Carnegie Mellon
Software Engineering Institute

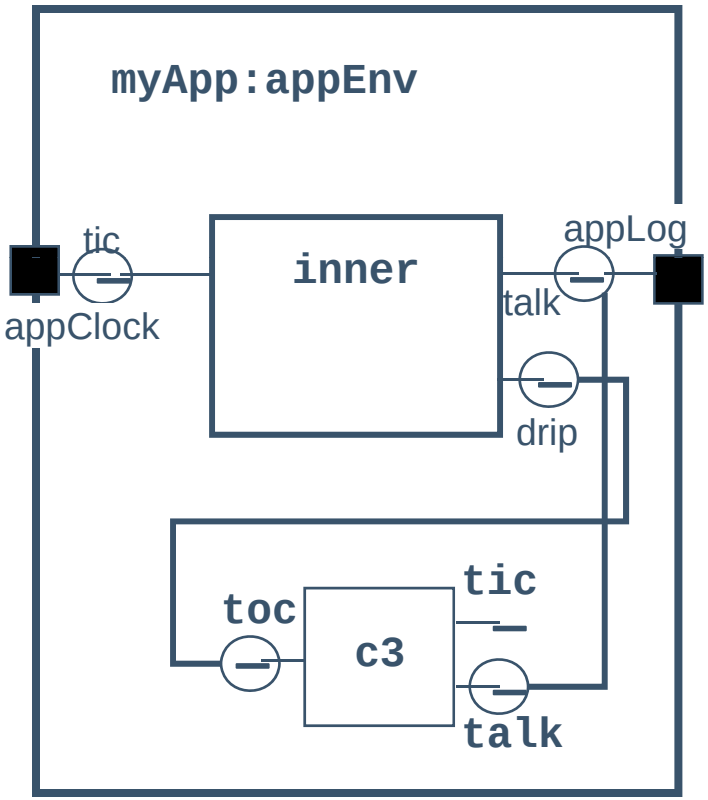
Pittsburgh, PA 15213-3890

CCL in Pictures

James Ivers, Kurt C. Wallnau

Sponsored by the U.S. Department of Defense

Example

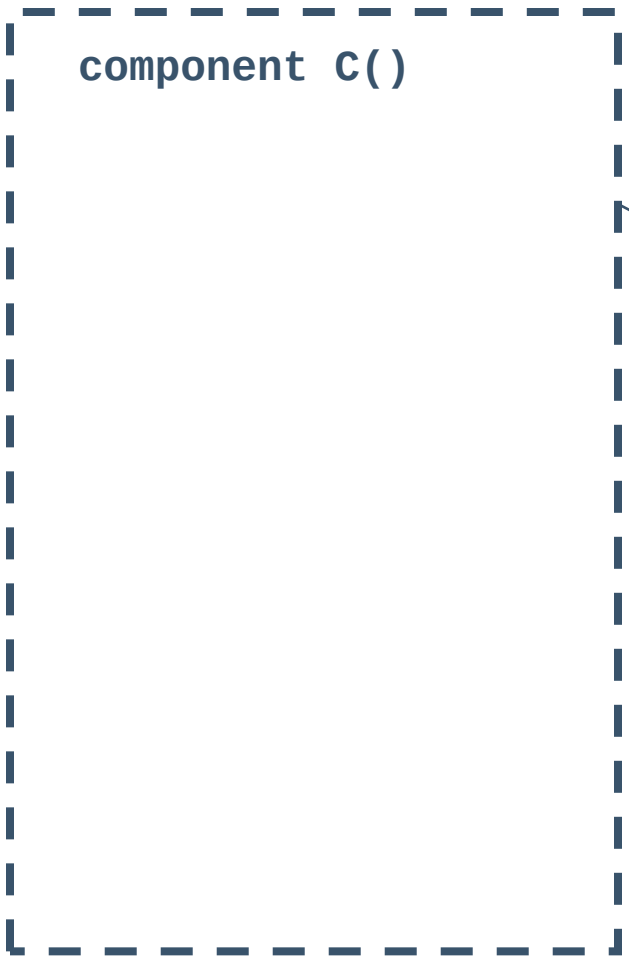


Starting with a simple case

- single runtime appEnv
- inner is a subassembly
  - but you can't tell from here

Carnegie Mellon
Software Engineering Institute

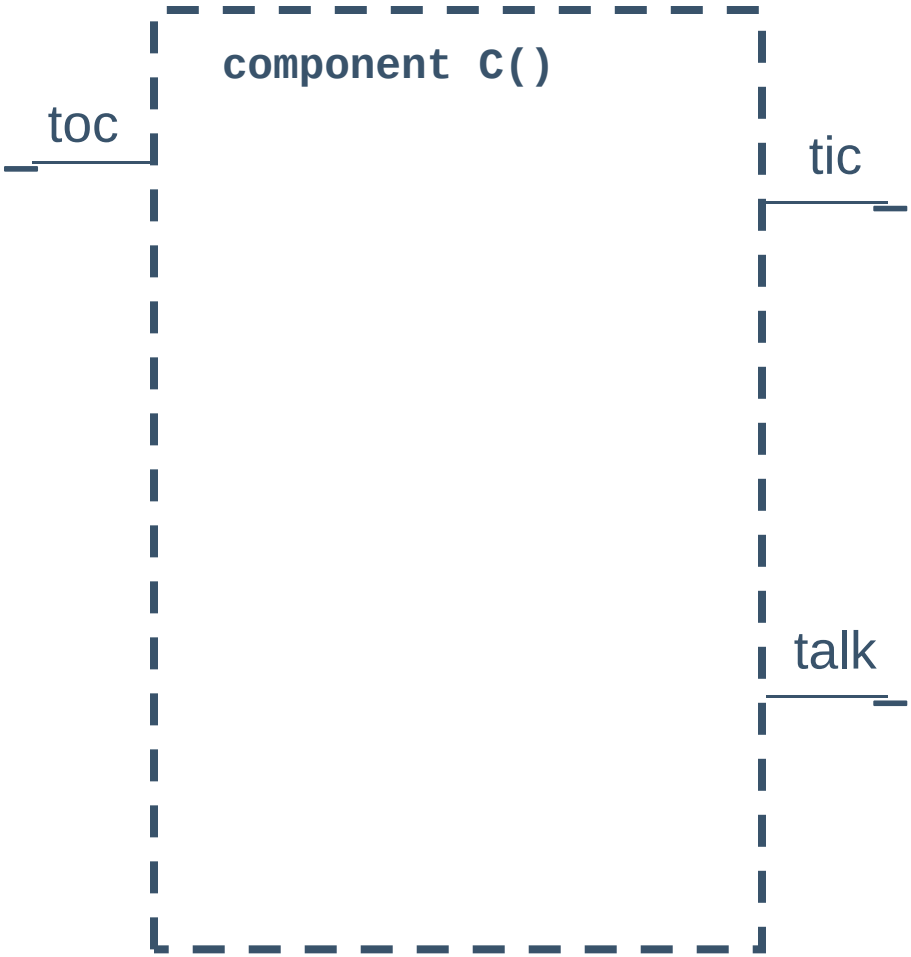
Example – Define Component Type




```
// CCL
component C() {
  …
}
```

