
Architecture Definition Languages

Ingolf H. Krueger
ikrueger@ucsd.edu

Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA



California Institute for Telecommunications
and Information Technologies
La Jolla, CA 92093-0405, USA



Overview

- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Modeling and Documenting Architectures – Review

How to model and implement

- *Units / components / subsystems,*
- *Connections / interactions / relationships,*
- *Quality attributes / development guidelines / constraints*

adequately?

⇒ Approaches:

- *Architecture Definition Languages (ADLs)*
- *Architectural styles and patterns*
- *Domain specific architectures*
- *Infrastructures*

Modeling



Implementation

Overview

- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Architectural Description Languages (ADLs)

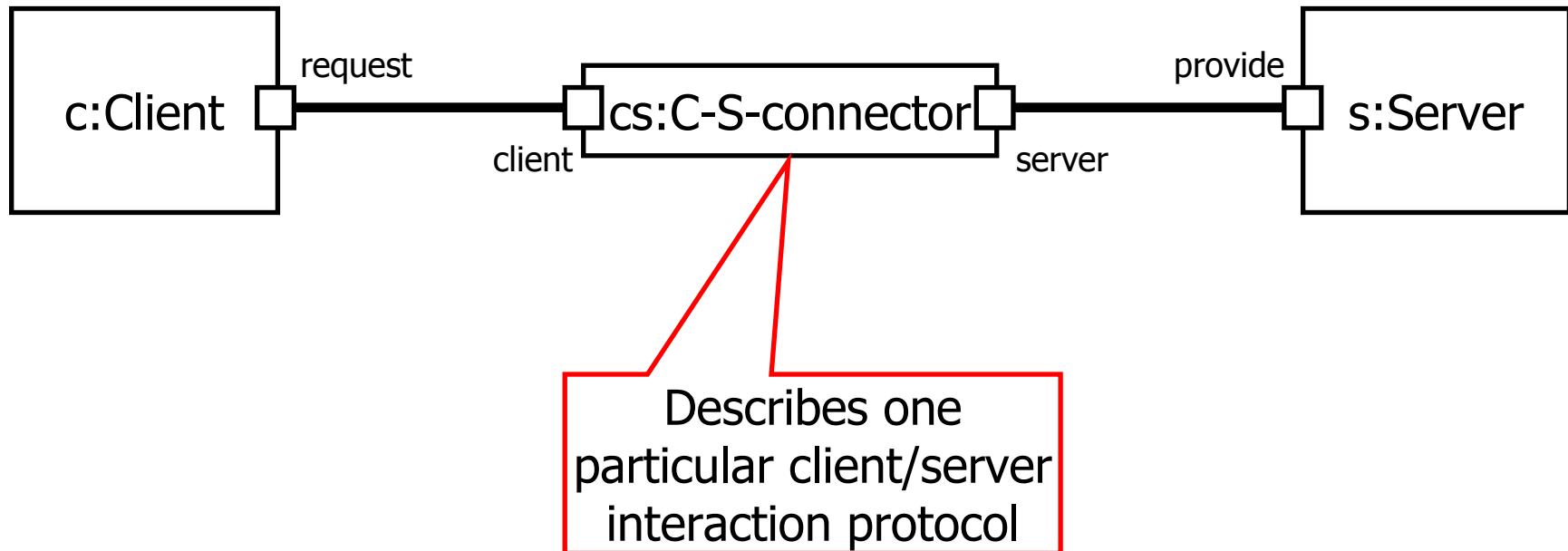
- ADLs focus on *explicit* representation of
 - Components
 - Connectors
 - Configurations / Topology
- Enable modeling on conceptual level
- Explicit modeling of connectors distinguishes ADLs from many approaches to module interconnection and object-orientation

Overview

- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Example: Wright

developed by Garlan et al. at CMU



adapted from:

R. Allen, D. Garlan: *Formal Connectors*, Technical Report CMU-CS-94-115, School of Computer Science, Carnegie Mellon University, 1994

Example: Wright

System SimpleExample

Component Server

Port provide [*provide protocol*]

Spec [*server specification*]

Component Client

Port request [*request protocol*]

Spec [*Client specification*]

Connector C-S-connector

Role client [*client protocol*]

Role server [*server protocol*]

Glue [*glue protocol*]

Example: Wright

Instances

s: Server

c: Client

cs: C-S-connector

Attachments

s.provide **as** cs.server

c.request **as** cs.client

end SimpleExample.

