Siempelkamp
Condition Monitoring System

Condition Monitoring System in Lua unter RTOS-UH

Tobias Aretz, Jan Bartels und Dennis Göbel
Siempelkamp Maschinen- und Anlagenbau GmbH

ODER:

Wieso Condition Monitoring für Holzwerkstoffanlagen auf eigenem Singleboardcomputer mit Echtzeitbetriebssystem in einer Skriptsprache, die zur Einbindung in andere Programme entwickelt wurde?

Inhalt

Siempelkamp
Condition Monitoring System

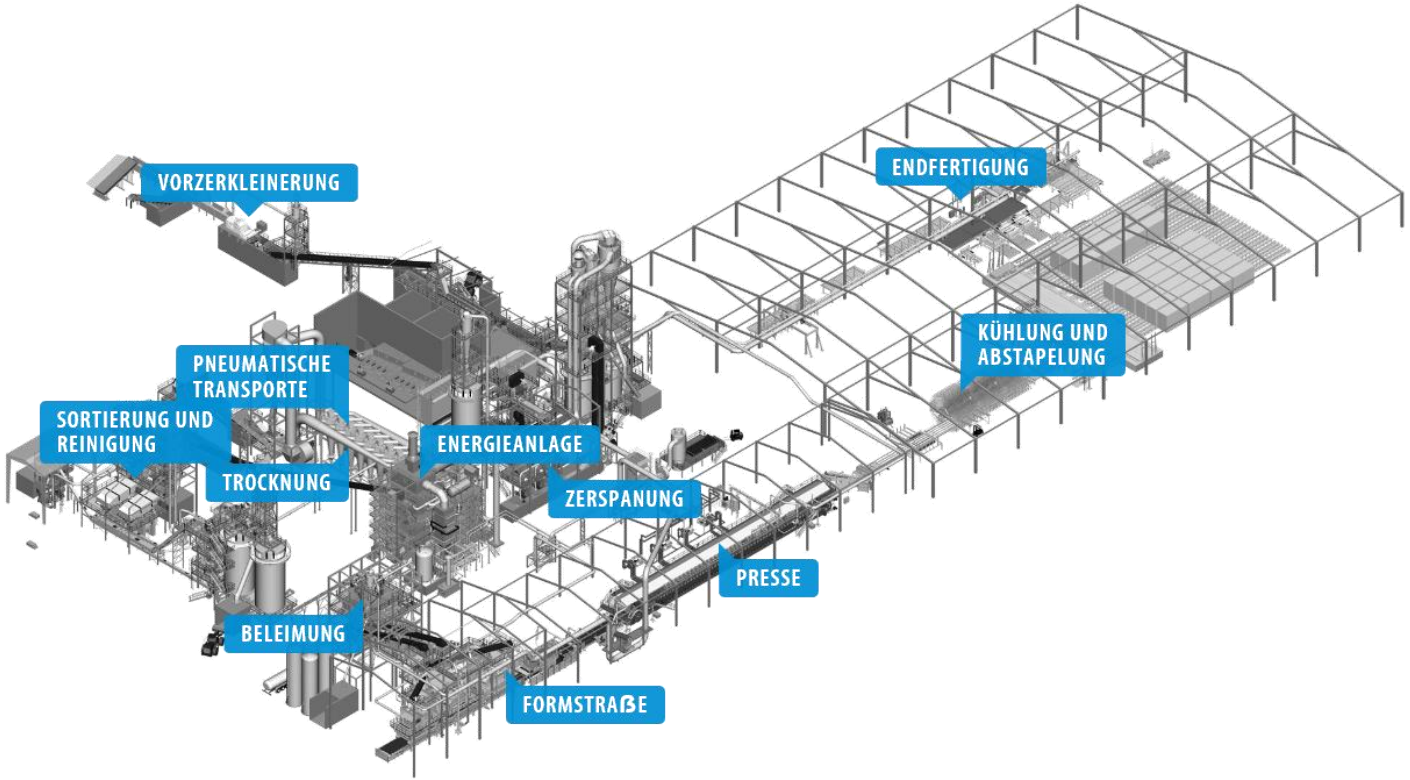
- Motivation zum Condition Monitoring
- Anforderungen
- Finden einer Lösung
- Lua auf RTOS-UH
- Fazit und Ausblick

Condition Monitoring an der ContiRoll

MOTIVATION

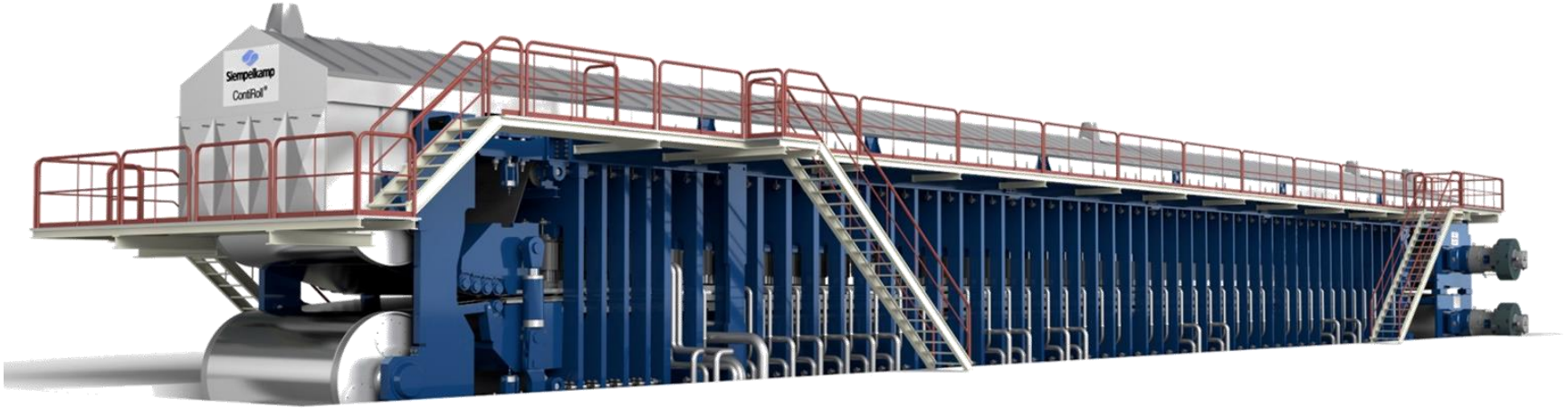
Holzwerkstoffanlage – vom Baumstamm zur Spanplatte

