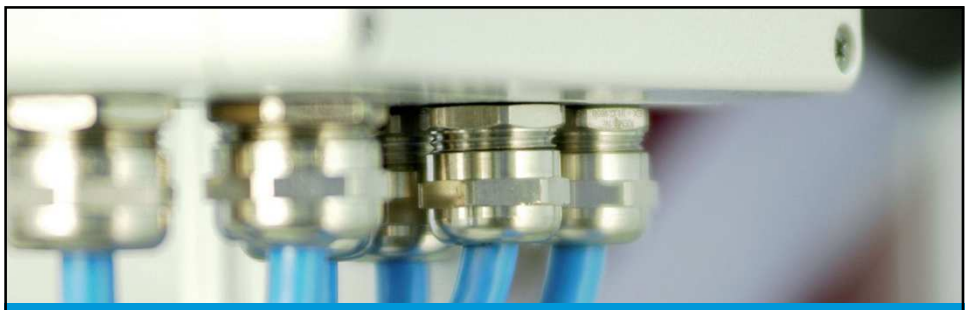


Automatisiertes Testen von verteilten Systemen über Petrinetze

Tim Ruß

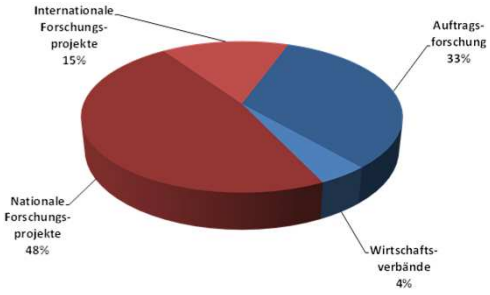
ifak

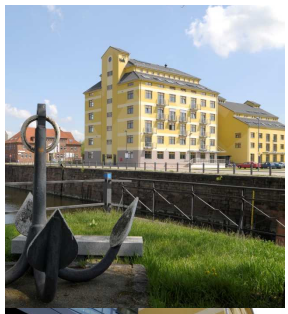
Kurzvorstellung ifak

ifak

Vorstellung ifak

- ifak: Institut für Automation und Kommunikation e.V., Magdeburg
- 50 Mitarbeiter plus Studenten und Gastwissenschaftler
- Zusammensetzung der Erträge aus F&E:





ifak

Geschäftsfelder des ifak

IKT & Automation
- Echtzeitkommunikation, Geräte- und Systemtest…

Wasser & Energie
- Kläranlagen, Integrierte Planungswerkzeuge…

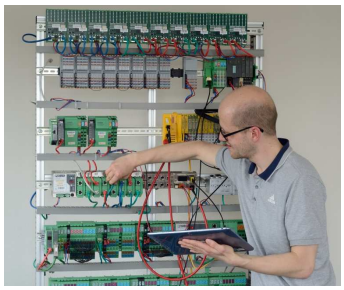
Messtechnik & Leistungselektronik
- Prozessmesstechnik, Kontaktlose Energie- und Datenübertragung…

Verkehr & Assistenz
- Intelligente Verkehrssysteme, Qualitätssicherung vernetzter Mobilitätssysteme, Elektromobilität…

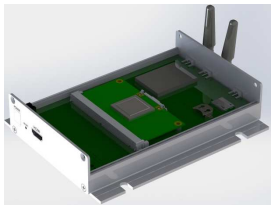
ifak

Einige Kommunikations-Domänen

- Kommunikation in der Automation
  - Ethernet, PROFINET…
  - Prüflabor Feldbussysteme