Interlude: CSP

- **Communicating Sequential Processes**
(C.A.R. Hoare; paper 1978, book appeared 1985)
- Process-algebraic approach to modeling concurrent systems
- System consists of concurrently operating processes
- Basic unit of behavior: occurrence of event
- Primitive and I/O events
 - coin
 - $\text{buffer?}x / \text{buffer!}(4*5)$
- Processes:
 - STOP, SKIP, $\surd$, RUN
 - or composed from events (and other processes)
 - αP : event alphabet of process P

Interlude: CSP

- Prefixing and repetition (recursion):
 - $\text{VEND} = \text{money_in?5} \rightarrow \text{VEND5}$
 - $\text{VEND5} = \text{money_in?10} \rightarrow \text{dispenser!soda} \rightarrow \text{VEND}$
- Choice:
 - internal: $(c?5 \rightarrow P) \hat{\cap} (c?5 \rightarrow Q)$
 - external: $(c?5 \rightarrow P) \square (c?5 \rightarrow Q)$
- Parallel Composition:
 - $P \parallel Q$
 - $\text{CUSTOMER} = \text{money_in!5} \rightarrow \text{money_in!10} \rightarrow$
 $\text{dispenser?soda} \rightarrow \text{CUSTOMER}$
 - $\text{CUSTOMER} \parallel \text{VEND}$
 - Synchronization via common events
(in intersection of processes' event alphabets)

Example: Wright

Connector C-S-connector

Role Client = (request!x -> result?y -> Client) $\int \checkmark$

Role Server = (invoke?x -> return!y -> Server) $\square \checkmark$

Glue = (Client.request?x -> Server.invoke!x
 -> Server.return?y
 -> Client.result!y
 -> Glue) $\square \checkmark$

Example: Wright

Shortcomings of Wright

- Connector specifications difficult to read
- Coordination only implicit by composing roles and glue
- Redundancy introduced by duality between roles and glue

Alternatives

- UML-RT for components and configurations
- MSCs for connector specifications

Overview

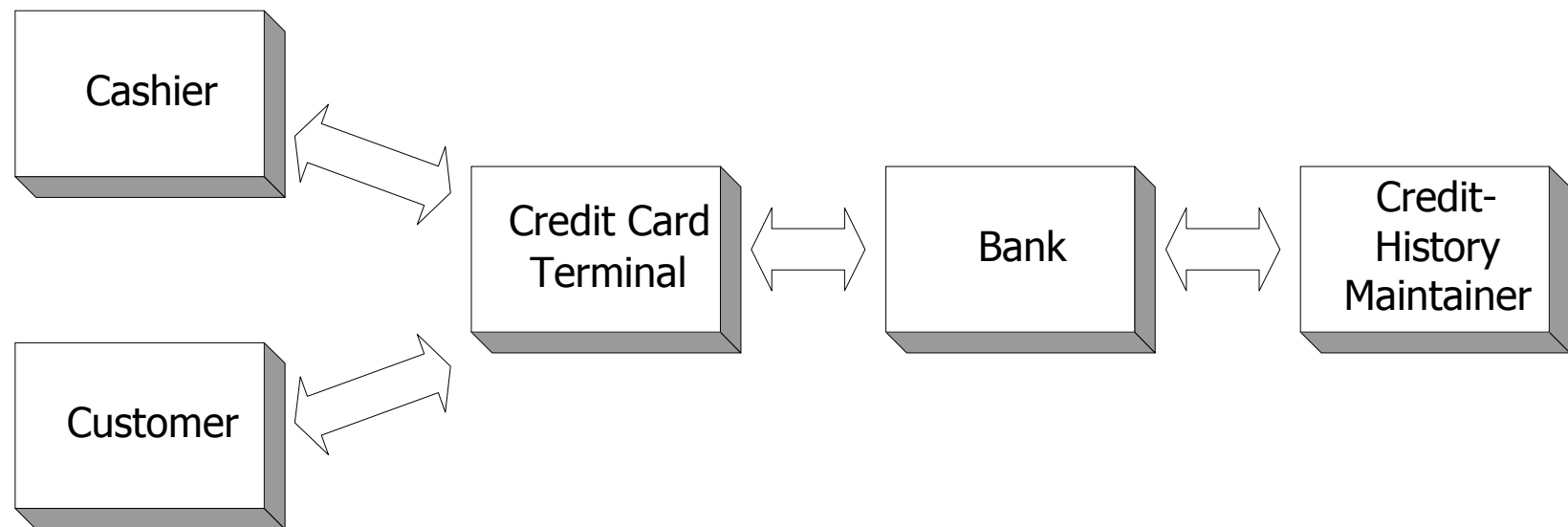
- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Example: Rapide

