– A Case Study [9] –

A fairy tale

(pictures from Matteo Garrone's "Pinocchio")

Tell them...

A fairy tale

(pictures from Matteo Garrone's "Pinocchio")

and they'll behave

A fairy tale

(pictures from Matteo Garrone's "Pinocchio")

unless they don't

A fairy tale

(pictures from Matteo Garrone's "Pinocchio")

So, keep an eye on'em

A fairy tale

(pictures from Matteo Garrone's "Pinocchio")

of course, it's for their own good

At a glance

API-based development

- difficult in theory...
- ...and in practice

Behavioural specifications (of API)

- help in
  - documenting
  - monitoring
- Case study: OpenDXL

Open Data Exchange Layer

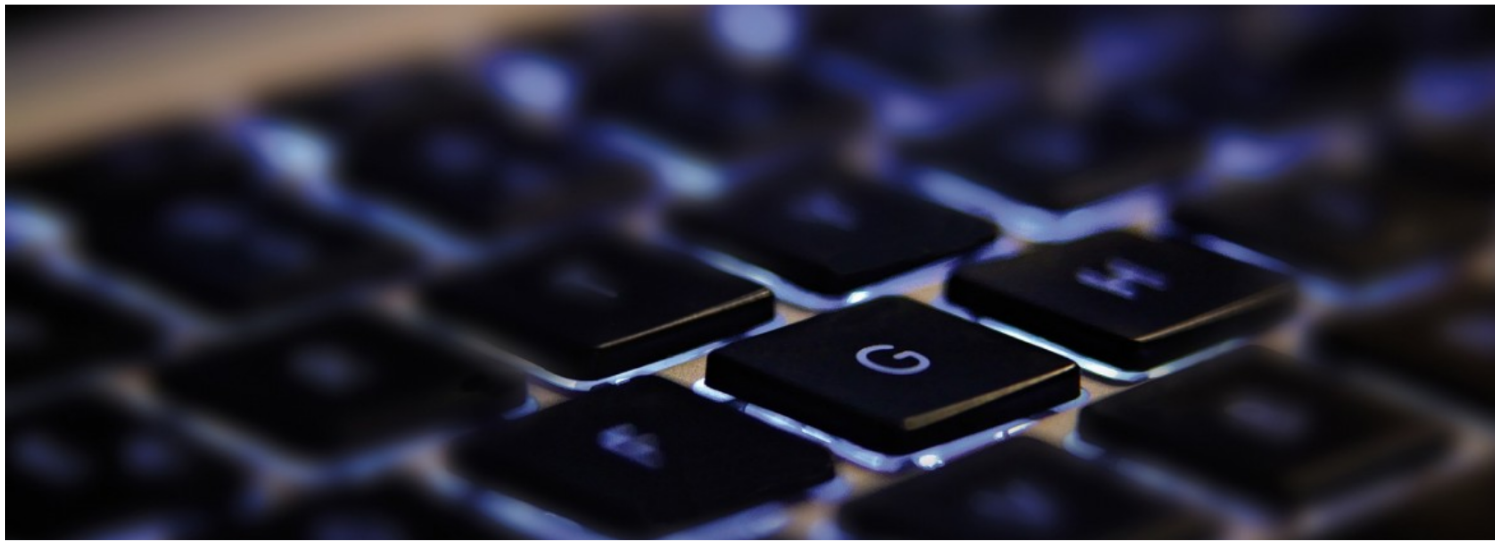
www.opendxl.com/index.php?article/11-introduction-to-opendxl/

Import bookmarks... gssi QuillBot Most Visited VARIOUS ecas haskell linux EU Research&Innovatio... Elsevier CS special issue... ISLP_w

