Generische Software

SCHWETER, Clemens

25. Oktober 2023

ALSTOM
• mobility by nature •

Vorstellung

Clemens Schweter @ Alstom

- 2004 Informatik Diplom an der TUB
- 18 Jahre als Freelancer für diverse Unternehmen tätig, u.a. für Alstom
  - Siemens, Thales, Bosch, Wacker, HDI, …
- Seit 2 Jahren festangestellt für Alstom
- 2010 Plattform „DaMaTo“ entwickelt
- 2011 Plattform „DAISY“ entwickelt

☺

Wir sind dort, wo Mobilität gefragt ist

Mehr als
80.000
Mitarbeiter:innen
weltweit

70
Länder

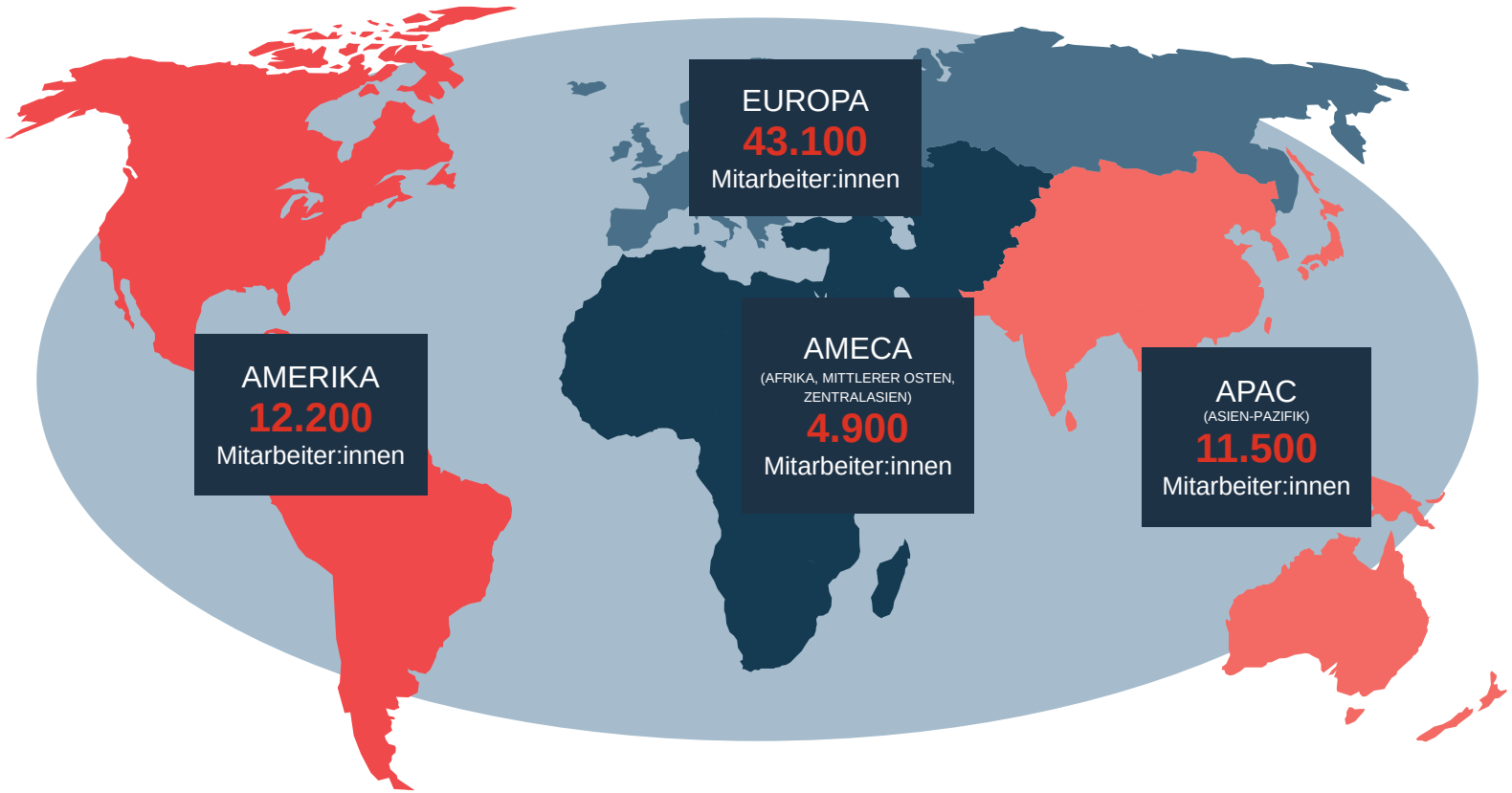
Über
250 Standorte

17.500
Ingenieur:innen

Über
9.500
Patente

Partner von
mehr als
300 Städten

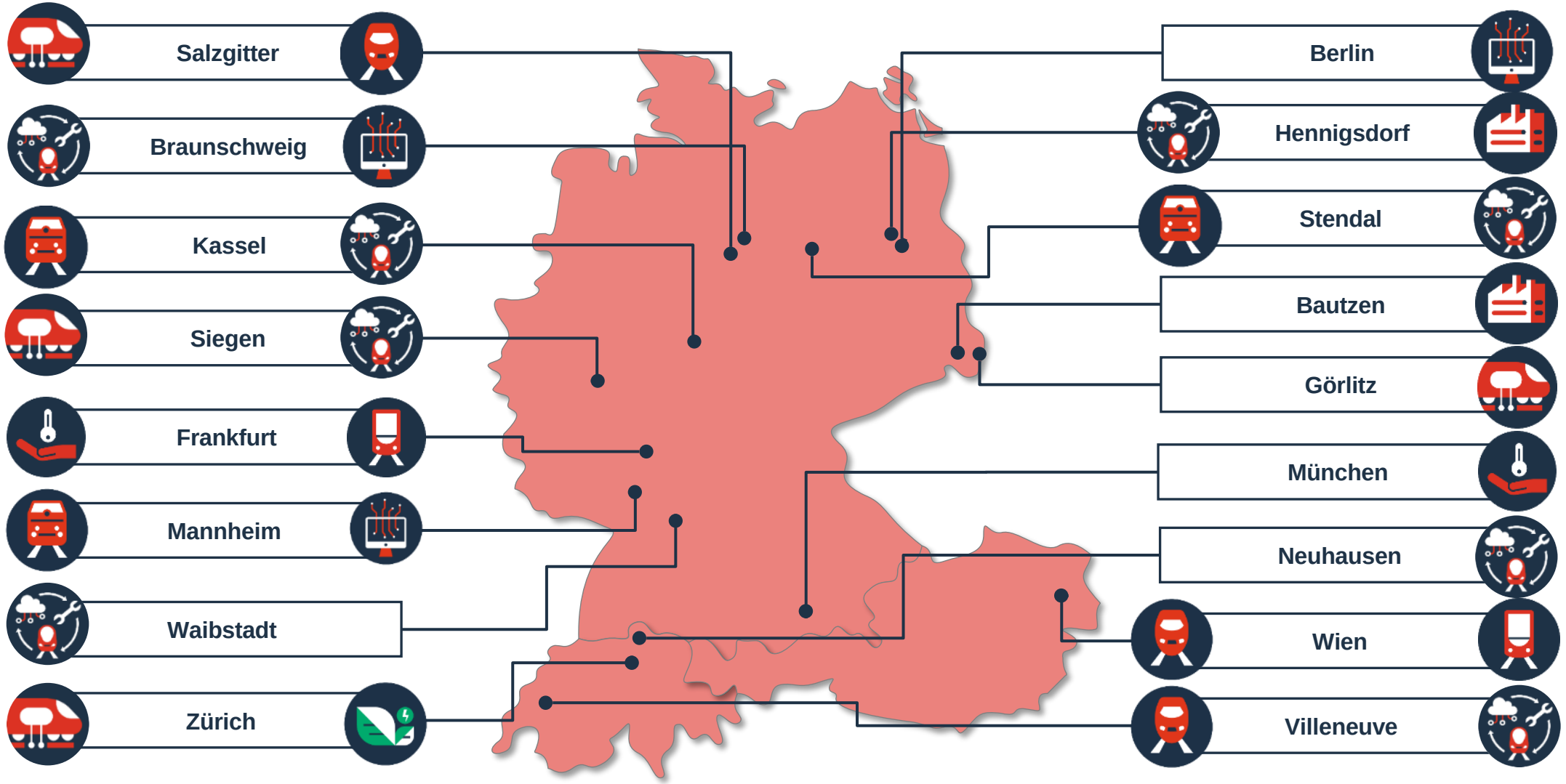
Mehr als **150.000** Fahrzeuge
im Einsatz



