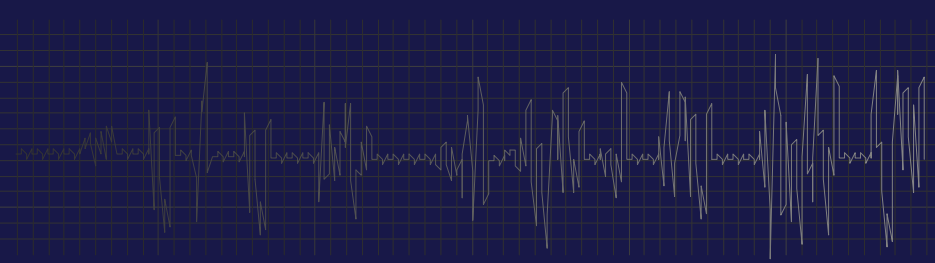


Deklarative Beschreibung verteilter signalverarbeitender Systeme

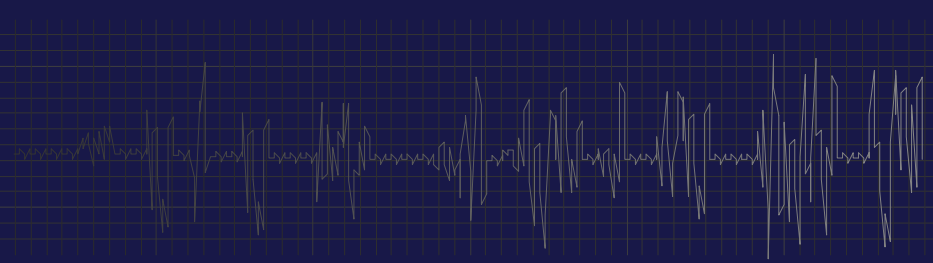
Dipl.-Inf. Axel Weiß (aweiss@informatik.hu-berlin.de)

Humboldt-Universität zu Berlin
Lehrstuhl für Signalverarbeitung und Mustererkennung



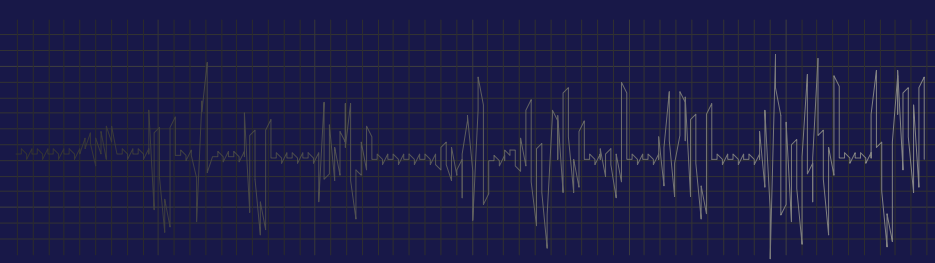
Gliederung

1. Motivation
2. Modellierung
3. Partitionierung
4. Toolchain
5. Zusammenfassung



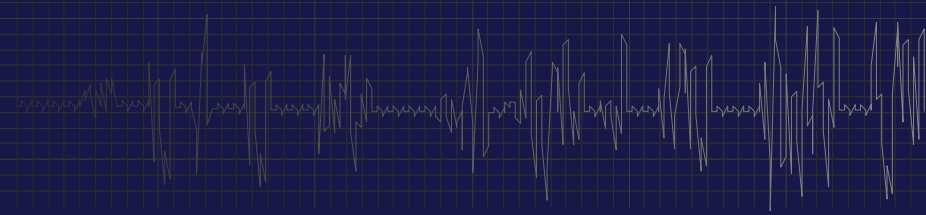
Motivation

- Tendenz signalverarbeitender Systeme:
 - zunehmende Verteiltheit
 - zunehmende Heterogenität
 - zunehmende Komplexität
 - abnehmende verfügbare Entwicklungszeit
- HW/SW-Codesign
- Anforderungen an die Systementwicklung:
 - plattformunabhängige Beschreibung des Gesamtsystems (Design)
 - Partitionierung des Designs bei invariantem Verhalten
 - separate Plattform-Beschreibung



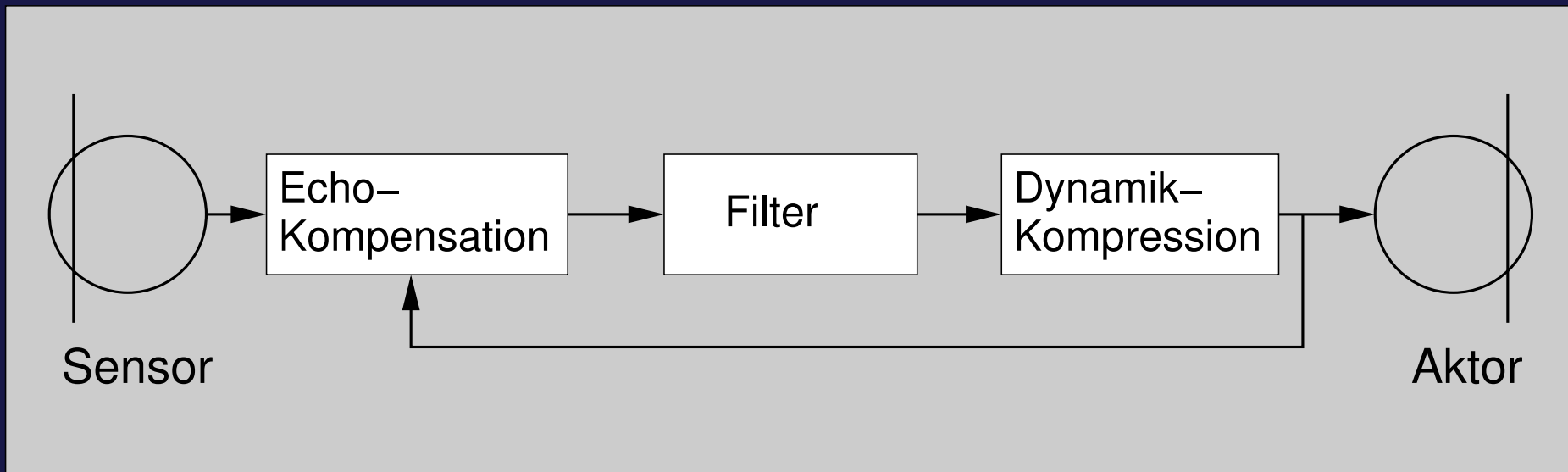
Gliederung

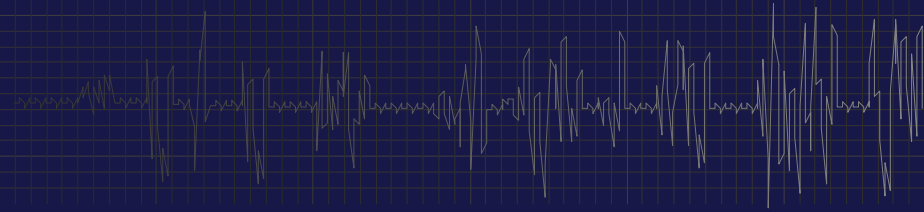
1. Motivation
2. Modellierung
 - Beispiele
 - Signalflussgraphen
 - Partitionierung
3. Partitionierung
4. Toolchain
5. Zusammenfassung