Solutions Forum Articles Learn Develop

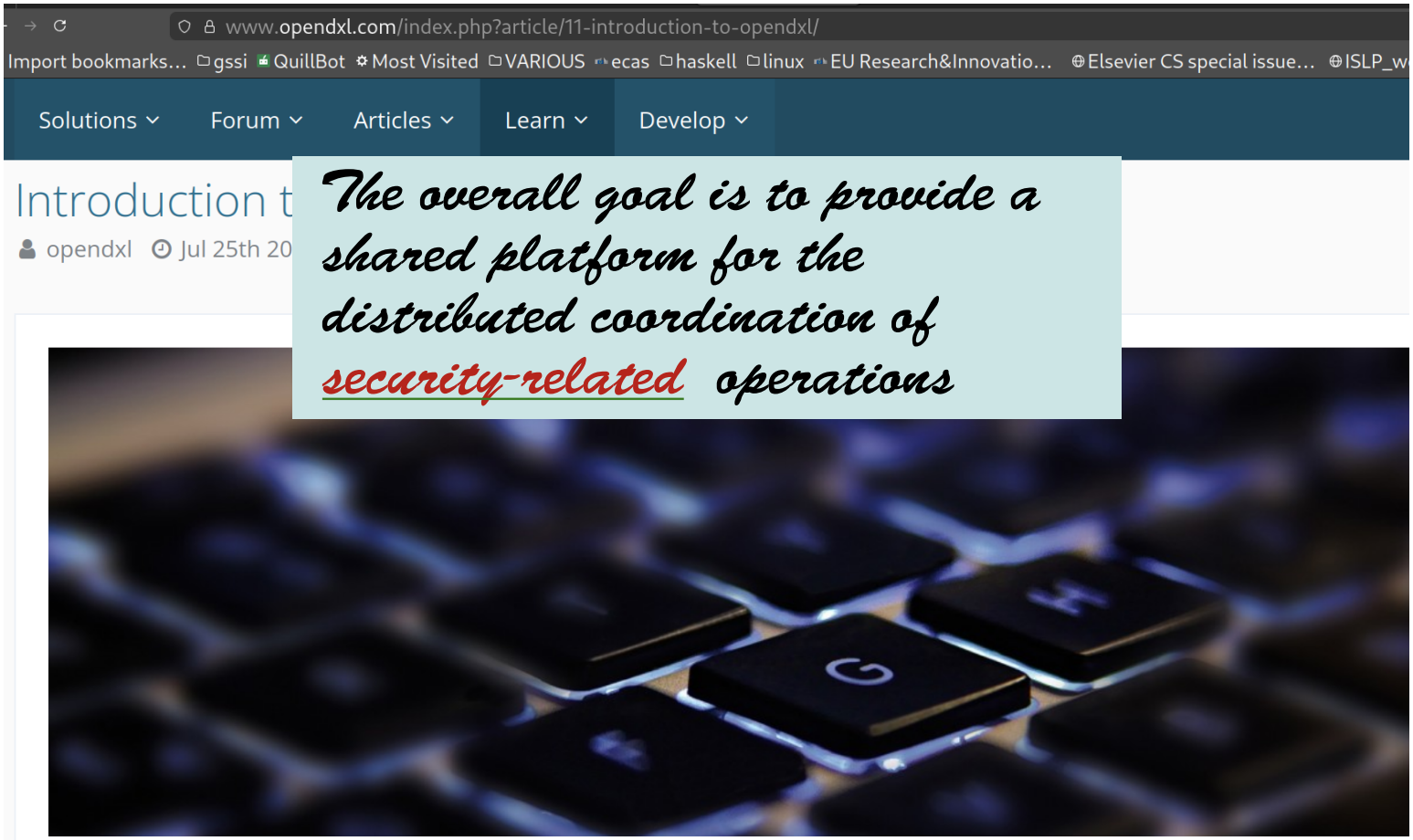
Introduction to OpenDXL

opendxl Jul 25th 2017 0 Comments 46,646 Views +9

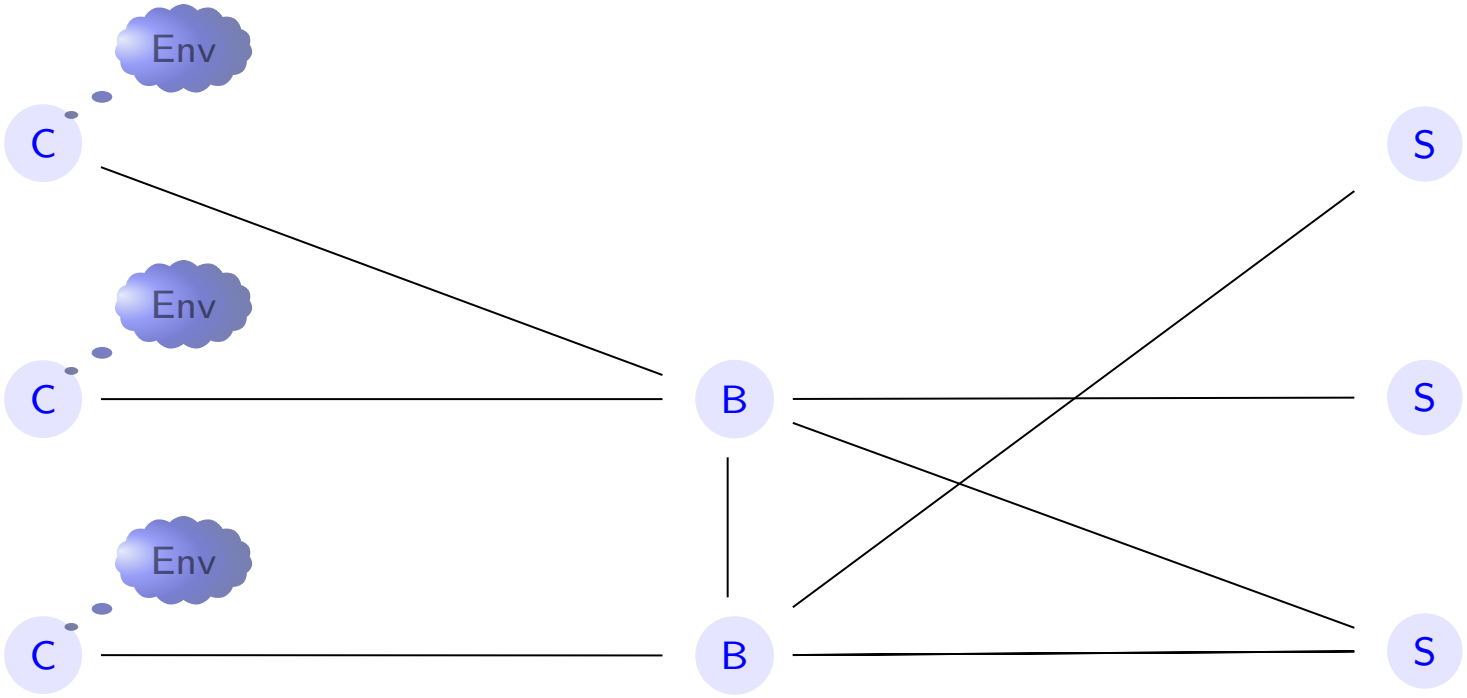


The goal of the Open Data Exchange Layer (OpenDXL) is to enable security devices to share intelligence and orchestrate security operations in real time.