Ein komplettes Produktportfolio zum Vorteil unserer Kunden

Schienenfahrzeuge & Komponenten	Digitale & Integrierte Systeme	Serviceleistungen
		
Peoplemover und Monorail	Signaltechnik für den Nahverkehr	Fahrzeuginstandhaltung
Straßen- und Stadtbahnen	Signaltechnik für den Fernverkehr	Asset Life Management
U-Bahnen	Schlüsselfertige Lösungen	Ersatzteile & Komponentenüberholung
Nahverkehrszüge	Smart Mobility	Digitale Lösungen
Regional- und Intercity-Züge	Cybersicherheit	Betrieb und Systemwartung
Hochgeschwindigkeitszüge	Serviceleistungen Signaltechnik und Infrastruktur	
Lokomotiven		
Komponenten	Infrastruktur und Telekommunikation	

Standorte DACH-Region



Welcome to Hennigsdorf

46 nationalities

113 years experience

1 km away from Berlin

More than 2,100 employees

650,000 m² total area

34% workload for export outside Germany

More than 800 engineers

43 % supply chain outside Germany

JOB

60 open positions
We are hiring!

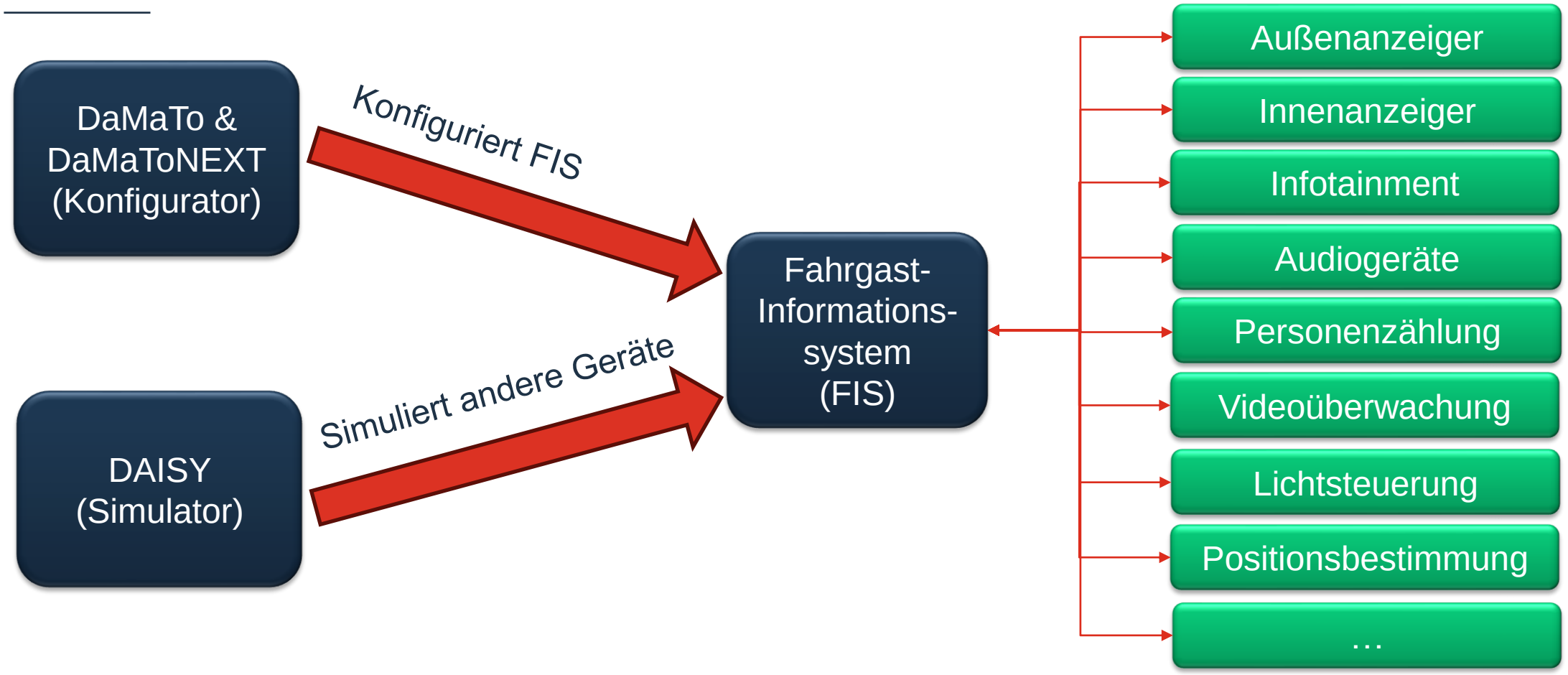
Hennigsdorf: Entwicklung, Prototypen-Fertigung, Testing und Service

