Describing Software Architectures
Revisiting The Early Decisions

XCD
Modular, Reusable, Realizable Architectures

Christos Kloukinas

Table of Contents

Software Architecture: Early Days — The Foundations

- "Foundations for the study of software architecture" (1992) — Dewayne Perry, and Alexander Wolf

Goal:

Specify complex distributed software systems

The Connector Wars

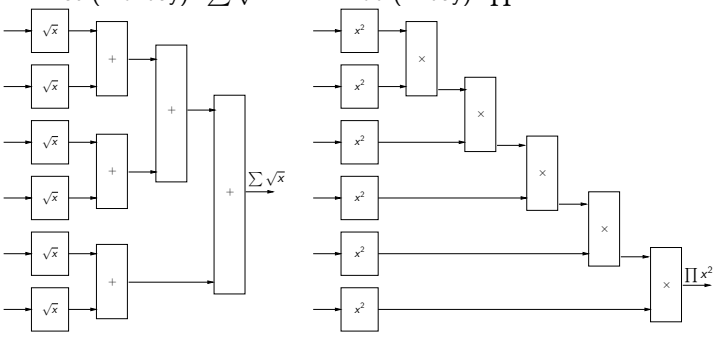
- Components as (potentially) *active actors* *Agreement!*
- *And...*
  - Components & Connect*ors* *Big Endians*
    "We *need* connectors (protocols) to describe systems"
    - Wright (1997) — Carnegie Mellon
    - CSP — Robert Allen, and David Garlan
  - Components & Connect*ions* *Little Endians*
    "All systems *can be* described with components alone, so connectors are *not needed*"
    - Darwin (1994) — Imperial College
    - π-Calculus ('94) — Jeff Magee, Susan Eisenbach, and Jeff Kramer
    - FSP ('97) — Jeff Kramer, and Jeff Magee
- Others since — components as *passive library functions*, connectors as *controllers*:
  - BIP (2005) — Joseph Sifakis *et al.*
  - Exogenous Connectors (2005) — Kung-Kiu Lau *et al.*

The Connector Wars — Personal View

- A *total* waste of very smart people's time... :-(
  1. "We *need* connectors (protocols) to describe systems"
  2. "All systems *can be* described with components alone,
  3. so connectors are *not needed*"
- 2 out of 3 are correct
  - Programs can be described without procedure calls (Turing Machine)
  - But we still need procedure calls to actually program...

No Connectors?

Alice (Monday): $\sum \sqrt{x}$ Bob (Friday): $\prod x^2$



- Yes, but what about map-reduce? `(reduce R (map M x))`
- Why re-invent the wheel each time?
  - We'll not get it right/optimal each time
- Define it once — Reuse-by-Call not Reuse-by-Copy (or Reinvent...)

Connectors — Advertised Pros

- Specify protocols *once*, reuse them by *applying* them
- Components become *protocol-agnostic*
- Increase *reusability* of components & connectors
  (resistors + sequential/parallel connector)
  (components + X/Y/Z replication protocol)
- Modularity makes specification *easier*
  - Less effort ⇝ more specs ⇝ less *architectural mismatch*
  - "Architectural Mismatch: Why Reuse Is So Hard"(1995) — David Garlan, Robert Allen, and John Ockerbloom
  - "Architectural Mismatch: Why Reuse Is *Still* So Hard" (2009) — D. Garlan, R. Allen and J. Ockerbloom
- Hopefully easier to *verify* each independently
- Hopefully easier to *understand* the system structure

Table of Contents

Connectors — Not So Modular/Reusable...