- ADL developed at Stanford by Luckham et al.
- Sublanguages:
 - types
 - architecture
 - constraint specification
 - executables
 - event patterns
- Focus on modeling, prototyping and simulation
- Semantic model: partially ordered event sets (posets)
- Tool support

Example: Rapide

the syntax on the following slides has been slightly adapted to fit the presentation; please refer to [LKA+95] for details of Rapide's syntax



[LKA+95] D.C. Luckham, J.J. Kenney, L.M. Augustin, J. Vera, D. Bryan, W. Mann: *Specification and Analysis of System Architecture Using Rapide*. IEEE Transactions on Software Engineering, 21(4):336—355, April 1995

Example: Rapide

type CreditCardTerminal **is interface**

extern

action debit(a: Amount);

action acceptAmount(a: Amount);

action acceptResult(r: Result);

action getPIN();

public

action displayAmount(a: Amount)

action enterPIN(p: PIN);

action readCard(c: Customer);

action setDebitStatus(ds: DebitStatus);

action amountOK();

Example: Rapide

behavior

```
(?c in Customer, ?a in Amount, ?p in PIN)  
readCard(c) -> displayAmount(a)  
=> acceptAmount(a)  
amountOK() => getPIN()  
enterPIN(p) => debit(a, c)  
setDebitStatus(ok) => acceptResult(success)  
setDebitStatus(nok) => acceptResult(failure)
```

Example: Rapide

```
type Bank is interface
  extern
    action inquiry(c: Customer);
    action setDebitStatus(ds: DebitStatus);
  public
    action block_account(c: Customer);
    action good_standing(c: Customer);
    action bad_standing(c: Customer);
    action debit(a: Amount);
    ...
end
```

Example: Rapide

```
type CHM is interface  
  extern  
    action block_account(c: Customer);  
    action good_standing(c: Customer);  
    action bad_standing(c: Customer);  
  public  
    action inquiry(c: Customer);  
  ...  
end  
  
type Customer is interface ... end  
  
type Cashier is interface ... end
```

Example: Rapide

architecture POS **is**

c: Customer;

cs: Cashier;

cct: CreditCardTerminal;

b: Bank;

chm: CHM;

connect

cct.debit **to** b.debit;

cct.getPIN **to** c.getPIN;

...

end

Overview

- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Background and Motivation

- Dramatic increase in distribution of software systems
 - Business/Enterprise Systems
 - Technical/Embedded Systems
- Shift from stand-alone to networked systems
- Internet/Wireless Networks have become key enabling technologies for advanced services
- Convergence between business and technical systems:
 - Telecommunication
 - Web Services
 - Embedded Systems