- Umfangreiche Kompetenzen in der Entwicklung von Vollbahnen
- Produktion von u.a. Prototypen, Vor- und Kleinserien
- Service Center

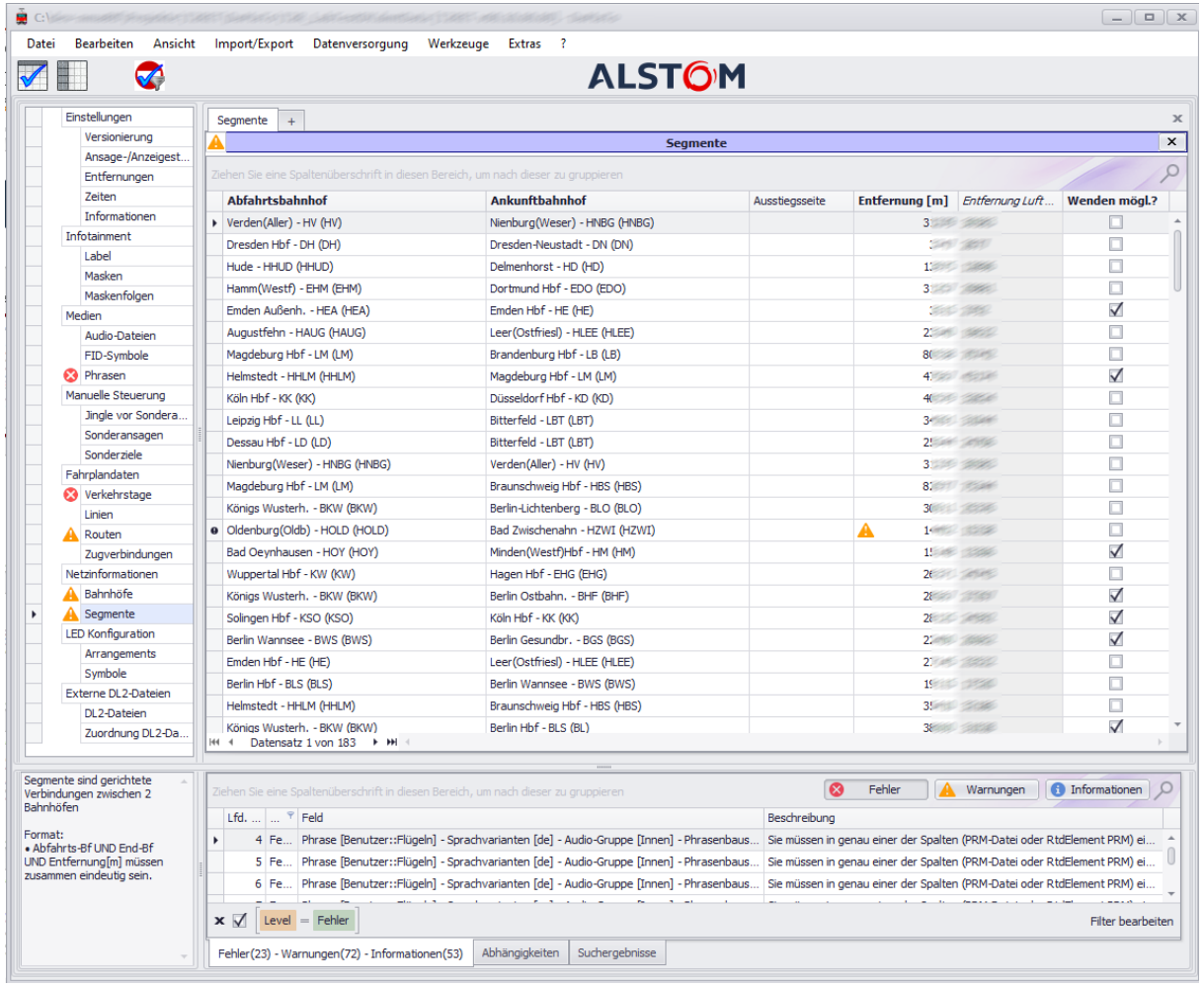
- Ca. 2.100 Mitarbeiter:innen
- 620.000 m^2 Gesamtfläche