the shell of a component type

Example – Define Component Type

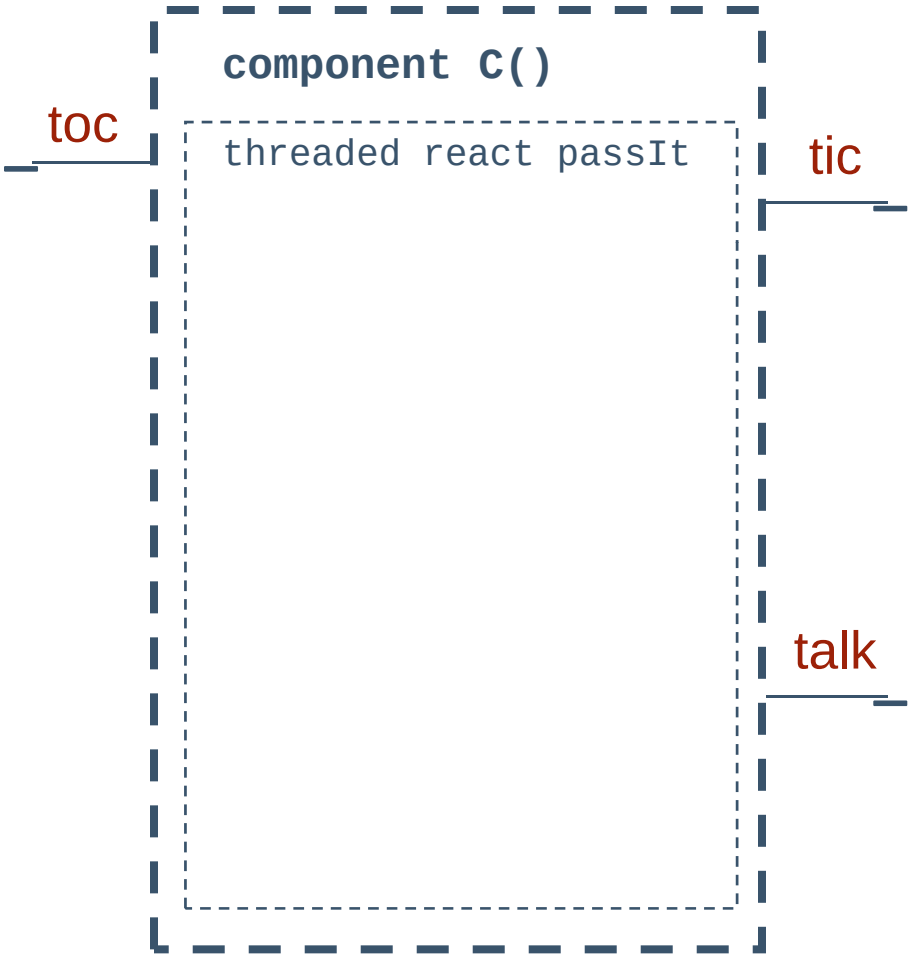



```
// CCL
component C() {
  sink asynch toc();
  source unicast tic();
  source unicast talk();
  …
}
```