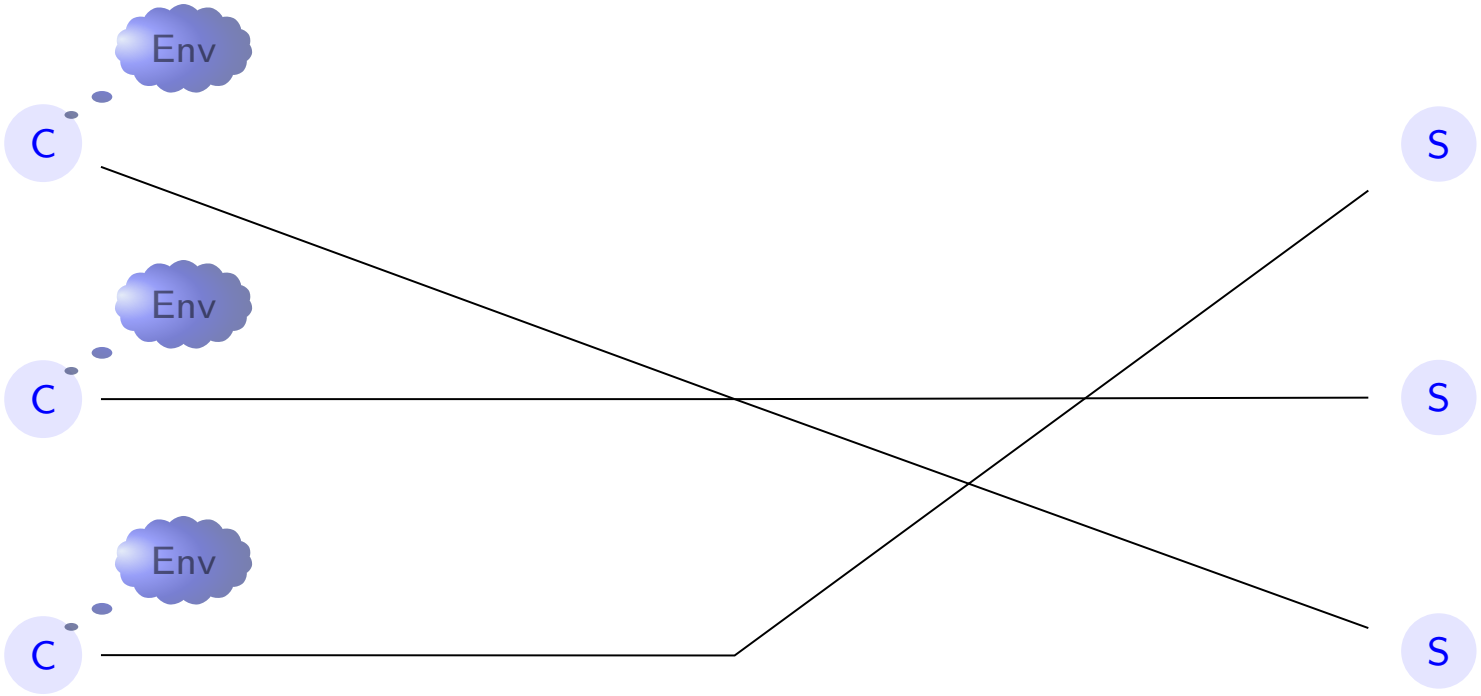
Open Data Exchange Layer



Architecture of OpenDXL



Architecture of OpenDXL

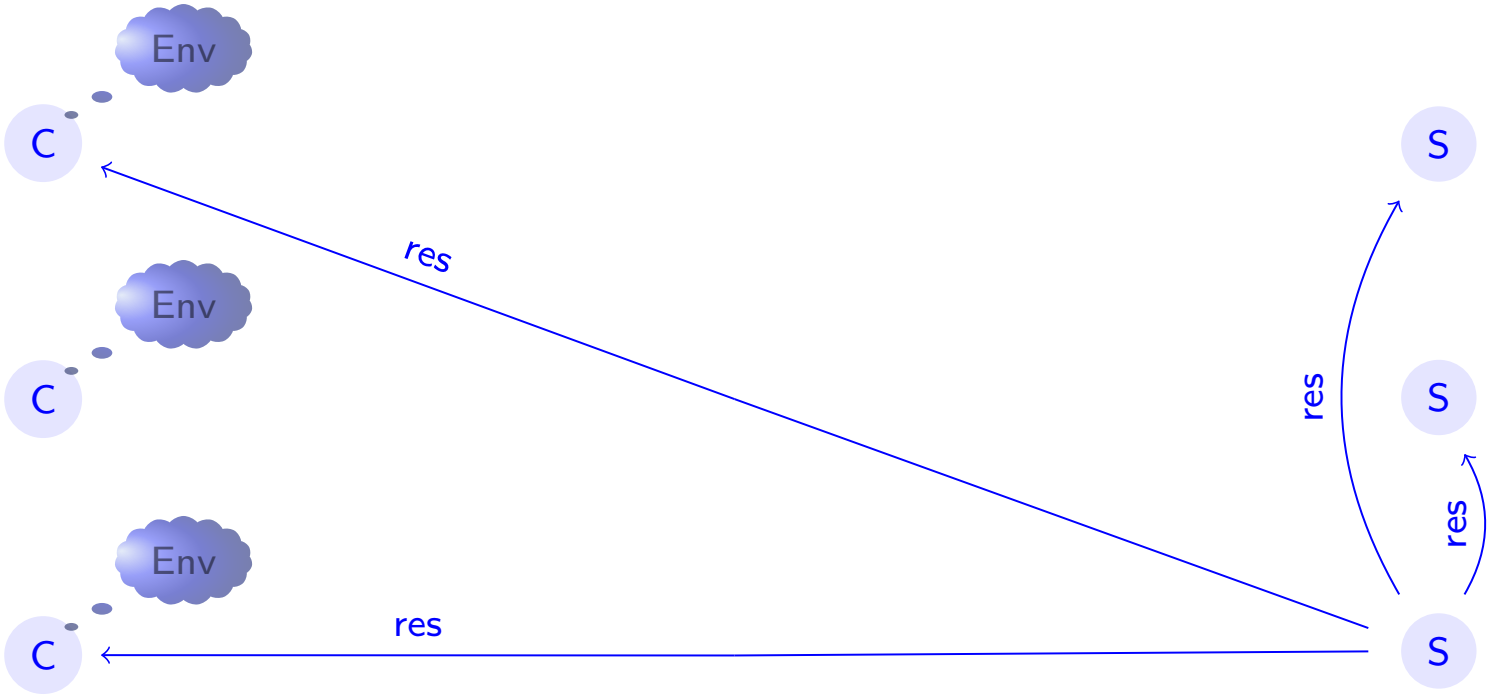


- Brokers abstracted away

Architecture of OpenDXL

- Brokers abstracted away
- Event-based communication

Architecture of OpenDXL



- Brokers abstracted away
- Event-based communication
- Application-level protocols are necessary

