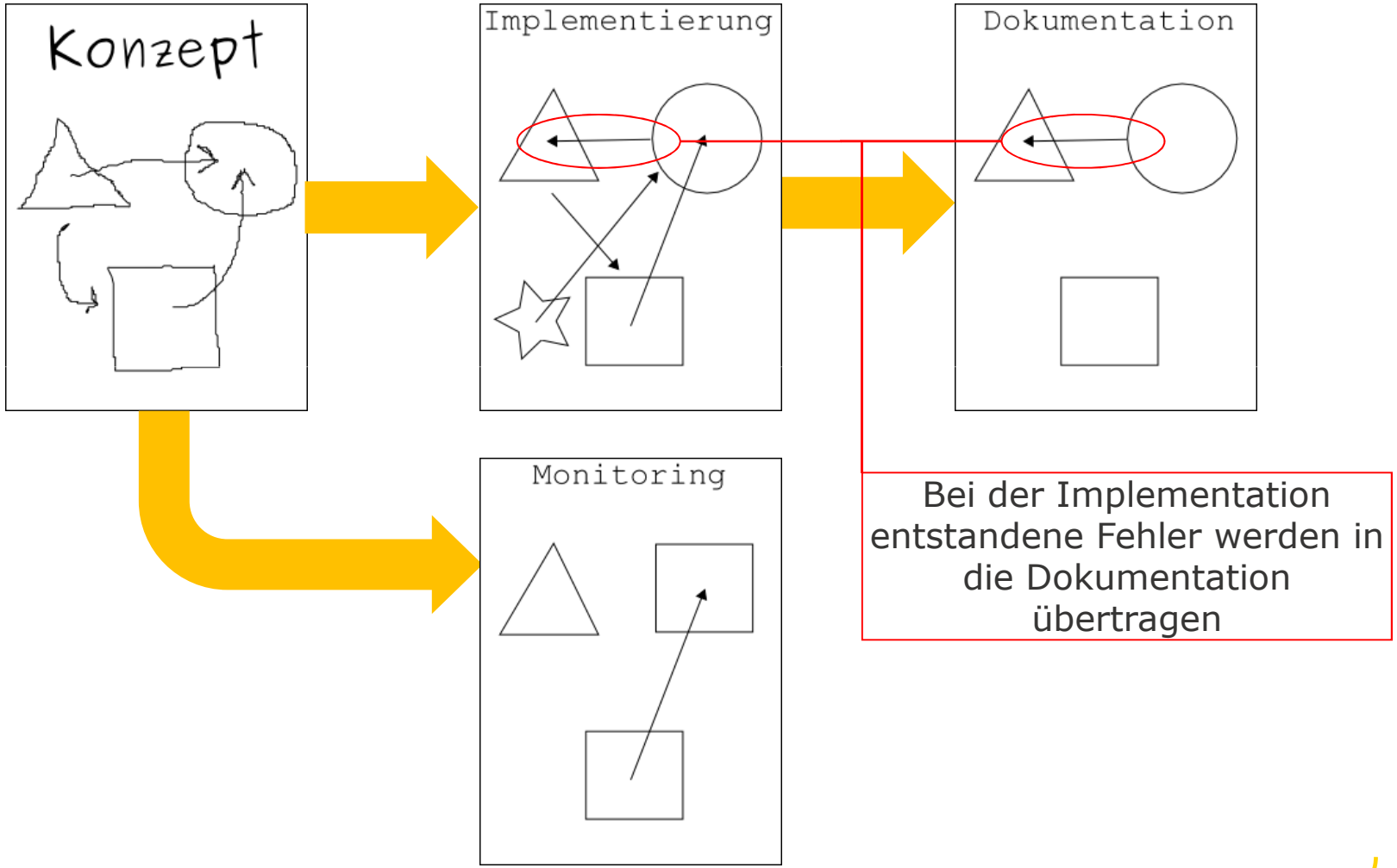
Der klassische Weg



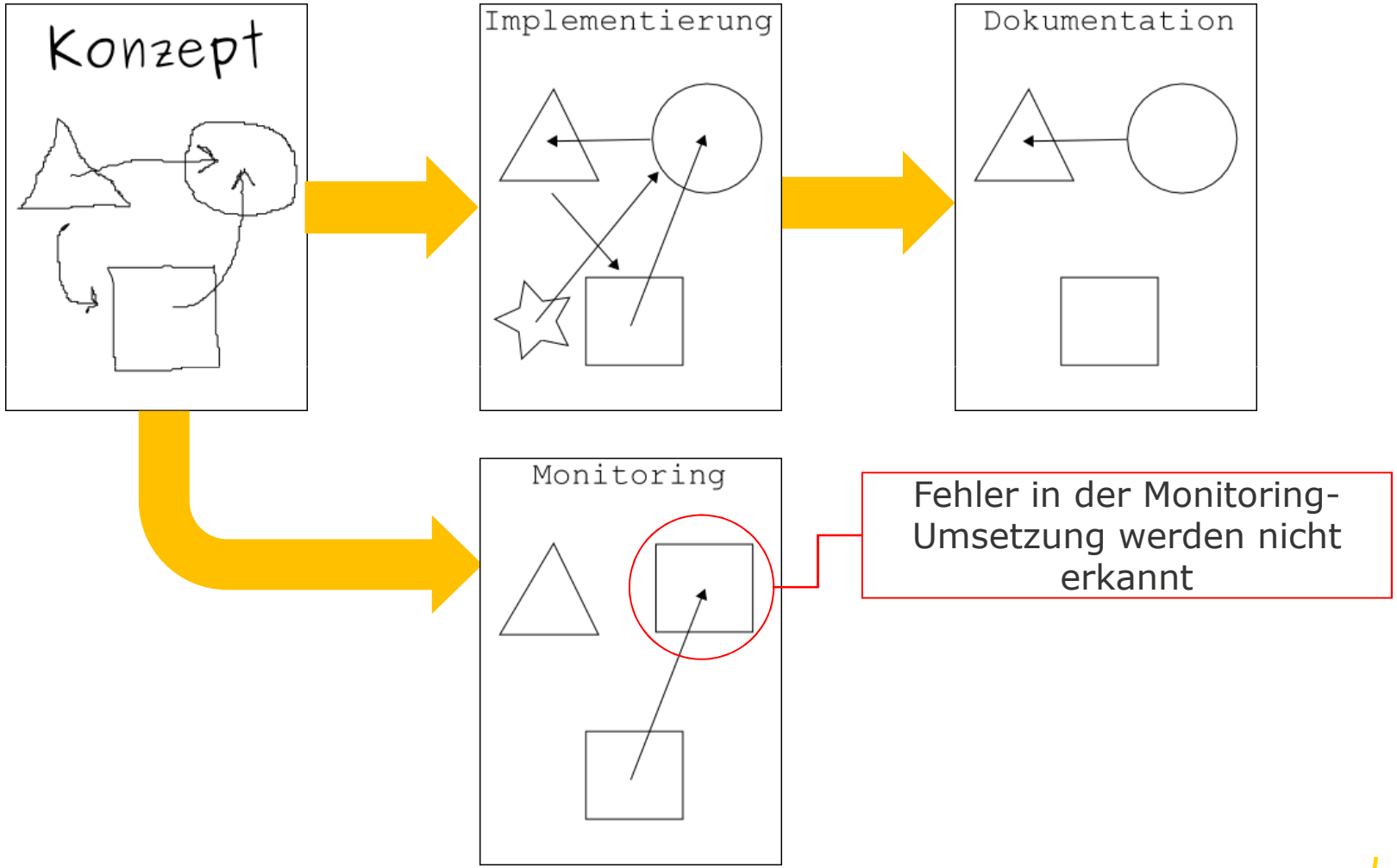
Problem 1: Unvollständigkeit



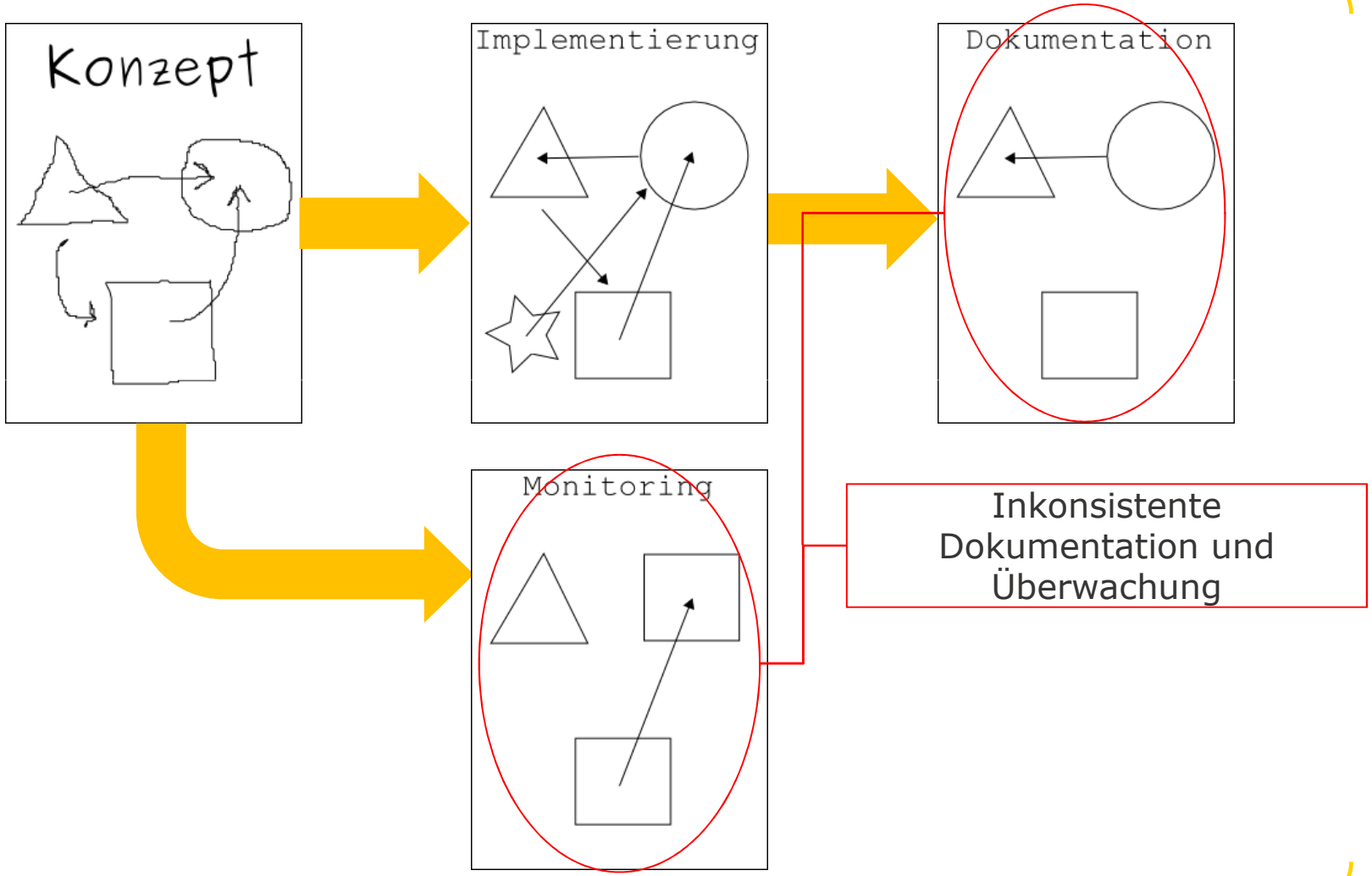
Problem 2: Fehlerverschleppung



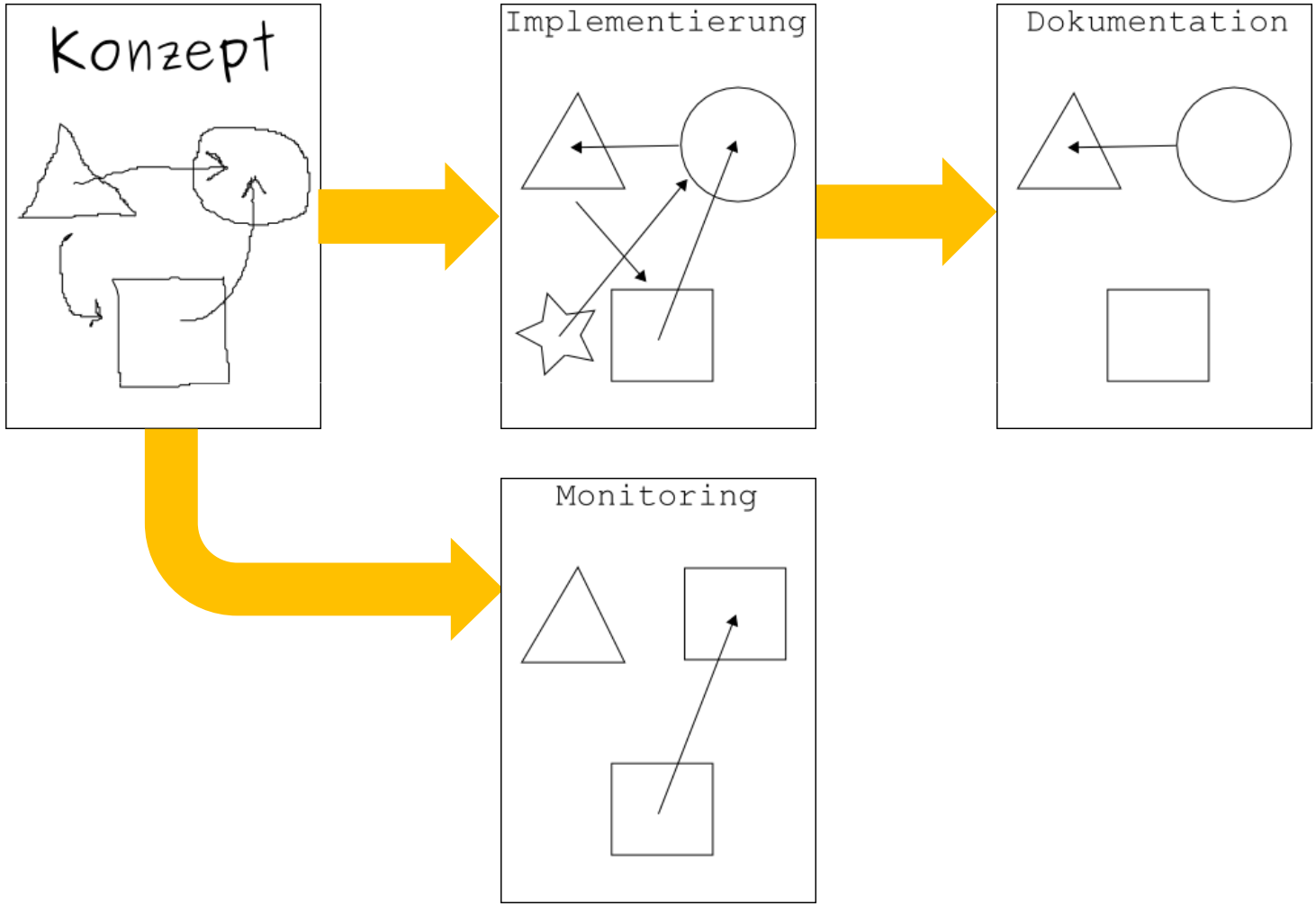
Problem 3: Fehleranfälligkeit



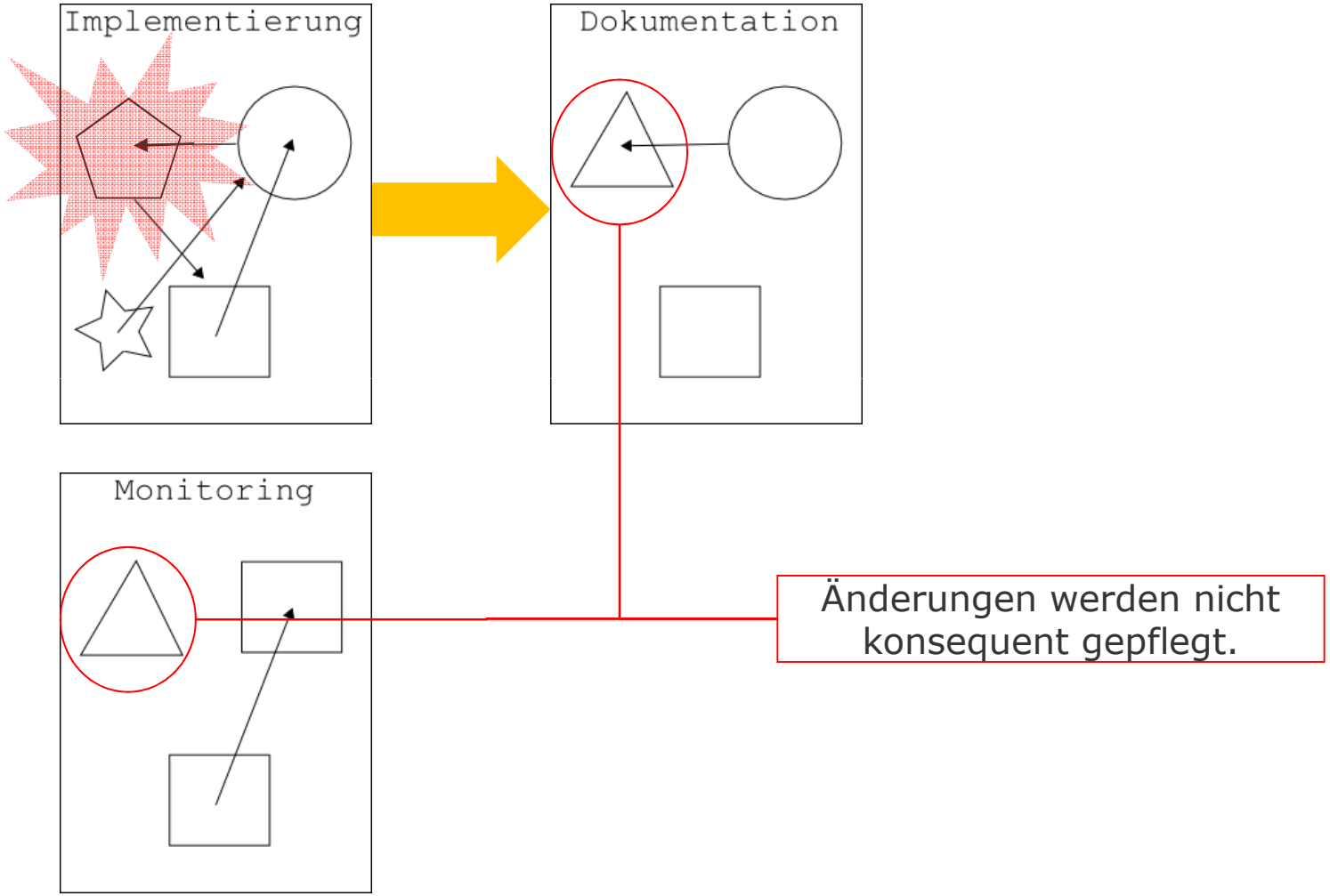
Problem 4: Inkonsistenz



Problem 5: Aktualität