- Kommunikation im Verkehr
  - WLANp, Car2X…
  - Kooperative Applikationen



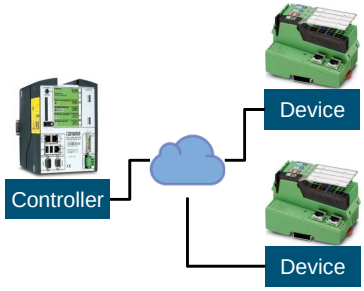
ifak

Motivation: Testen von Netzwerkkommunikation

ifak

Motivation: Industrieautomation mit PROFINET-Protokoll

- Steuerungen (Controller) tauschen Daten mit Feldgeräten (Devices) aus
- Typische Testzeitpunkte:
  - Bei Inbetriebnahme einer Anlage
  - Nach Aufspielen neuer Stack-Versionen
- Typische Testziele:
  - Erfolgreicher Verbindungsaufbau, ggf. zu mehreren Devices
  - Robustheit gegenüber Störungen



ifak

Motivation

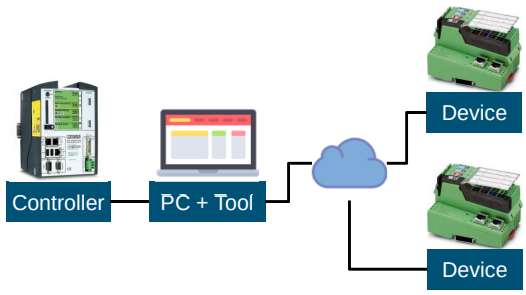
- Wunsch: Manuelle Testabläufe **automatisieren**
- Bsp.: PROFINET-Verbindungsaufbau in Wireshark:

Time	Source	Destination	Protocol	Length	Info
12.087517	PhoenixC_51:1e:be	PN-MC_00:00:00	PN-DCP	60	Ident Req, Xid:0x120d13ad, NameOfStation:"bk01"
12.613632	PhoenixC_51:1e:be	PN-MC_00:00:00	PN-DCP	60	Ident Req, Xid:0x120e13ae, AliasName:"port-001.rfc-470-pn-51-1e-be"
14.190131	PhoenixC_51:1e:be	PN-MC_00:00:00	PN-DCP	60	Ident Req, Xid:0x120f13af, NameOfStation:"bk01"
14.196615	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	PN-DCP	122	Ident Ok , Xid:0x120f13af, NameOfStation:"bk01", Dev-Options(9), DeviceVendorValu
14.219544	PhoenixC_19:2c:50	LLDP_Multicast	LLDP	119	Chassis Id = bk01 Port Id = port-001 TTL = 20 RTClass3 Port Status = OFF
14.222106	PhoenixC_51:1e:c0	LLDP_Multicast	LLDP	114	Chassis Id = rfc-470-pn-51-1e-be Port Id = port-001 TTL = 20 RTClass3 Port Status
15.303088	PhoenixC_51:1e:be	Broadcast	ARP	60	Who has 172.16.48.200? Tell 172.16.48.206
15.306568	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	ARP	60	172.16.48.200 is at 00:a0:45:19:2c:4f
16.324137	PhoenixC_51:1e:c0	LLDP_Multicast	LLDP	114	Chassis Id = rfc-470-pn-51-1e-be Port Id = port-001 TTL = 20 RTClass3 Port Status
17.446753	PhoenixC_51:1e:be	Broadcast	ARP	60	Who has 172.16.48.200? Tell 172.16.48.206
17.450708	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	ARP	60	172.16.48.200 is at 00:a0:45:19:2c:4f
17.453146	172.16.48.206	172.16.48.200	PNIO-CM	641	Connect request, ARBlockReq, IOCRBlockReq, IOCRBlockReq, ExpectedSubmoduleBlockRe
17.475674	172.16.48.200	172.16.48.206	PNIO-CM	244	Connect response, OK, ARBlockRes, IOCRBlockRes, IOCRBlockRes, AlarmCRBlockRes, Mo
17.570402	172.16.48.206	172.16.48.200	PNIO-CM	212	Write request, IODWriteReqHeader, Api:0x0, Slot:0x0/0x1, Index:(0x1), 6 bytes
17.578695	172.16.48.200	172.16.48.206	PNIO-CM	206	Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x0/0x1, Index:(0x1), OK
17.599673	PhoenixC_51:1e:be	PhoenixC_19:2c:4f	PNIO	60	RTC1(legacy), ID:0xc010, Len: 40, Cycle:65504 (Valid,Primary,Ok,Run)
17.632142	172.16.48.206	172.16.48.200	PNIO-CM	209	Write request, IODWriteReqHeader, Api:0x0, Slot:0x0/0x1, Index:(0x2), 3 bytes
17.638700	172.16.48.200	172.16.48.206	PNIO-CM	206	Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x0/0x1, Index:(0x2), OK
17.682703	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	PNIO	60	RTC1(legacy), ID:0xc000, Len: 40, Cycle:57312 (Valid,Primary,Ok,Run)
17.694034	172.16.48.206	172.16.48.200	PNIO-CM	215	Write request, IODWriteReqHeader, Api:0x0, Slot:0x2/0x1, Index:(0x7000), 9 bytes
17.700654	172.16.48.200	172.16.48.206	PNIO-CM	206	Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x2/0x1, Index:(0x7000), OK
17.755729	172.16.48.206	172.16.48.200	PNIO-CM	215	Write request, IODWriteReqHeader, Api:0x0, Slot:0x3/0x1, Index:(0x7000), 9 bytes
17.762605	172.16.48.200	172.16.48.206	PNIO-CM	206	Write response, OK, IODWriteResHeader, Api:0x0, Slot:0x3/0x1, Index:(0x7000), OK
17.848499	172.16.48.206	172.16.48.200	PNIO-CM	174	Control request, IODControlReq Prm End.req, Command: ParameterEnd
17.855028	PhoenixC_51:1e:be	PhoenixC_19:2c:4f	PNIO	60	RTC1(legacy), ID:0xc010, Len: 40, Cycle: 8160 (Valid,Primary,Ok,Run)
17.860667	172.16.48.200	172.16.48.206	PNIO-CM	174	Control response, OK, IODControlRes Prm End.rsp, Command: Done
17.883689	172.16.48.206	172.16.48.200	PNIO-CM	206	Control request, IOXBlockReq Application Ready.req, Command: ApplicationReady, Mo
17.931000	172.16.48.206	172.16.48.200	PNIO-CM	174	Control response, OK, IOXBlockRes Application Ready.rsp, Command: Done
17.938683	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	PNIO	60	RTC1(legacy), ID:0xc000, Len: 40, Cycle:65504 (Valid,Primary,Problem,Run)
18.041997	172.16.48.206	172.16.48.200	PNIO-CM	206	Read request, IODReadReqHeader, Api:0x0, Slot:0x0/0x0, Index:PDRealData, 8192 byt
18.053681	172.16.48.200	172.16.48.206	PNIO-CM	418	Read response, OK, IODReadResHeader, Api:0x0, Slot:0x0/0x0, Index:PDRealData, 212
18.110248	PhoenixC_51:1e:be	PhoenixC_19:2c:4f	PNIO	60	RTC1(legacy), ID:0xc010, Len: 40, Cycle:16352 (Valid,Primary,Ok,Run)
18.194685	PhoenixC_19:2c:4f	PhoenixC_51:1e:be	PNIO	60	RTC1(legacy), ID:0xc000, Len: 40, Cycle: 8160 (Valid,Primary,Problem,Run)
18.365608	PhoenixC_51:1e:be	PhoenixC_19:2c:4f	PNIO	60	RTC1(legacy), ID:0xc010, Len: 40, Cycle:24544 (Valid,Primary,Ok,Run)

ifak

Lösung: Testsoftware

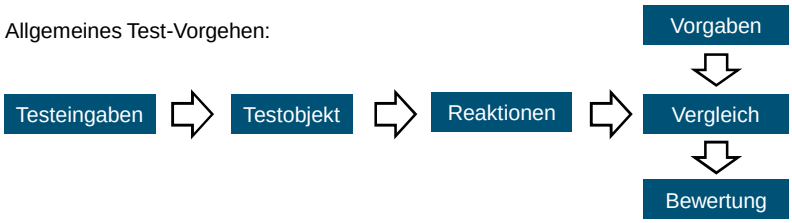
- Projekt: PROGES (2012 – 2014) - „Programmierbarer Fehlergenerator für Ethernet-basierte Automatisierungsnetze“
- Anforderungen:
  - Erfüllung von Sollverhalten nachweisen
  - Gezielt Datenverkehr manipulieren
- Lösung: PC mit Testsoftware als Man-in-the-middle direkt hinter Steuerung
  - Überwachung
  - Manipulation