Siempelkamp
Condition Monitoring System



Die kontinuierliche Presse „ContiRoll“ als Herzstück der Anlage

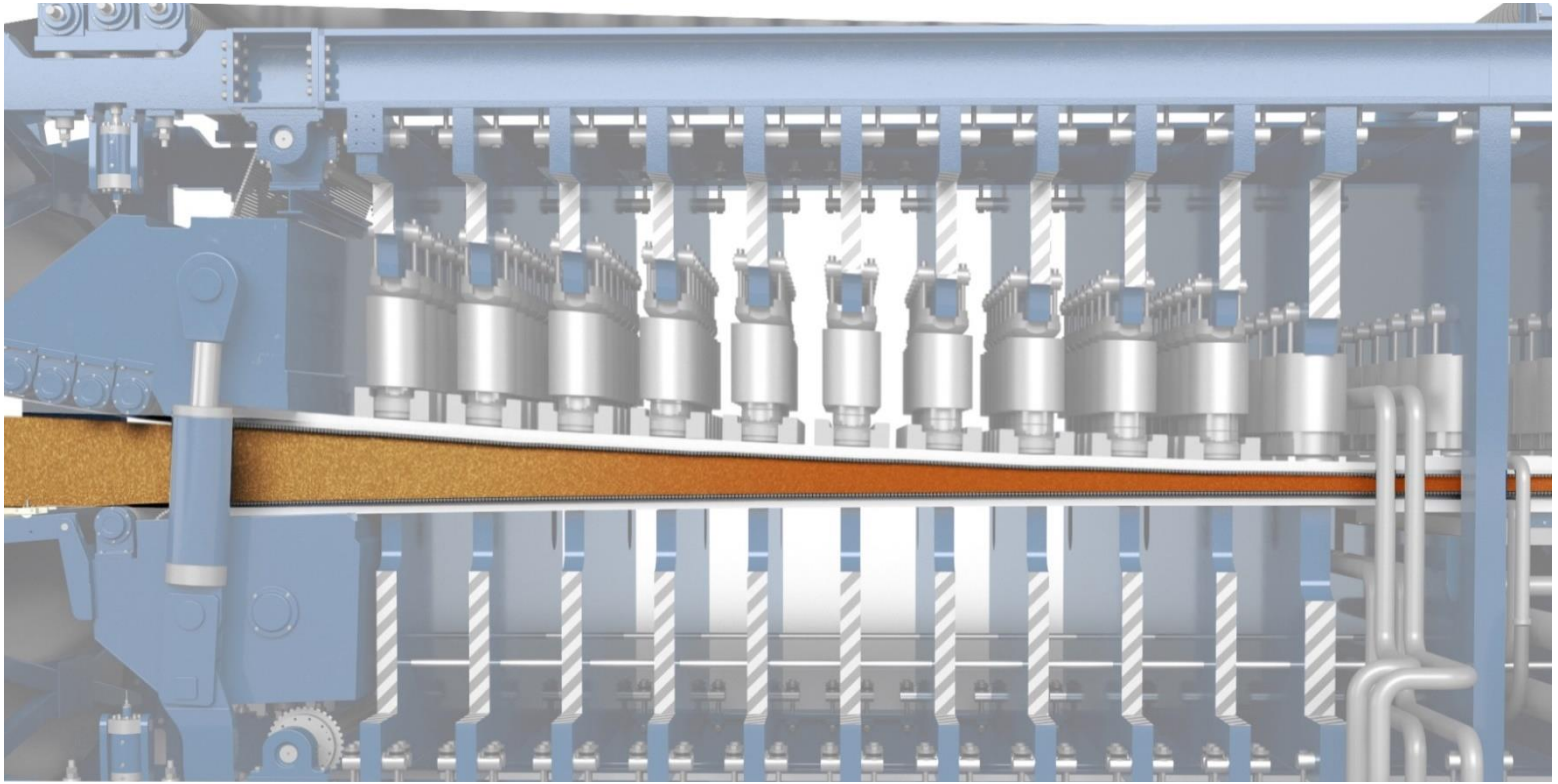
Siempelkamp
Condition Monitoring System



- Länge zwischen 30-60 m
- Breite zwischen 4 und 10 Fuß
- Geschwindigkeit bis 2000 mm/s
- Plattendicke zwischen 1,5 und 42 mm
- ca. 60-100 hydraulische Achsen
- 300 bar Systemdruck
- > 200 °C Heizöltemperatur
- Tagesleistung bis zu > 1.000 m³

Der Prozess in der ContiRoll

Siempelkamp
Condition Monitoring System



Wieso Condition Monitoring für Holzwerkstoffanlagen?

Siempelkamp
Condition Monitoring System

- Uptime zählt
- Ausnutzung der Lebensdauer von Komponenten
- Teils langwierige Ersatzteilbeschaffung
- Beurteilung des Anlagenzustandes unabhängig von Qualifikation des Personals

Wieso Condition Monitoring an der ContiRoll?

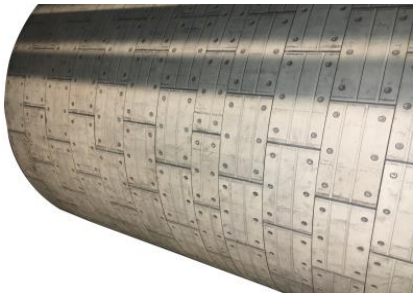
- ContiRoll ohne Redundanz
- *Hersteller-Know-How und Erfahrung*
- *keine Lösung für spezifische Schädigungen am Markt verfügbar*

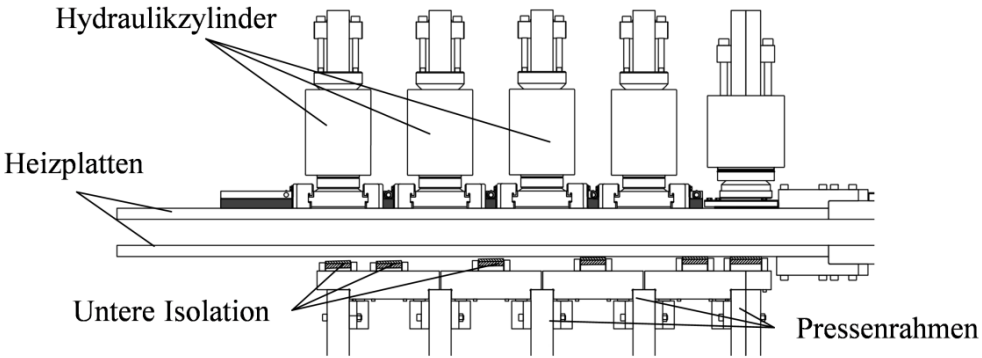
Wahl der Hardwarebasis, Betriebssystem und Programmiersprache