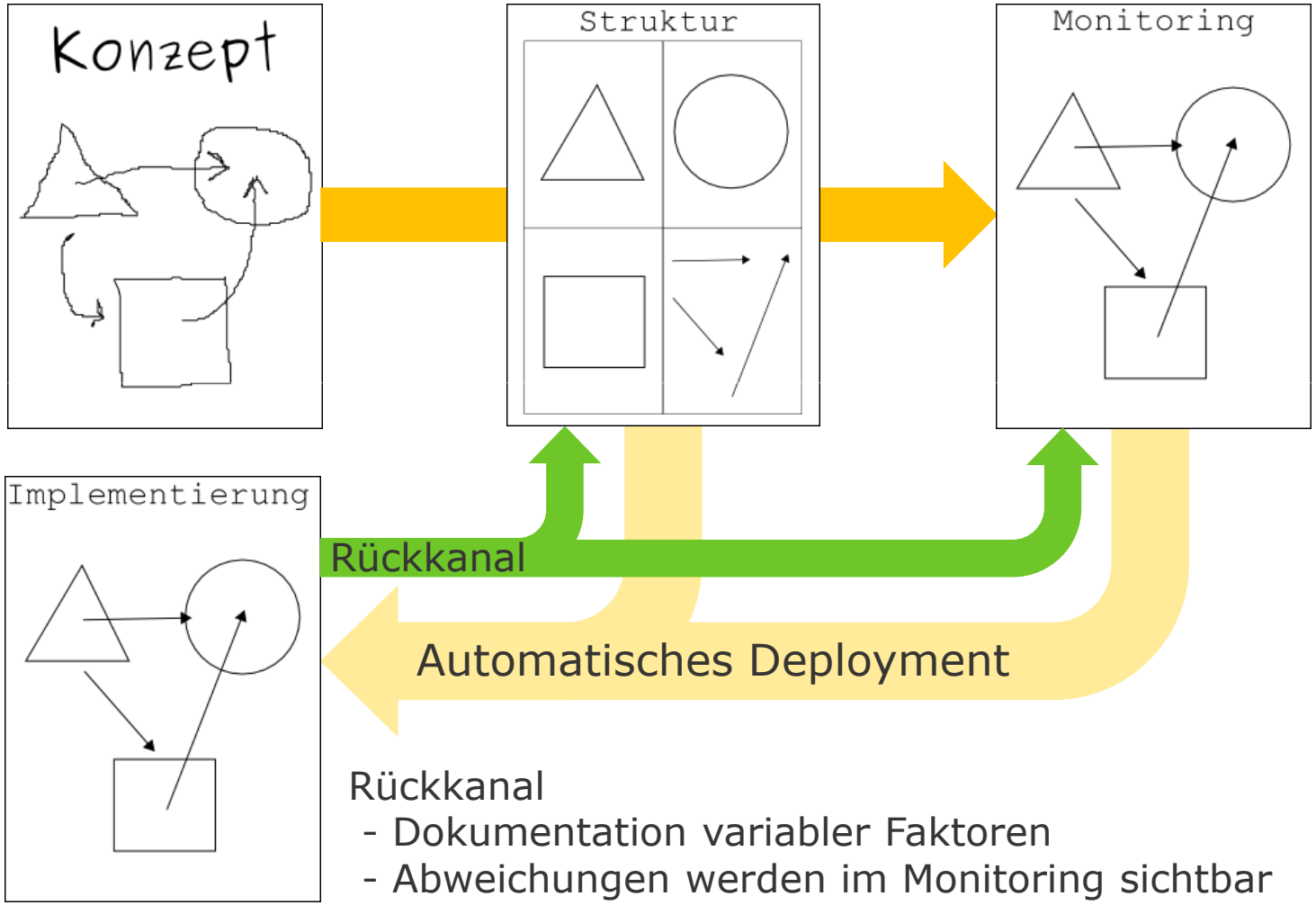


Problem 5: Aktualität



Lösungsansatz

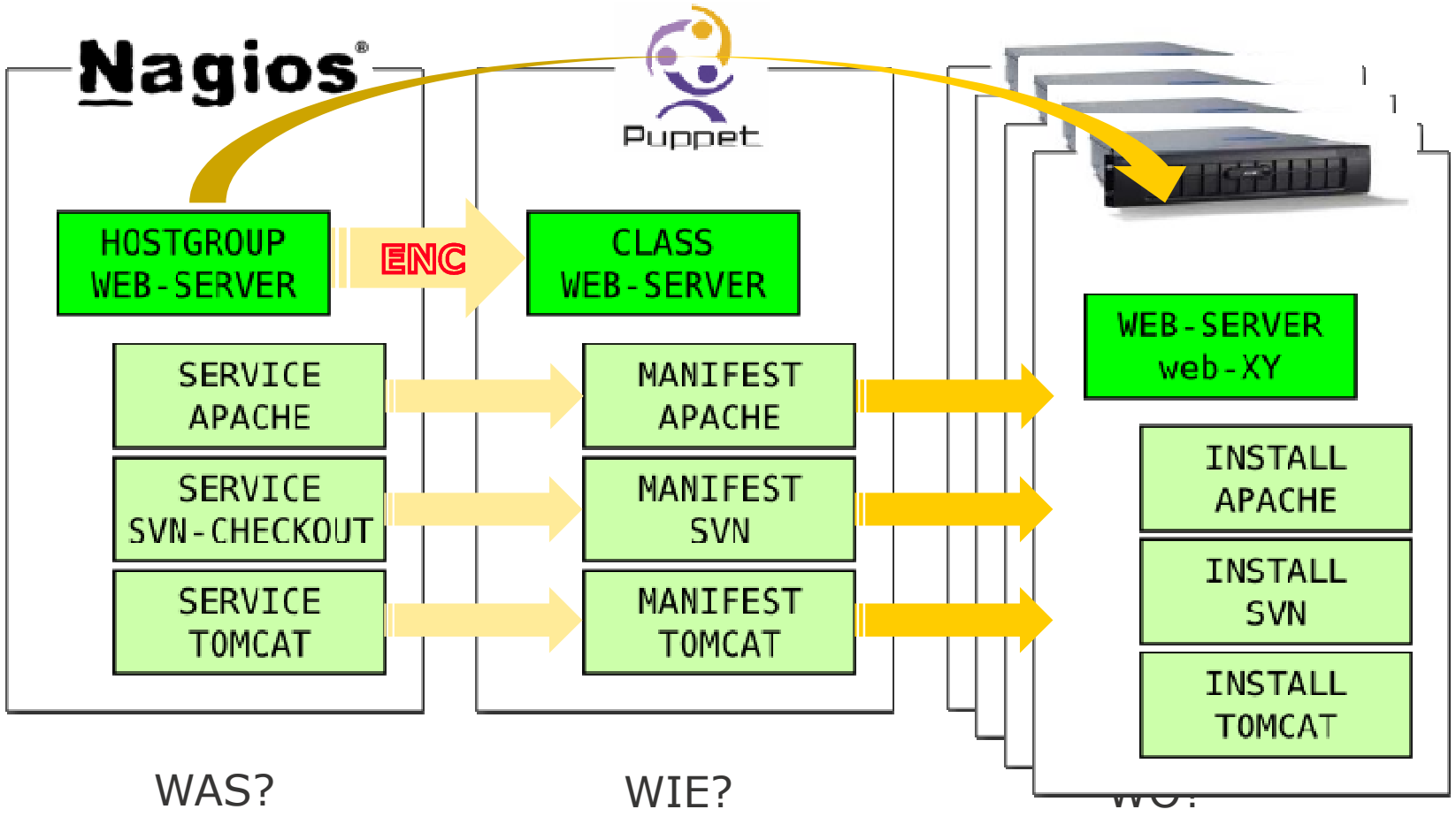
Lösungsansatz



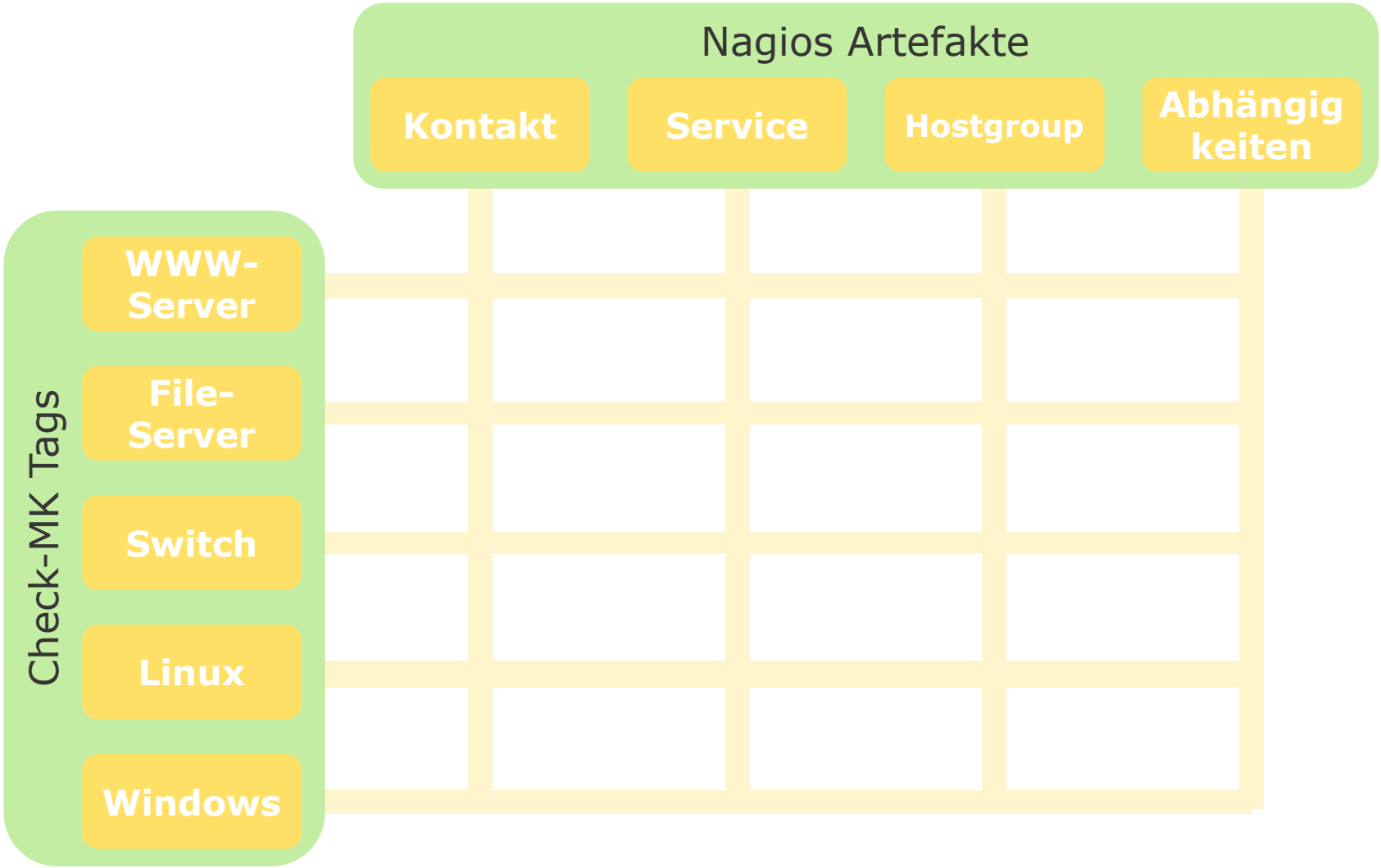
Rückkanal
- Dokumentation variabler Faktoren
- Abweichungen werden im Monitoring sichtbar

Umsetzungs-konzept

Zusammenführung der Dokumentation



Nagios: Konfigurationsstruktur mit Check_MK