- All ADLs supporting connectors follow Wright but ...
- Wright Connector: $Glue \parallel n_1{:}R_1 \parallel \cdots \parallel n_k{:}R_n$
- Wright System: $Glue \parallel n_1{:}P_1 \parallel \cdots \parallel n_k{:}P_n$
  - Roles (R_i) *disappear* in the system
  - Ports (P_i) must be *compatible* with respective Roles
    - So ports need to *repeat* role specifications... :-(
    - Writing a protocol role once should be enough
  - *"Solution"*: Move role constraints to the glue!
    So now role specs are essentially empty — an interface

The Glue

- Additional *global* interaction constraints
- How does one describe these?
  - *Local* views (*e.g.*, distributed algos)
  - *Global* view (ADLs, Session Types)
    - *State explosion* — *hard to describe this way*
      ⇒ *Complexity*
    - *Must split to implement...*
      ⇒ *Realizability?*

Complexity — Spin Examples State Space (spinroot.com)

Example	States	No Partial Reduction
abp.pml	12	12
calculator.pml	52	568
dtp.pml	44363	223512
eratosthenes.pml	2093	25295
for_example.pml	24	24
for_select_example.pml	176	176
hajek.pml	2201	5788
leader0.pml	97	15779
life.pml	66002	66002
loops.pml	15	15
manna_pnueli.pml	86	117
peterson.pml	40	55
rtos1.pml	11	11
sat.pml	14	14
snoopy.pml	1249	3591
sort.pml	135	107713
welfare.pml	53	53
werkplaats.pml	759	1600
	Not very promising...	*Must be God...*

The main problem is the relative increase — leader0.pml

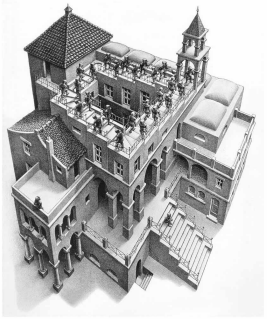
Implementing The Glue — *Need to Split...*

- Problem is *undecidable* in general:
  "Inference of message sequence charts" (2003) — R. Alur, K. Etessami, and M. Yannakakis
- So, what do?
  - Session Types
    "Multiparty asynchronous session types" (2008) — Kohei Honda, Nobuko Yoshida, and Marco Carbone
    `http://www.scribble.org/` `http://scribble.doc.ic.ac.uk/`
    - Pick *some* semantics for interaction
    - Find *some* subset that can be split, so that they behave *exactly* the same
    - Consider the rest to be *unrealizable*
  - ADLs:
    - Ignore the problem!
    - Promise a *decentralized* architecture
    - Provide a *glue* for it as solution
    - *Not show if/how that glue can be realized in a decentralized manner ?!?!?*

Requirements vs Architecture

Requirement "I want infinite stairs"

Architecture



M. C. Escher
(1898 – 1972)

- Architecture needs to be realizable
- *In a way that respects communication integrity*
  ⇒ No hidden interaction channels!
- Otherwise, performance/reliability/etc. analyses are invalid

Table of Contents

The XCD ADL: Main Ideas (http://staff.city.ac.uk/c.kloukinas/Xcd)

- Components as (potentially active) actors
- Support arbitrary connectors
  ⇒ Modular specifications
  Simpler component specifications *How?*
  ⇒ Reusable components & reusable connectors
- Only local constraints can be imposed
  ⇒ Realizable architectures + communication integrity *How?*
- Formal (extended Design-by-Contract approach — JML-inspired)
  ⇒ Uses SPIN to verify (among others):
  - Are provided services (local) interaction constraints satisfied?
  - Are provided services functional pre-conditions complete?
  - Are there (global) deadlocks?

Component Interaction Constraints

Cleanliness is next to Godliness...

Machine *A* | Machine *B*

Same interfaces but...

```
1 @interaction {
2     waits: ! washing; }
3 void open_door();
```

A blocks my call till it's safe

```
1 @interaction {
2     accepts: ! washing; }
3 void open_door();
```

B may reject it with undefined behaviour (might electrocute me!)