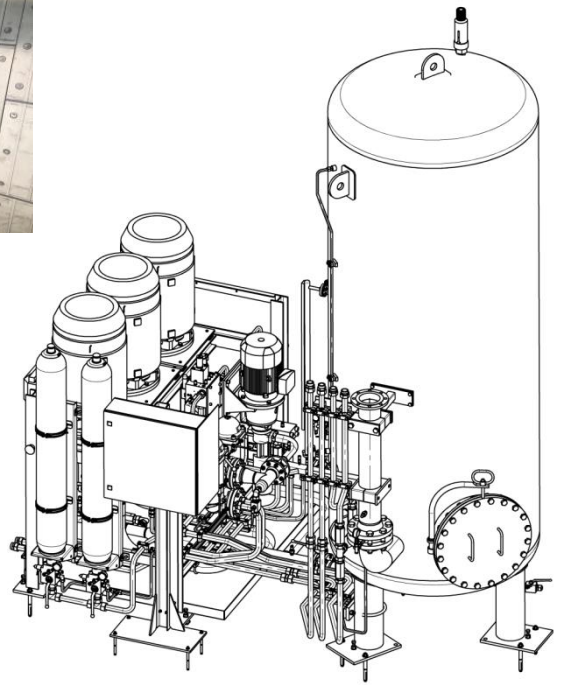
ANFORDERUNGEN

Aufgabenstellung aus dem Engineering: Zustandsüberwachung an Komponenten unterschiedlichster Art

Siempelkamp
Condition Monitoring System

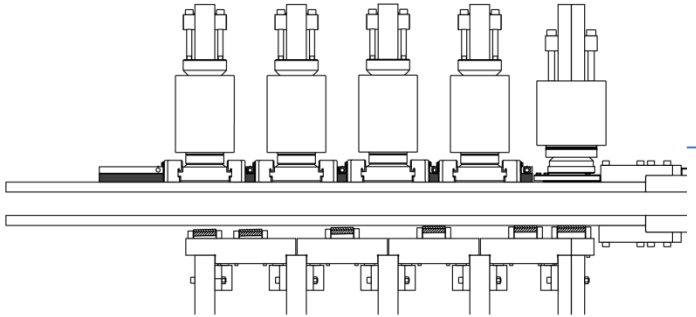






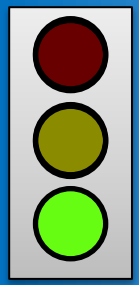
Spezifische Überwachung mit interner Logik: Höhenprofil der Isolationskassetten

Siempelkamp
Condition Monitoring System



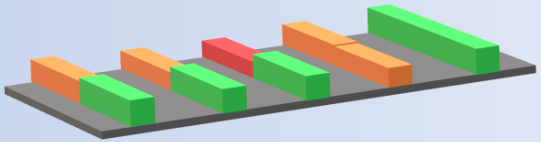
PROFI NET INDUSTRIAL ETHERNET

SCMS



Information

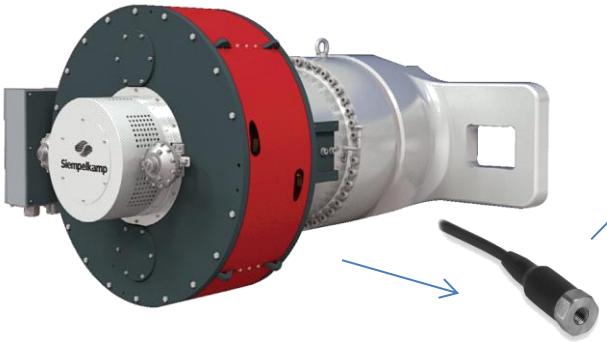
Auswertung durch Siempelkamp-Know-How

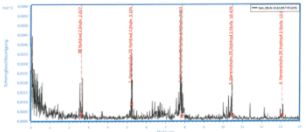


Quellen: MTS Sensor Technologie GmbH & Co. KG

Generische Überwachung mit komplexer externer Logik: Schwingungsanalyse am Antriebsstrang