Modellierung – Beispiele

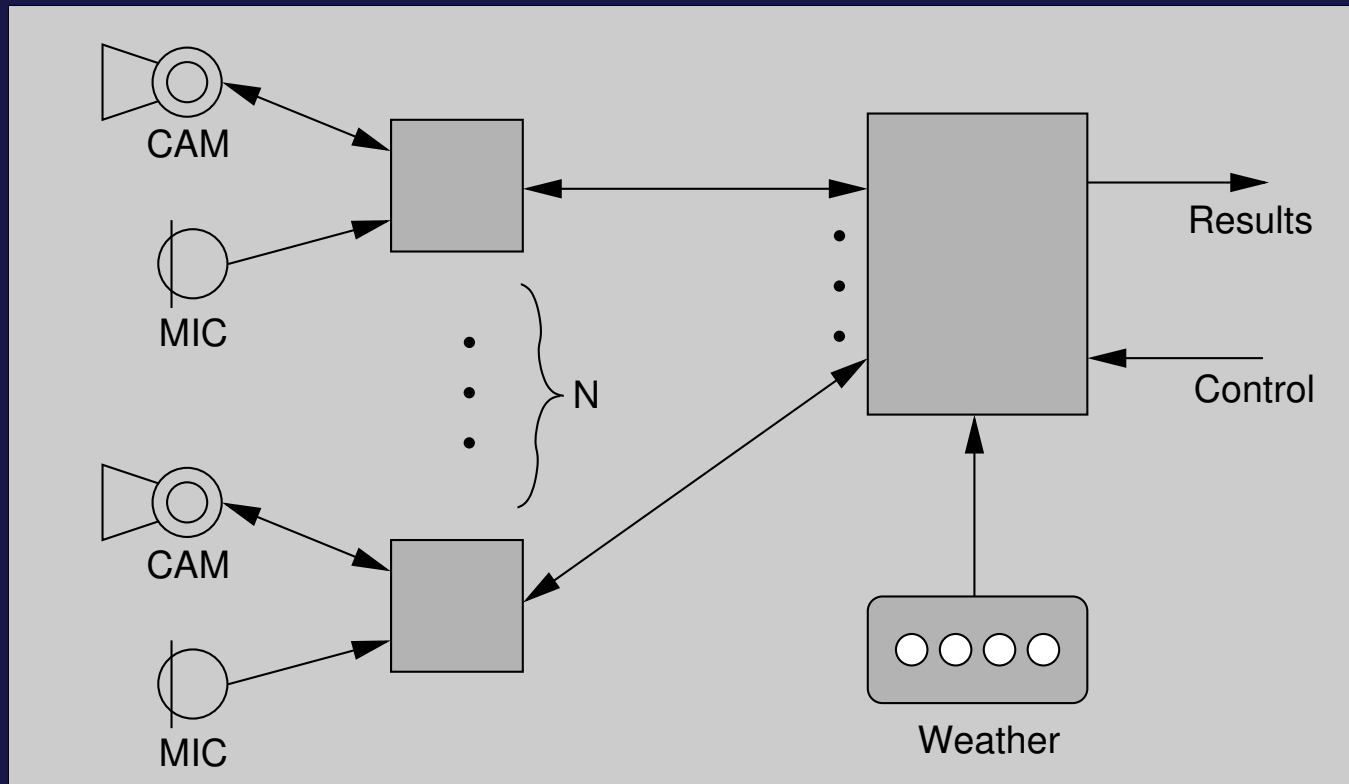
- Beispiel 1: Hörgerät

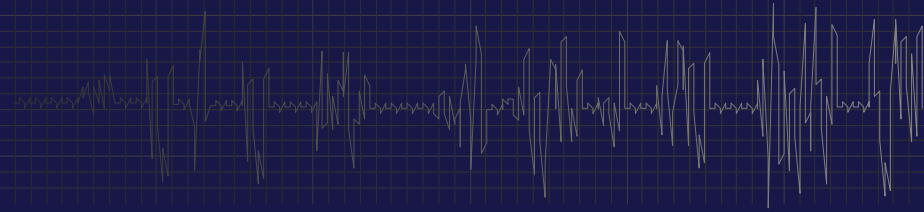




Modellierung – Beispiele

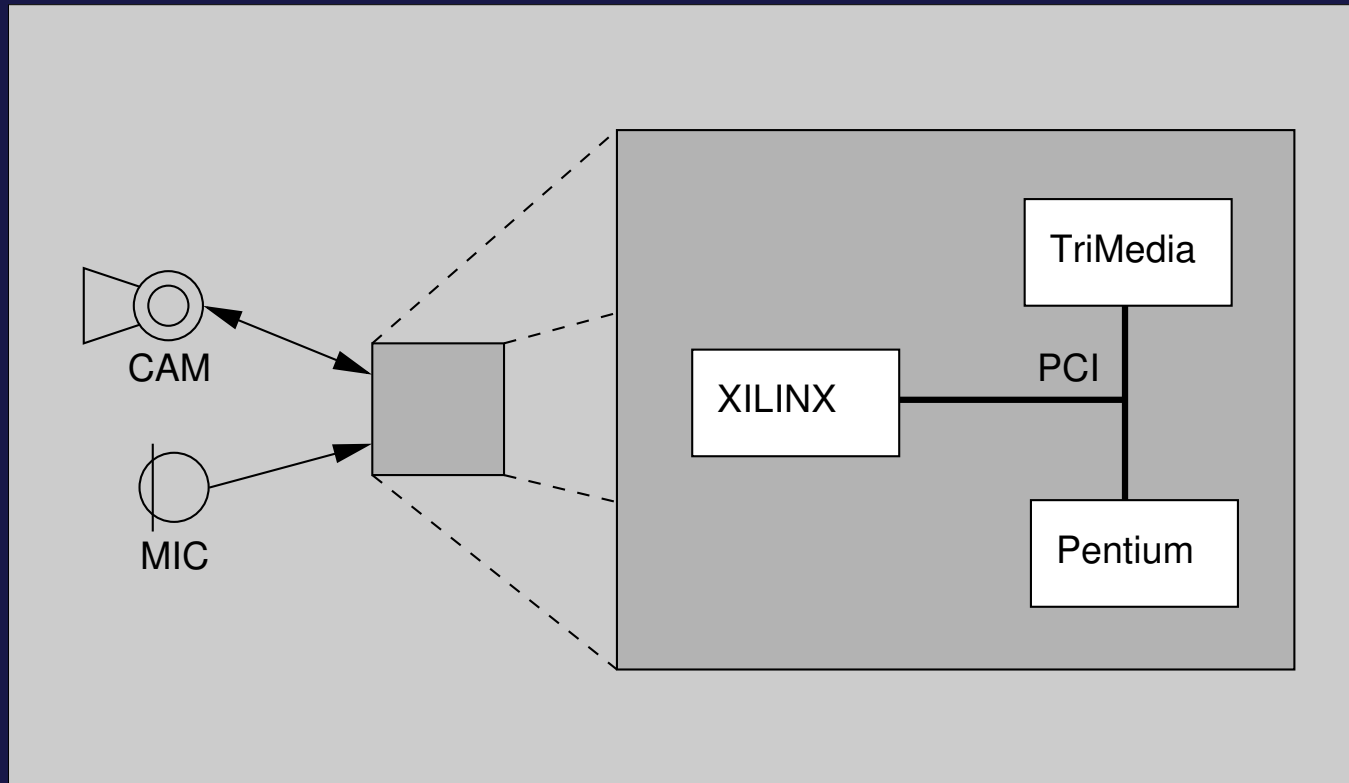
- Beispiel 2: INTEGROS-Projekt
 - Plattformen (Grobstruktur)

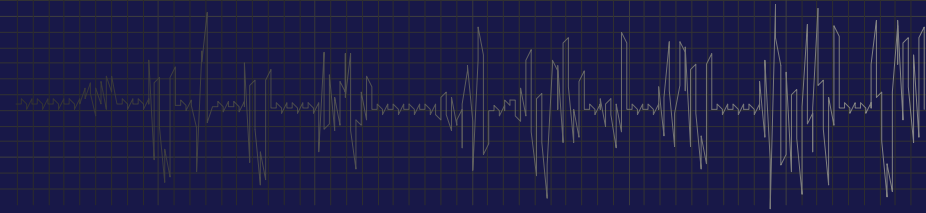




Modellierung – Beispiele

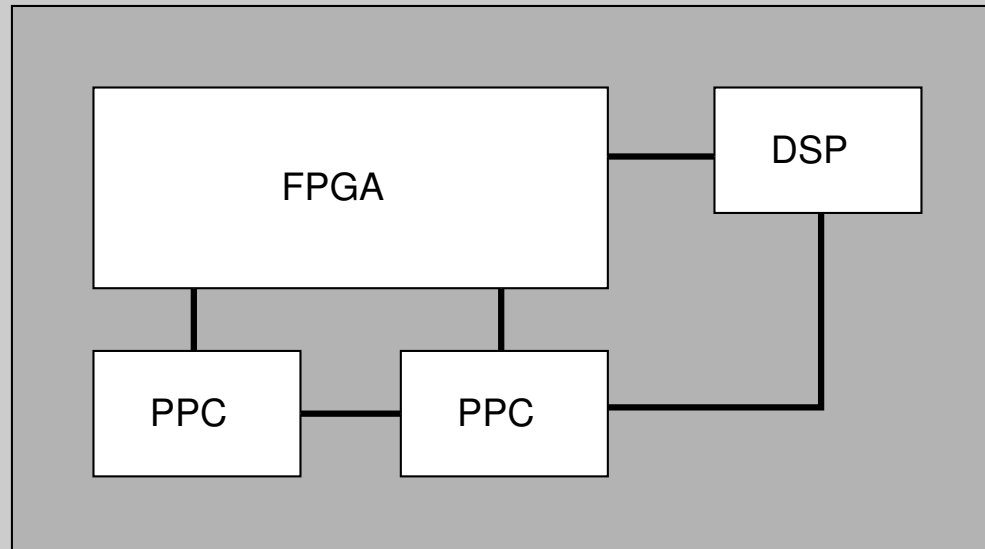
- Beispiel 2: INTEGROS-Projekt
 - Plattformen (Kameramodul)