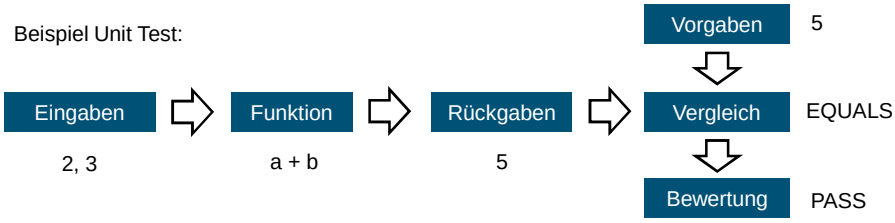
Ansatz: Modellbasierter Test von sequentiellem Datenverkehr

Ansatz: Testen der beobachtbaren Reaktionen

Allgemeines Test-Vorgehen:



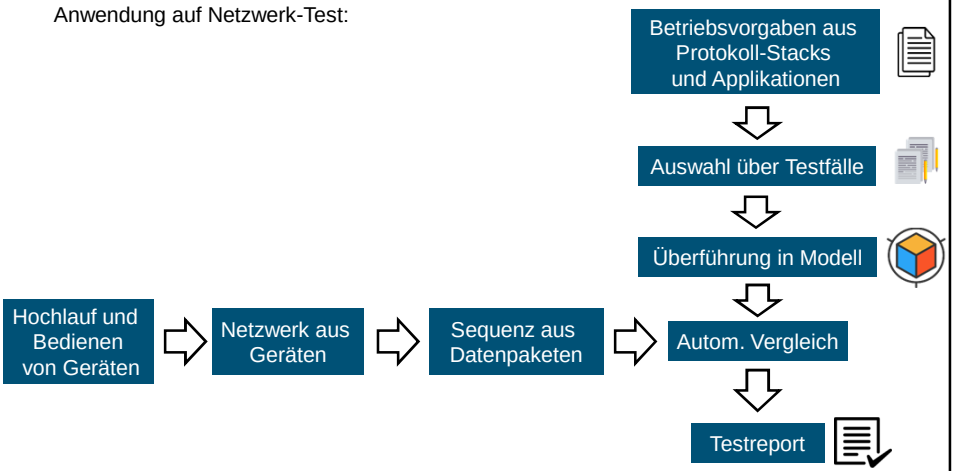
Beispiel Unit Test:



ifak

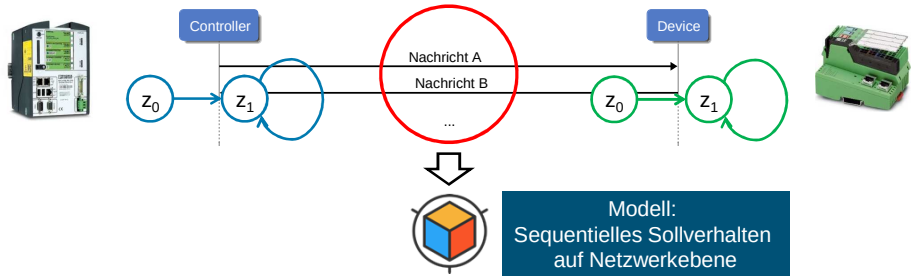
Ansatz: Testen der beobachtbaren Reaktionen

Anwendung auf Netzwerk-Test:



ifak

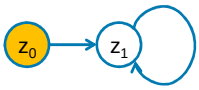
Ansatz: Testen der beobachtbaren Reaktionen



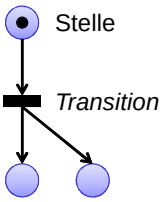
- Protokolle wie PROFINET über Automatenmodelle im Stack implementiert
- Sequentieller Datenverkehr beobachtbar
- Vollständiger Test aller Automaten aufgrund der Komplexität nicht möglich
- Alternative: Test ausgesuchten Datenverkehrs

Ansatz: Sollverhalten als Modelle

- Zustand in Sequenzdiagrammen nicht speicherbar
- Daher Überführung in gerichteten Graphen
- Netzwerk-Nachrichten ändern Zustand
- **Endlicher Zustandsautomat** (FSM): Nur 1 Zustand zur gleichen Zeit. Jedoch Modellieren paralleler Komm.-Beziehungen gefordert.

