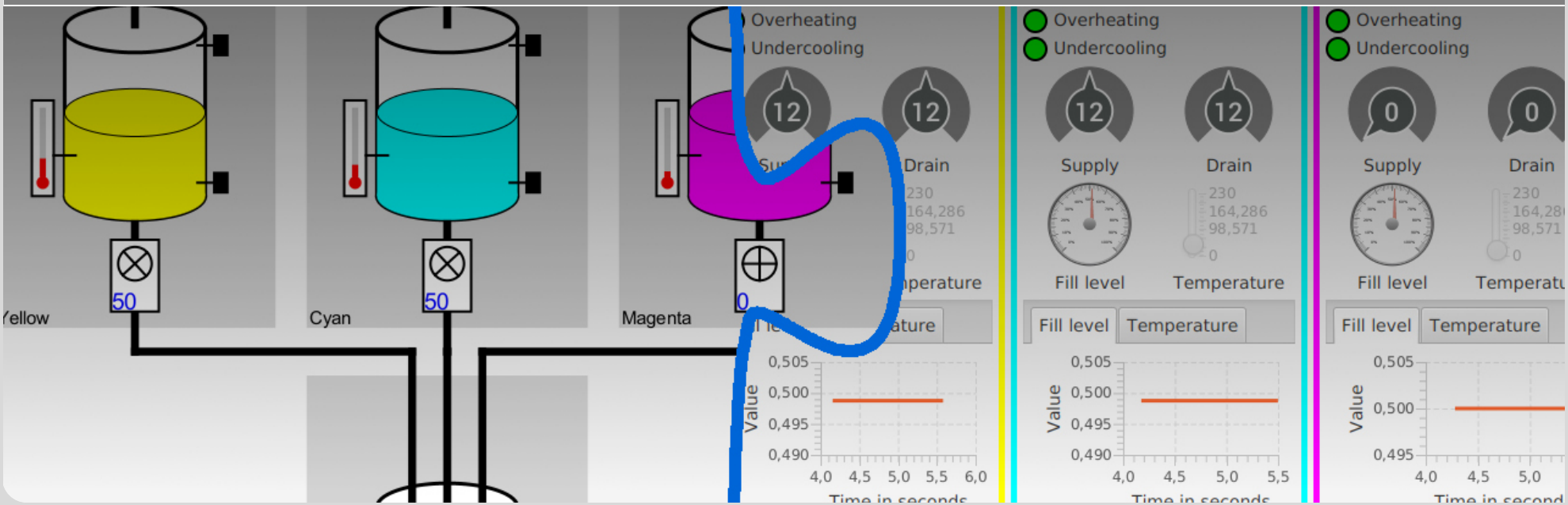
Karlsruher Institut für Technologie

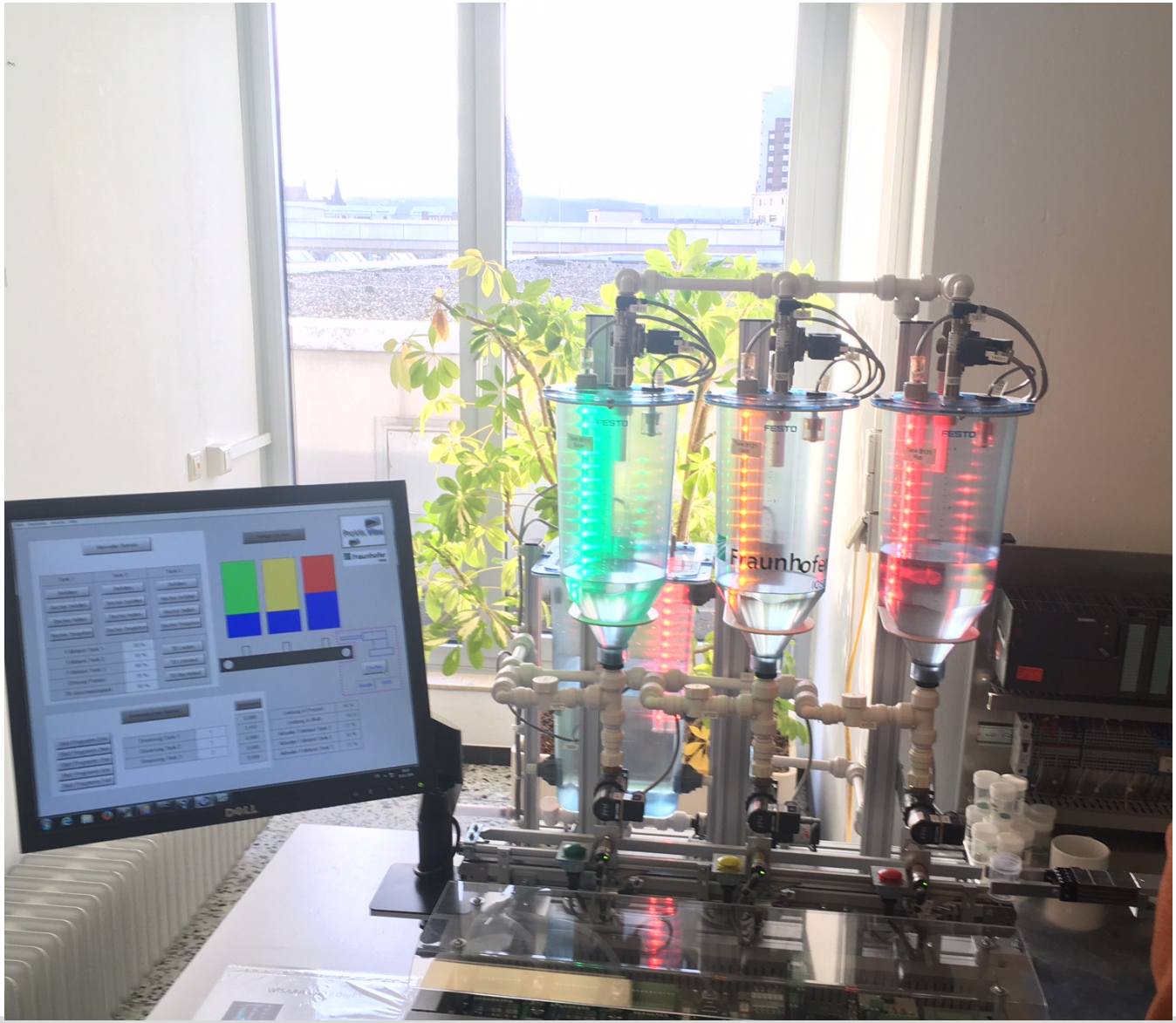
OPC UA Simulator for Industrial Plants

PSE Projekt

M. Armbruster, D. Kahles, H. Lehmann, M. Schwarzmann, N. Wilhelm | 13. März 2017



Wofür benötigt man OSIP?



Was ist OSIP?