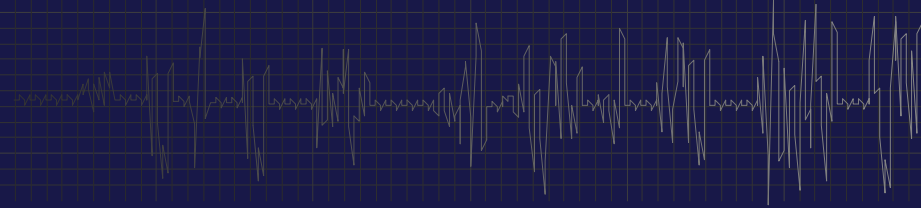




Modellierung – Beispiele

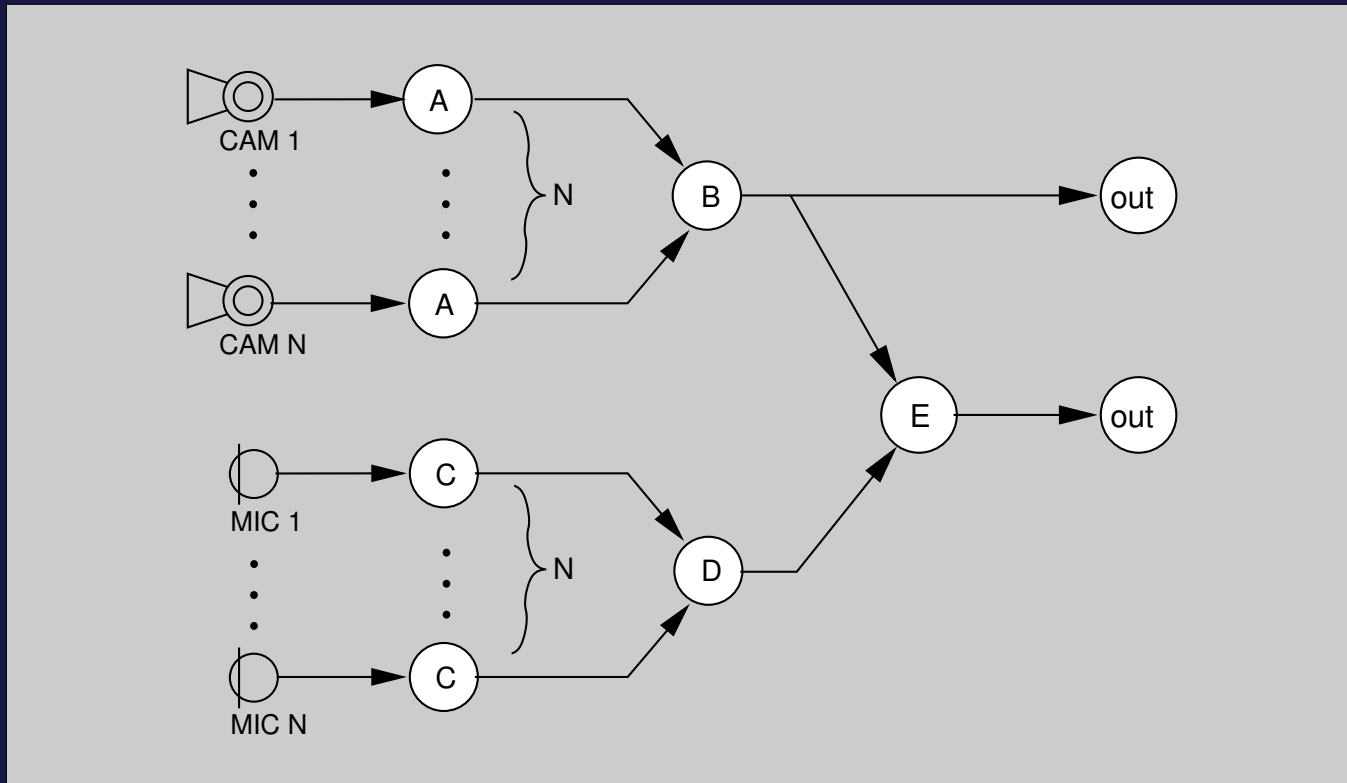
- Beispiel 2: INTEGROS-Projekt
 - Plattformen (Kreuzungsrechner)

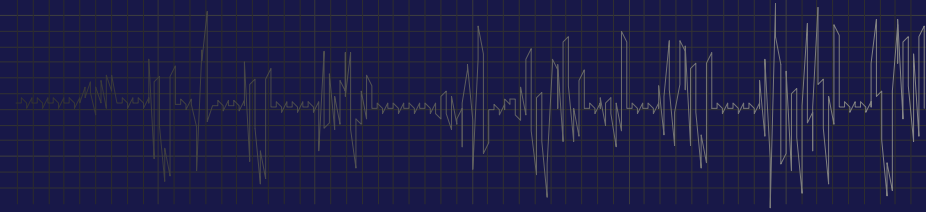




Modellierung – Beispiele

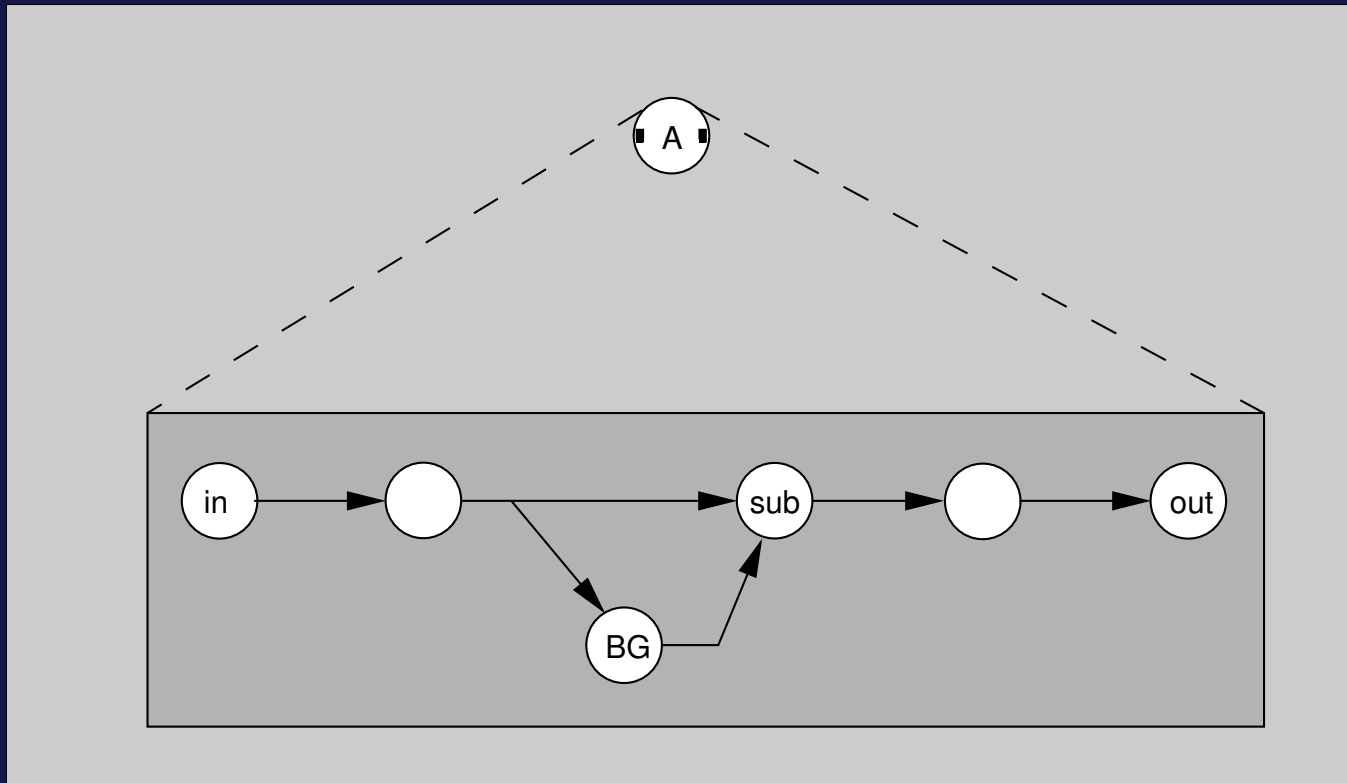
- Beispiel 2: INTEGROS-Projekt
 - Signalflussgraph

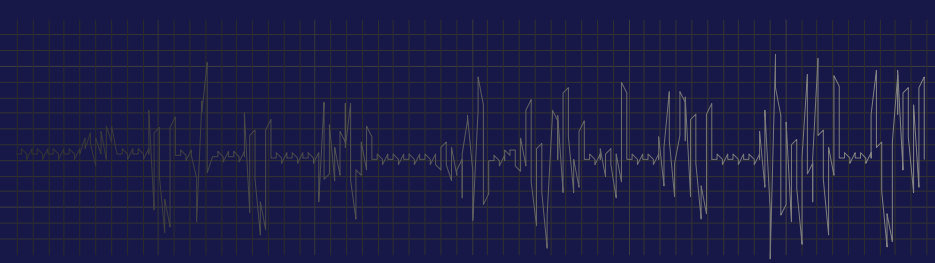




Modellierung – Beispiele

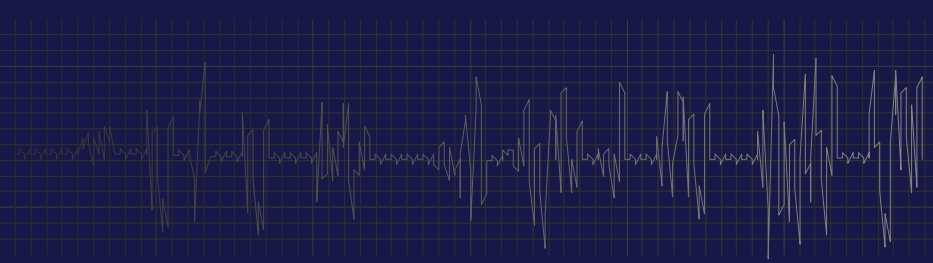
- Beispiel 2: INTEGROS-Projekt
 - Knotenexpansion (hierarchische Struktur)