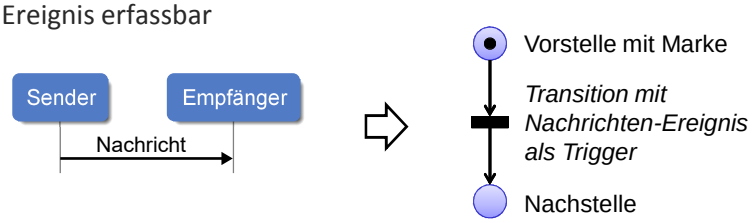


- **Bedingungs-Ereignis-Netz** (einfachster Typ)
- **Stellen-Transitions-Netz** (Stellen mit erhöhter Kapazität)
- **Petrinetz** (Marken mit Attributen)

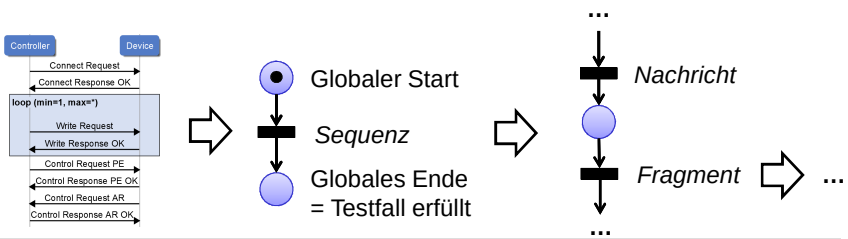


Ansatz: Sollverhalten als Modelle

- Sender und Empfänger nicht explizit modellieren, da nur Sende-Ereignis erfassbar



- Top-Down-Entwurf

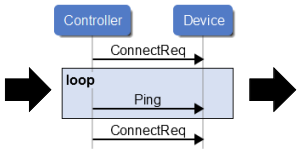


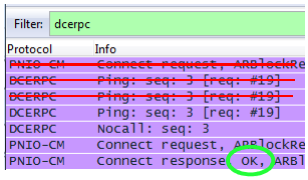
ifak

Manipulation des Datenverkehrs

- Weiteres Feature: Manipulation zur Fehlergenerierung
  - Fehlerhafte Geräte nachstellen
  - Protokoll-Stack prüfen (Reaktion bei Paketverlust etc.)
- Umsetzung: Action an Transition
  - Weiterleiten
  - Löschen (nicht weiterleiten)
  - Ändern von Parametern

```
1  // Beteiligte Komponenten
2  #import KommA_Components
3  // Relevante Profinet Services
4  #import KommA_Profinet
5
6  // LöscheVerbindungsaufbaus auf der Leitung
7  ConnectReq(Controller->Device): erase;
8
9  // Controller fragt 3 mal mit einem Ping nach dem Server
10 loop_begin [min=1, max=2]
11   // Zwei Pings davon werden auf der Leitung gelöscht
12   Ping(Controller->Device): erase;
13 loop_end
14
15 // Prüfe Wiederholung des Connect-Request
16 ConnectReq(Controller->Device)
```