Example: Automotive Domain

- Increasingly complex automotive systems
 - up to 1400 s/w functions, 80 ECUs
 - highly networked – up to 5 different buses
- Software in the Automotive Domain
 - major driving factor for innovation
 - 40% of vehicle cost determined by software
 - 50-70% of development cost of ECUs s/w related

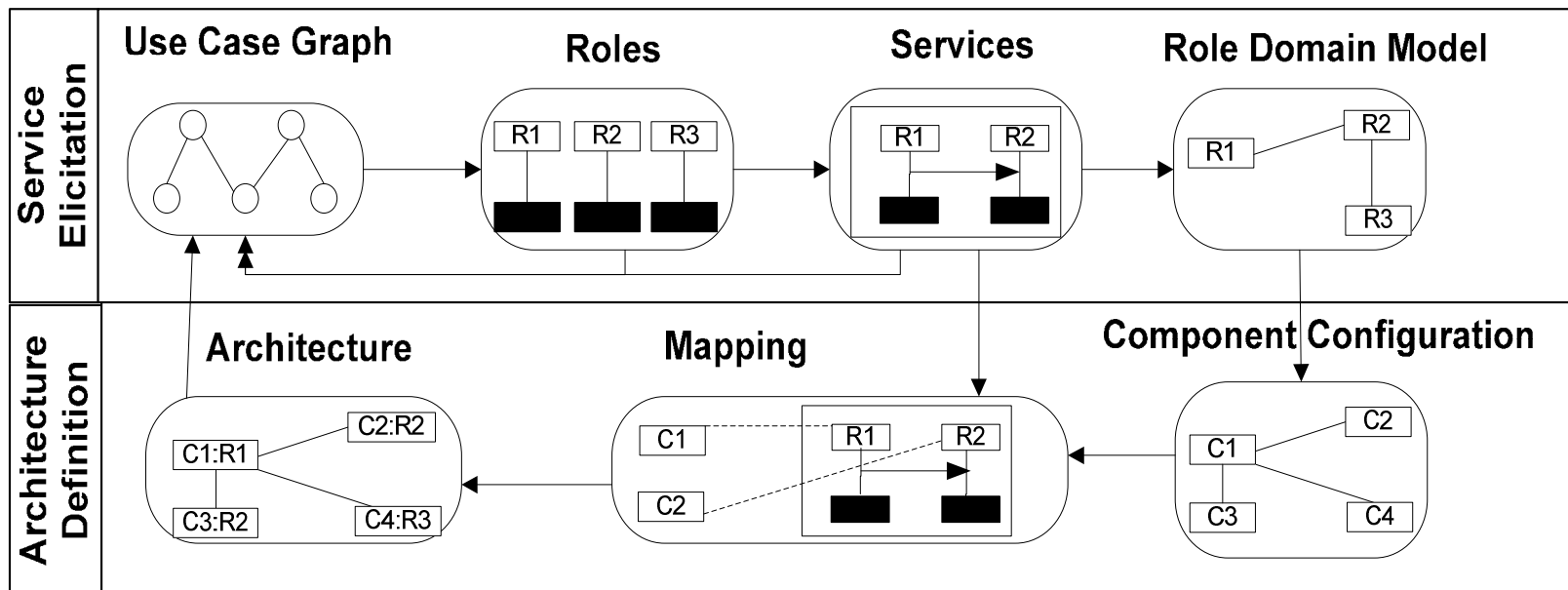
***Cars have become complex, distributed, networked,
tightly integrated software systems***

Services – Introduction and Concepts

- Interaction patterns are cross-cutting concerns
 - shift focus from components to their interactions
- Services
 - projection of overall system behavior
 - defined by interactions between participating entities
 - first class elements of modelling and implementation
 - described using extended MSCs