stimulus received on sink pins and emitted on source pins

pin signature is covered in a separate module

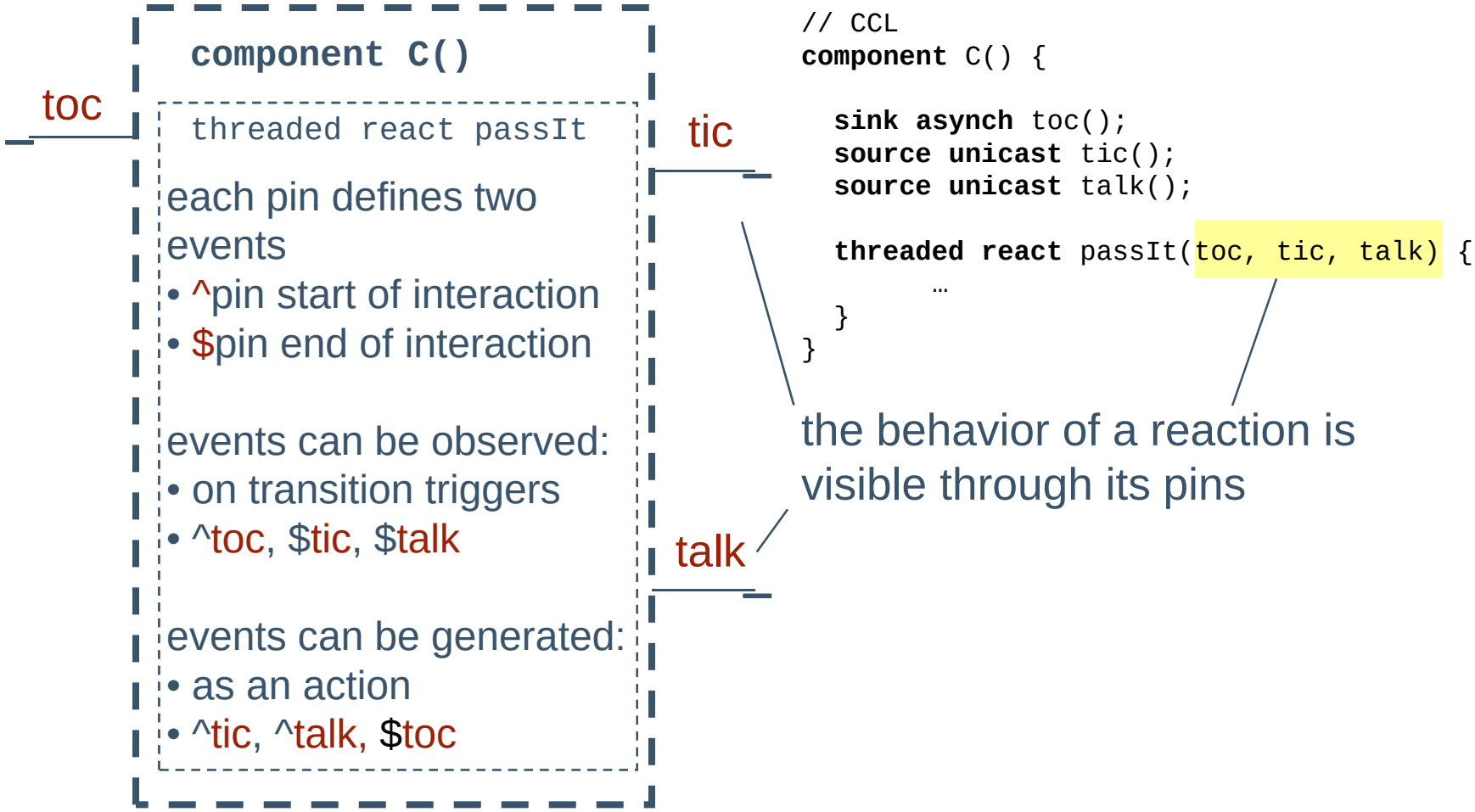
Example – Define Component Type



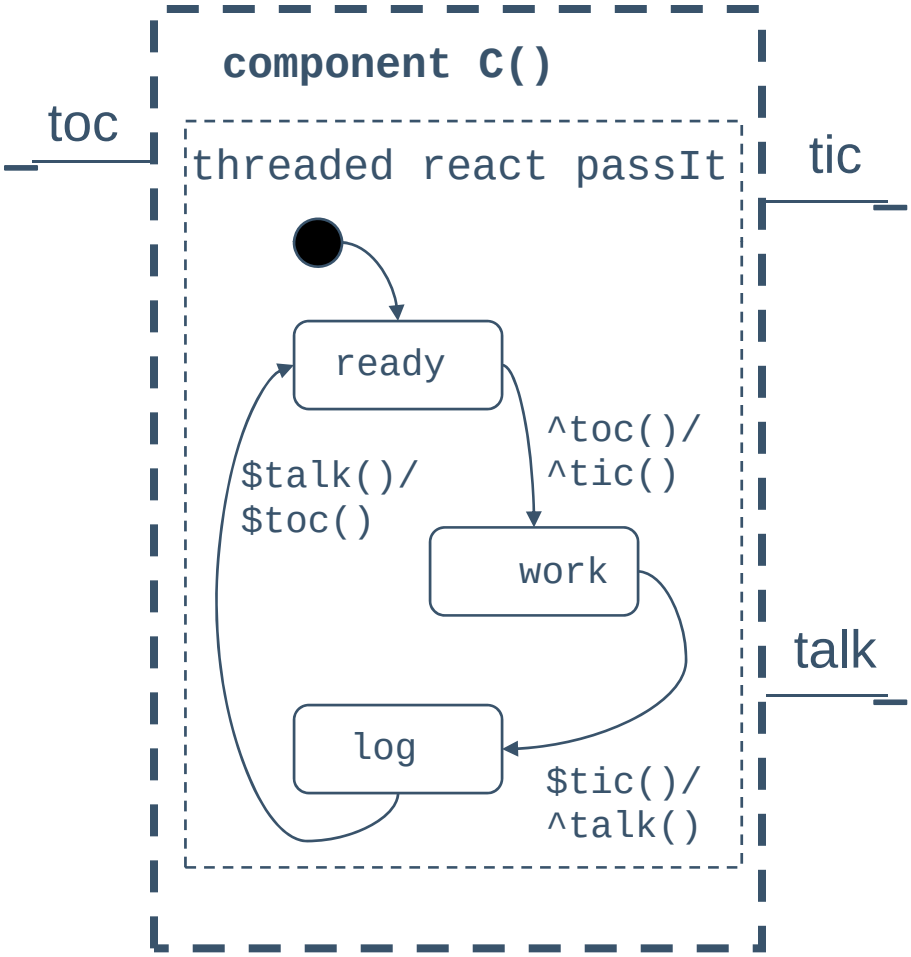

```
// CCL
component C() {

  sink asynch toc();
  source unicast tic();
  source unicast talk();

  threaded react passIt(…) {
        …
  }
}
```

- component behavior is specified by reaction
- reactions may be units of concurrent execution
- components may have one or more reactions

Example – Define Component Type



Example – Define Component Type




```
// CCL
component C() {

  sink asynch toc();
  source unicast tic();
  source unicast talk();

  threaded react passIt(toc, tic, talk) {
        start->ready{}
        ready->work{
           trigger ^toc(); action ^tic();}
        work->log{
           trigger $tic(); action ^talk();}
        log->ready{
           trigger $talk(); action $toc();}
  }
}
```

behavior is specified as executable state machines ≅ UML statecharts

details about the CCL action language and writing reactions are provided in a separate module.

Example – Define Environment Type

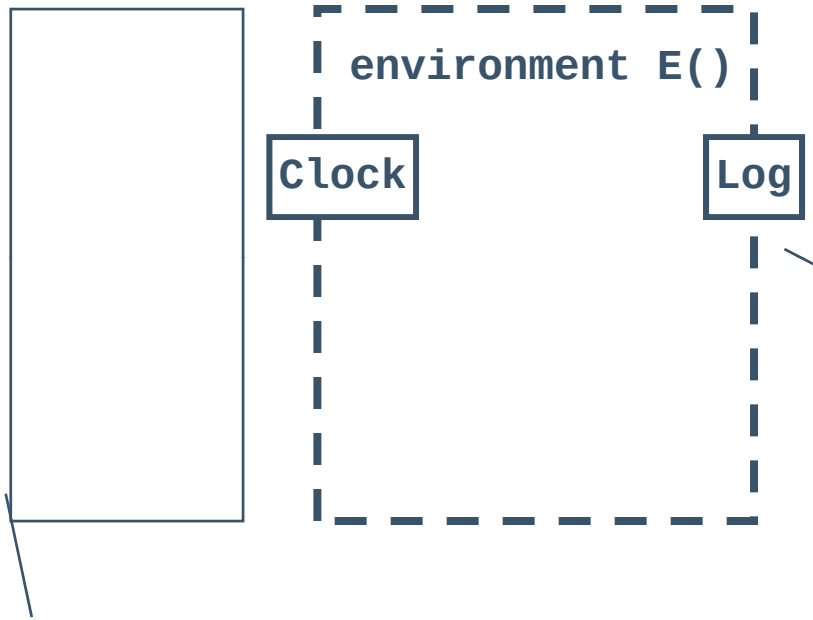

```
// CCL
environment E() {

}
```

the shell of an environment type

an virtual palette associated with each environment type

Carnegie Mellon
Software Engineering Institute

Example – Define Environment Type

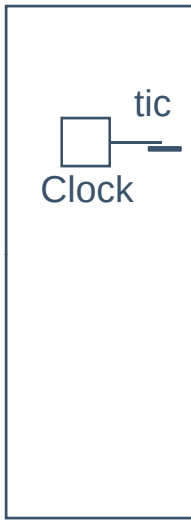



```
// CCL
environment E() {

  service Clock() {}
  service Log() {}
}
```