Threat Intelligence Exchange service

Threat Intelligence Exchange service

- OpenDXL's nature is service-oriented

Threat Intelligence Exchange service

- OpenDXL's nature is service-oriented

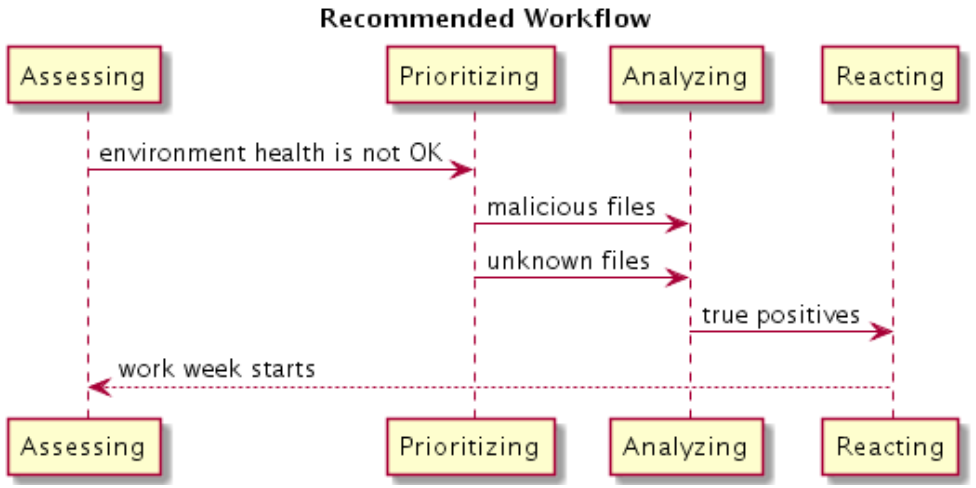
TIE's features

coordination of activities involving

- assessment of the security threats
  of configuration files, certificates, unsigned or unknown files, etc.
- prioritisation of analysis steps
  focusing on malicious or unknown files
- customisation of security queries
  based on reputation-based data such as product or company names
- reaction to suspicious indicators

Documenting TIE

Semi-formal diagrams



Verbal recommendations

a client "must have permission to send messages to the `/mcafee/service/tie/reputation/set` topic"

Sounds good...in theory

In practice

- "Stuff" developed @McAfee works fine
  - McAfee provides the service
  - and clients
- but it's a SOA: 3rd-party clients misbehave sometimes
- hence, defensive programming of TIE services

Sounds good...in theory

In practice

- "Stuff" developed @McAfee works fine
  - McAfee provides the service
  - and clients
- but it's a SOA: 3rd-party clients misbehave sometimes
- hence, defensive programming of TIE services

Caveat

3rd-party code may not be available for analysis
Hence, post-mortem analysis of execution logs to identify misbehaviour and communicate it to 3rd-parties

Idea

Adding more precision:

1. draw the protocol (global choreography)
2. turn the "drawing" into a behavioural type
3. project to component specs (local types)
4. turn local specs into state machines

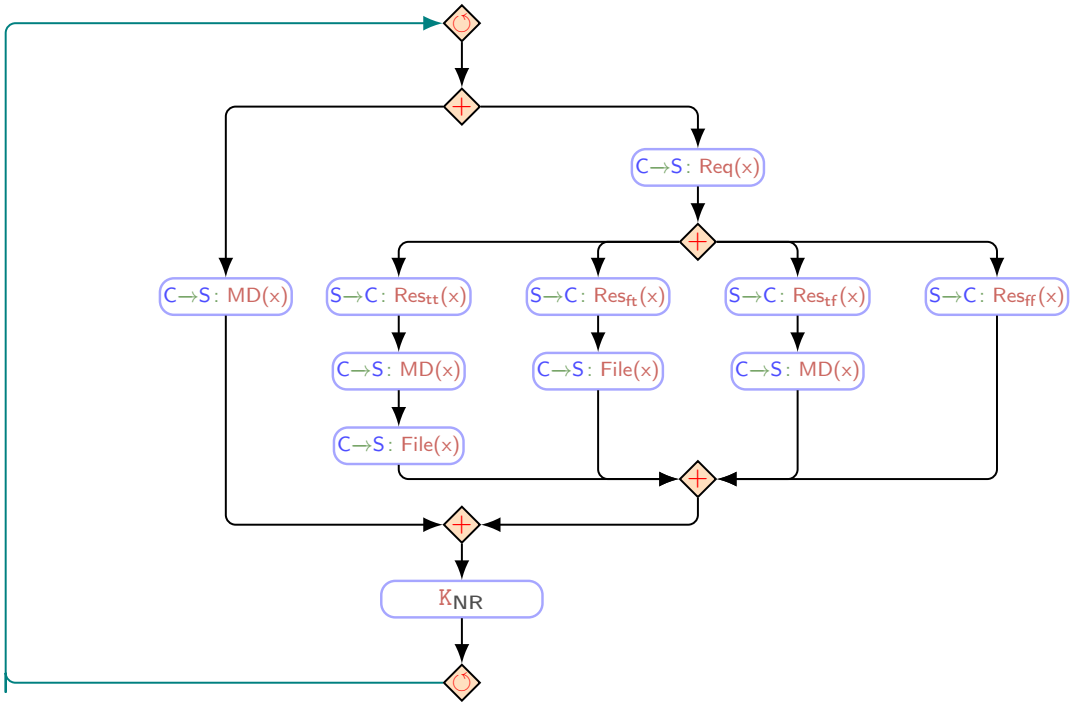
Advantages

- global choreographies: formal & precise, yet intuitive
- algorithmically generate monitors
- enhance "program comprehension"