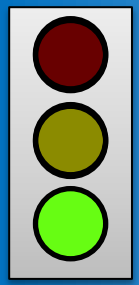
Siempelkamp
Condition Monitoring System





PROFI INDUSTRIAL ETHERNET NET

SCMS



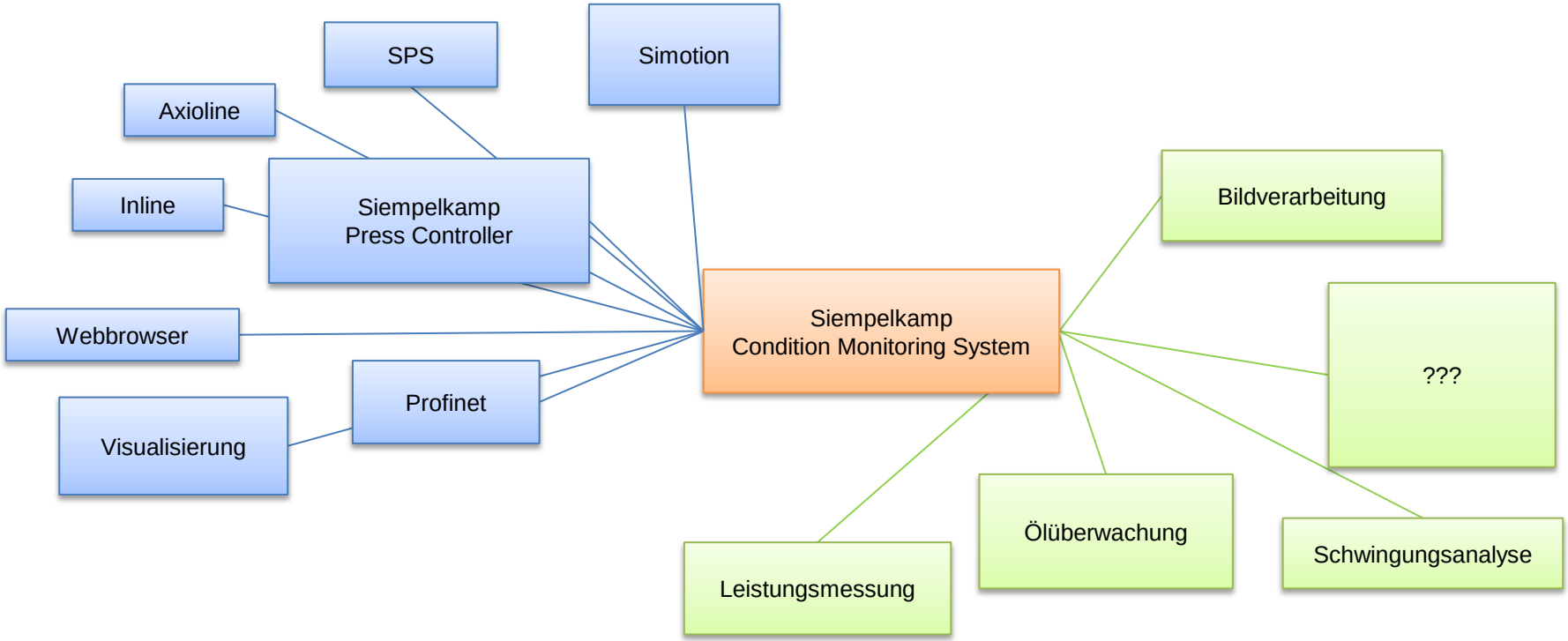
Information

Homogenisierung der Daten

Quellen: GfM Gesellschaft für Maschinendiagnose

Das Systemumfeld des CMS ist durch bestehende Anlagenautomatisierung, Überwachungs- und Bedienungsanforderungen vorgegeben

Siempelkamp
Condition Monitoring System



Auszug aus den Anforderungen und Bewertungskriterien

Siempelkamp
Condition Monitoring System

Rahmenbedingungen

- **Systemumfeld der bestehenden Automatisierungstechnik**
- **Bereits ausgewählte Subsysteme**
- *Eigene Hardware/proprietäres System*

Quality of Life

- Konfigurieren statt Programmieren
- Anwendungsentwicklung entkoppelt von Basis-System (**geringe Einstiegshürde in Entwicklung**)
- Deployment von Updates
- *Nutzung von Techstack und Workflows des Unternehmens*

Funktionen

- Überwachung großer Maschinen und kleinerer „Nebenaggregate“
- **Integration neuer Subsysteme**
- **Eigene Algorithmen/KPIs zur Zustandsbewertung**
- Stand-Alone und offline lauffähig
- Bereitstellung von modernen Schnittstellen (Web-API) für Integration auf Kundenseite
- Anwenderinteraktion über Webbrowser
- Gleichartige Darstellung der Informationen, unabhängig vom Ursprung

Wahl der Hardwarebasis, Betriebssystem und Programmiersprache

FINDEN EINER LÖSUNG

Für Auswahl der Systemplattform und Sprache maßgeblich treibende Anforderungen

Siempelkamp
Condition Monitoring System

- Proprietäres System gewünscht
- Geringe Einstiegshürde in spezifische Anwendungsentwicklung
- Kommunikation mit vorhandener Automatisierungstechnik und neuen Systemen
- Minimierung der Anzahl neuer Komponenten im Techstack

Bestehendes System „Siempelkamp Press Controller“ (SPC)

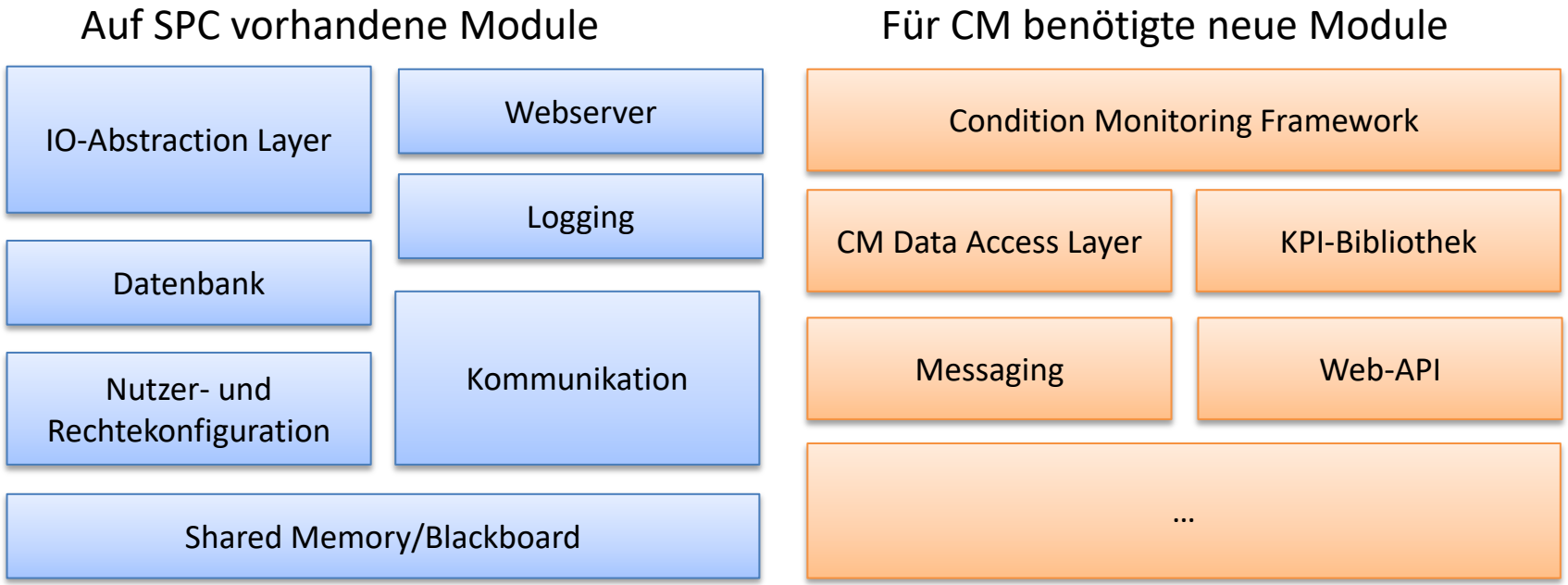
Siempelkamp
Condition Monitoring System

Auf SPC vorhandene Module

- IO-Abstraction Layer
- Webserver
- Logging
- Datenbank
- Kommunikation
- Nutzer- und Rechtekonfiguration
- Shared Memory/Blackboard