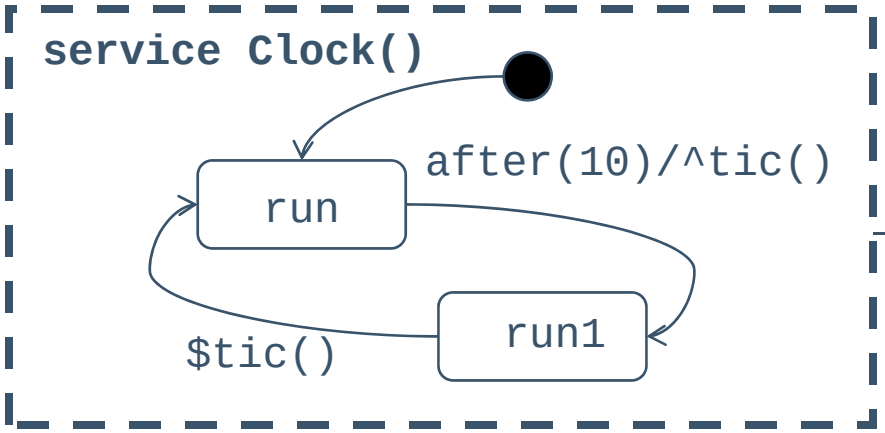
Service types

an virtual palette of services types
associated with each environment type

Example – Define Environment Type



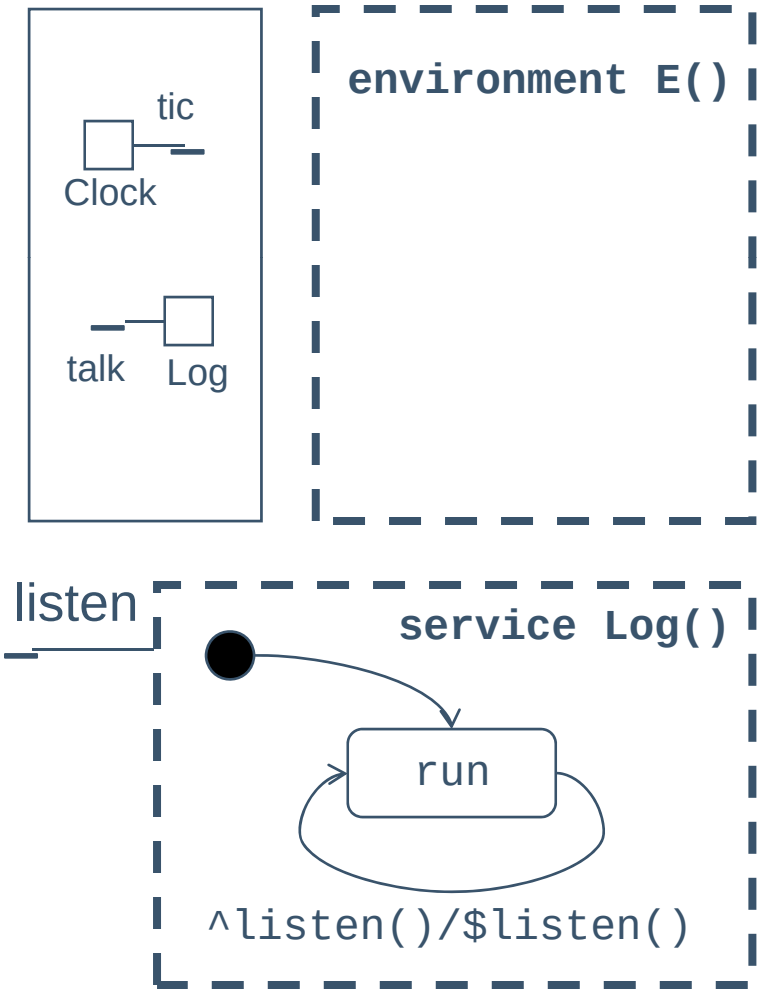

```
// CCL
environment E() {
  service Clock() {
     source unicast tic();
     threaded react ticking(tic) {
        start->run{}
        run->run1{trigger after(10); action ^tic();}
        run1->run{trigger $tic();}
      }
   }
}
```



services are environment-provided component types

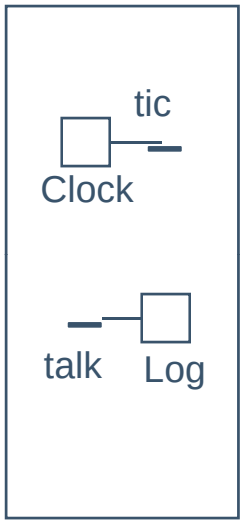
services may have different well-formedness rules

Example – Define Environment Type




```
// CCL
environment E() {

  service Clock() {
     source unicast tic();
     threaded react ticking(tic) {
        start->run{}
        run->run1{trigger after(10); action ^tic();}
        run1->run{trigger $tic();}
      }
   }

   service Log() {
     sink asynch listen();
     threaded react logging(listen) {
        start->run{}
        run->run{trigger ^listen(); action $listen();}
     }
   }
}
```

Example – Define Environment Type




```
// CCL
environment E() {

  service Clock() {
     source unicast tic();
     threaded react ticking(tic) {
        start->run{}
        run->run1{trigger after(10); action ^tic();}
        run1->run{trigger $tic();}
      }
   }

   service Log() {
     sink asynch listen();
     threaded react logging(listen) {
        start->run{}
        run->run{trigger ^listen(); action $listen();}
     }
   }

   sink handler (asynch, mutex)  { } //details omitted
}
```

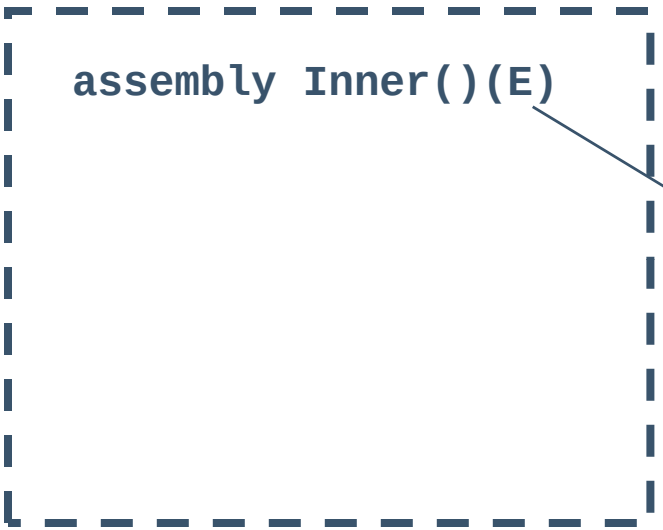
sink logic {–,>, {–,>,}, {…} …}

source logic {–}

sink handlers is not covered here...

Carnegie Mellon
Software Engineering Institute

Example – Define Inner Assembly Type

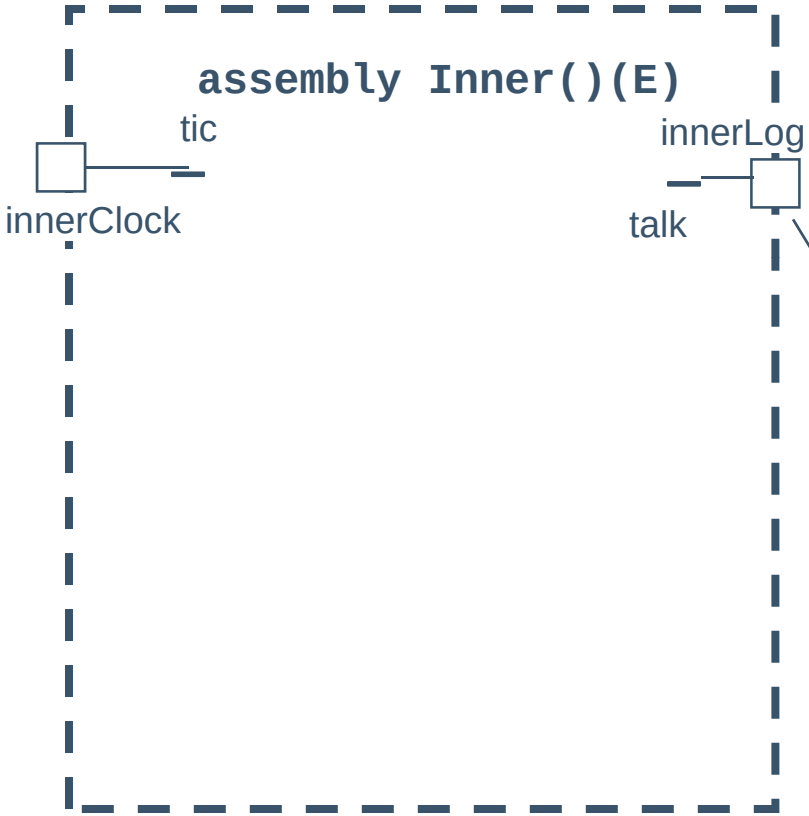



```
// CCL
assembly Inner ()(E) {
…
}
```