Modellierung – Signalflussgraphen

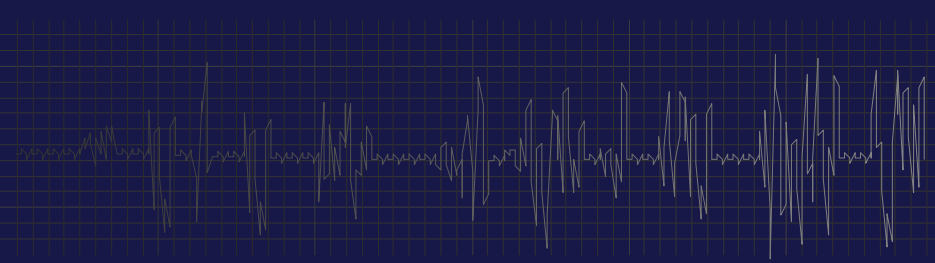
- Knoten repräsentieren Aktionen mit Signalen
- Kanten repräsentieren Signaltransport
- Knoten können aus Signalflussgraphen bestehen (hierarchische Strukturierung)
- Signalflussgraphen enthalten keine Information über die ausführende Plattform
- Ein- und Ausgabeknoten sind abstrakt
- syntaktische Trennung von
 - Verhalten (Signalflussgraph)
 - Umsetzung (Partitionierung)



Modellierung – Partitionierung

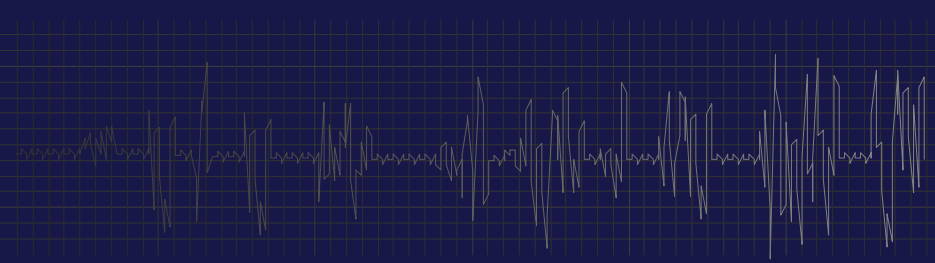
Partitionierung: Abbildung eines Signalflussgraphen auf ausführende Plattformen

- referenziert genau einen Signalflussgraph
- referenziert alle verwendeten Plattformen
- definiert Prozesse (synchrone Einheiten) auf jeder Plattform
- ordnet jeden (elementaren) Knoten des Signalflussgraphen genau einem Prozess zu
- konkretisiert die abstrakten Ein- und Ausgabeknoten

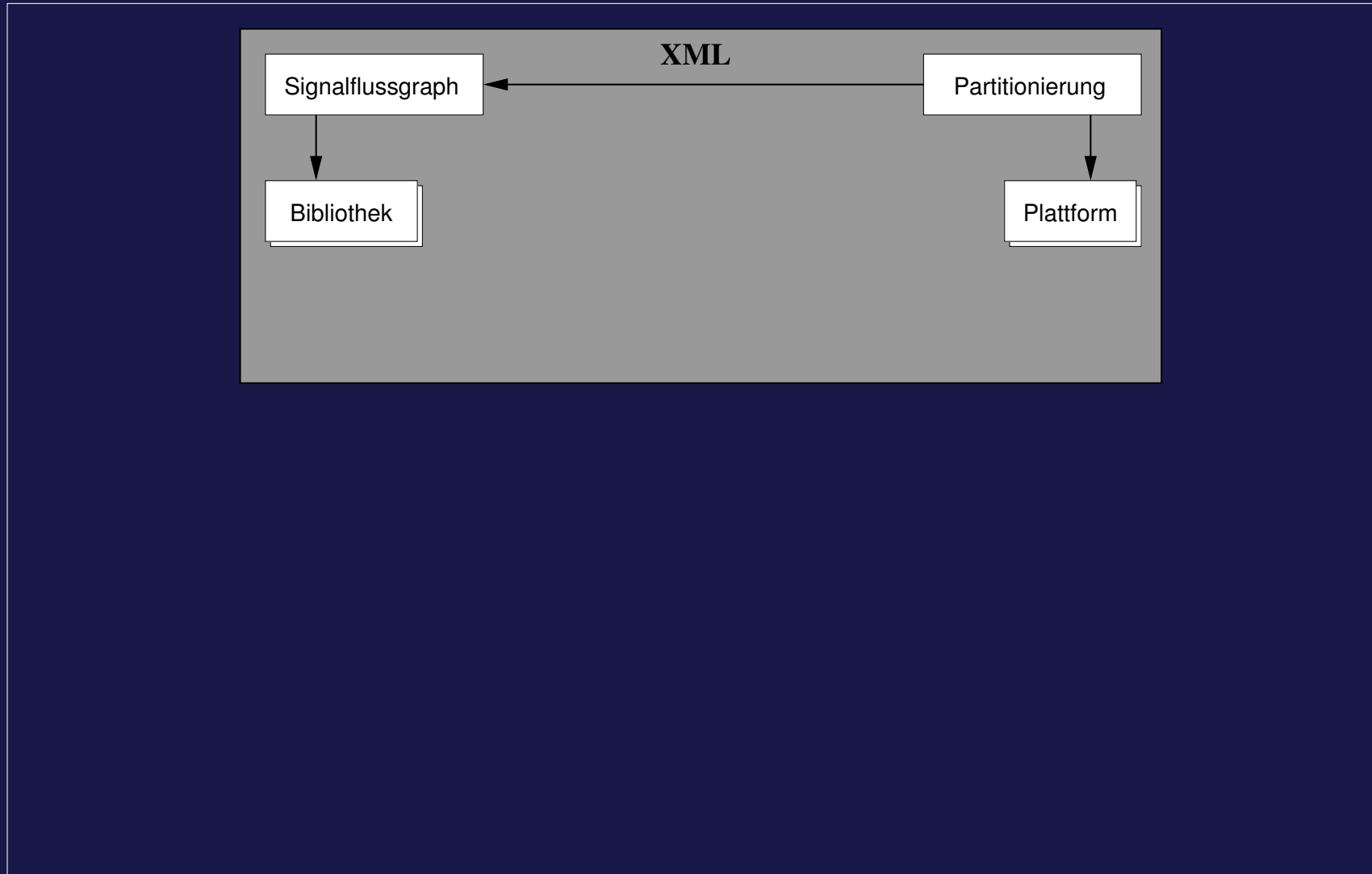


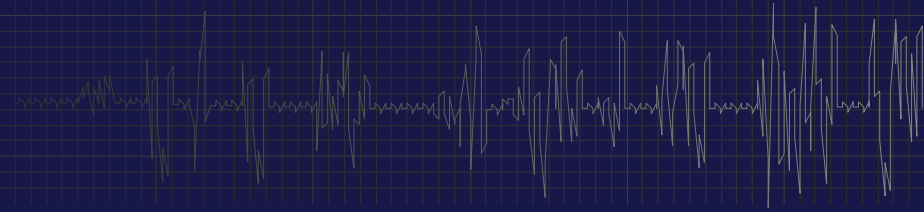
Gliederung

1. Motivation
2. Modellierung
3. Partitionierung
 - Übersicht
 - Signaltransport
 - Transport-Optimierung
4. Toolchain
5. Zusammenfassung