Zusätzlich benötigte Module

Siempelkamp
Condition Monitoring System



Die Sprache LUA erfüllt alle kritischen Voraussetzungen

Siempelkamp
Condition Monitoring System

- Prototypen-/Objektorientierte dynamische Skriptsprache
- MIT-Lizenz
- **Interpreter implementiert in ANSI-C**
- Incremental Garbage Collector
- Freie Bibliotheken (json, Template-Engine, Sockets, sendmail)
- Wrapper für bestehende Systemfunktionen simpel zu implementieren

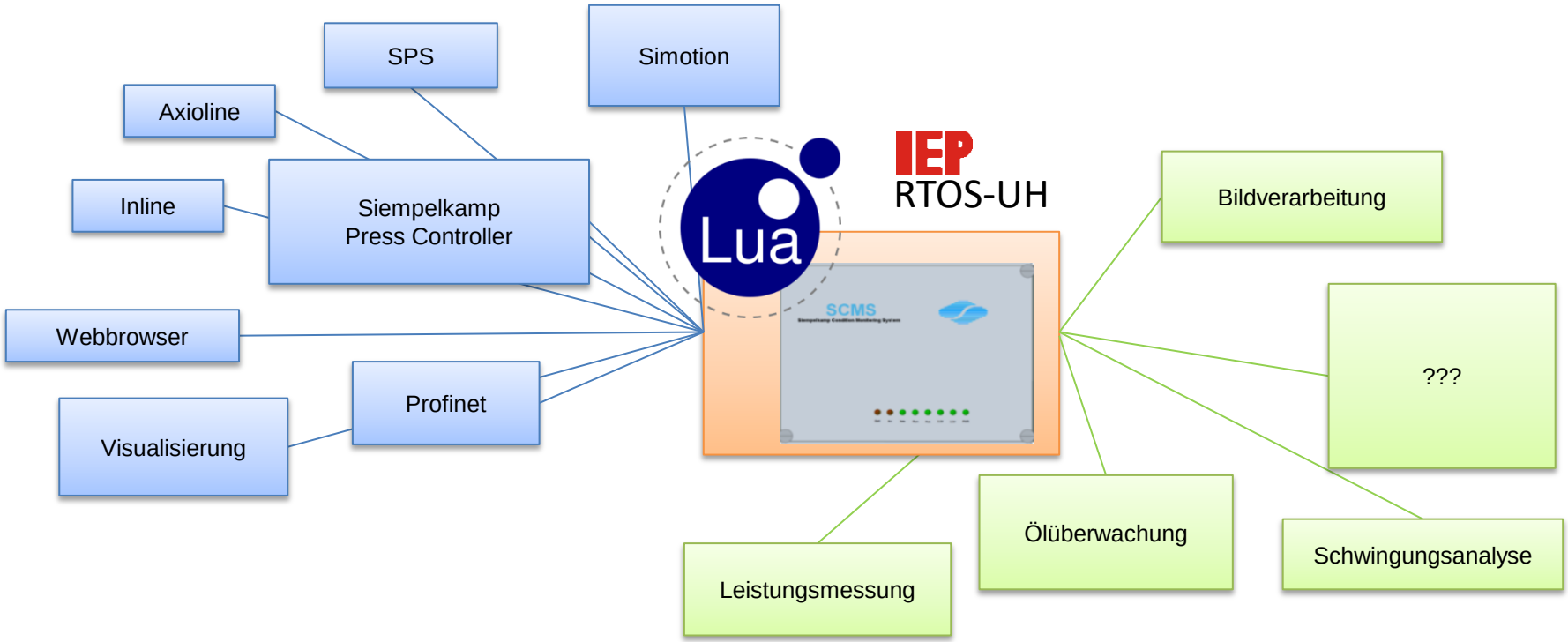
```
local bucketDataEngine = require "dataaccess.timingbucketdataengine"
self.BucketData = bucketDataEngine:getData()[tostring(bucketID)]

self.KPIs = {}
self:setupKPIs()

self.Options = {}
self.Options.IgnoreOutdated = false
```

Systemplattform und Sprache

Siempelkamp
Condition Monitoring System



Quellen: IEP Ingenieurbüro für Echtzeitprogrammierung

LUA AUF RTOS-UH

Lua: Codeerzeugung zur Laufzeit

Siempelkamp
Condition Monitoring System

```
local expressionString = "math.sin (values.Diff) * 0.5"

local func, errormsg = load(
        "return " .. self.ExpressionString,
        self.Name .. ":expressionFunction",
        "t",
        {values = valuesToPass, math = math, os = os}
    )

if (func == nil) then
    error(errormsg)
else
    self.expressionFunction = func
end
```

Code-Chunk zur Laufzeit aus string

Durchreichen der Umgebung

Überladen der Methode

Lua + RTOS-UH: dynamische Taskerzeugung

Siempelkamp
Condition Monitoring System

```
local codeTemplate =
        [[
            local scheduler = require "cmsframework.decoupledkpischeduler"
            pcall (scheduler.start, scheduler, "{{=KPIName}}")
        ]]
```

Code zur Laufzeit

```
local codeCompiledTemplate = liluat.compile(codeTemplate)
self.CodeTask = liluat.render(codeCompiledTemplate, self)
```

Template Engine

```
self.Task = rtos.rt_create_task(self.CodeTask)
```

Task aus Code

Lua + RTOS-UH: Timing einer Task „von innen“

Siempelkamp
Condition Monitoring System

```
rtos.rt_timed_activate_quick(rtos.rt_my_TID(), 0, 0x80000000,
                             self.KPI.DecoupledCycleTime, 0x7FFFFFFF)

rtos.rt_set_prio(13)
```

Einplanen und Prio setzen

```
while not self.KPI.IsFinished and self.KillPtr:read() == 0 do
        rtos.rt_wait_for_activation()
         -- KPI durchlaufen
end
```

Warten auf Aktivierung

```
-- Selber ausplanen
rtos.rt_prevent_task_quick(rtos.rt_my_TID())
```