Let's tie our TIE

No Greek letters or esoteric symbols were used in the making of this global view

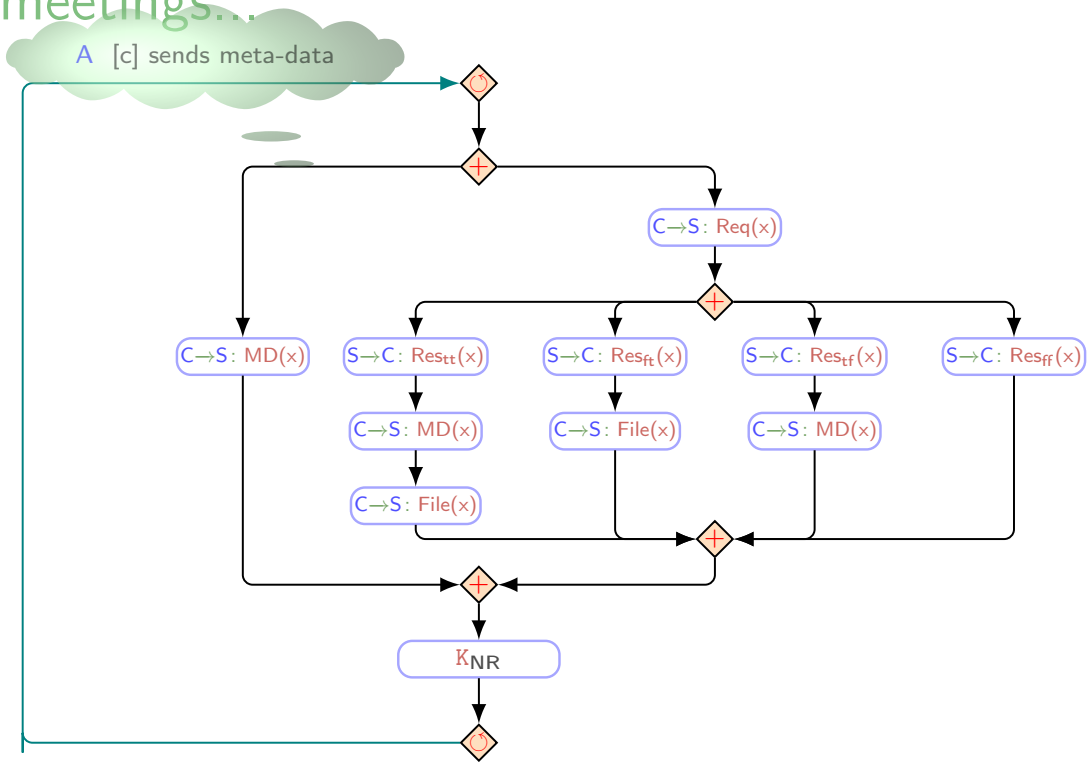
After a couple of meetings...



Let's tie our TIE

No Greek letters or esoteric symbols were used in the making of this global view

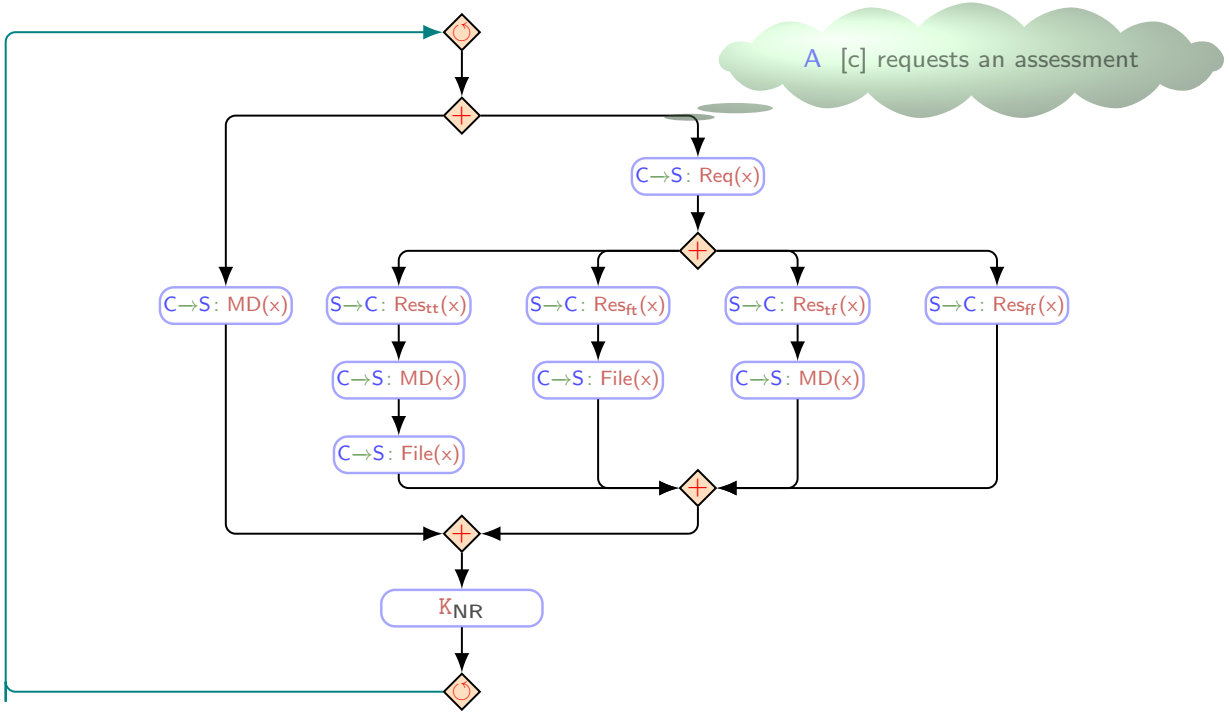
After a couple of meetings...



Let's tie our TIE

No Greek letters or esoteric symbols were used in the making of this global view

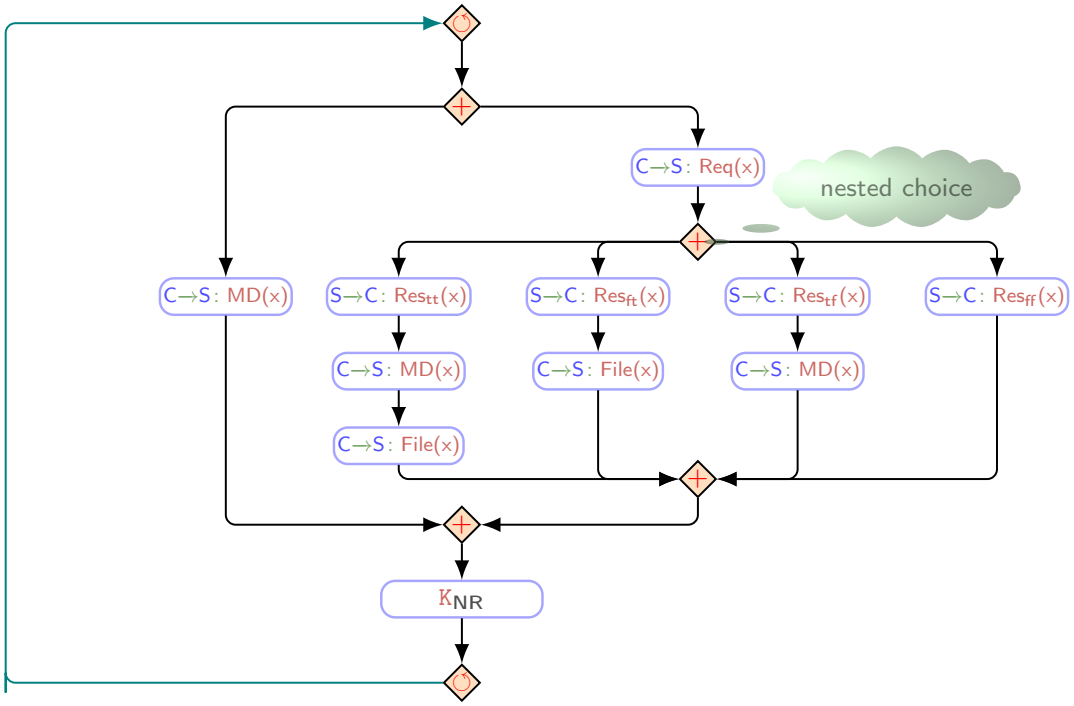
After a couple of meetings...