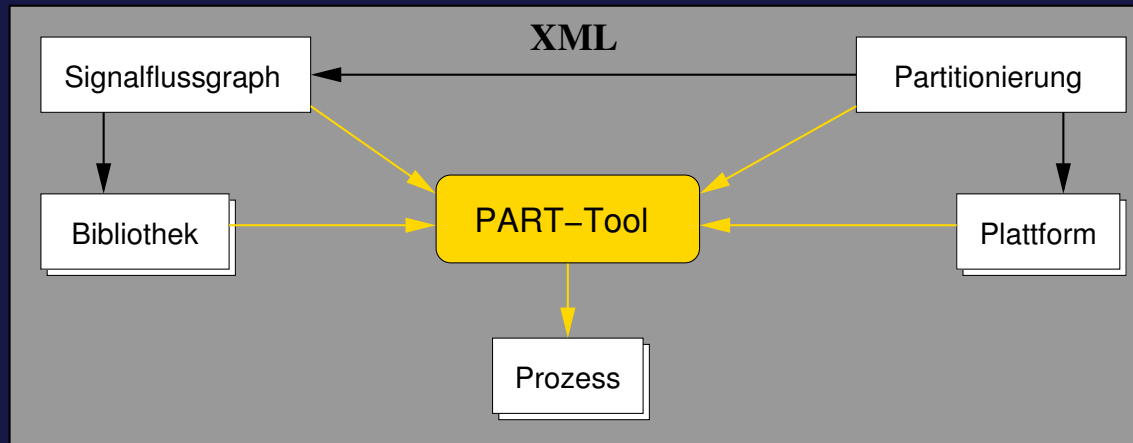


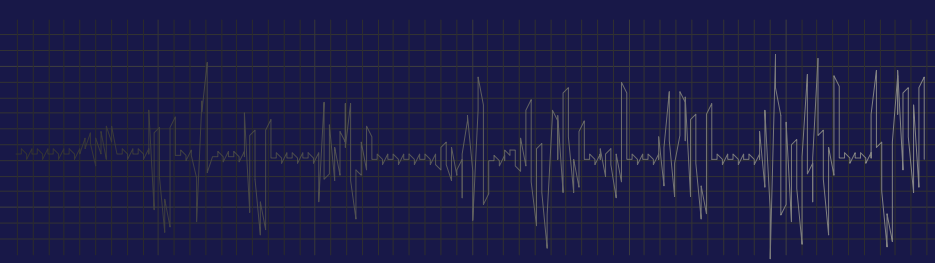
Partitionierung – Übersicht





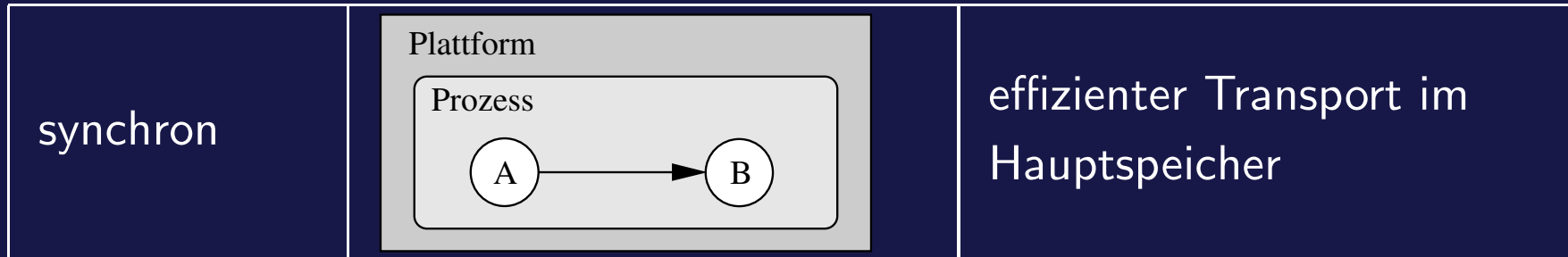
Partitionierung – Übersicht

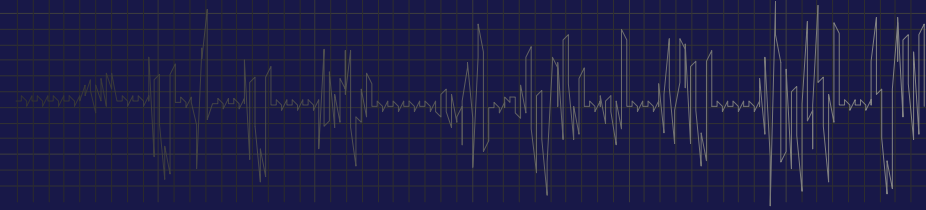




Partitionierung – Signaltransport

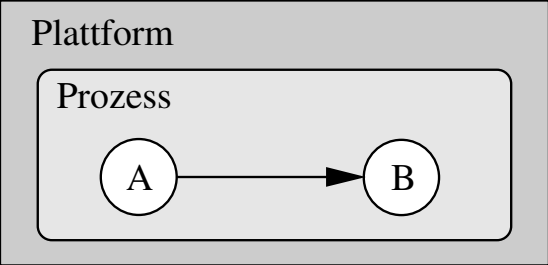
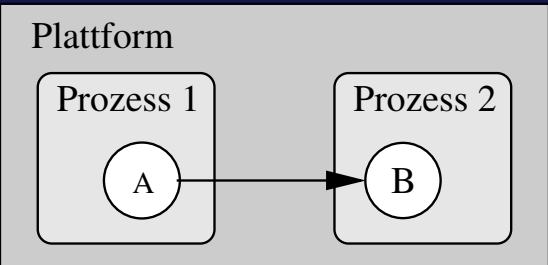
- Kantentypen

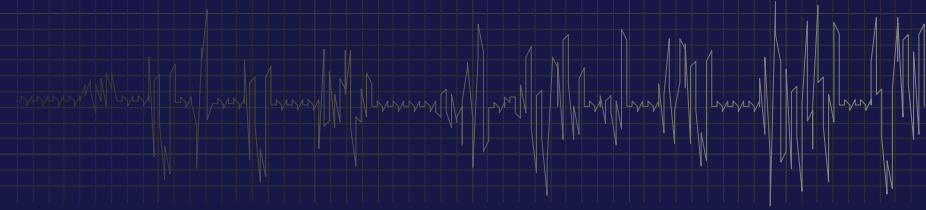




Partitionierung – Signaltransport

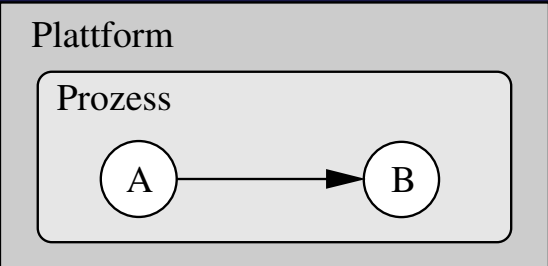
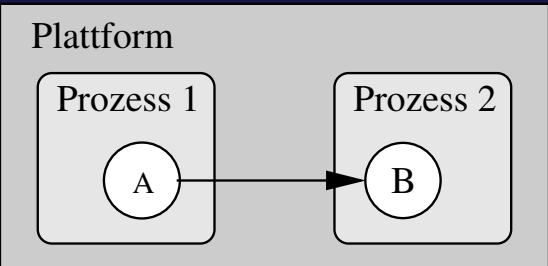
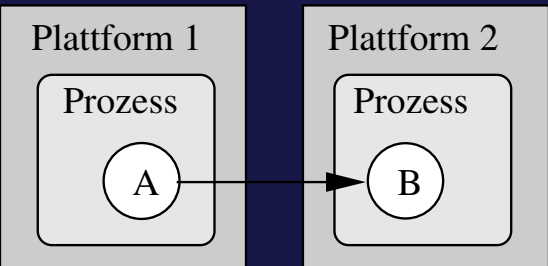
- Kantentypen

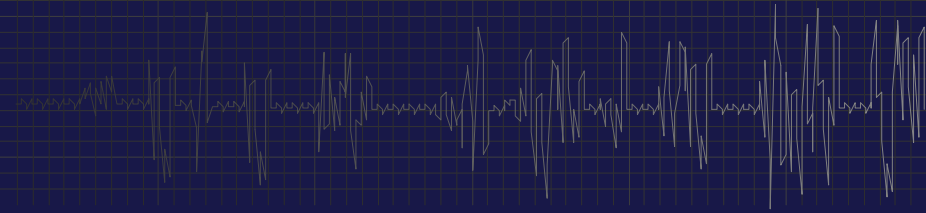
synchron		effizienter Transport im Hauptspeicher
Prozess- übergreifend		Synchronisation beim Transport über <i>shared memory</i>



Partitionierung – Signaltransport

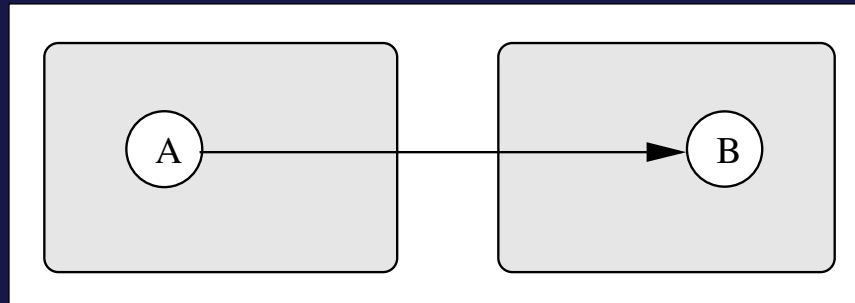
- Kantentypen

synchron		effizienter Transport im Hauptspeicher
Prozess- übergreifend		Synchronisation beim Transport über <i>shared memory</i>
Plattform- übergreifend		<i>dual-port-RAM</i> oder <i>I/O</i> , Konvertierung der Zahlenrepräsentation