Service-Oriented Development Overview

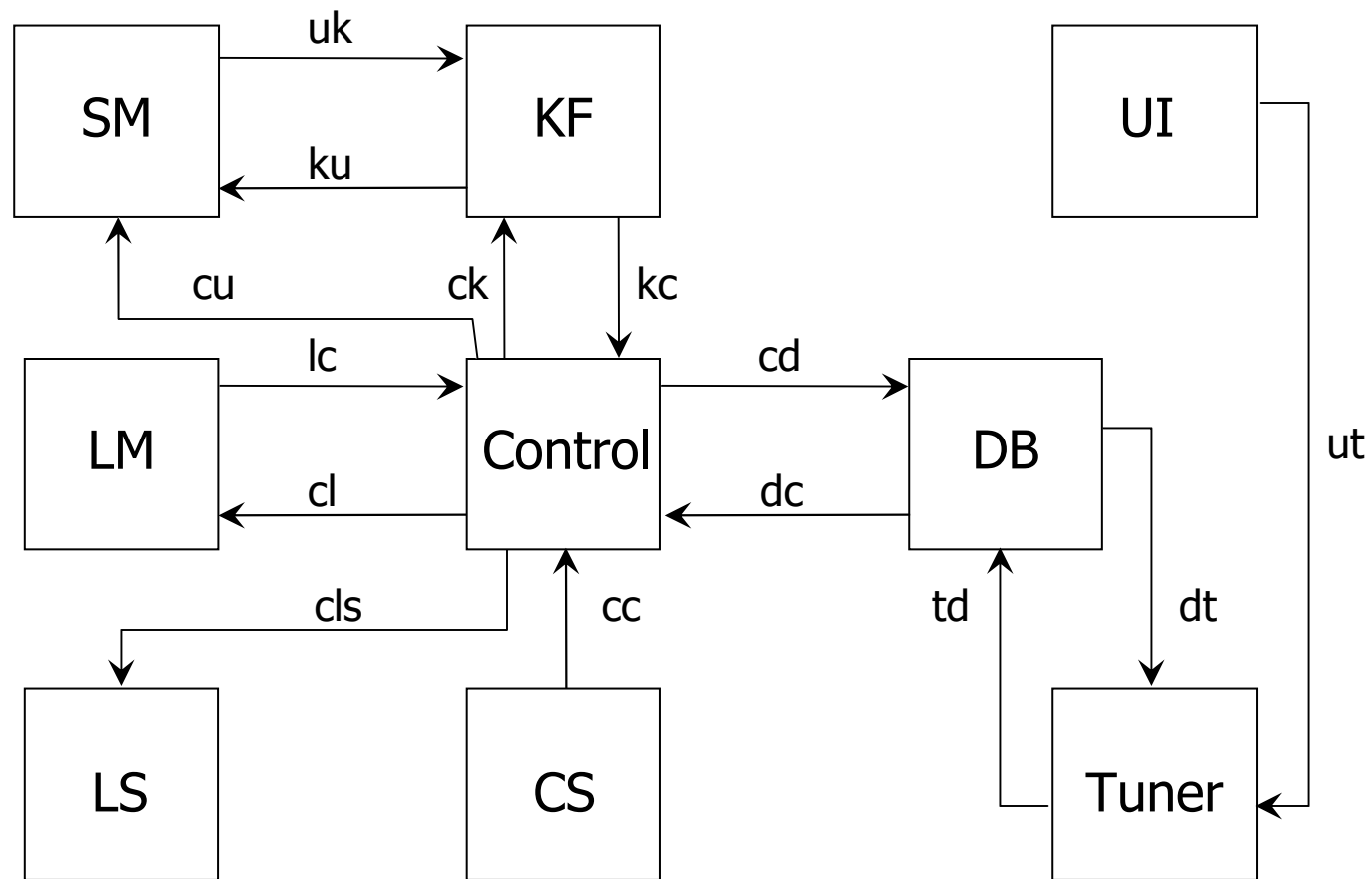
- Iterative development process
 - Service elicitation
 - Architecture definition