each assembly type is parameterized by an environment type

the environment type supplies interaction mechanisms and services

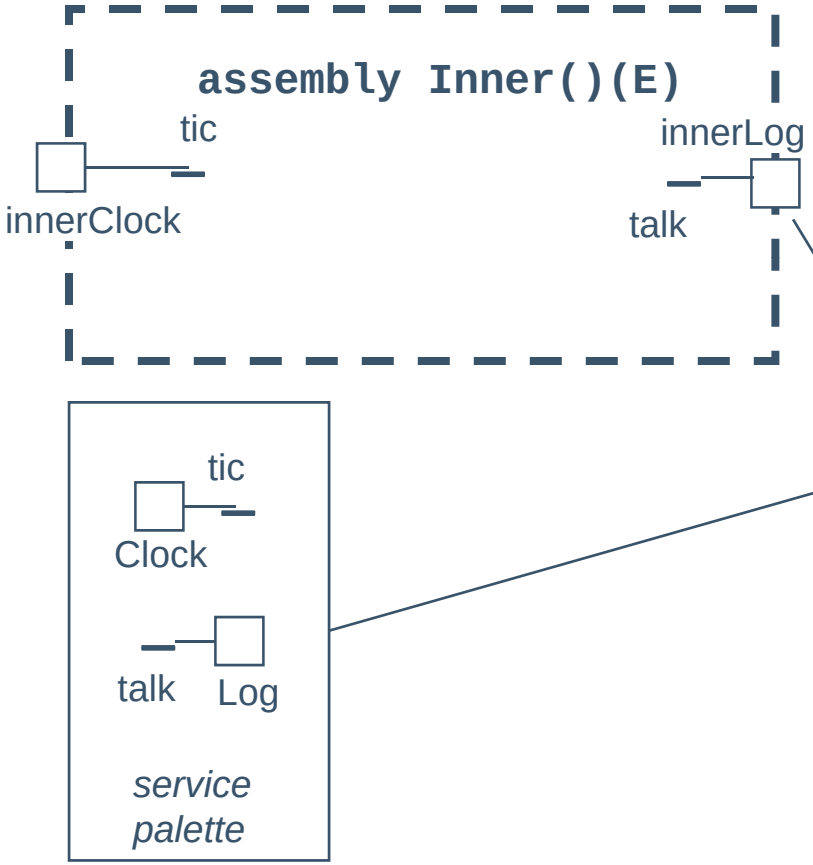
Example – Define Inner Assembly Type




```
// CCL
assembly Inner ()(E) {

   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }
}
```

assembly types may assume environment provided service instances

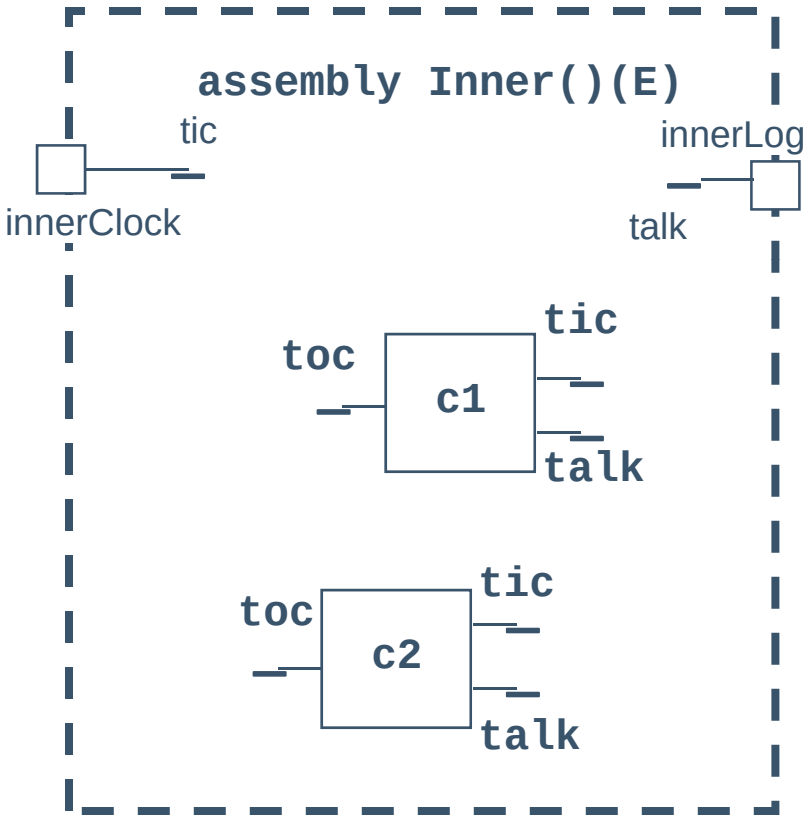
Example – Define Inner Assembly Type




```
// CCL
assembly Inner ()(E) {
   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }
}
```

assembly types may assume environment provided service instances

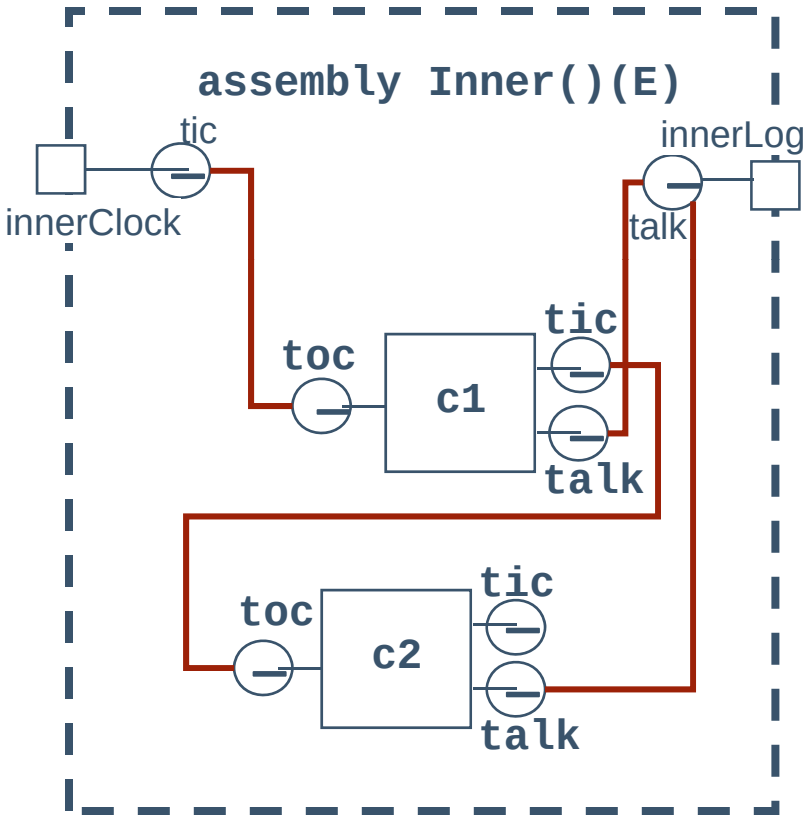
Example – Define Inner Assembly Type




```
// CCL
assembly Inner ()(E) {

   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }

   C c1(), c2();  // instantiation

}
```

assemblies also comprise component instances

Example – Define Inner Assembly Type



details about interaction and well-formedness checking is covered in a different module.