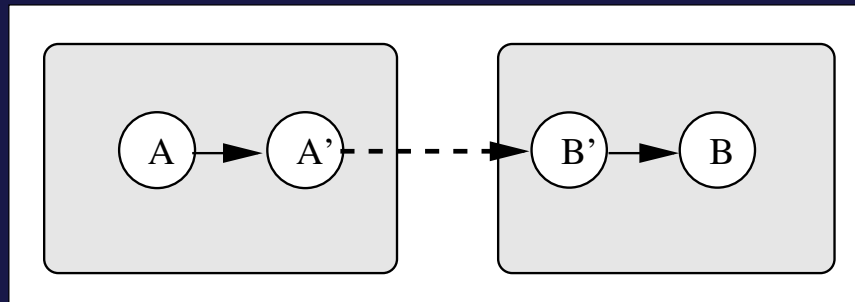


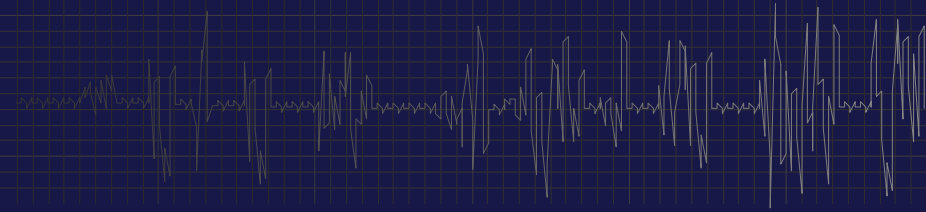
Partitionierung – Transport-Optimierung

- Das Prinzip:



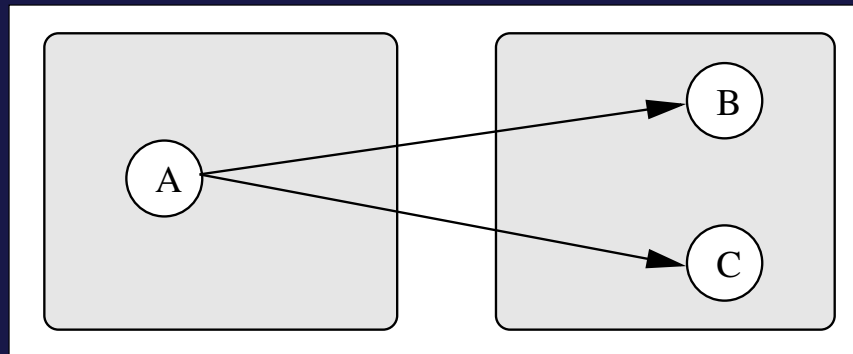
wird ersetzt durch



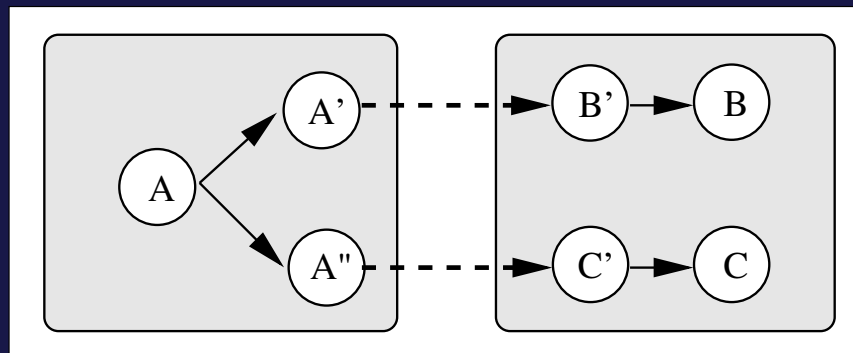


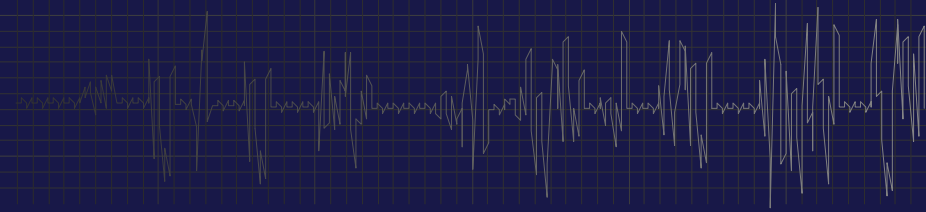
Partitionierung – Transport-Optimierung

- Das Problem: doppelte Kanten



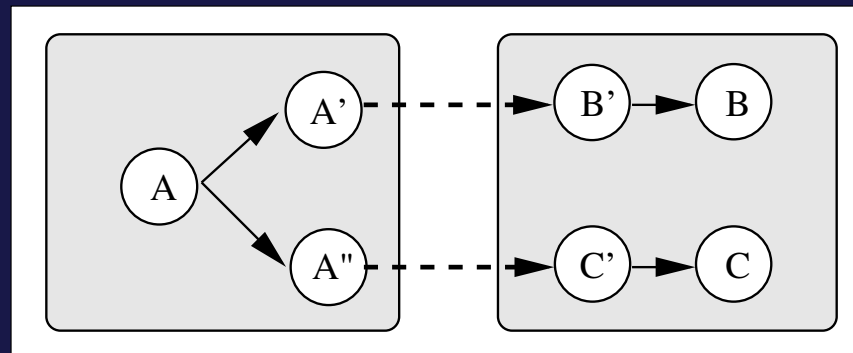
wird ohne Optimierung ersetzt durch



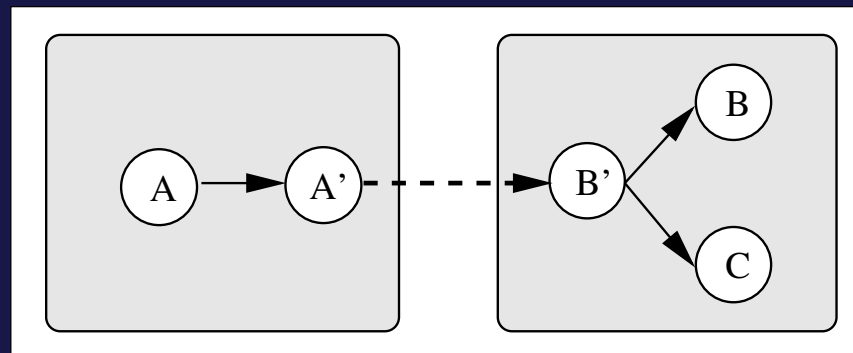


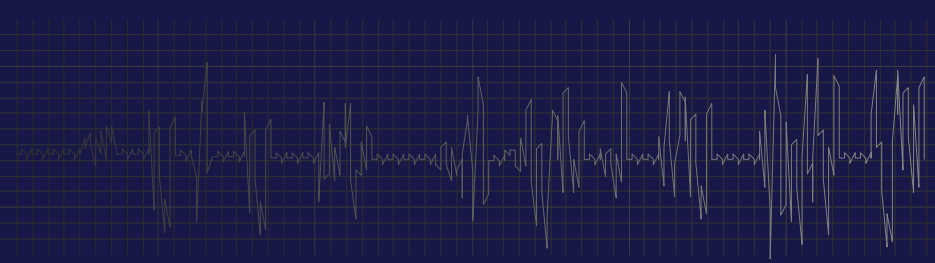
Partitionierung – Transport-Optimierung

- Die Lösung: Vermeidung doppelter Kanten



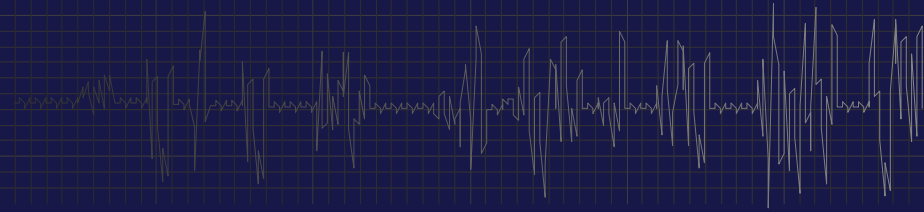
wird ersetzt durch



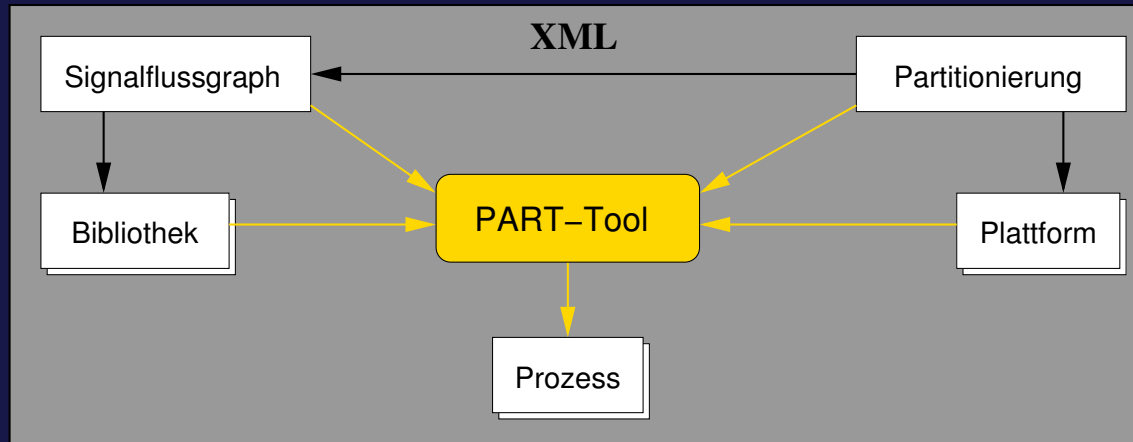


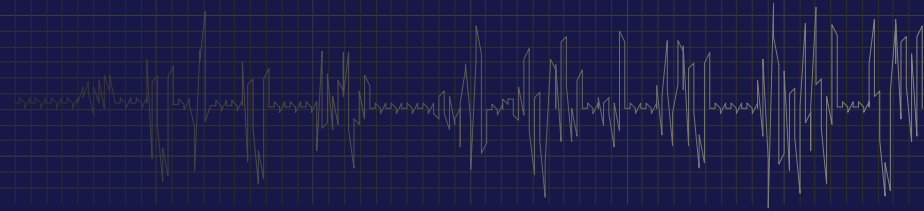
Gliederung

1. Motivation
2. Modellierung
3. Partitionierung
4. **Toolchain**
5. Zusammenfassung

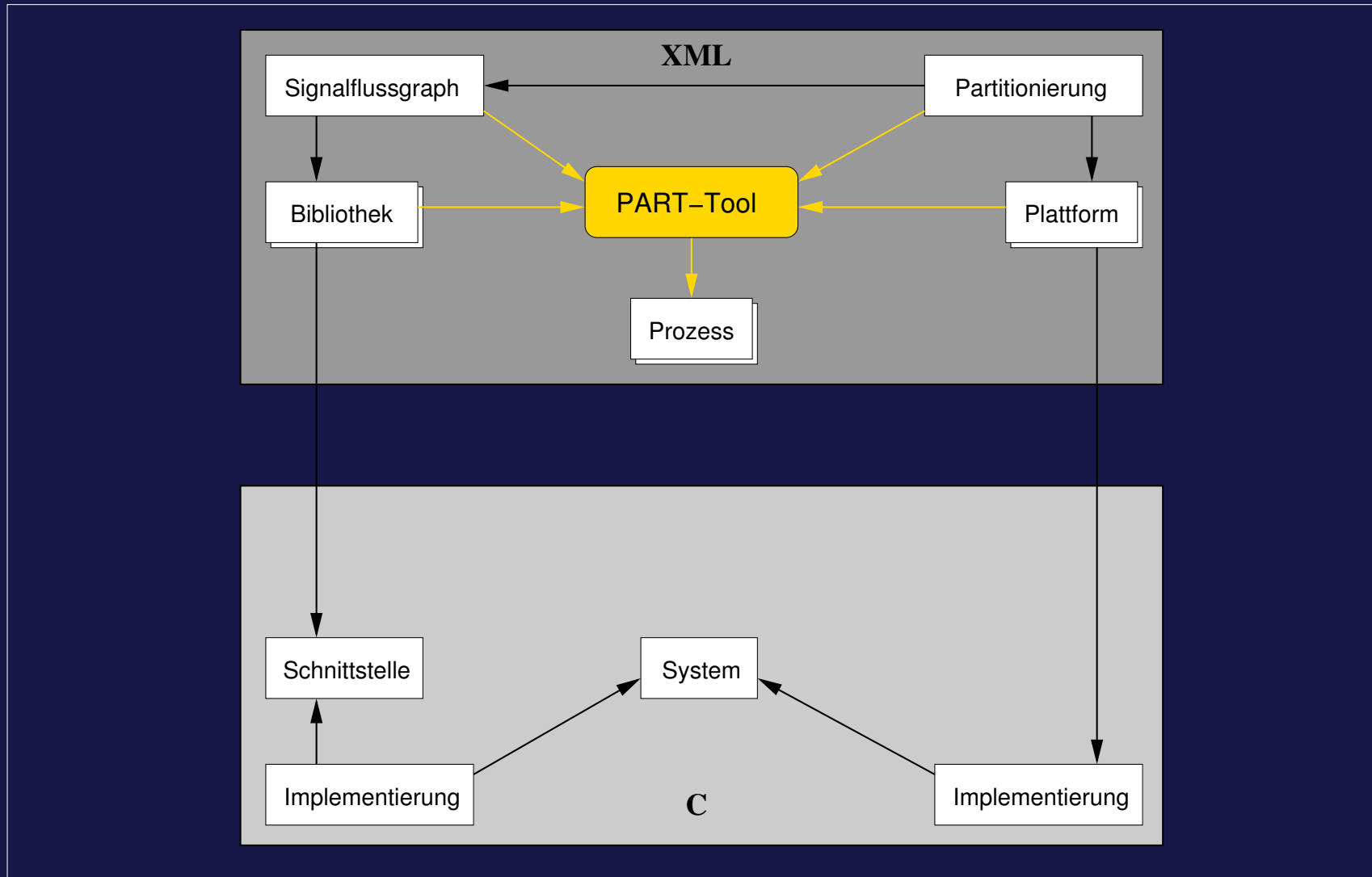


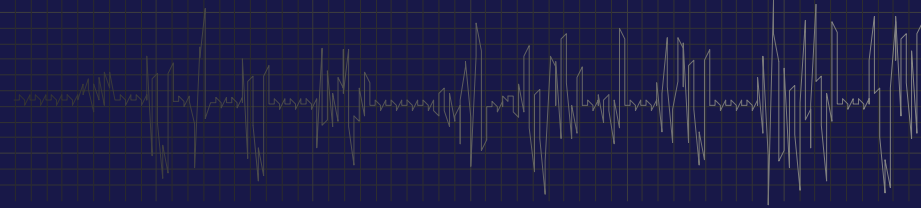
Toolchain – Übersicht



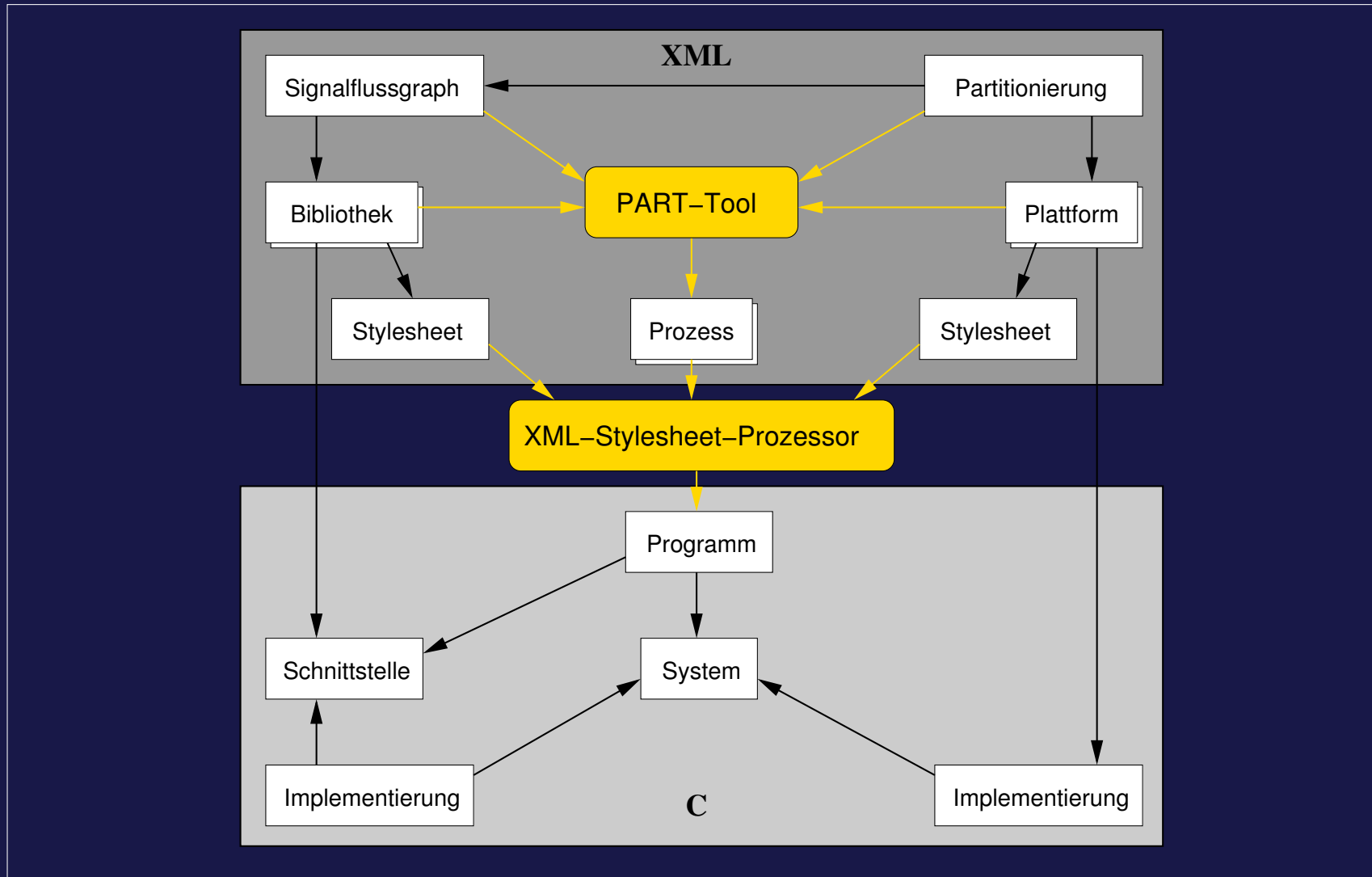


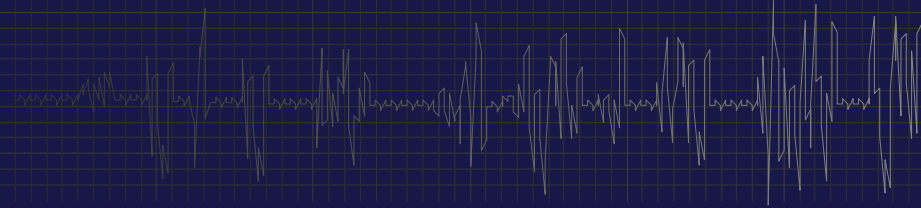
Toolchain – Übersicht



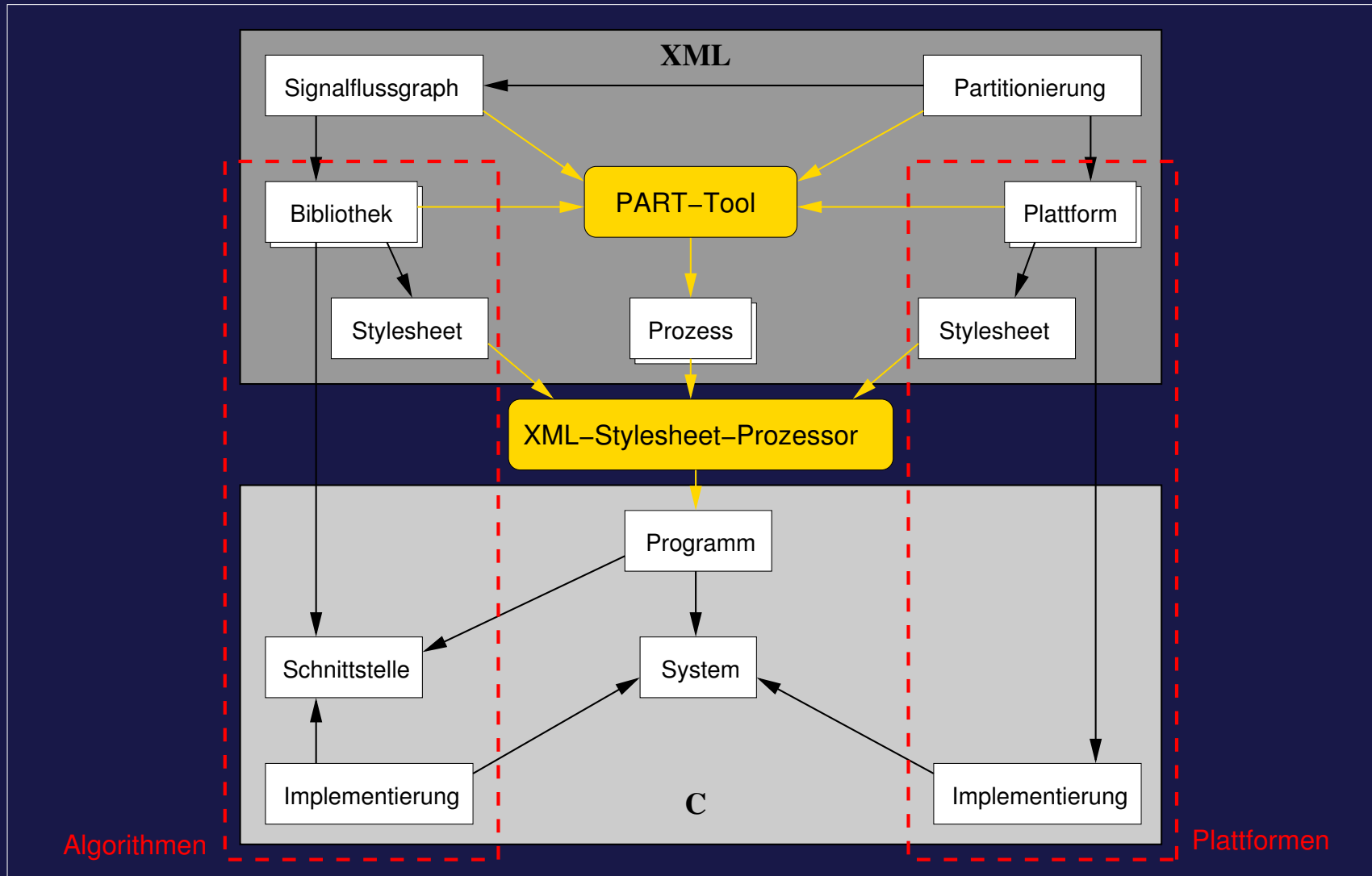


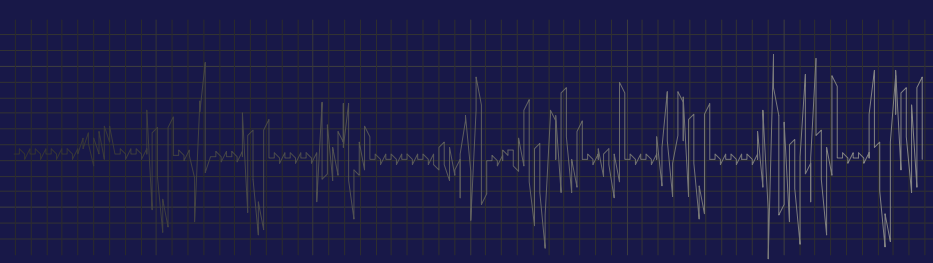
Toolchain – Übersicht