- 3 interne Teststrecken (70 km/h)
- 1 externe Teststrecke (140 km/h)
- Typ-, System- und Zulassungstests

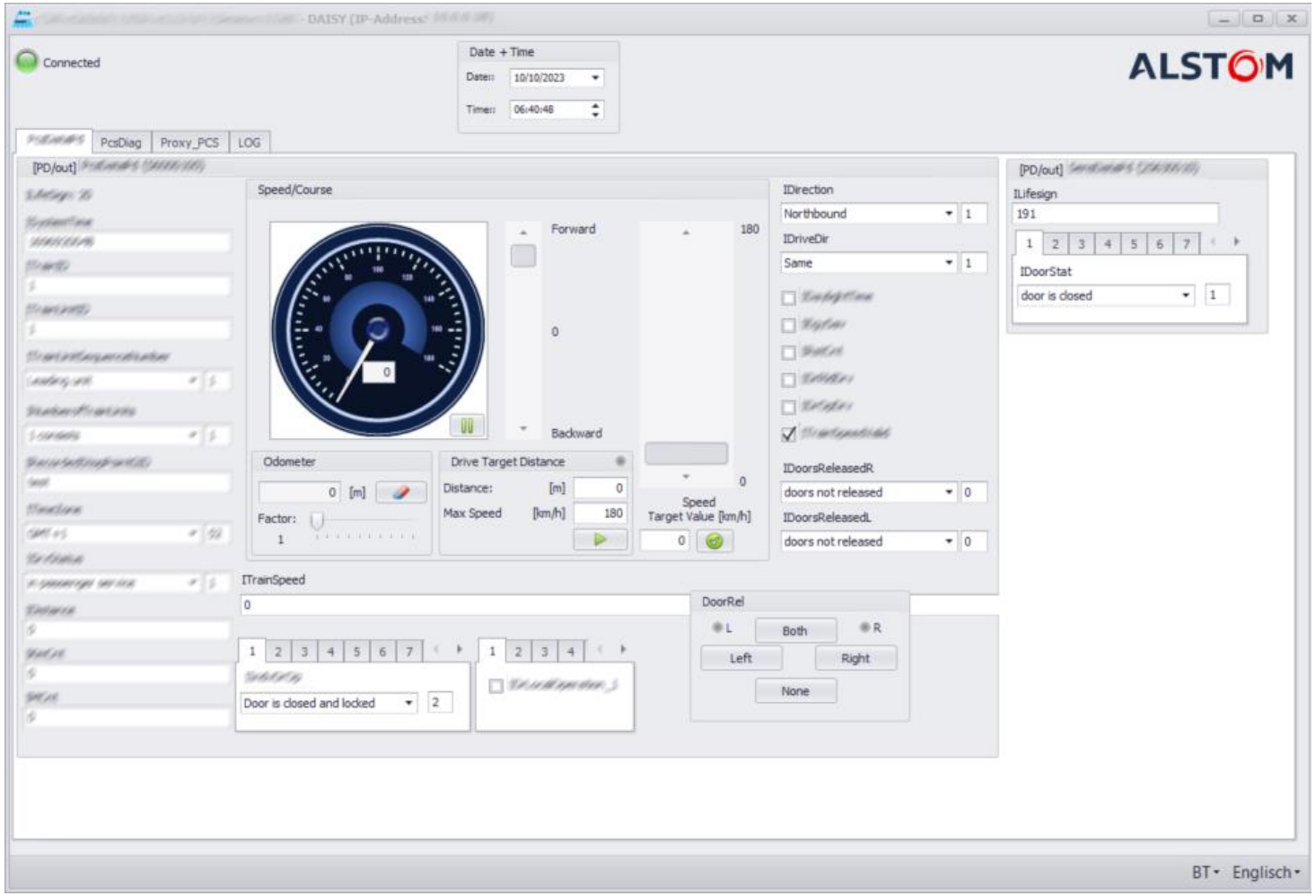
Funktionen in einem Zug



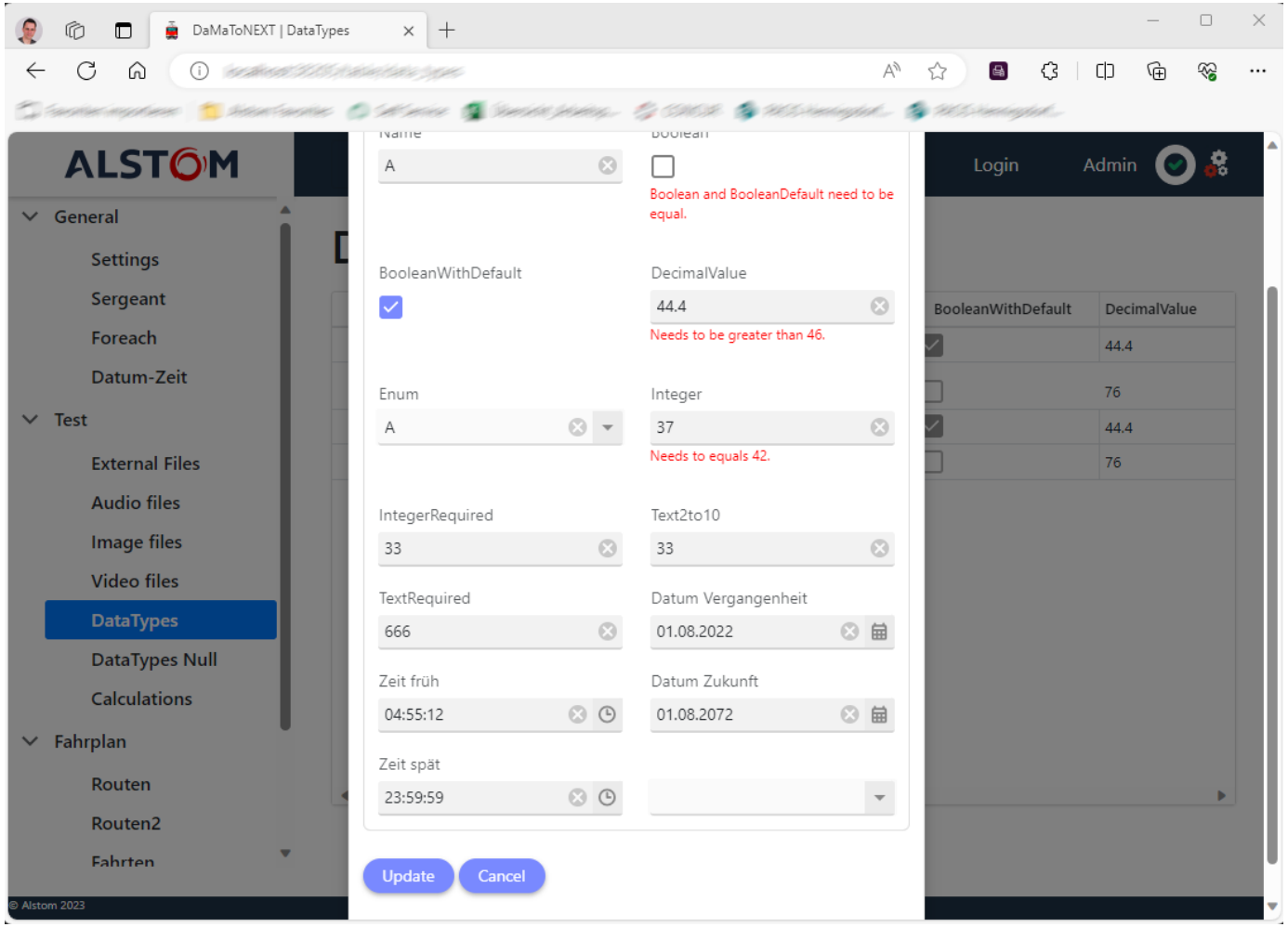
DaMaTo – Strecke zwischen zwei Bahnhöfen konfigurieren



DAISY - Beispielprojekt



DaMaToNEXT - Datenbearbeitung



Problem

- Viele Kundenprojekte gleichzeitig.
- Alle Projekte irgendwie gleich, aber doch nicht.
- Projekte laufen sehr lang, sind komplex und Anforderungen ändern sich.
- Projekte haben lange Wartungszeit (spätere Erweiterungen, Änderungswünsche, …).
- Zeit für Benutzerhandbücher und Dokumentation oft nicht mehr vorhanden.
- Anforderungen und Quellcode konsistent halten, erfordert manuelle Disziplin.