Let's tie our TIE

No Greek letters or esoteric symbols were used in the making of this global view

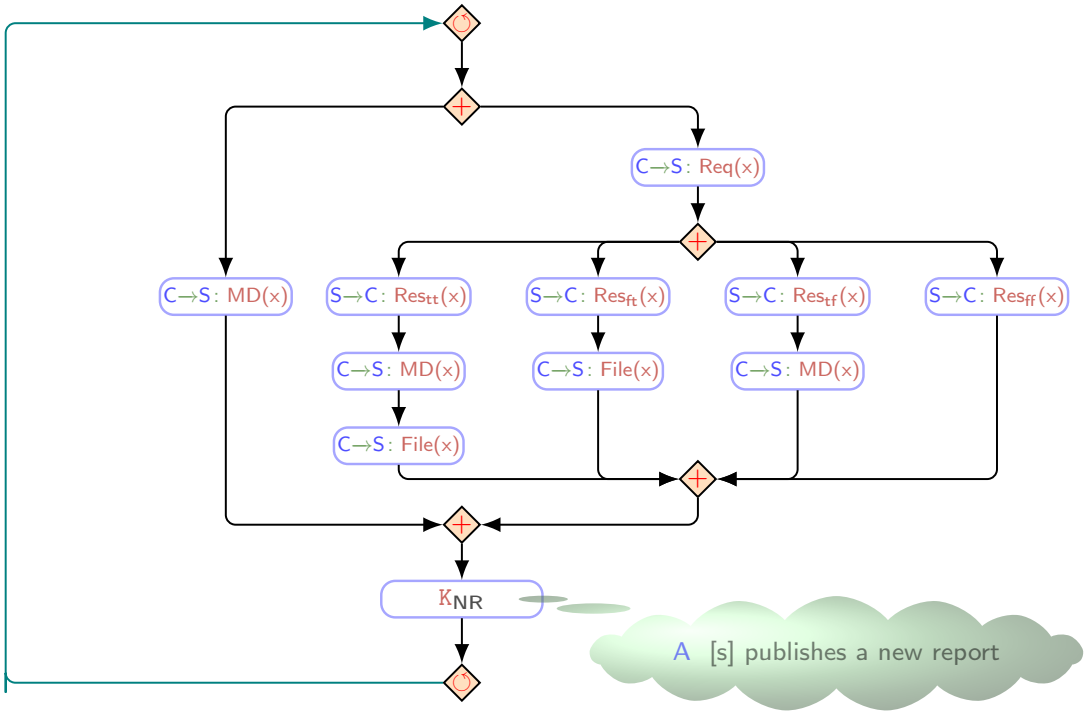
After a couple of meetings...



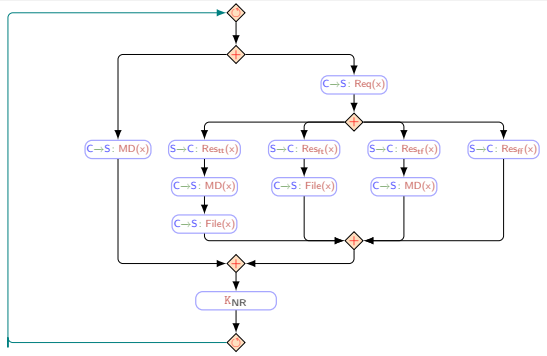
Let's tie our TIE

No Greek letters or esoteric symbols were used in the making of this global view

After a couple of meetings...



TIE...formally



$$\mathtt{K}_{\mathsf{TIE}} \triangleq \mu_{\mathsf{C}}\, X.\mathtt{K}_{\mathsf{Body}} \prec X$$

$$\mathtt{K}_{\mathsf{Body}} \triangleq (\mathtt{K}_{\mathsf{MD}}(\mathtt{x},\ell) + \mathtt{K}_{\mathsf{REQ}}(\mathtt{x},\ell)) \prec \mathtt{K}_{\mathsf{NR}}(\mathtt{x})$$

$$\mathtt{K}_{\mathsf{MD}}(\mathtt{x},\ell) \triangleq \mathsf{C} \to \mathsf{S} : (\mathtt{MD} \cdot \nu \mathtt{x} : \mathtt{Dgt} \cdot \nu \ell : \mathtt{loc})\,@\,\mathtt{b}$$

$$\mathtt{K}_{\mathsf{req}}(\mathtt{x},\ell) \triangleq \mathsf{C} \to \mathsf{S} : (\mathtt{Req} \cdot \nu \mathtt{x} : \mathtt{Dgt} \cdot \nu \ell : \mathtt{loc})\,@\,\mathtt{b}\,.\,\mathtt{K}_{\mathsf{info}}(\mathtt{x},\ell)$$

$$\begin{aligned}\mathtt{K}_{\mathsf{info}}(\mathtt{x},\ell) \triangleq\ & \mathsf{S} \to \mathsf{C} : (\mathtt{Res}_{\mathtt{tt}} \cdot \mathtt{x} : \mathtt{Dgt})\,@\,\ell\,.\,\mathtt{K}_{\mathtt{tt}}(\mathtt{x},\ell) \\ +\ & \mathsf{S} \to \mathsf{C} : (\mathtt{Res}_{\mathtt{tf}} \cdot \mathtt{x} : \mathtt{Dgt})\,@\,\ell\,.\,\mathtt{K}_{\mathtt{tf}}(\mathtt{x},\ell) \\ +\ & \mathsf{S} \to \mathsf{C} : (\mathtt{Res}_{\mathtt{ft}} \cdot \mathtt{x} : \mathtt{Dgt})\,@\,\ell\,.\,\mathtt{K}_{\mathtt{ft}}(\mathtt{x},\ell) \\ +\ & \mathsf{S} \to \mathsf{C} : (\mathtt{Res}_{\mathtt{ff}} \cdot \mathtt{x} : \mathtt{Dgt})\,@\,\ell\end{aligned}$$