Beispielkonfiguration mit Check_MK

```
all_hosts += [
hostxy|tag-xy| ]
```

```
host_groups += [
 ( hg_xy, [tag-xy], ALL_HOSTS ]
```

```
legacy_checks += [
((check_XY, XY-Check, False), [tag-xy], ALL_HOSTS)]
```

```
define command {
        command_name      check_XY
        command_line      /path/to/check_xy.pl}
```

Erweiterte Dokumentation in Nagios

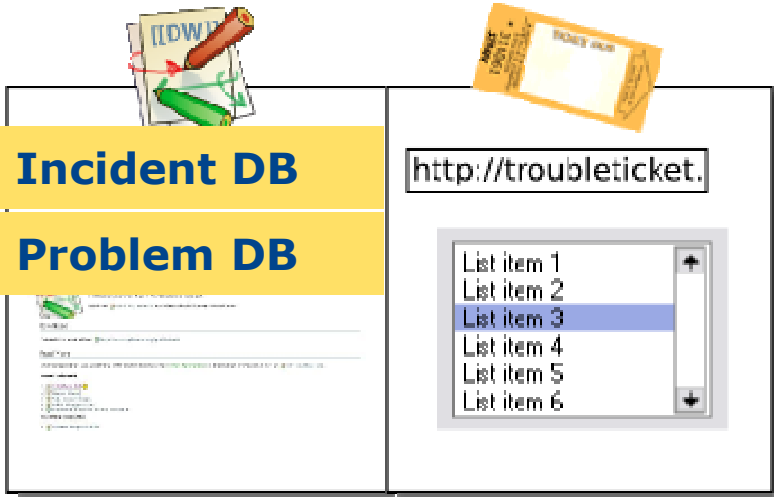
- **Funktionalität**
- **Abhängigkeiten**
- **Gruppierungen**
- **Verantwortliche**
- **Prioritäten**

- **SDH**
- **AdminLog**
- **HW-Support**

Erweiterte Dokumentation in Nagios

- **Funktionalität**
- **Abhängigkeiten**
- **Gruppierungen**
- **Verantwortliche**
- **Prioritäten**

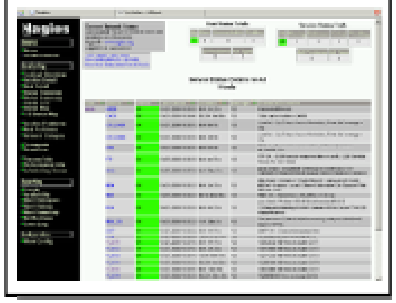
- **SDH**
- **AdminLog**
- **HW-Support**