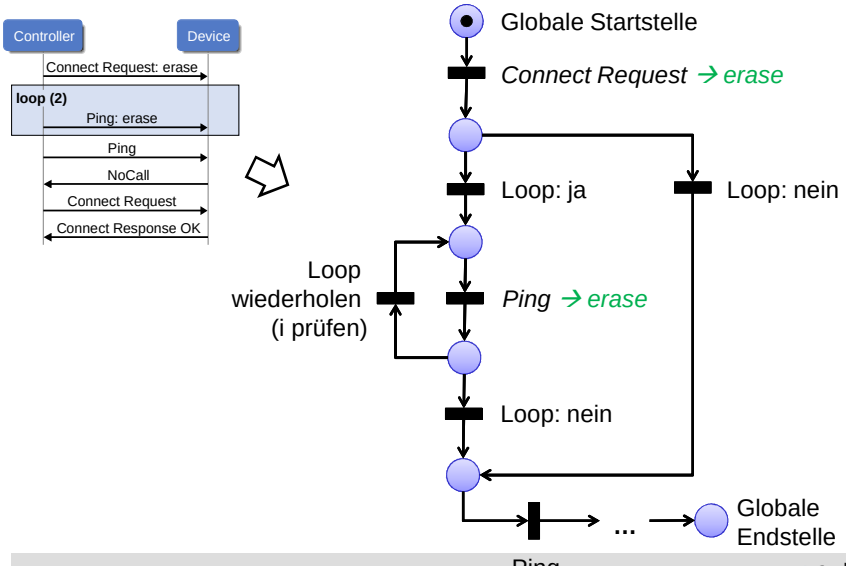


Sollverhalten textuell **Sollverhalten grafisch** **Verhalten im Netz**

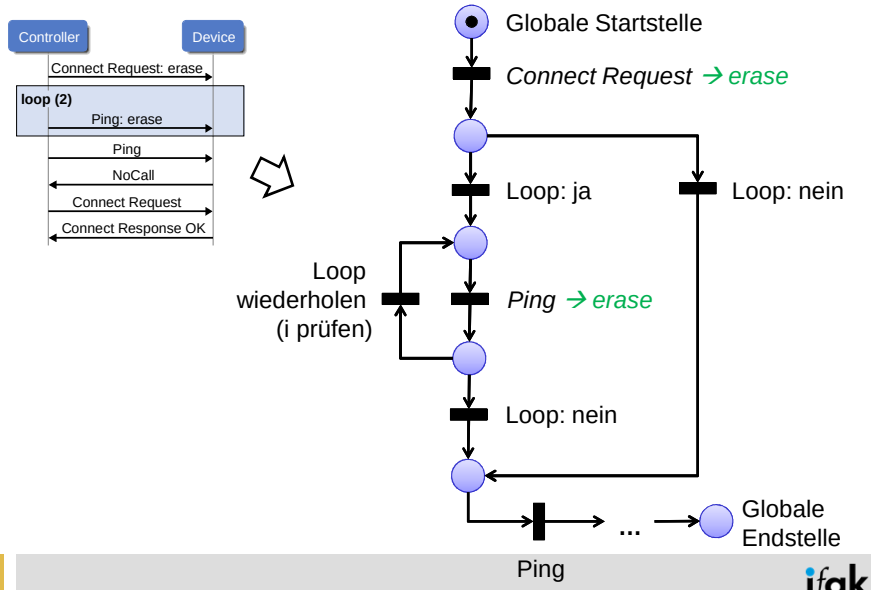
ifak

Markenspiel an einem Beispiel

Beispiel Markenspiel



Beispiel Markenspiel



ifak

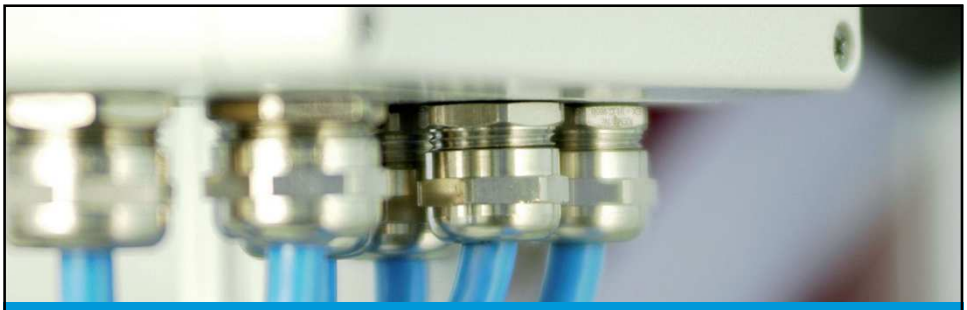
Fähigkeiten und Grenzen des Ansatzes

ifak

Fähigkeiten und Grenzen der Lösung

- Gutverhalten
  - Passiv mitschneiden
  - Punkt des Datenabgriffs frei wählbar (z. B. über TAP), hinter Steuerung am geeignetsten
- Negativverhalten
  - Gezielte Manipulation
  - Man-in-the-middle nötig
  - Zusätzliche Jitter und Latenzen
- Nur ausgewählte Testfälle: Automatisierung bisher manueller Prozesse – keine Abdeckung der gesamten Spezifikation
- Netzwerk-Tests ergänzend zu applikativen Tests

ifak



Automatisiertes Testen von verteilten Systemen über Petrinetze

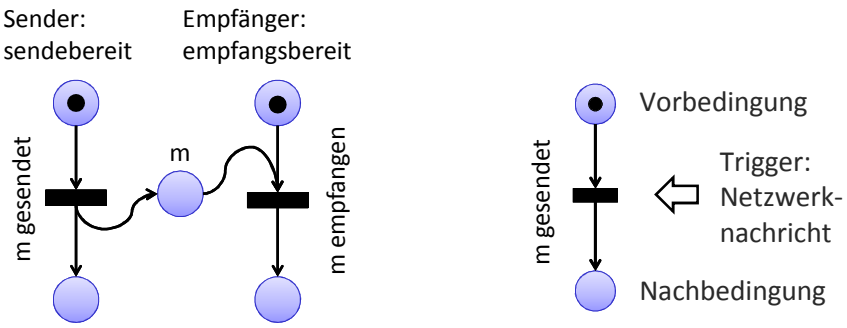
Bei Fragen – einfach fragen!

ifak

Grafiken: http://www.flaticon.com

Zusatz-Folien

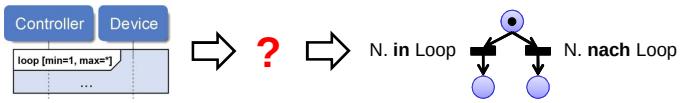