Control vs Data

Behavioural types

Suitable devices for specification and analysis

- focus on control (mostly)
- assume point-to-point channels

Control vs Data

Behavioural types

Suitable devices for specification and analysis

- focus on control (mostly)
- assume point-to-point channels

VS

Klaimographies

Behavioural types with

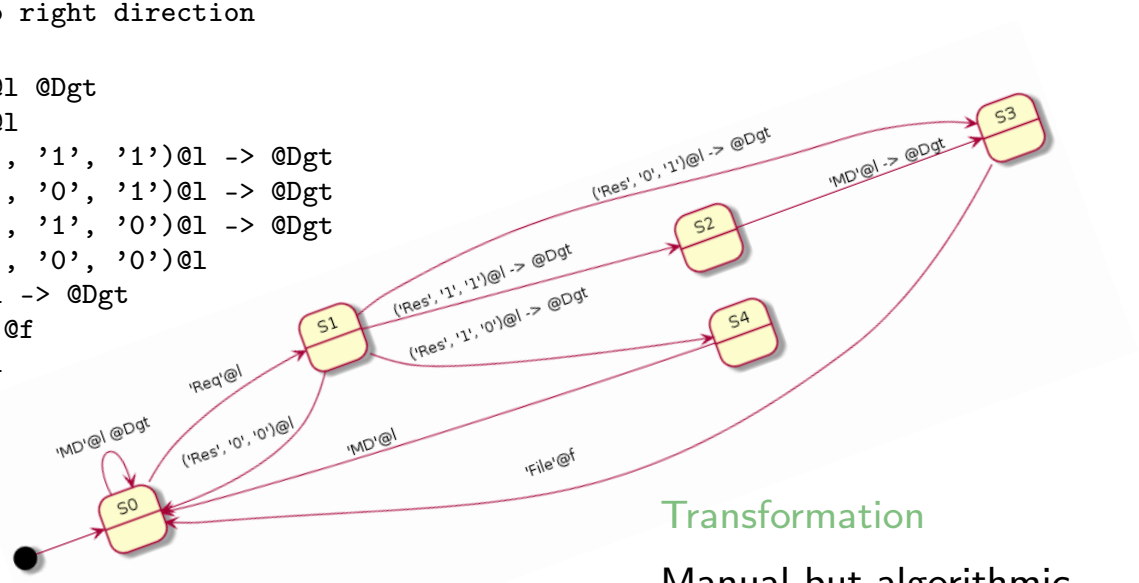
- focus on data (mostly)
- interactions based on generative communication
- unit & multi-roles

Monitors from projections

UML-like

```
@startuml left to right direction
[*] -> S0
S0 -> S0:  ’MD’ @l @Dgt
S0 -> S1:  ’Req’@l
S1 -> S2:  (’Res’, ’1’, ’1’)@l -> @Dgt
S1 -> S3:  (’Res’, ’0’, ’1’)@l -> @Dgt
S1 -> S4:  (’Res’, ’1’, ’0’)@l -> @Dgt
S1 -> S0:  (’Res’, ’0’, ’0’)@l
S2 -> S3:  ’MD’@l -> @Dgt
S3 -> S0:  ’File’@f
S4 -> S0:  ’MD’@l

@enduml
```



Transformation

Manual but algorithmic

- '@...' for data-flow analysis
- huge logs (can't fit memory)

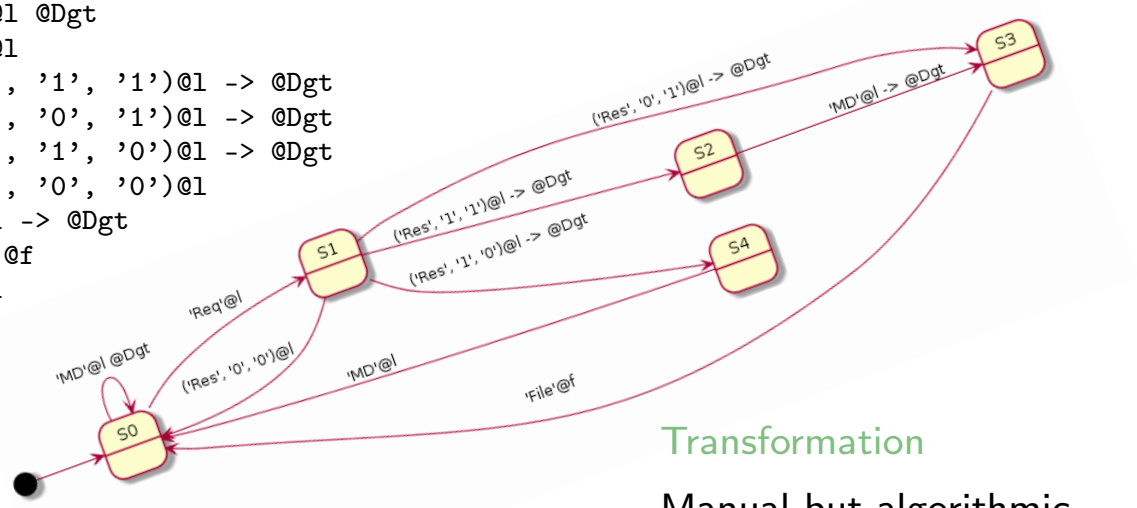
Monitor checks expected data correspondences (e.g., 'file' corresponds to a 'file req')

Monitors from projections

UML-like

```
@startuml left to right direction
[*] -> S0
S0 -> S0:  'MD' @l @Dgt
S0 -> S1:  'Req'@l
S1 -> S2:  ('Res', '1', '1')@l -> @Dgt
S1 -> S3:  ('Res', '0', '1')@l -> @Dgt
S1 -> S4:  ('Res', '1', '0')@l -> @Dgt
S1 -> S0:  ('Res', '0', '0')@l
S2 -> S3:  'MD'@l -> @Dgt
S3 -> S0:  'File'@f
S4 -> S0:  'MD'@l

@enduml
```



Violations found:
- file sent without prior reqs
- further info reqs, not honoured

Transformation

Manual but algorithmic

- '@...' for data-flow analysis
- huge logs (can't fit memory)

Monitor checks expected data correspondences (e.g., 'file' corresponds to a 'file req')

Thou shalt be abstractly precise

Effectiveness & Reproducibility

(meta-)communication with practitioners has to be smooth and amenable to be "re-played"

- non-deterministic & visual abstractions & handwaving even
  - facilitate communication
  - provide insights & "inspirations"
- but semantics is necessary
  - to attain precision
  - to change mind

Thou shalt strive for generality

How tight to TIE are we?