Erweiterte Dokumentation in Nagios

Nagios®

http://nagios.intern



SDH

AdminLog

HW-Support

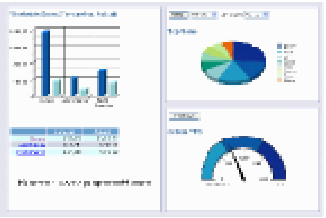
http://dokuwiki.inte

Kennzahlen

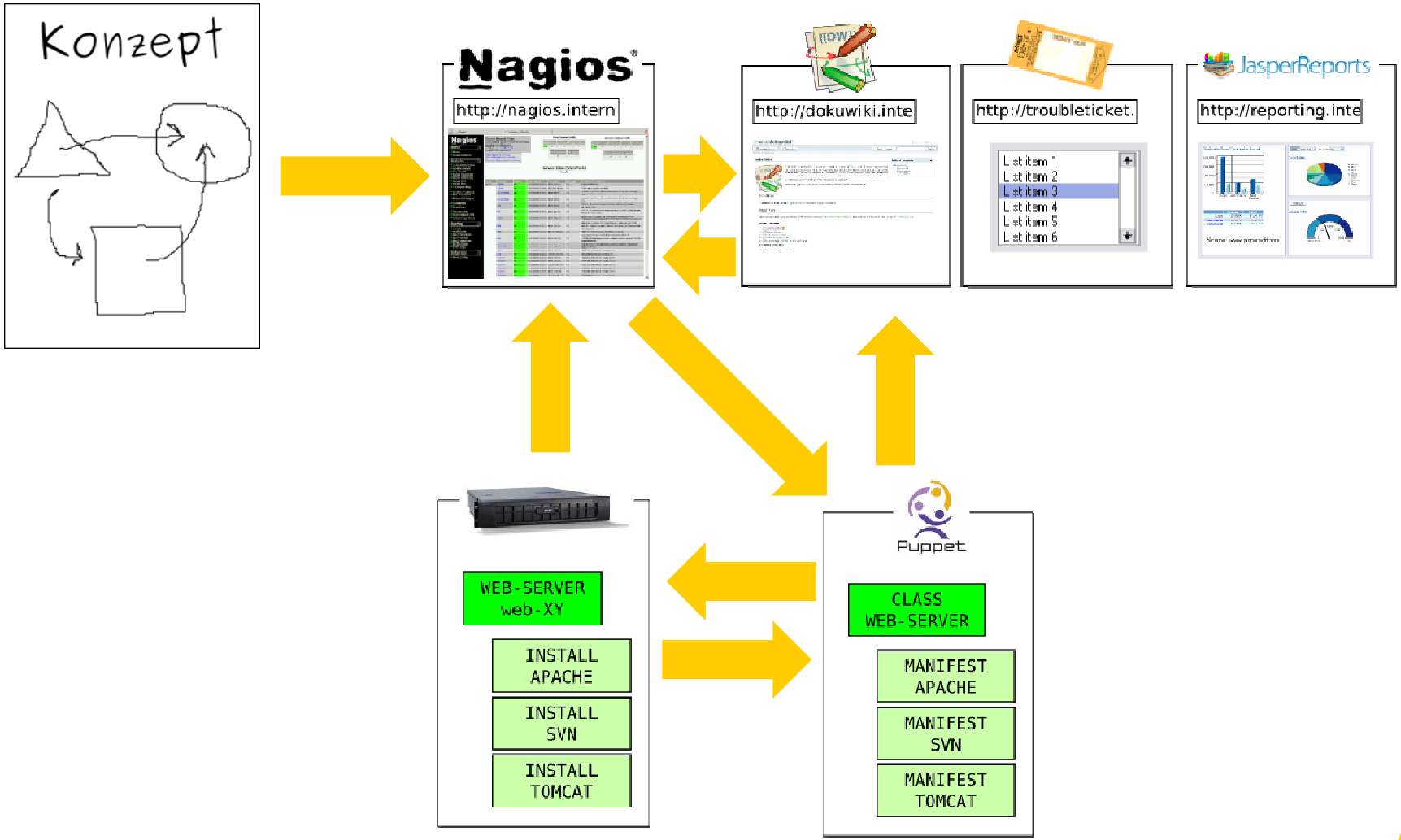
Auslastung

JasperReports

http://reporting.inte

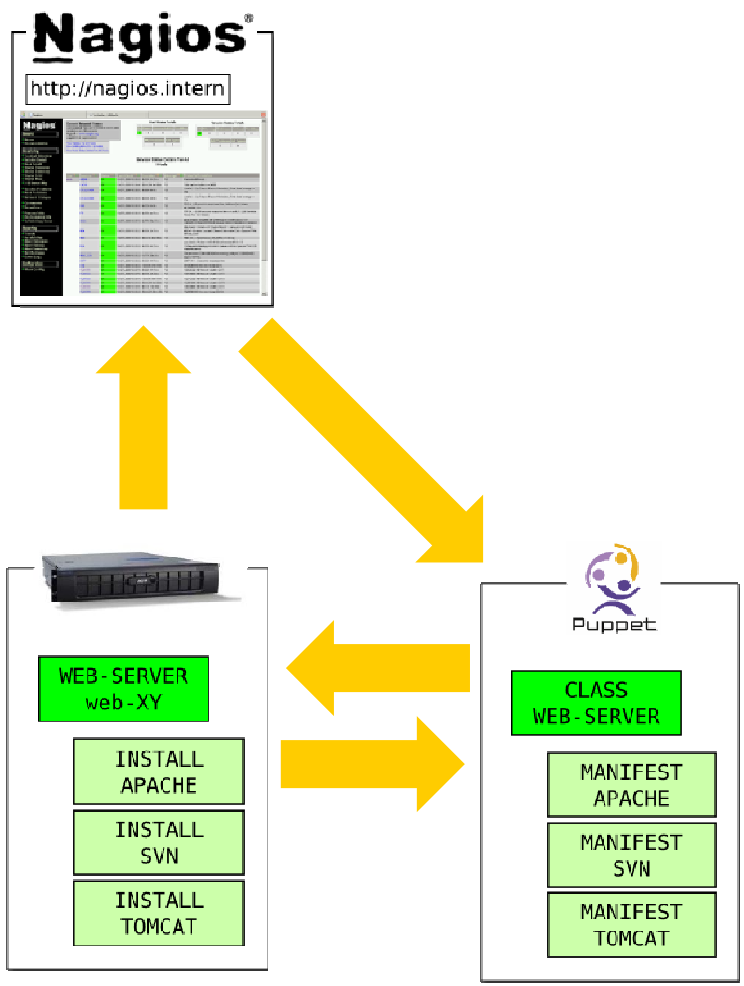


Der komplette Entwurf des Systems



Technische Umsetzung

Von der Struktur zur Implementierung

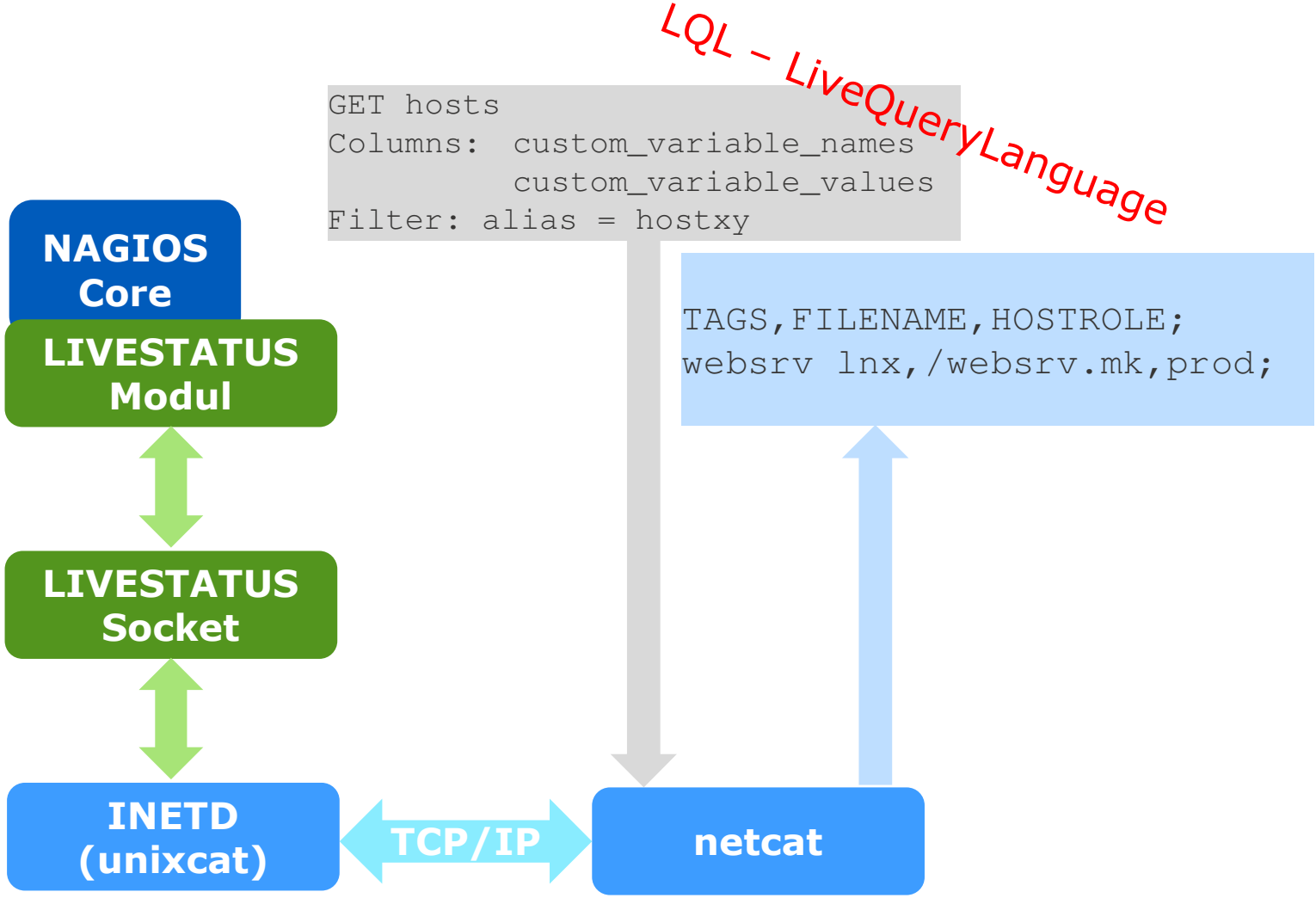


Struktur im Monitoring bestimmt die Implementierung

Kommunikation Puppet-System ist keine Einbahnstraße

Monitoring zeigt den Zustand des Systems

Check_MK Livestatus



Von der Struktur zur Implementierung cont.