accepts Constraints

- Exist already in many places — *cf.* Java's RuntimeException
  https://docs.oracle.com/javase/7/docs/api/java/lang/RuntimeException.html
  "Direct Known Subclasses:
  AnnotationTypeMismatchException, ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException, DataBindingException, DOMException, EmptyStackException, EnumConstantNotPresentException, EventException, FileSystemAlreadyExistsException, FileSystemNotFoundException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException, IllegalStateException, IllformedLocaleException, ImagingOpException, IncompleteAnnotationException, IndexOutOfBoundsException, JMRuntimeException, LSException, MalformedParameterizedTypeException, MirroredTypesException, MissingResourceException, NegativeArraySizeException, NoSuchElementException, NoSuchMechanismException, NullPointerException, ProfileDataException, ProviderException, ProviderNotFoundException, RasterFormatException, RejectedExecutionException, SecurityException, SystemException, TypeConstraintException, TypeNotPresentException, UndeclaredThrowableException, UnknownEntityException, UnmodifiableSetException, UnsupportedOperationException, WebServiceException, WrongMethodTypeException"

Expected to crash the program

Interaction Constraints — Where do They Come From?

- A function can be:
  - *total* — no interaction constraint
    `// no @interaction`
    ```
    @functional { ensures:
     \result := (i = 0) ? 0 : ( (i<0) ? -1 : 1 ); }
    int sign(int i);
    ```
  - *partial* — accepts defines the *domain*
    ```
    @interaction { accepts: j!=0 && !(i=INT_MIN && j=-1); }
    @functional { ensures: \result := i/j; }
    int div(int i, int j);
    ```
  - And we also have *partial application*:
    ```
    @interaction { waits: B!=0 && !(a=INT_MIN && B=-1); }
    @functional { ensures: \result := a/B; }
    int divByB(int a); // B is a global/object var
    ```

Note:
- `waits` here is our choice
- `waits` can wait for ever if it's never satisfied
- `//divByB(a)=div(a, B);` B at which point in time?
- No guarantee in general on how long a call will take, so no guarantee either of which value of B will be used

Java Thread as an XCD Component

```
component Thread {
 bool started := false; // component data.
 bool died := false;

 aliveP() {return started && !died;} // helper function

 provided p { // all ports have their own thread
  @functional  {ensures: \result := aliveP();}
  bool isAlive();

  @interaction {waits: !aliveP();}
  void join();

  @interaction {accepts: ! started;}
  @functional  {ensures: started := true;}
  void start();
  // ... other methods
 };
};
```

?

- Constraints are more modular *(JML allows but doesn't enforce it)*
- Functional constraints must be complete! Call already accepted!!!

Software Components & DbC

- Software Components have:
  - Provided method ports ✔
  - Required method ports ✘
  - Consumer event ports ✘
  - Emitter event ports ✘
- DbC created for objects
  - No events
  - No required methods
  - Ignores clients' needs
- Separation of interaction/functional contracts

  Restaurant Provide a service from 7pm-11pm — Italian menu

  Client Require a service from 9pm-12pm — pizza or Peking duck
- Extension for required methods & event consumption/emission

```
provided port1 {// d(ata)
@interaction{waits/accepts:φ_I(d,p);}
@functional{

  requires: φ_R(d,p);
   ensures: φ_E(d',d,p);}
void service(p); };
// φ_I(d,p) ∧ φ_R(d,p) ∧ φ_E(d',d,p)
```

```
required port2 {// p(arams)
@interaction{waits:φ_I(d);}
@functional{
  promises: φ_P(p,d);

   ensures: φ_E(d',d,p);}
void service(p); };
// φ_I(d) ∧ φ_P(p,d) ∧ φ_E(d',d,p)
```

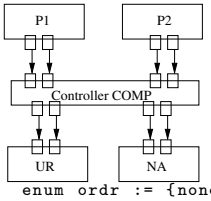

XCD Connector Structure & Roles

- XCD Connectors are collections of roles — no glue/global data
- XCD roles add extra data and port constraints to components
- Component port must have all role port actions