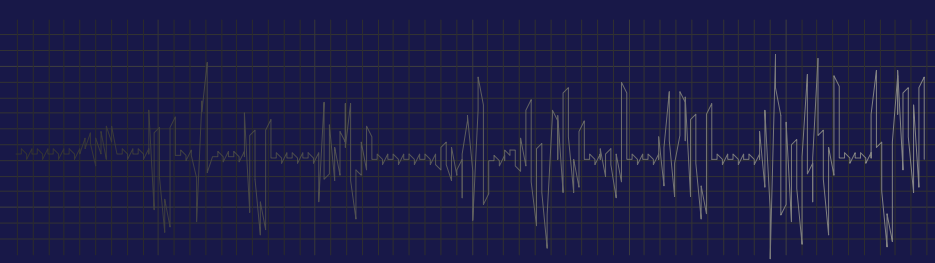
Toolchain – Übersicht





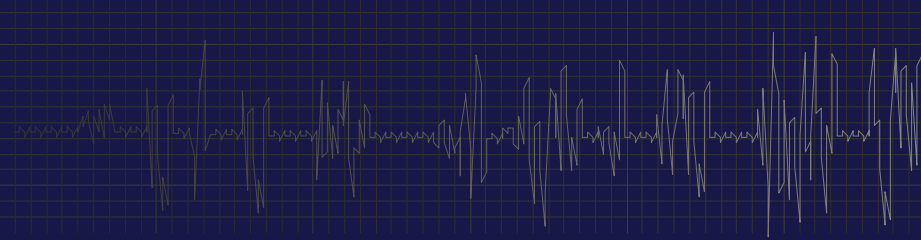
Toolchain

- Aufgaben:
 - Konsistenzprüfungen:
 - * Eindeutigkeiten
 - * Referenzen
 - * Auflösung von Signaltypen
 - Zusammenstellung der *build*-Verzeichnisse
 - für jeden Prozess ein Programm generieren \Rightarrow (*cross*)-*compiler*
- Werkzeuge:
 - GNU-Tools (*make, sed, awk, u.a.*)
 - Stylesheet-Prozessor (*Xalan*)
 - XML-Bibliothek (*Xerces-C*) für PART



Gliederung

1. Motivation
2. Modellierung
3. Partitionierung
4. Toolchain
5. **Zusammenfassung**



Zusammenfassung

- Durch die deklarative Beschreibung signalverarbeitender Systeme gelingen
 - die Trennung von Verhalten und Umsetzung
 - die Isolierung signalverarbeitender Algorithmen und Teilsysteme
 - die Automatisierung des Signaltransports.
- Vorteile/Nutzen:
 - portierbare, verteilte Anwendungen
 - wiederverwendbare Lösungen und Teillösungen
 - schnelle Entwicklung neuer Applikationen
 - schnelle Anpassung an neue Plattformen
- *open-source*-Implementierung: www.freesp.de