Auf welchen Host werden welche Puppet-Manifeste angewendet?

Bisherige Antwort:

```
node node086 inherits basenode {
    include puppetclass_aaa
    include puppetclass_bbb
    include puppetclass_zzz
}
```

ENC – External Node Classifier:

$$ENC(fqdn) = \{classes, environment, parameter\}$$

ENC

```
classes:
    classA:
    classC:
parameters:
    parameterAAA: foobar
    parameterXYZ:
         key:
            - value1
environment: production
```

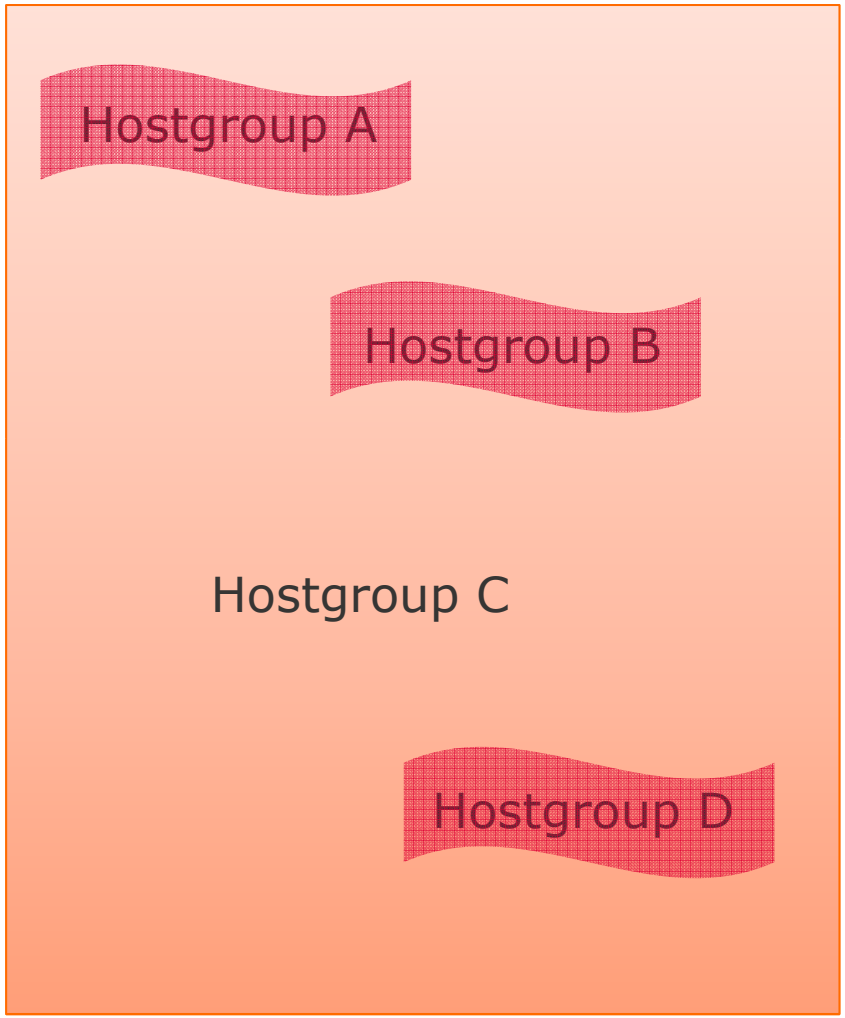
Antwort auf die Frage „wer bekommt was“ als YAML

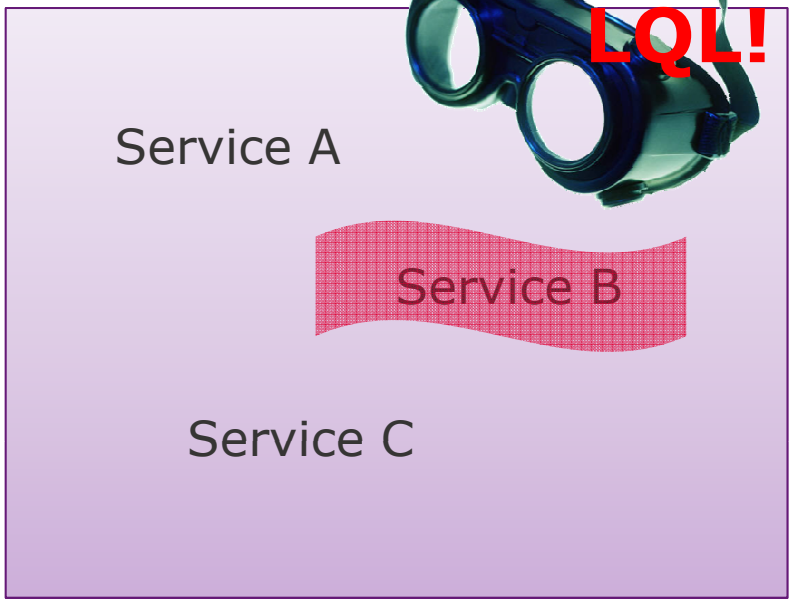
Livestatus-ENC

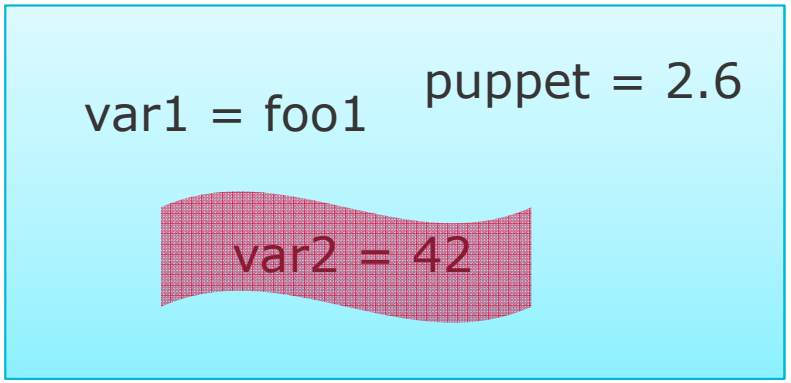
$$ENC(fqdn) = \begin{pmatrix} \{hostgroup\}|_{host=fqdn} \\ \{service\}|_{host=fqdn} \\ \{hostparameter\}|_{host=fqdn} \\ \dots \end{pmatrix}$$

Antwort auf die Frage „wer soll was sein“…

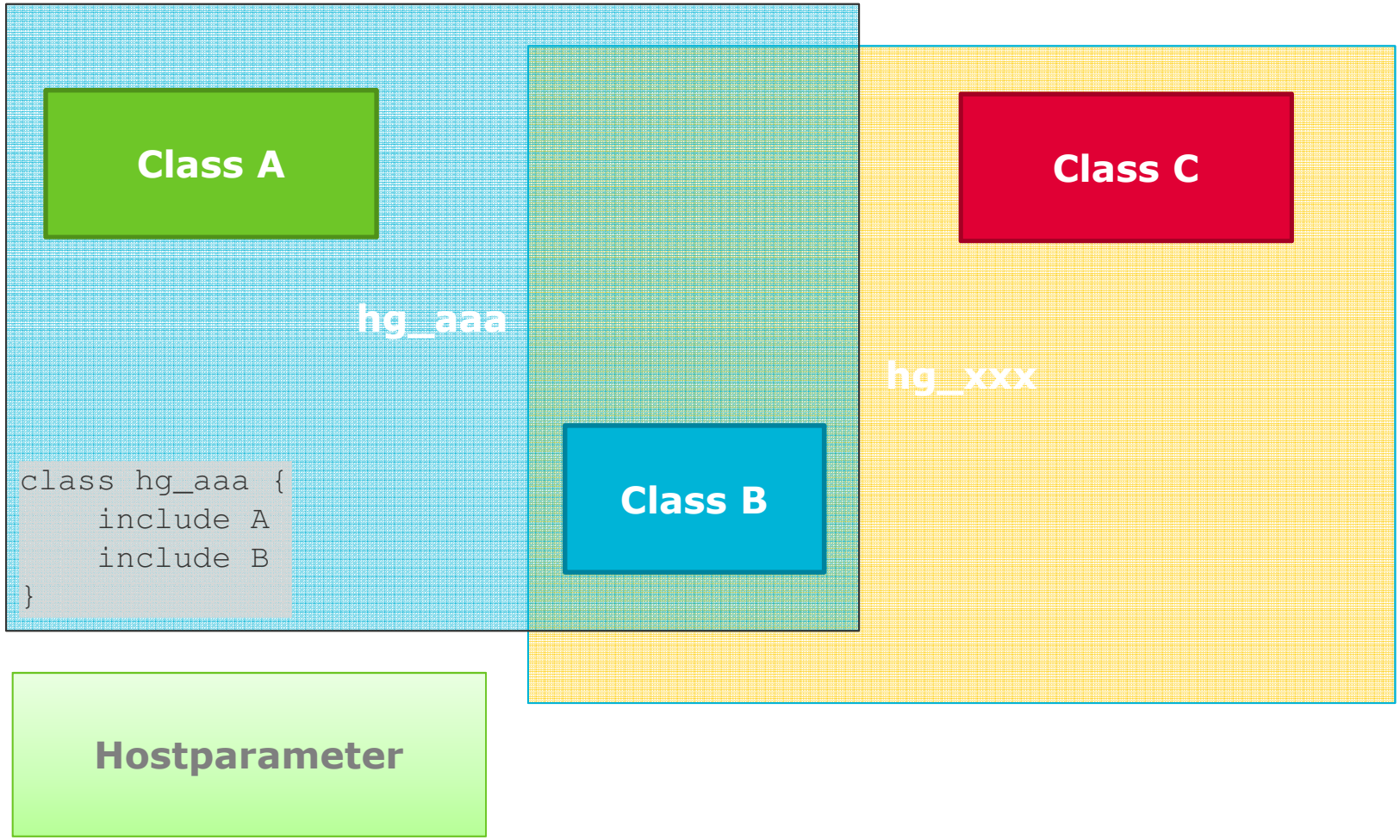
Livestatus-ENC cont.