  Wright: Component ports *implement* role ports

  XCD: Component ports *interpret* role ports
- *Role's not a wrapper — it's a script to be interpreted*
  - Once a component is assigned a role
    - Role data are *added* to the component data
    - Role port method constraints are *added* to the component port method constraints
  - Wrapper components can do this for provided ports but not for *required* ports
- Now centralized solutions must be specified *explicitly*, with explicit centralized controller components

Alur's Plant — A Centralized XCD Connector




```
enum ordr := {none, incFirst,
              dblFirst};
role roleController {
  ordr order := none;
  bool p1_incNARcvd := false;
  bool p1_incURRcvd := false;
  bool ur_incUREmtd := false;
  bool na_incNAEmtd := false;
 all_received(){return
    p1_incURRcvd && p1_incNARcvd
 && p2_dblURRcvd && p2_dblNARcvd;}

provided port_variable P1toUR {
  @interaction{
  waits: !p1_incURRcvd;
  ensures: p1_incURRcvd :=true;
    order := pre(order) == none
      ? incFirst : pre(order); }
  void incUR(); }
```

```
provided port_variable P1toNA {
  @interaction{
  waits:  !p1_incNARcvd;
  ensures: p1_incNARcvd :=true; }
  void incNA(); }
required port_variable CtoURinc {
  @interaction{
  waits: all_received()
    && !ur_incUREmtd
    && ( (order==incFirst)
      || (order==dblFirst
          && dbl_emitted()) );
  ensures: ur_incUREmtd := true; }
  void incUR(); }
required port_variable CtoNAinc {
  @interaction{
  waits: ur_incUREmtd && !na_incNAEmtd;
  ensures: // clear flags if dblFirst
    p1_incURRcvd
    := !(pre(order) == dblFirst);
    ...
  ur_dblUREmtd
    := pre(order) == dblFirst
       ? false : pre(ur_dblUREmtd);
  na_dblNAEmtd
    := pre(order) == dblFirst
       ? false : pre(na_dblNAEmtd);
  order
    := pre(order) == dblFirst
```

Evaluation

Case Study	Issues	State-vector (Bytes)	States Stored	States Matched	Memory (MB)	Time (sec)
Decentralized Nuclear Plant	glue	240	137	73	130	0.00
Centralized Nuclear Plant		424	168349	407776	186	1.21
Lunar Lander v. 1	Ovrflw	372	118	78	131	0.01
Lunar Lander v. 2		392	4223125	8072166	3793	15.50
Gas Station (1 customer)		188	1003	1401	130	0.00
Gas Station (2 customers)		288	1136214	2793961	382	3.23
Gas Station (3 customers)		368	25056808	89254880	7024†	78.00
BITSTATE (3 customers)		368	62792292	207452380	24	242.00
BITSTATE (4 customers)		456	66989014	289982810	25	321.00
BITSTATE (5 customers)		544	69607515	356984080	26	365.00
Aegis v. 1	L-DDLCK	620	13834057	71301546	7024†	52.00
BITSTATE v. 1	L-DDLCK	620	64408848	266469200	37	330.00
BITSTATE v. 2		548	63568962	268078040	35	304.00
English auction v. 1 (1 part.)	DDLCK, Ovrflw	140	296	295	130	0.00
English auction v. 2 (1 part.)	Ovrflw	144	776	1642	130	0.00
English auction v. 2 (2 part.)	Ovrflw	232	1293488	3732650	367	5.00
English auction v. 2 (3 part.)	Ovrflw	312	27315867	96797687	7024†	134.00
BITSTATE v. 2 (3 part.)	Ovrflw	312	57105380	189090640	20	310.00

"States Stored" unique global states.
"States Matched" states revisited during the search.
† run out of memory (7024 MB).

All case studies available at the XCD web site

Table of Contents

Summary

XCD — a new ADL
`http://staff.city.ac.uk/c.kloukinas/Xcd`

- Support for connectors
  - Arbitrary, complex connectors
  - Always *realizable*
  - *Modular* — Interaction constraints are transferred from Connectors to Components
- Formal
  - Architectures can be model-checked
  - Reasonable results for a number of classic case studies
- A less steep learning curve than classic ADLs (?)
  - Closer to a programming language than process algebras
  - Extended DbC

- All problems solved? Far from it!