Example: Central Locking System

operation of locks / signaling

fetch driver presets on entry



crash detection / management

set tuner presets

Service: CLS Unlocking 1

service UNLK-1

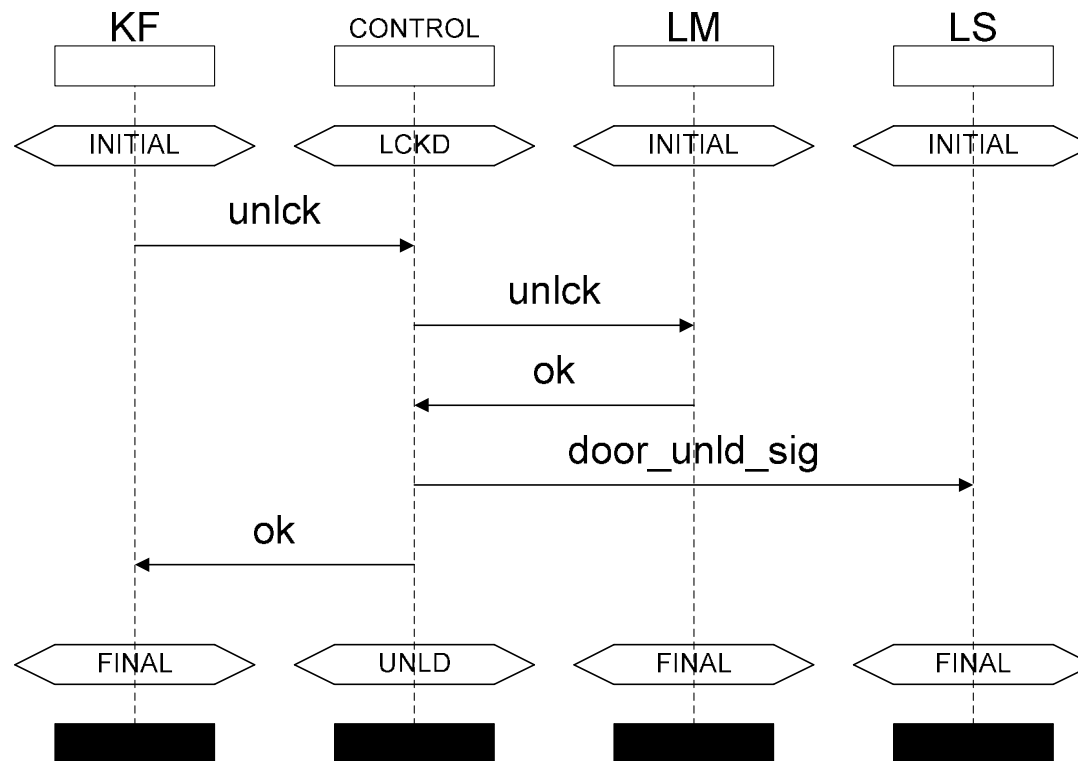
description

Unlocks car and signals success

roles

KF, Control, LM, LS

interaction



Service: CLS Unlocking 2

service UNLK-2

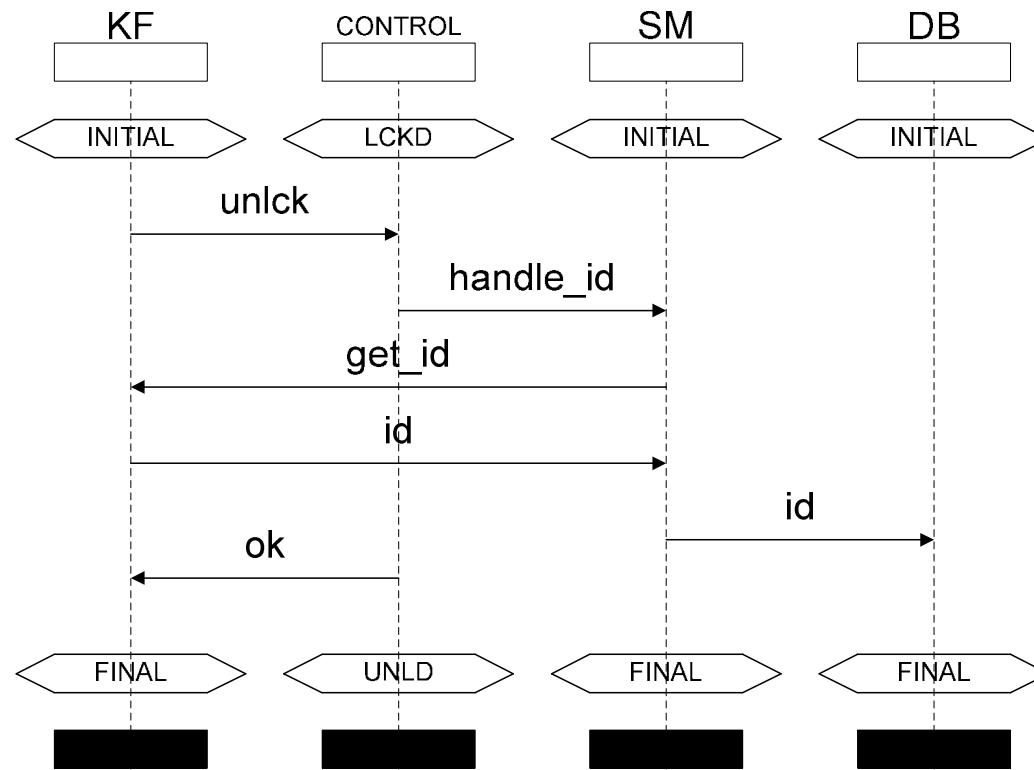
description

Transfers key ID for storage in DB

roles

KF, Control, SM, DB

interaction



Architecture: CLS

component type Keyfob
plays KF
in service CLS

component type CrashMgmt
plays CS
in service CLS

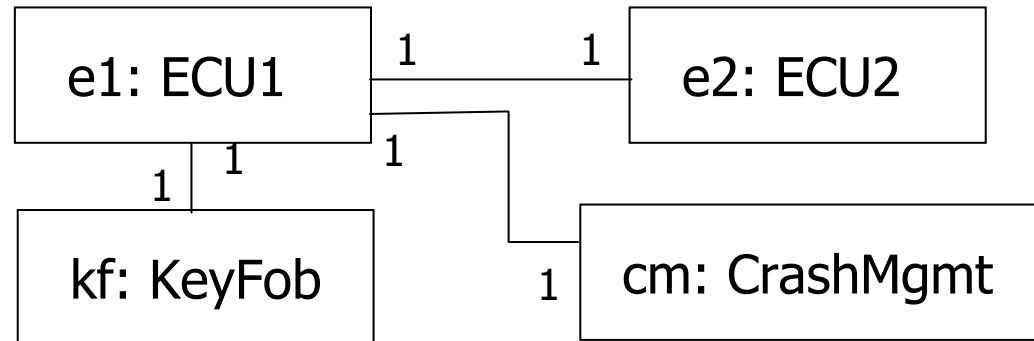
component type ECU1
plays Control, LM, LS
in service CLS

component type ECU2
plays DB, UI, Tuner
in service CLS

components

kf : KF;
e1 : ECU1;
e2 : ECU2;
cm : CrashMgmt;

configuration



implements
CLS

Overview

- Modeling and Documenting Architectures – Review
- Architecture Definition Languages (ADLs)
- Example: Wright
- Example: Rapide
- Example: Service-Based ADL
- Summary and Outlook

Summary and Outlook

- ADLs provide explicit means for specifying components, connectors, configuration
- Wright:
 - components, connectors, instances, attachements
 - formal foundation: CSP specifications
- Rapide:
 - rich set of specification languages, enabling multiple layers of abstraction/hierarchical specifications
 - complex event patterns/constraint specifications
 - formal foundation: posets
- Service-ADL:
 - Focus shift from components to interactions/services
- Formally founded ADLs enable systematic analysis/synthesis on the level of architecture
- ADLs have been “in” and “out” over the years
- Increased interest in using UML-RT/UML 2.0 as ADLs