LS-ENC Klassenkonzept



LS-ENC: Verwendung der Parameter

Manifest:

```
class ntp {

    # default variables
    $ntp_servers_default = ['time1.domain.tdl','time2.domain.tdl`]
    $ntp_servers_options_default = 'iburst'
    # …
    package { 'ntp':
        ensure      => installed
    }


    file { '/etc/ntp.conf':
        ensure      => file,
        content     => template('ntp/ntp.conf.erb'),
        ….
    }

    ….
}
```

LS-ENC: Verwendung der Parameter cont.

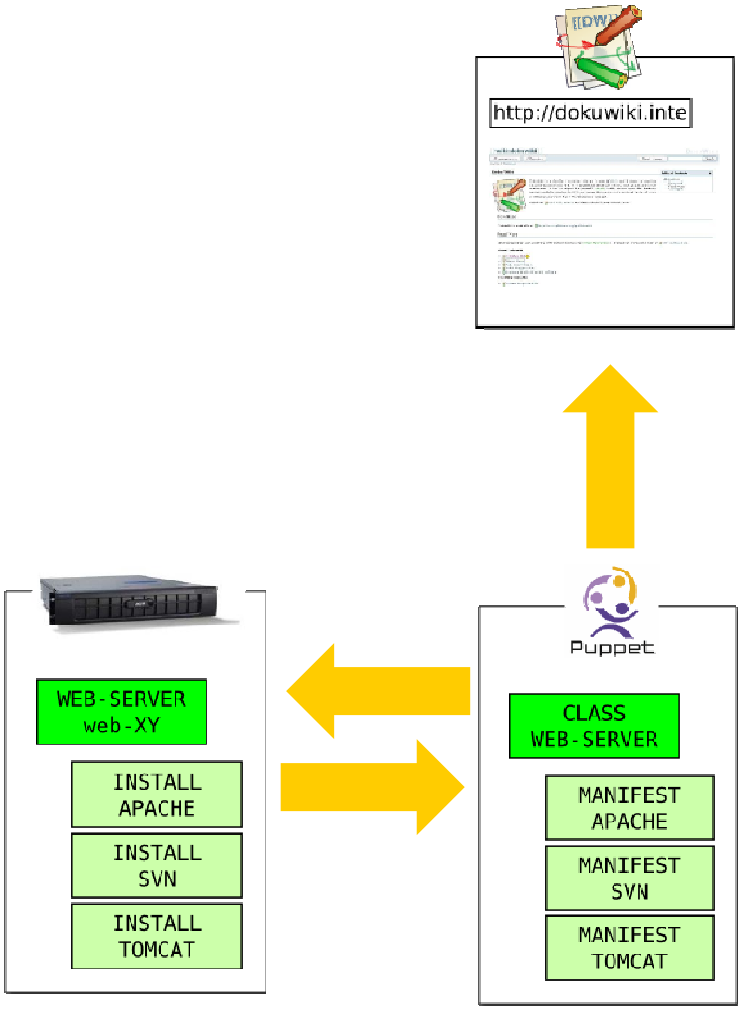
Template:

```
<%
# use specific or if not defined the default vars
has_variable?("ntp_servers„)
    ? servers = ntp_servers
    : servers = ntp_servers_default
%>

# servers
<%servers.each do |server| -%>
server <%= server %> <%= servers_options %>
<%end -%>
```

Hostvariablen haben Vorrang vor Klassenvariablen!
oder es muss ein anderes Konzept geben…