Ein generisches Konzept muss her!

Ziele und Wünsche

- Geringer Entwicklungsaufwand!
- Geringe Fehlerquote!
- Anforderungen ohne manuellen Aufwand konsistent zu Programm halten.
- Benutzerhandbücher und Dokumentation entsteht von allein.
- Auch alte Projekte können jederzeit erweitert oder geändert werden.
- Probleme nur einmal lösen, nicht „das Rad jedes Mal neu erfinden“.

Ein generisches Konzept muss unbedingt her!

Lösungsansätze

Jedes Projekt „auf der grünen Wiese“ starten

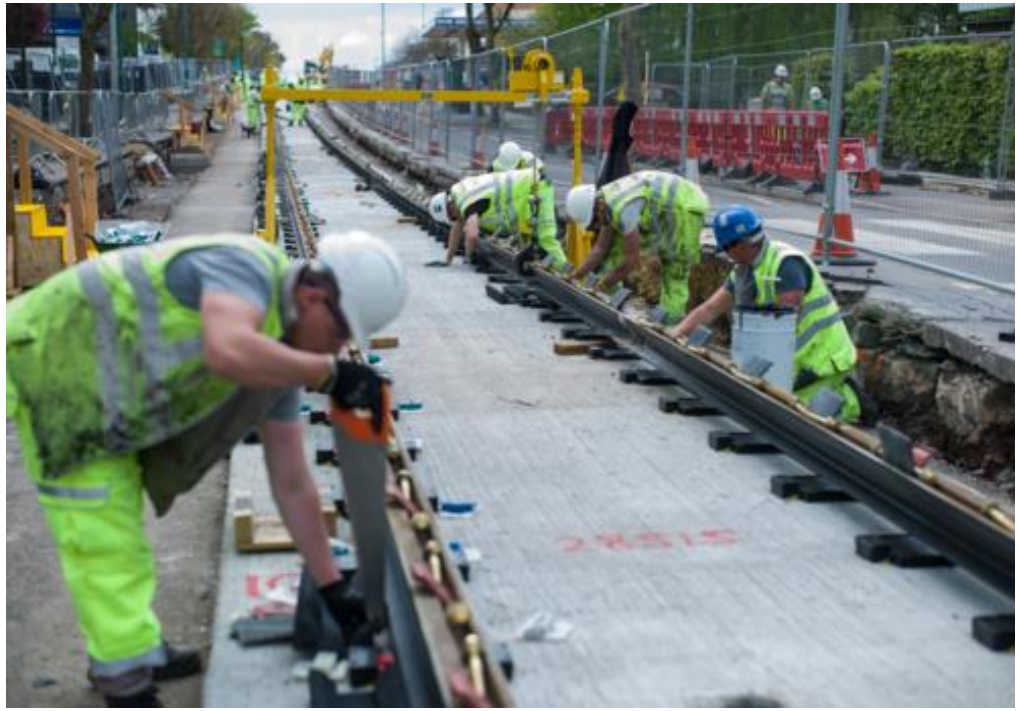
- Mit jedem Projekt wird komplett vor vorne begonnen.
- Man hat keine Altlasten, muss aber jedes Problem erneut lösen.
- Keine Wiederverwendbarkeit, keine Synergien, …
- Unter Umständen werden sogar unterschiedliche Teams für unterschiedliche Projekte beauftragt.

Keine Lösung

Lösungsansätze

#ifdef

- Eine Codebasis, aber je nach Compiler-einstellungen ergeben sich unterschiedliche Ergebnisse.
- Quellcode ist schwer wartbar.
- Änderungen an einem Projekt haben u. U. zur Folge, dass sich alte Projekte nicht einmal mehr übersetzen lassen.
- Jedes Projekt muss komplett von vorn bis hinten durchgetestet werden, da die Auswirkungen nicht abschätzbar sind.



Diese Projekte gibt es wirklich!

Lösungsansätze

Unterschiedliche Entwicklungszweige

- Die generelle Software wird auf einer Entwicklungslinie gestartet.
- Werden Änderungen vorgenommen, so wird der entsprechende Code Teil in einem separaten Branch weiterentwickelt.
- Problem: Branche laufen mit der Zeit auseinander.