Modellierung von Kommunikationspartnern?



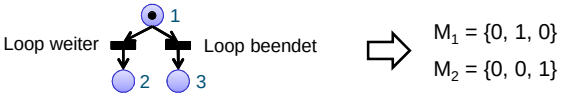
Problem: Beobachtbarkeit
des Empfangsereignisses

Post-Event-Entscheidbarkeit

- Bisher: **Entscheidbarkeit durch Netzwerkereignisse** (z. B. Loop beenden, Alternative entscheiden etc.)



- Durch zusätzliche Stellen & Transitionen nicht mehr (einfach) möglich
- Idee: **Mehrere, gleichzeitig aktive Markierungen**
- Dann erzeugen, wenn zwei Transitionen feuern könnten

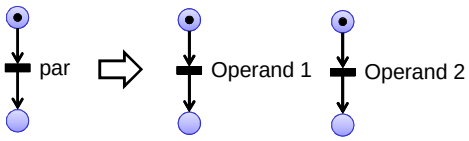


- Dann M_1 UND M_2 gegen Netzwerk-Nachrichten prüfen
- Eine M. „gewinnt“, sobald mit ihr weitere N.-Transition feuert

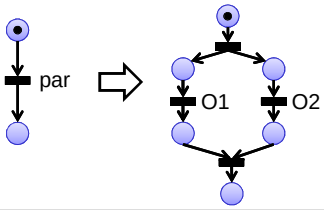
ifak

Überführung Sequenzen - Netze

- Anfangs:
  - Transitionen modellieren ausschließlich Netzwerk-Nachrichten
  - Dadurch minimale Zwischenstellen
  - Dadurch Fragmente in Fragmenten nie klar getrennt (eines beeinflusst ggf. das andere)



- Jetzt: Klare Abgrenzung: Zwischenstellen beginnen und beenden Fragmente



ifak