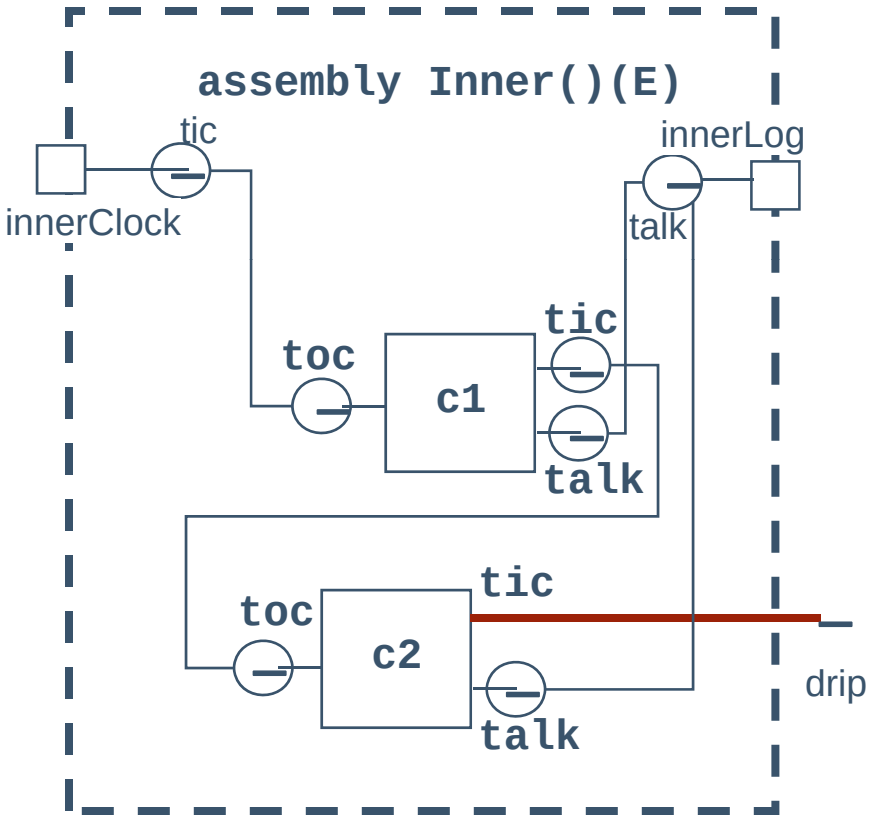
```
// CCL
assembly Inner ()(E) {

   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }

   C c1(), c2();  // instantiation

   innerClock:tic ~> c1:toc;
   c1:tic ~> c2:toc;
   c1:talk ~> innerLog:listen;
   c2:talk ~> innerLog:listen;
}
```

interactions take place between a source pin and one or more sink pins

connections must conform on

- source/sink pin signature
- connector-imposed restrictions
- reasoning framework restrictions

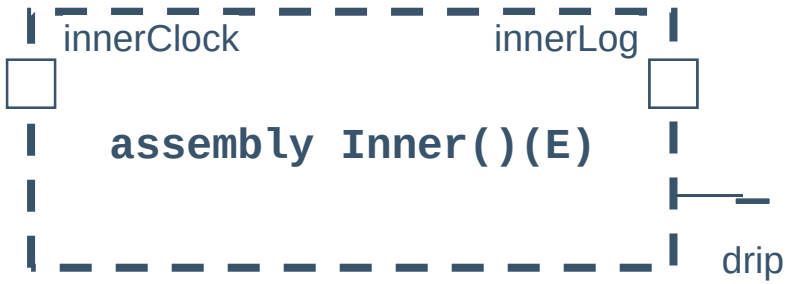
Example – Define Inner Assembly Type




```
// CCL
assembly Inner ()(E) {

   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }

   C c1(), c2();  // instantiation

   innerClock:tic ~> c1:toc;
   c1:tic ~> c2:toc;
   c1:talk ~> innerLog:listen;
   c2:talk ~> innerLog:listen;

   expose {c2:tic as drip}
}
```

an assembly type by default has no visible interactive behavior

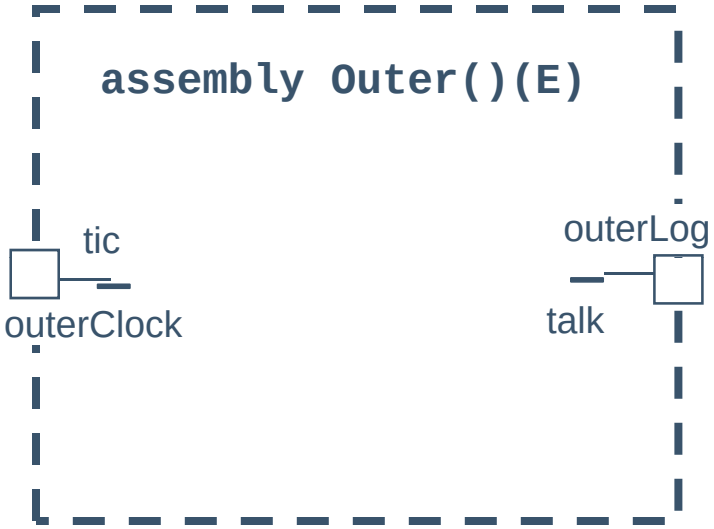
Example – Define Inner Assembly Type



The resulting assembly type `Inner` looks like a component

- with the addition of resource assumptions
- assumptions must be discharged at instantiation time
  - by passing along the assumption
  - by satisfying the assumption

Example – Define Outer Assembly Type

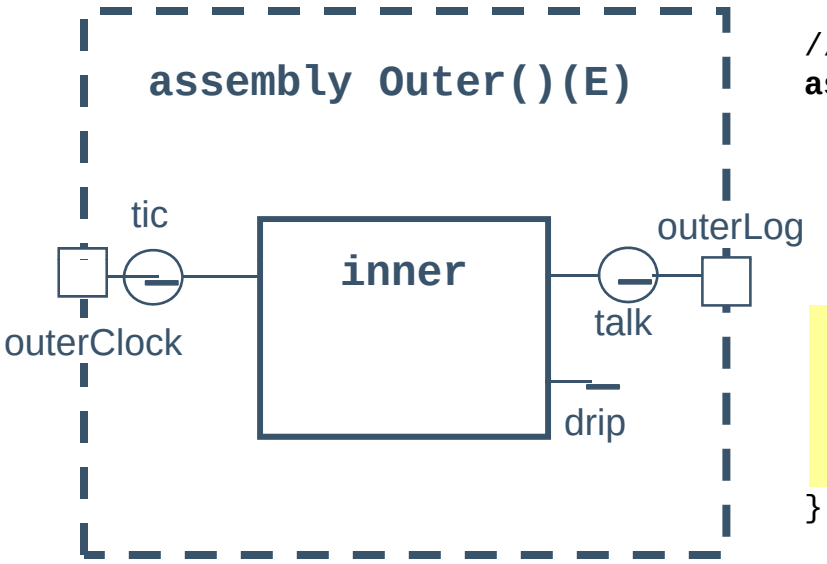



```
// CCL
assembly Outer ()(E) {

   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }
}
```

as before…an assembly makes assumptions about its environment

Carnegie Mellon
Software Engineering Institute

Example – Define Outer Assembly Type



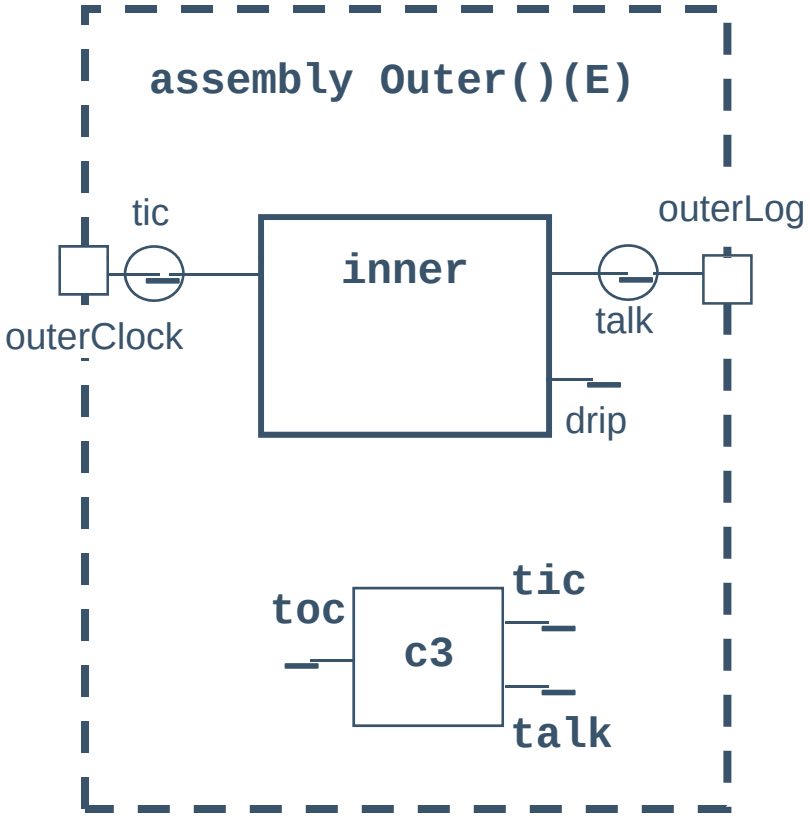

```
// CCL
assembly Outer ()(E) {

   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }

   Inner inner(){
      Inner:innerClock = outerClock;
      Inner:innerLog = outerLog;
   }
}
```