Irgendwann hat man etliche Projekte, die nichts mehr gemeinsam haben!

Lösungsansätze

Objektorientierte Architektur

- Funktionen werden in Basisklassen abstrahiert und pro Projekt angepasst.
- Teil des Quellcodes ist in allen Projekten gleich und nur die Anpassungen werden pro Projekt implementiert.
- Prinzipiell ein guter Ansatz, ABER
  - Die Abstraktion muss ständig angepasst werden, was doch wieder Änderungen in anderen Projekten zur Folge hat.
  - Segen für die Entwickler und Fluch für die Tester.

Ansatz ist schon besser, aber hat noch viele Nachteile und ist kompliziert!

Lösungsansätze

Generische Software über Konfiguration

- Software bietet nur Funktionsblöcke an, die mit einer Konfigurationsdatei parametriert werden.
- Die Funktionsblöcke selber können projektunabhängig getestet werden (bspw. mit (automatischen) Unit Tests.

Dieser Ansatz wird genauer betrachtet!

Funktionsweise

- Parameter der Funktionsblöcke werden in einer Konfigurationsdatei abgelegt.
- Softwarearchitektur:
  - Aus der Konfigurationsdatei wird Quellcode generiert, der dann kompiliert wird.
  - Für jeden Funktionsblock gibt es Quellcode, der dann ausgeführt wird.
  - Jeder Funktionsblock enthält die Funktion zum Generieren der Textbauteile für die Dokumentation.
- Automatische Versionsupdates
  - Bspw. XML-Konfiguration mit XSLT auf neue Struktur konvertieren.
- Benutzerhandbuch setzt sich zusammen aus projektspezifischen Textbausteinen und den generierten Textbausteinen aus den Funktionsblöcken.

XML als Beschreibungssprache für die Anwendung!

Bewertung

Generische Software über Konfiguration - Vorteile

- Funktionsblöcke sind unabhängig zueinander.
- Neue Funktionsblöcke können entwickelt und benutzt werden, ohne Auswirkungen auf andere Projekte.
- Funktionsblöcke als kleine Einheiten können ausgiebig getestet werden. Die parametrierte Anwendung ist dann sehr fehlerarm.

Wiederverwendbare konfigurierbare Funktionsblöcke!

Bewertung

Generische Software über Konfiguration - Nachteile

- Es wird sozusagen eine proprietäre Entwicklungssprache geschaffen.
- Keine Dokumentation und Hilfe im globalen Internet verfügbar.
- Interne Dokumentation oft nur spärlich vorhanden.
- Debug-Möglichkeiten müssen in der Plattform selber entwickelt werden (und können gar nicht so umfangreich sein, wie bei einer IDE "von der Stange")

Dieser Ansatz hat aber auch seinen Preis, aber es lohnt sich!

Grenzen von generischer Software

- Projekte müssen ähnlich sein.
- Unter Umständen nicht performant genug (im embedded Umfeld).
- Unter Umständen zu hoher Speicherverbrauch (im embedded Umfeld).
- Hohes Abstraktionsvermögen nötig.
- Diszipliniertes Arbeiten (Kosten, Zeit und Qualität).
- Gerade am Anfang ist Investitionsaufwand nötig.

Muss genau geprüft werden, ob dieser Ansatz für das System geeignet ist.

Qualitätssicherungsmaßnahmen

Unit Tests und Anwendungstest für den langfristigen Erfolg!

- Unit Tests der Funktionsblöcke (Test-driven development).
- 4 Augen-Prinzip!
- Auch die Testabdeckung ständig im Auge behalten (80% für nicht-sicherheitsrelevante Software)
- Manuelle „Chaos-Konfigurationen“ durch Test-Ingenieur.
- Bei der Entwicklung der Funktionsblöcke darf an Zeit nicht gespart werden.
- Stabiles Fundament = Stabile Software

www.alstom.com

ALSTOM

• mobility by nature •

Join #TeamAlstom

Get in Touch!

- Opportunities for Working Students
  - at our engineering site in Hennigsdorf
  - at our DACH headquarter at Ernst-Reuter-Platz (signalling & corporate functions)
- Internship
- Thesis
- VIE (French International Internship Program)
- Many exciting junior positions!