Ausplanen

FAZIT

Fazit zur Kombination Lua + RTOS-UH auf SPC

Siempelkamp
Condition Monitoring System

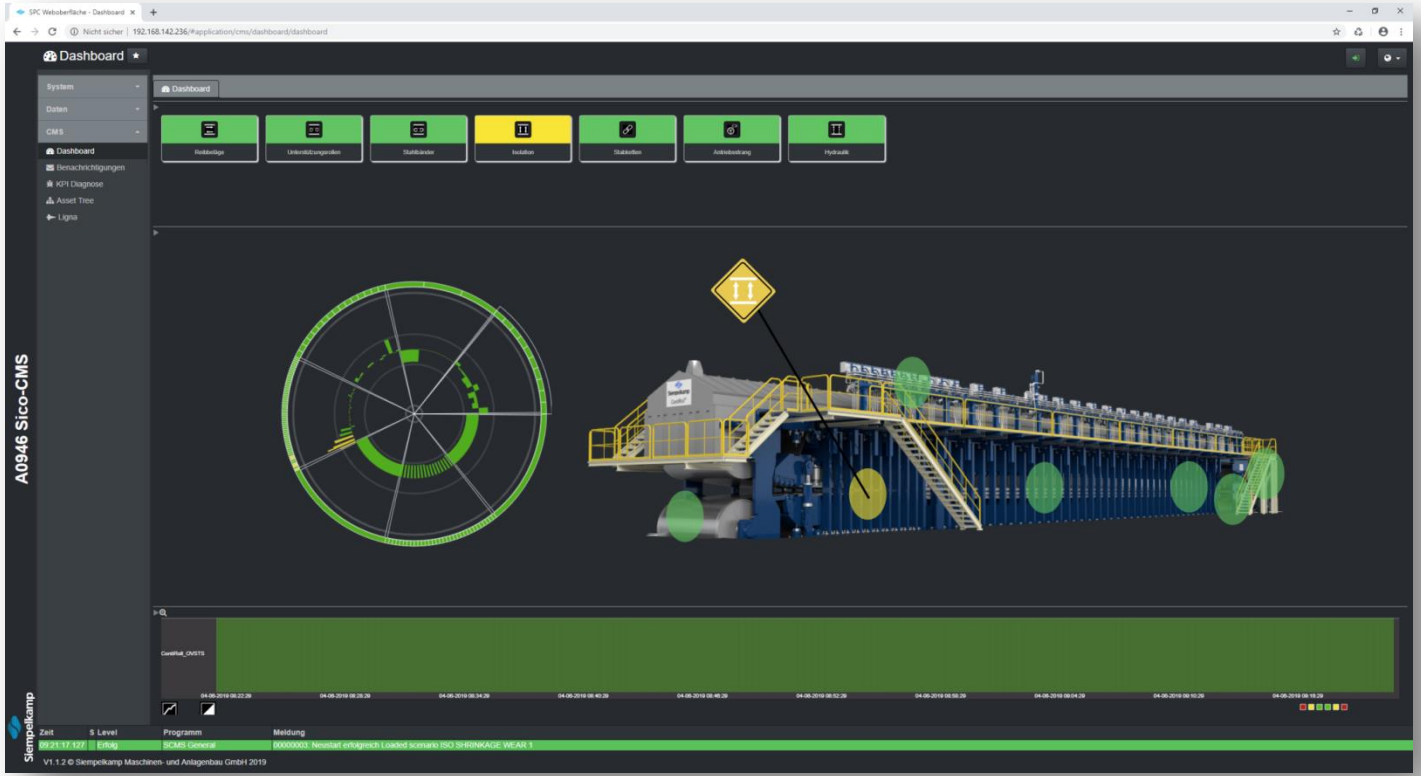
Bewertung nach ~1,5 Jahren Entwicklung in Lua auf RTOS-UH:

- Nur geringe Anpassungen am Interpreter-Code bei Portierung
- Performance-Optimierung für Speicher notwendig, aber dank Community-Arbeit ohne eigene Implementierung gelungen
- Untersuchung des Umsetzbarkeit von Multi-Threading Lua-Interpreter geplant

- Einfacher Einstieg für Anwendungsentwickler
- Bestehende Pearl- und C-Systemfunktionen integriert
- Bisher alle Module unkompliziert umsetzbar
- Automatisierte Tests wegen Plattform/Umgebung nicht in CI möglich

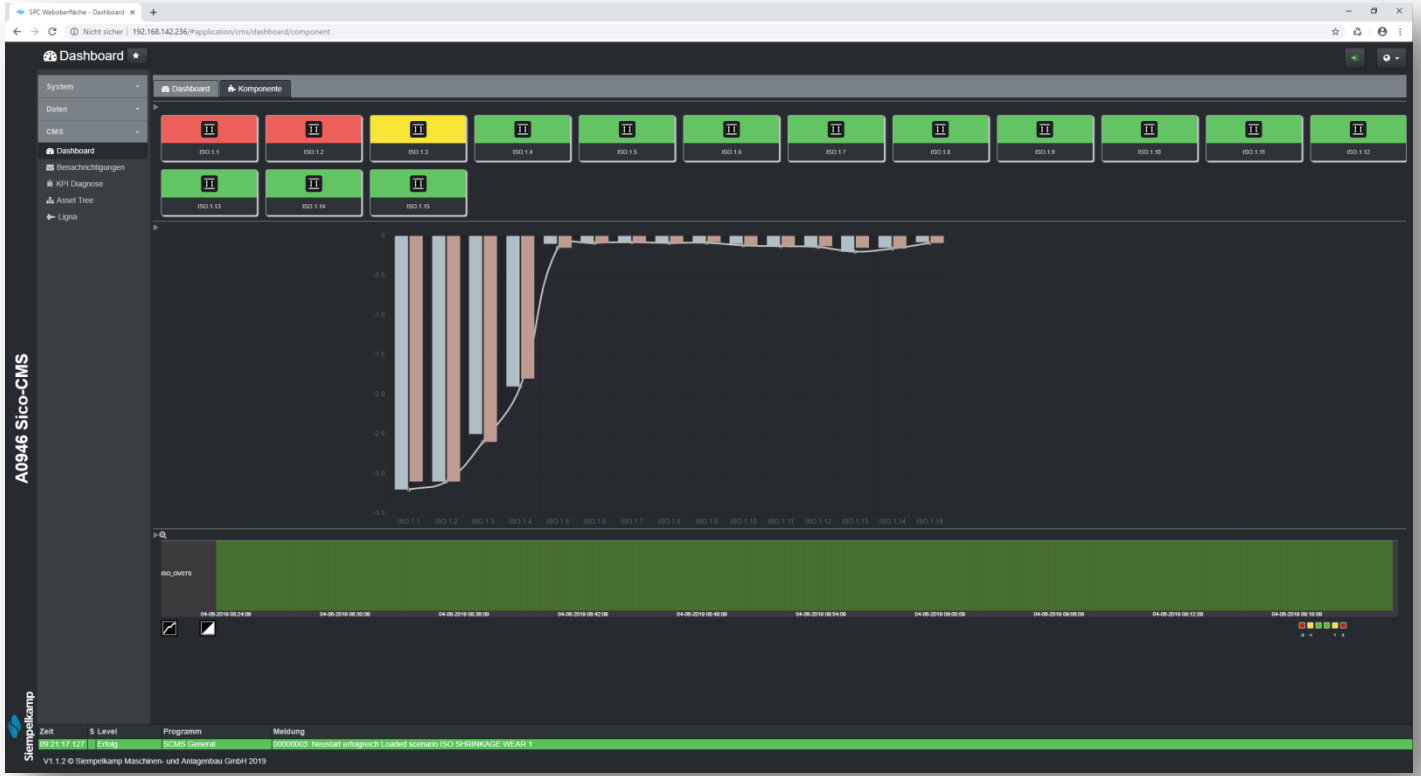
Ergebnis für Anwender: Gesamtübersicht über die Maschine