Inner assumption satisfied by Outer assumption

Assumptions are satisfied at instantiation time

- syntax: assumed resource = provided resource
- assumed resource in the scope of instantiated assembly type
- provided resource in the scope of the instantiator

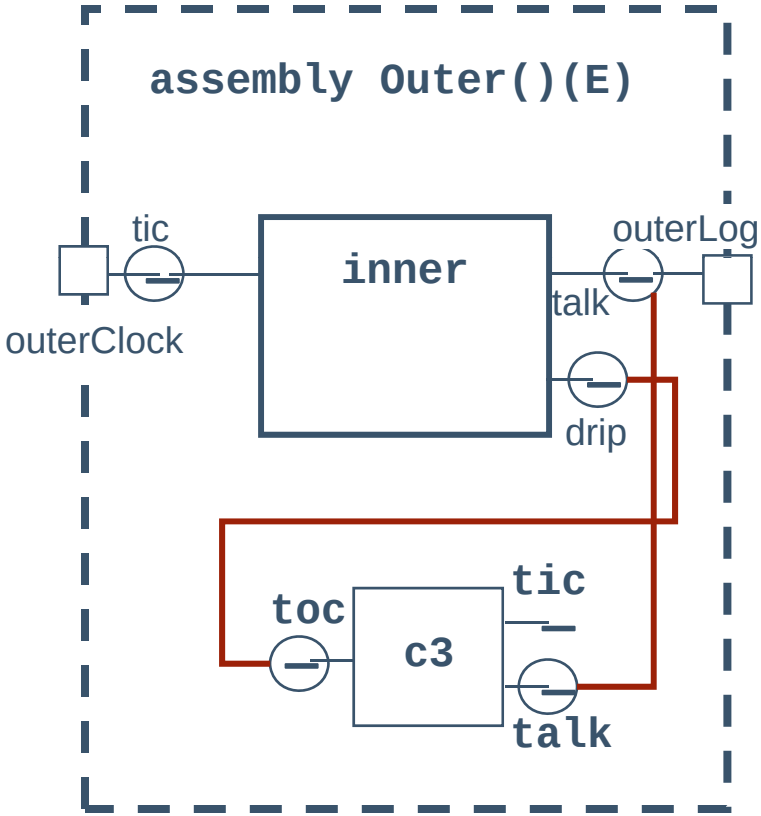
Example – Define Outer Assembly Type




```
// CCL
assembly Outer ()(E) {

   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }

   Inner inner(){
      Inner:innerClock = outerClock;
      Inner:innerLog = outerLog;
   }

   C c3();
}
```

components and assemblies can engage in *apparent* peer interaction

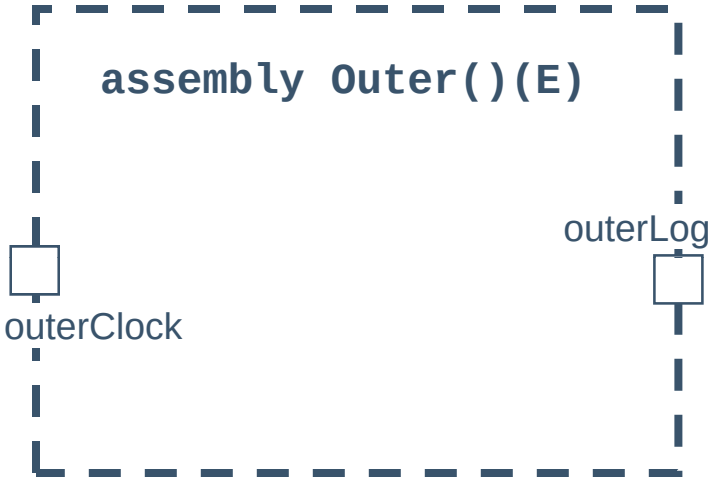
Example – Define Outer Assembly Type




```
// CCL
assembly Outer ()(E) {

   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }

   Inner inner(){
      Inner:innerClock = outerClock;
      Inner:innerLog = outerLog;
   };

   C c3();

   inner:drip ~> c3:toc;
   c3:talk ~> outerLog:listen;
}
```

interaction, as before

Example – Define Outer Assembly Type

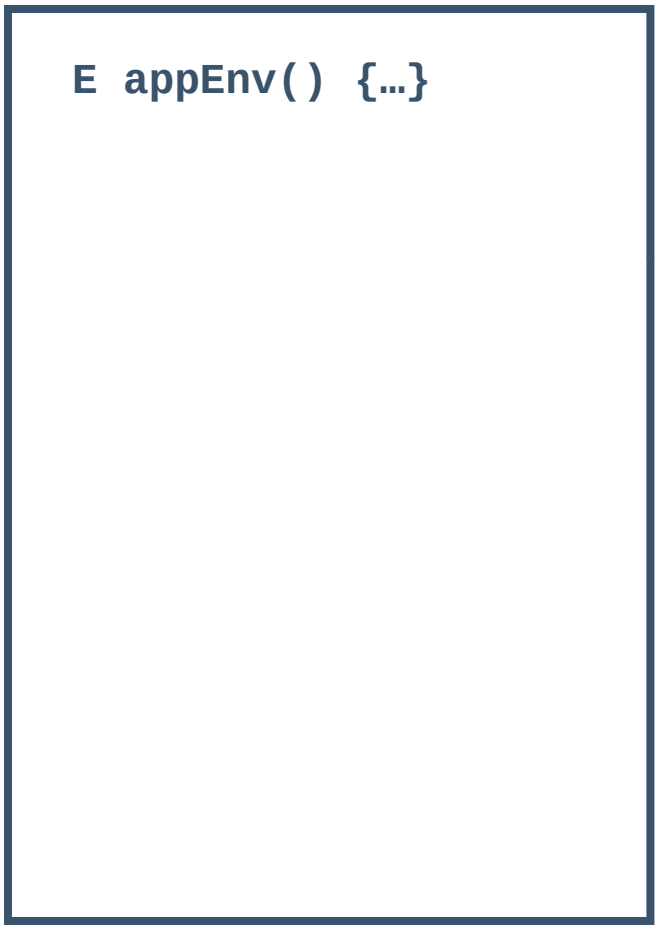



```
// CCL
assembly Outer ()(E) {

   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }

   Inner inner(){
      Inner:innerClock = outerClock;
      Inner:innerLog = outerLog;
   };

   C c3();

   inner:drip ~> c3:toc;
   c3:talk ~> outerLog:listen;

   expose {}
}
```