Future Work (`http://staff.city.ac.uk/c.kloukinas/Xcd`)

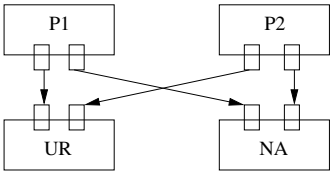
- Formal semantics using the $\mathbb{K}$ Framework
  ("An Overview of the $\mathbb{K}$ Semantic Framework" (2010) — Grigore Rosu, and Traian Florin Serbanuta)
  - Production of analyser automatically
  - Facilitate language extensions (recursive connectors, comp/port arrays, *etc.*)
  - Facilitate support for different event queue policies
  - Facilitate translation to code
- Theory
  - Verification of components
    - Construct testing environments to complete a sub-system
  - Compute interface of a composite component: Controller synthesis
  - Is interface A a refinement of B?
- Extend
  - Timed, Probabilistic automata

Thank You!

Table of Contents

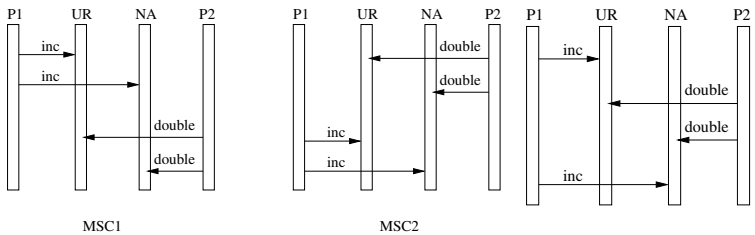
Realizability Problem — Not A Problem?

Unrealizable Architecture — A Nuclear Plant



(a) A decentralized architecture

[AEY03] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. IEEE TSE, 29(7):623-633, 2003



(b) The plant's (unrealizable) MSCs

(c) An unavoidable bad behaviour

Unrealizable Wright Connector

```
1 connector Plant_Connector =
2 role P1 = ur → na → P1. // increase both
3 role P2 = ur → na → P2. //   double both
4 role UR = inc → UR □ double → UR.
5 role NA = inc → NA □ double → NA.
6 glue G = P1.ur→UR.inc → P1.na→NA.inc
7         → P2.ur→UR.double → P2.na→NA.double
8         → G
9       □ P2.ur→UR.double → P2.na→NA.double
10        → P1.ur→UR.inc → P1.na→NA.inc
11        → G. // → link, → global constraint
12 SYSTEM = (P1:P1 ‖ P2:P2 ‖ UR:UR ‖ NA:NA ‖ G).
```

Wright's *(unrealizable)* connector for the nuclear plant

- Property is verified! But no way to realize it. . . :-(
- Architect needs to be told requirement isn't satisfied
- *Global constraints are requirements*
- Architecture shouldn't simply repeat the requirements

Realizable Plant

```
connector Realizable_Plant_Connector =
role P1 = ur → na → P1.                    // same
role P2 = ur → na → P2.                    // same
role UR = inc → UR □ double → UR. // same
role NA = inc → NA □ double → NA. // same
glue G = (G1 || G2 || G3 || G4).// Real glue
// where Gi's are simple links: A sends to B
G1= P1.ur → UR.inc    →G1.
G2= P1.na → NA.inc    →G2.
G3= P2.ur → UR.double →G3.
G4= P2.na → NA.double →G4.
SYSTEM = (P1:P1 || P2:P2 || UR:UR || NA:NA || G).
GlueProperty = UR.inc    → NA.inc
                 → UR.double → NA.double
               → GlueProperty
             □   UR.double → NA.double
                 → UR.inc    → NA.inc
               → GlueProperty.
```

- *Of course, GlueProperty is no longer satisfied. . .*
- At least now we *know* that the "decentralized" design doesn't work!
- And so do the designers/developers/clients . . .