- klaimographies were not designed for OpenDXL
- a reviewer noted: “event-based middleware are becoming the norm”
- choreography can go bottom-up (as noted by another reviewer)

Thou shalt use appropriate tools

Other FM?

Sure, but ...

- Model checking
  but it is not easy for lay-users to express properties in some temporal logic
- Other behavioural types
  often too irksome for non-experts
- Other FM (Petri nets, event structures,...)
  too low level (and (sadly) not much studied anymore)

References I

[1] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*.
MIT Press, Cambridge, MA, USA, 1986.

[2] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines.
JACM, 30(2):323–342, 1983.

[3] Alex Coto, Roberto Guanciale, and Emilio Tuosto. An abstract framework for choreographic testing.
In *ICE*, volume abs/2009.07990, 2020.

[4] Alex Coto, Roberto Guanciale, and Emilio Tuosto. Choreographic development of message-passing applications - A tutorial.
In Simon Bliudze and Laura Bocchi, editors, *Coordination Models and Languages - 22nd IFIP WG 6.1 International Conference, COORDINATION 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15-19, 2020, Proceedings*, volume 12134 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2020.

[5] Alex Coto, Roberto Guanciale, and Emilio Tuosto. An abstract framework for choreographic testing.
Journal of Logic and Algebraic Methods in Programming, 123:100712, 2021. Extended version of [3].

[6] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata.
In *ESOP 2012*, pages 194–213, 2012.

[7] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process.
J. ACM, 32(2):374–382, apr 1985.

[8] Cedric Fournet and George Gonthier. The reflexive CHAM and the join-calculus.
In *Conference Record of POPL '96: The* 23^{rd} *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 372–385, St. Petersburg Beach, Florida, January 1996.

References II

[9] Leonardo Frittelli, Facundo Maldonado, C. Hernán Melgratti, and Emilio Tuosto. A Choreography-Driven Approach to APIs: the OpenDXL Case Study.
In *COORDINATION*, volume 12134 of *LNCS*. Springer, June 2020.

[10] Roberto Guanciale and Emilio Tuosto. An abstract semantics of the global view of choreographies.
In *Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016.*, pages 67–82, 2016.

[11] Roberto Guanciale and Emilio Tuosto. Realisability of pomsets.
Journal of Logic and Algebraic Methods in Programming, 108:69–89, 2019.

[12] Roberto Guanciale and Emilio Tuosto. Pomcho: a tool chain for choreographic design.
Science of Computer Programming, 202:102535, 2021.

[13] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Fair join pattern matching for actors.
In Jonathan Aldrich and Guido Salvaneschi, editors, *38th European Conference on Object-Oriented Programming, ECOOP 2024, Vienna, Austria, September 16-20, 2024*, volume 313 of *LIPIcs*, pages 17:1–17:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[14] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Fair join pattern matching for actors (artifact).
Dagstuhl Artifacts Ser., 10(2):8:1–8:3, 2024.

[15] Carl Hewitt, Peter Boehler Bishop, and Richard Steiger. A Universal Modular ACTOR Formalism for Artificial Intelligence.
In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, USA, August 20-23, 1973*, pages 235–245. William Kaufmann, 1973.

[16] Ayman Hussein, Philipp Haller, Ioannis Karras, Hernán C. Melgratti, Alceste Scalas, and Emilio Tuosto. Joinactors: A modular library for actors with join patterns.
Art Sci. Eng. Program., 11(1), 2026.

References III

[17] Nickolas Kavantzas, Davide Burdett, Gregory Ritzinger, Tony Fletcher, and Yves Lafon. Web services choreography description language version 1.0.
http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217. Working Draft 17 December 2004.

[18] Roland Kuhn and Alan Darmasaputra. Behaviorally typed state machines in typescript for heterogeneous swarms.
In René Just and Gordon Fraser, editors, *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, pages 1475–1478. ACM, 2023.

[19] Roland Kuhn, Hernán C. Melgratti, and Emilio Tuosto. Behavioural types for local-first software.
In Karim Ali and Guido Salvaneschi, editors, *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States*, volume 263 of *LIPIcs*, pages 15:1–15:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[20] Roland Kuhn, Hernán C. Melgratti, and Emilio Tuosto. Behavioural types for local-first software (artifact).
Dagstuhl Artifacts Ser., 9(2):14:1–14:5, 2023.

[21] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From Communicating Machines to Graphical Choreographies.
In *POPL15*, pages 221–232, 2015.

[22] Hubert Plociniczak and Susan Eisenbach. Jerlang: Erlang with joins.
In *Coordination Models and Languages: 12th International Conference, COORDINATION 2010, Amsterdam, The Netherlands, June 7-9, 2010. Proceedings 12*, pages 61–75. Springer, 2010.

[23] Carlos G. Lopez Pombo, Agustín E. Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
In Erika Ábrahám, Clemens Dubslaff, and Silvia Lizeth Tapia Tarifa, editors, *Theoretical Aspects of Computing – ICTAC 2023*, pages 119–138. Springer, 2023.

References IV

[24] Carlos López Pombo, Agustín E. Martinez Suñé, and Emilio Tuosto. A dynamic temporal logic for quality of service in choreographic models.
TCS, 1043:115247, 2025.

[25] John A. Trono. A new exercise in concurrency.
SIGCSE Bull., 26(3):8–10, September 1994.

[26] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies.
Journal of Logic and Algebraic Methods in Programming, 95:17–40, 2018. Revised and extended version of [10]. available at http://www.cs.le.ac.uk/people/et52/jlamp-with-proofs.pdf.