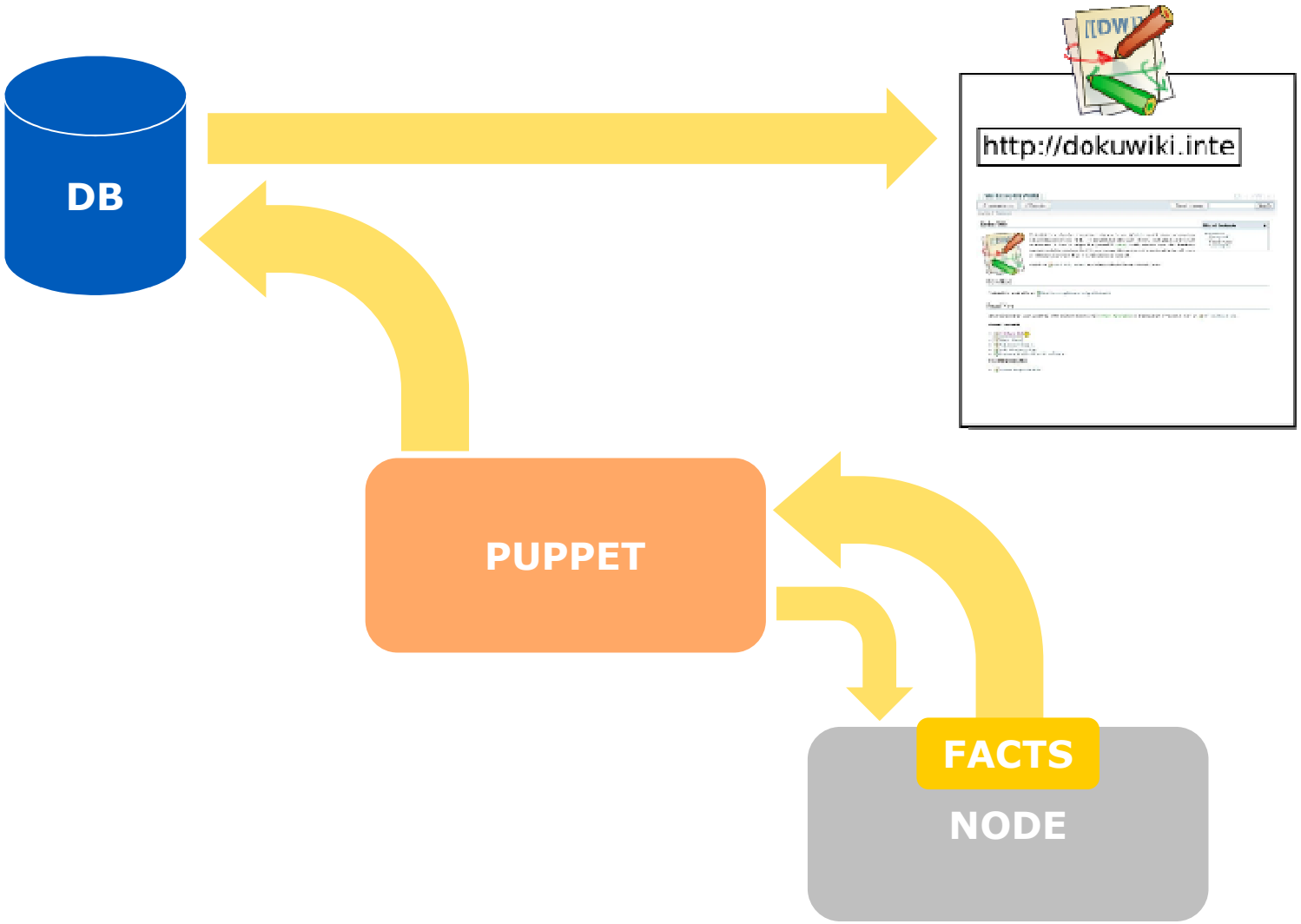
Autogenerierte Dokumentation



Dokumentation
dynamischer Fakten

Dokumentation
manueller Eingriffe

Facts



Puppet: Exported Resources

Virtual resource

Definition:

```
@user {
    luke: ensure => present
}
```

Anwendung:

```
User <| title == luke |>
```

Exported resource

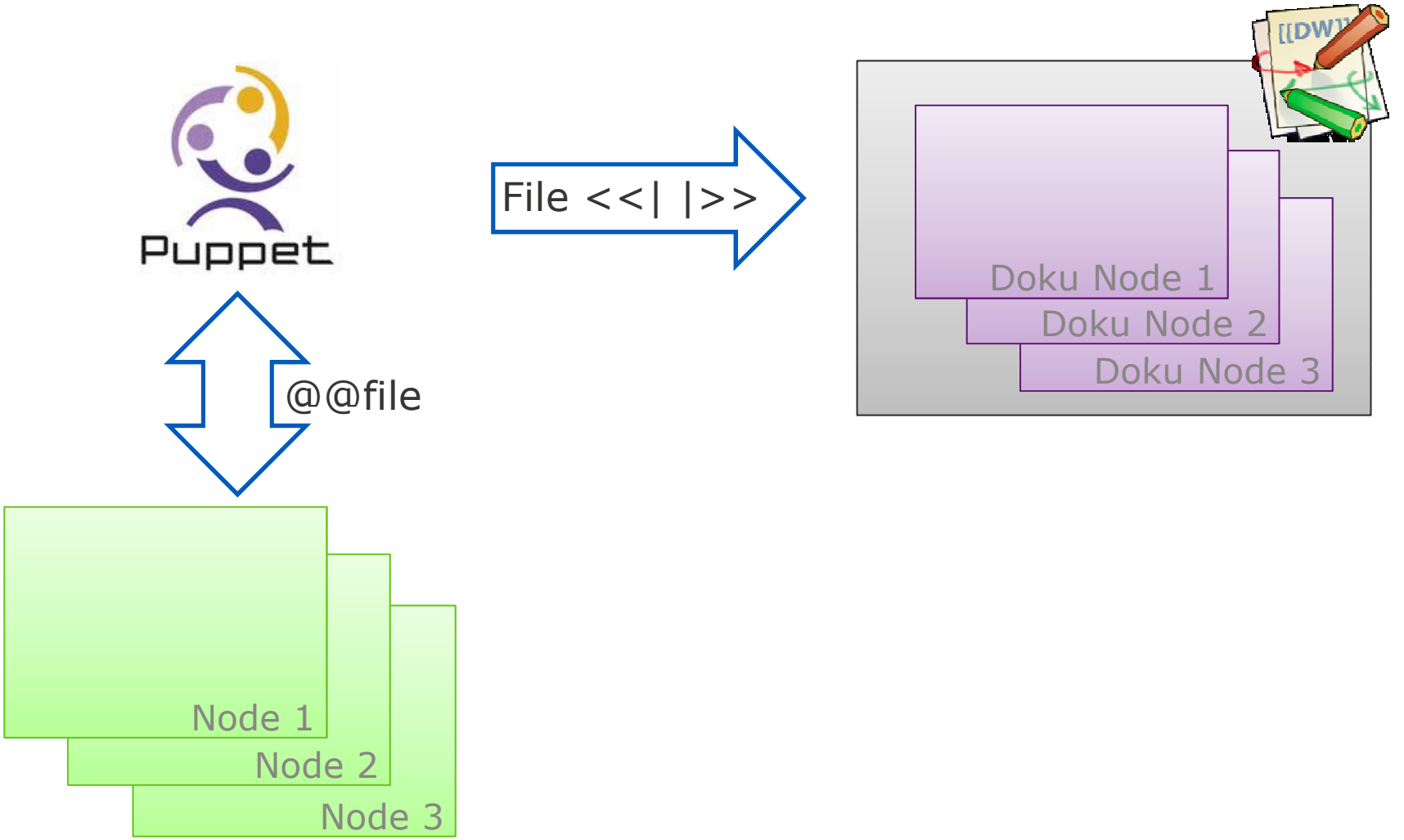
Definition:

```
@@sshkey { $hostname:
    type => dsa,
    key => $sshdsakey
}
```

Anwendung:

```
Sshkey <<| |>>
```

Autogenerierte Dokumentation cont.



Autogenerierte Dokumentation cont.

- Server-Datenblatt realisiert als exportierte file-Ressource
  - ✓ Verwendung von erb-Templates
  - ✓ Verfügbarkeit aller Facts des exportierenden Nodes im Template
  - ✓ automatische Erstellung bei Anwendung des Manifests
  - ✓ automatischer Aktualisierung des Datenblatts bei Fact-Änderungen

```
@@file {„dokuwiki-$fqdn“:
    path => „/srv/www/htdocs/data/pages/$hostname“,
    content => template(‚serverdoku/datasheet.erb‘),
    …
}
```

```
===== Server <%= $hostname -%> =====
IP-Address: <%= ipaddress -%>
FQDN: <%= $fqdn -%>
…
```

AdminLog

Zwang zur Dokumentation

```
tmplog=$(mktemp /tmp/tmplog.XXXX)
$EDITOR $tmplog
cat $tmplog >> /usr/share/adminlog/logentries
```

.bash_logout

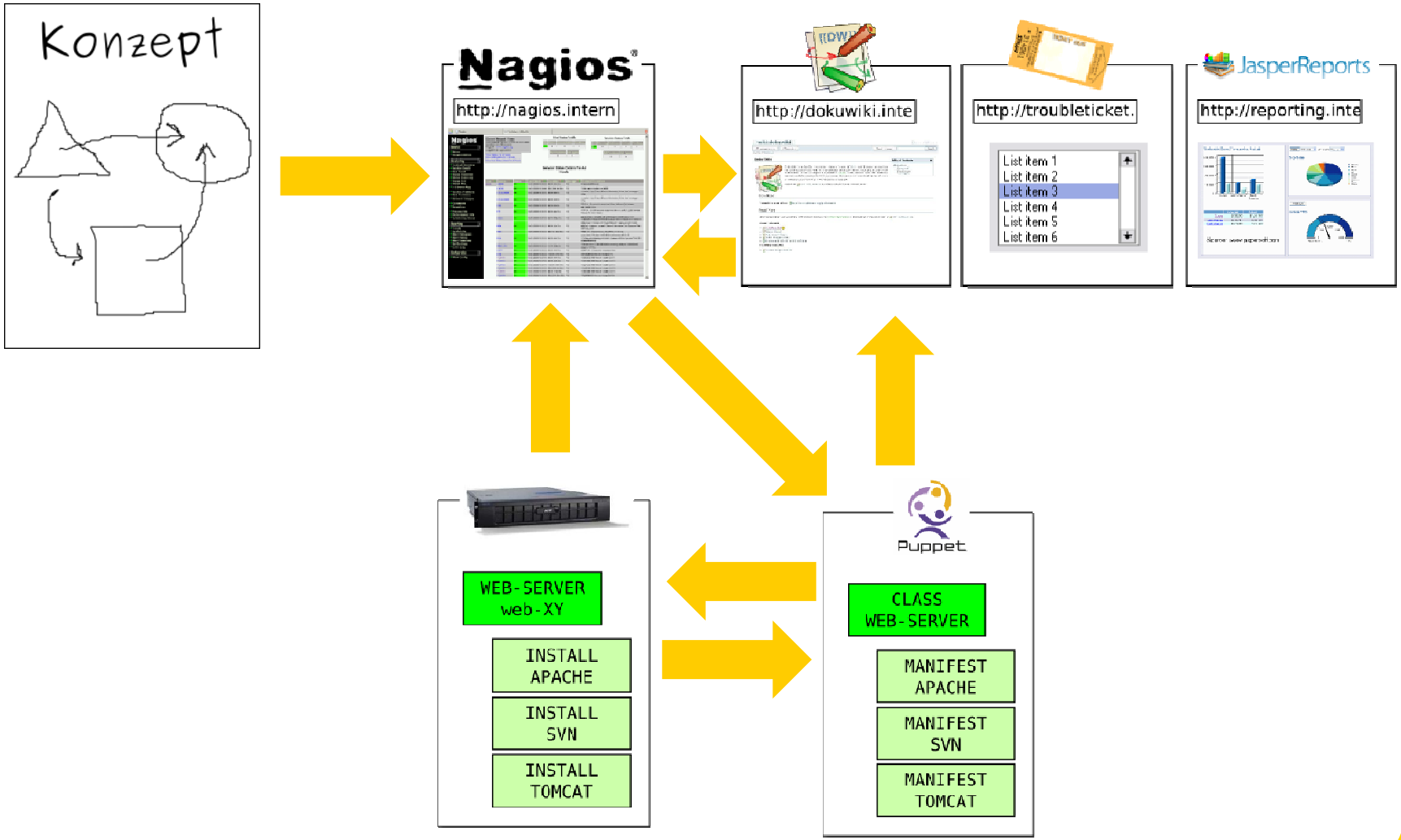
Interpretation der lokalen Logeinträge als Fact

```
export FACTER_adminlog=$(cat \
 /usr/share/adminlog/logentries )
```

.bashrc

Einbindung in Datenblatt-Template

Der komplette Entwurf des Systems



Danke

Christoph Oelmüller, Piotr Orlowski