Siempelkamp
Condition Monitoring System



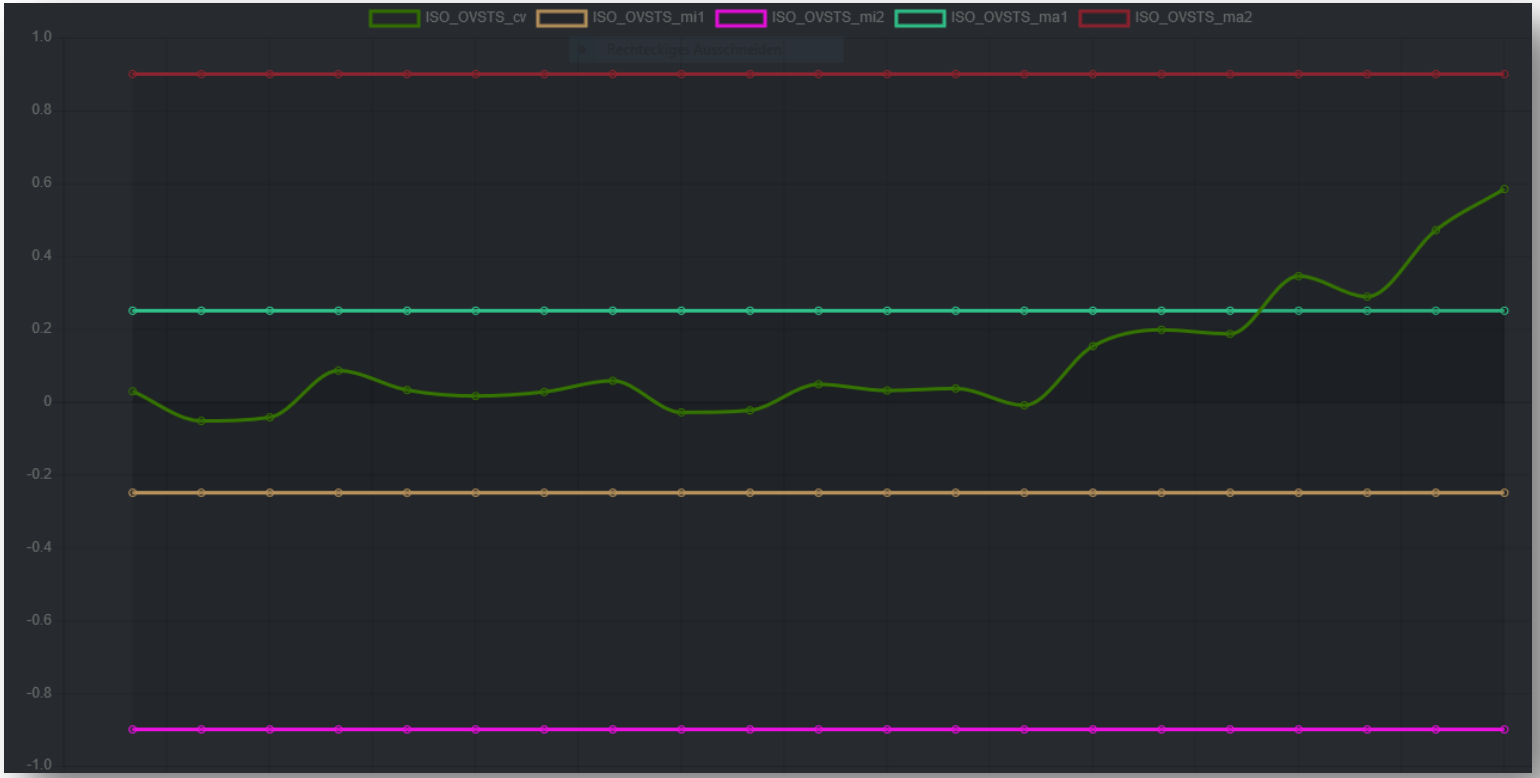
Ergebnis für Anwender:
Verschleiß von Isolationskassetten am Presseneinlauf