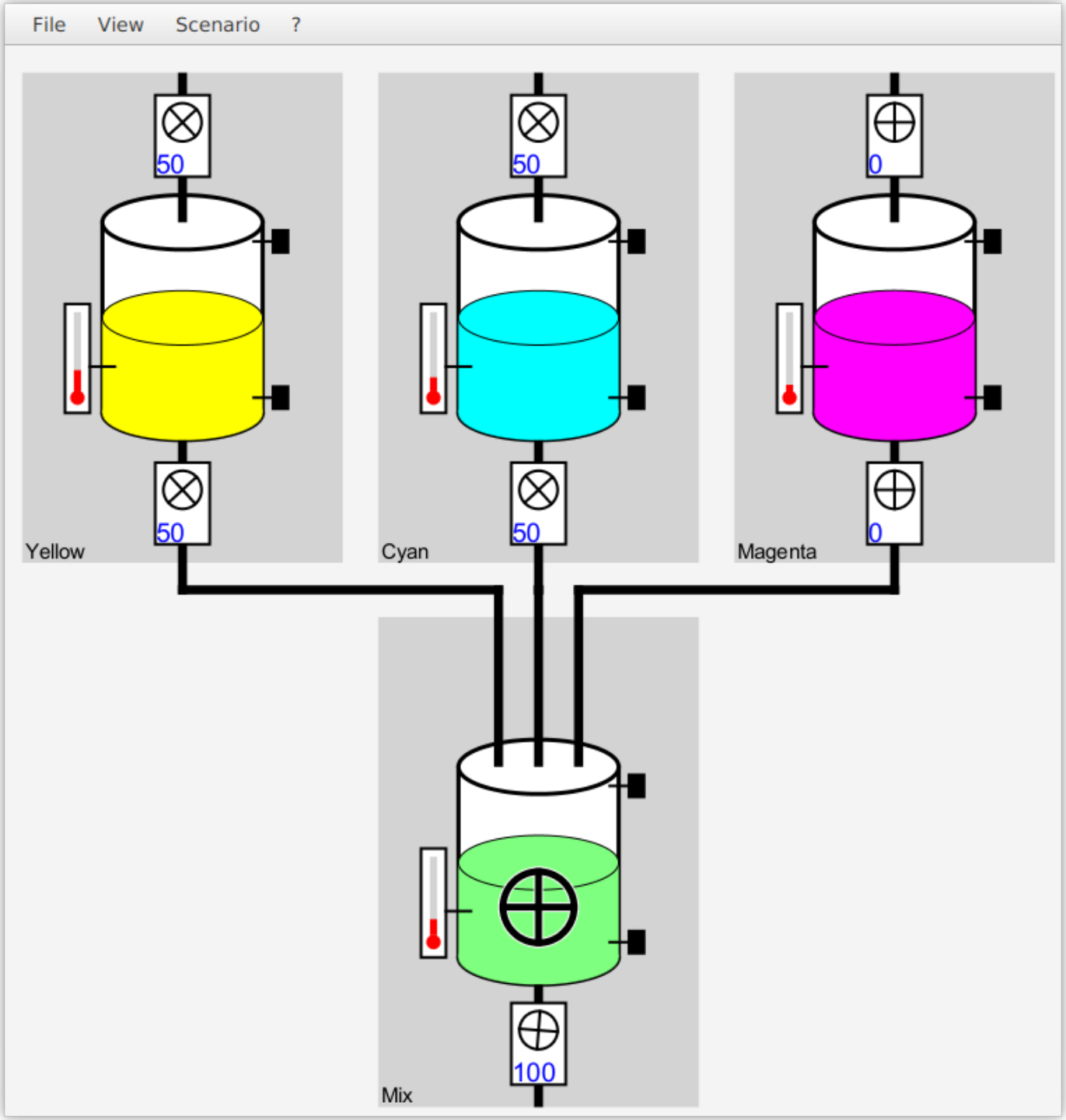
- Zwei Anwendungen: *Simulation* eines chemischen Produktionsprozesses mit vier Tanks und *Überwachungskonsole*
- Kommunikation ausschließlich über OPC UA Protokoll
- Anwendungen per Netzwerk auf getrennten Computern lauffähig

Was ist OSIP?

- Zwei Anwendungen: *Simulation* eines chemischen Produktionsprozesses mit vier Tanks und *Überwachungskonsole*
- Kommunikation ausschließlich über OPC UA Protokoll
- Anwendungen per Netzwerk auf getrennten Computern lauffähig

Was ist OSIP?

- Zwei Anwendungen: *Simulation* eines chemischen Produktionsprozesses mit vier Tanks und *Überwachungskonsole*
- Kommunikation ausschließlich über OPC UA Protokoll
- Anwendungen per Netzwerk auf getrennten Computern lauffähig


File
View
Scenario
?
50
50
Yellow
50
50
Cyan
0
0
Magenta
100
Mix

File ?

Yellow tank
Overflow
Underflow
Overheating
Undercooling
12
Supply
12
Drain
Fill level
230
164,286
98,571
0
Temperature
Fill level
Temperature
0,505
0,500
0,495
0,490
Value
4,0 4,5 5,0 5,5 6,0
Time in seconds

Cyan tank
Overflow
Underflow
Overheating
Undercooling
12
Supply
12
Drain
Fill level
230
164,286
98,571
0
Temperature
Fill level
Temperature
0,505
0,500
0,495
0,490
Value
4,0 4,5 5,0 5,5
Time in seconds

Magenta tank
Overflow
Underflow
Overheating
Undercooling
0
Supply
0
Drain
Fill level
230
164,286
98,571
0
Temperature
Fill level
Temperature
0,505
0,500
0,495
Value
4,0 4,5 5,0 5,5
Time in seconds

Mix tank
Overflow
Underflow
Overheating
Undercooling
Color
0
Motor speed
25
Drain
Fill level
230
164,286
98,571
0
Temperature
Fill level
Temperature
0,505
0,500
0,495
0,490
Value
0 25 50 75
Time in seconds

Alarms
Console

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Verwendete Technologien

- OPC UA Protokoll nicht selbst implementiert
  $\rightarrow$ Open Source Implementierung *Milo*
- Programmiersprache: *Java*
- Bauen des Projekts: *Maven*
- Entwicklung der GUI: *JavaFX*
- *Git* mit Codereview und CI auf *GitHub.com*
- Plattformen: Ubuntu 16.04 und Windows 10
- Unter Ubuntu: einfaches Starten in Docker-Containern

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

Was haben wir bis jetzt?

- Circa 800 Commits
- Verschiedene Entwurfsmuster
  - Model-View-Controller
  - Fassade
  - Strategie
  - ...
- Circa 15.000 LOC, davon 7.500 SLOC
- Fast alle geplanten Features umgesetzt

It’s Showtime

Karlsruher Institut für Technologie

Demo

Resümee

- Controller (und generell Implementierungsphase) aufwändiger als gedacht
- Entwurf bei den Schnittstellen zwischen Controller und View unvollständig
- Code Review und CI vermeiden viele Probleme
- Unittests für Nicht-GUI-Komponenten vermeidet Probleme
- GitHub-Issuetracker sehr nützlich, um Übersicht zu behalten

Resümee

- Controller (und generell Implementierungsphase) aufwändiger als gedacht
- Entwurf bei den Schnittstellen zwischen Controller und View unvollständig
- Code Review und CI vermeiden viele Probleme
- Unittests für Nicht-GUI-Komponenten vermeidet Probleme
- GitHub-Issuetracker sehr nützlich, um Übersicht zu behalten

Resümee

- Controller (und generell Implementierungsphase) aufwändiger als gedacht
- Entwurf bei den Schnittstellen zwischen Controller und View unvollständig
- Code Review und CI vermeiden viele Probleme
- Unittests für Nicht-GUI-Komponenten vermeidet Probleme
- GitHub-Issuetracker sehr nützlich, um Übersicht zu behalten

Resümee

- Controller (und generell Implementierungsphase) aufwändiger als gedacht
- Entwurf bei den Schnittstellen zwischen Controller und View unvollständig
- Code Review und CI vermeiden viele Probleme
- Unittests für Nicht-GUI-Komponenten vermeidet Probleme
- GitHub-Issuetracker sehr nützlich, um Übersicht zu behalten

Resümee

- Controller (und generell Implementierungsphase) aufwändiger als gedacht
- Entwurf bei den Schnittstellen zwischen Controller und View unvollständig
- Code Review und CI vermeiden viele Probleme
- Unittests für Nicht-GUI-Komponenten vermeidet Probleme
- GitHub-Issuetracker sehr nützlich, um Übersicht zu behalten