Siempelkamp
Condition Monitoring System



Ergebnis für Anwender:
Zeitlicher Verlauf eines Schädigungszustandes

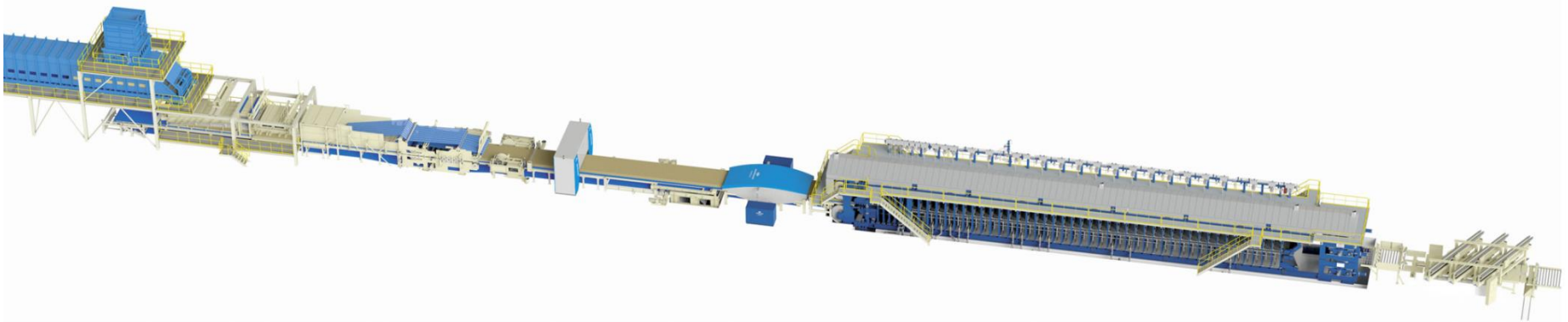
Siempelkamp
Condition Monitoring System



AUSBLICK

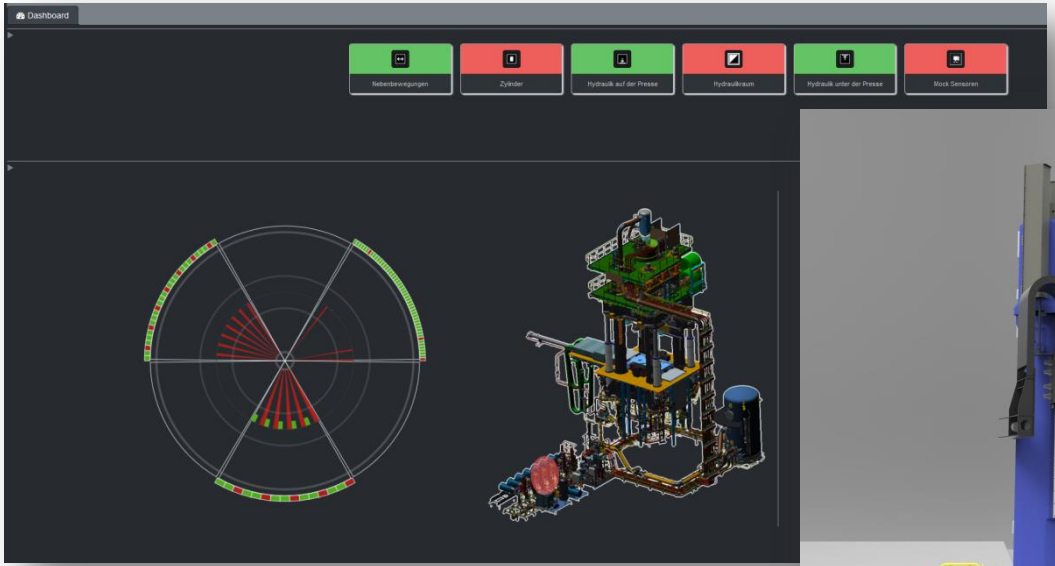
Ausblick: weitere Komponenten und Maschinen im Holzbereich

Siempelkamp
Condition Monitoring System



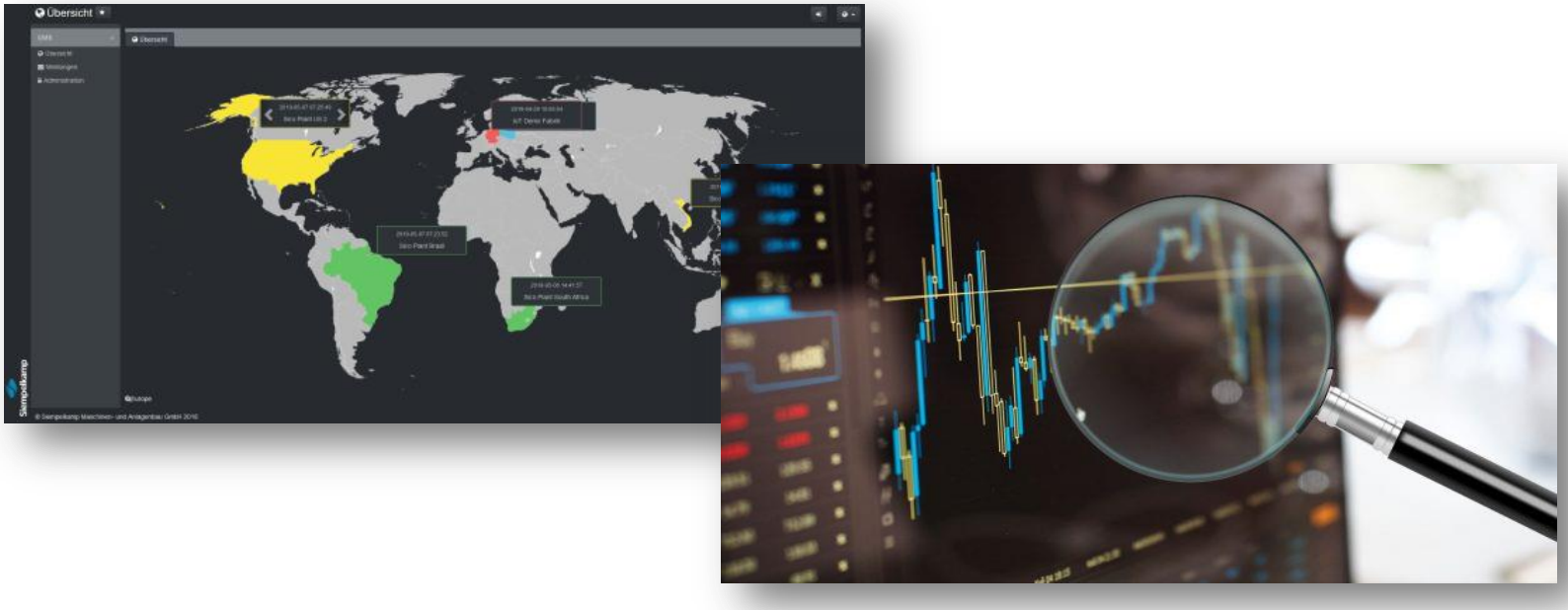
Ausblick: Nutzung in Metallumformanlagen

Siempelkamp
Condition Monitoring System



Ausblick: IoT und Machine Learning

Siempelkamp
Condition Monitoring System




Vielen Dank für Ihre Aufmerksamkeit !
Siempelkamp
Leadership in Technology