this outermost assembly has
no visible interactive behavior

Carnegie Mellon
Software Engineering Institute

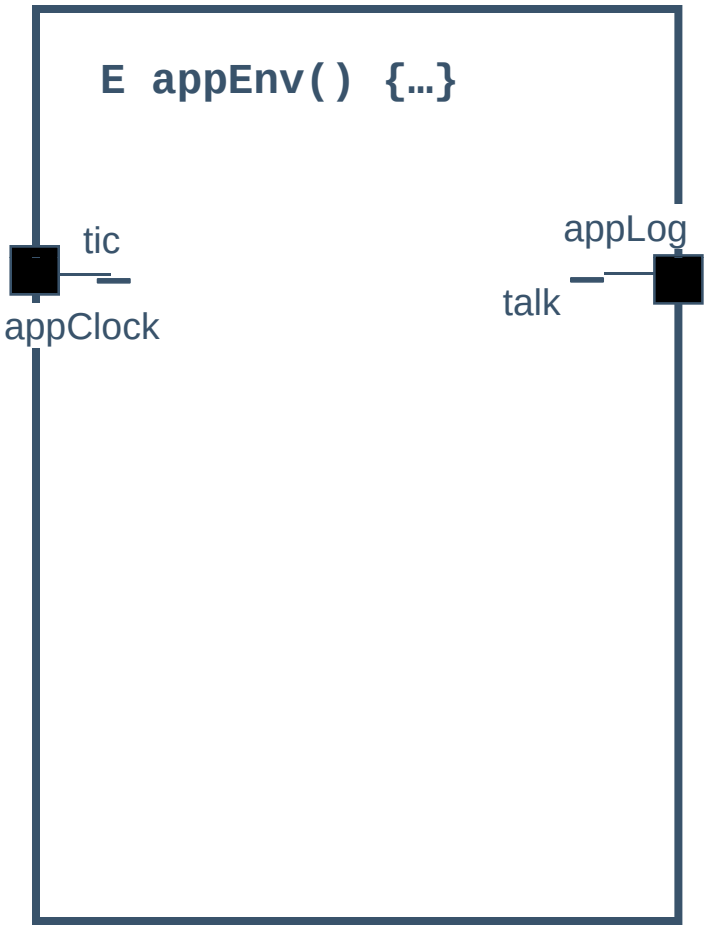
Example – Instantiate Application




```
// CCL
E appEnv() {
   …
}
```

instantiate the environment

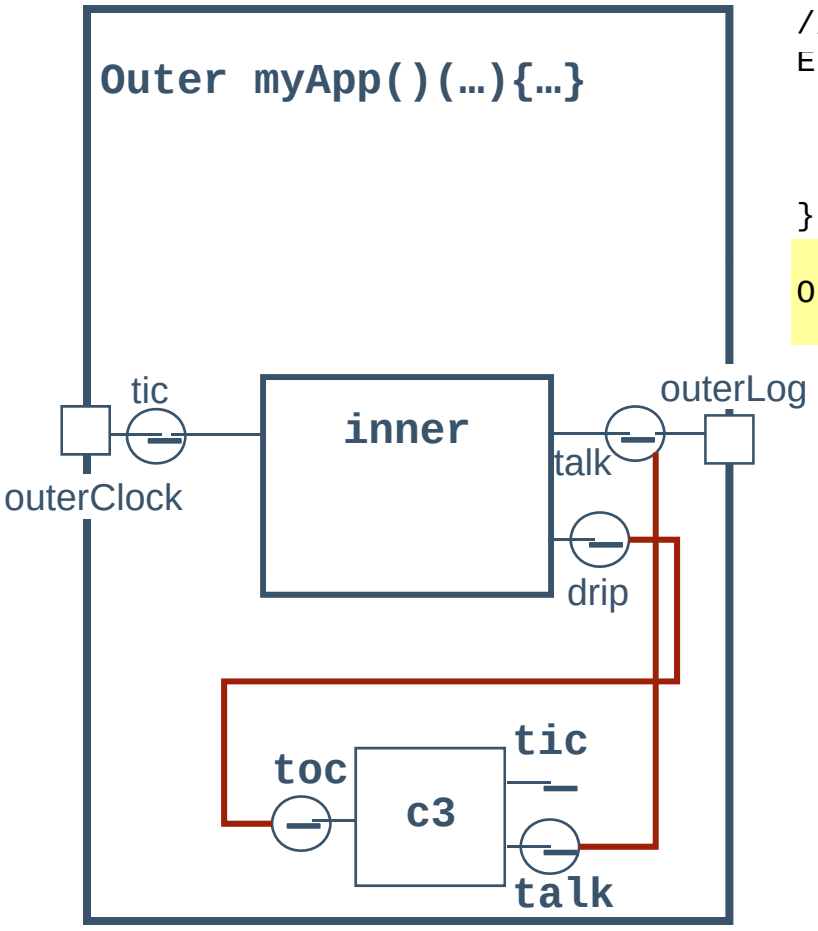
Example – Instantiate Application




```
// CCL
E appEnv() {
    E:Clock appClock();
    E:Log appLog();
};
```

elaboration clause instantiates the services of appEnv

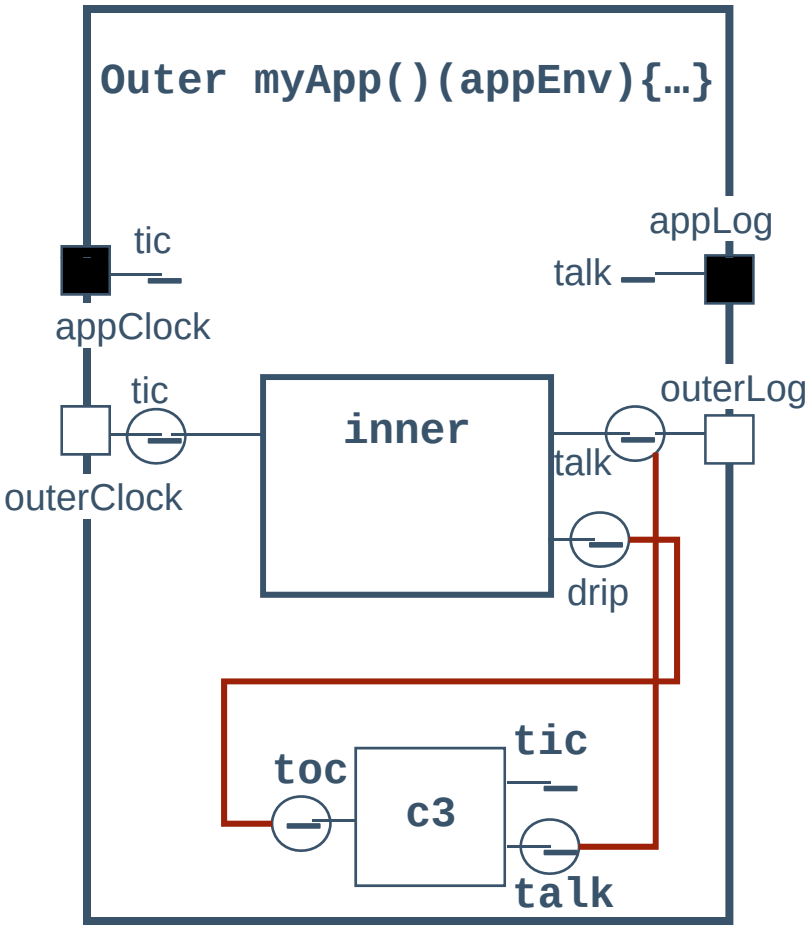
Example – Instantiate Application




```
// CCL
E appEnv() {
   E:Clock appClock();
   E:Log appLog();
   }
};

Outer myApp()() {…}
```

instantiate the assembly

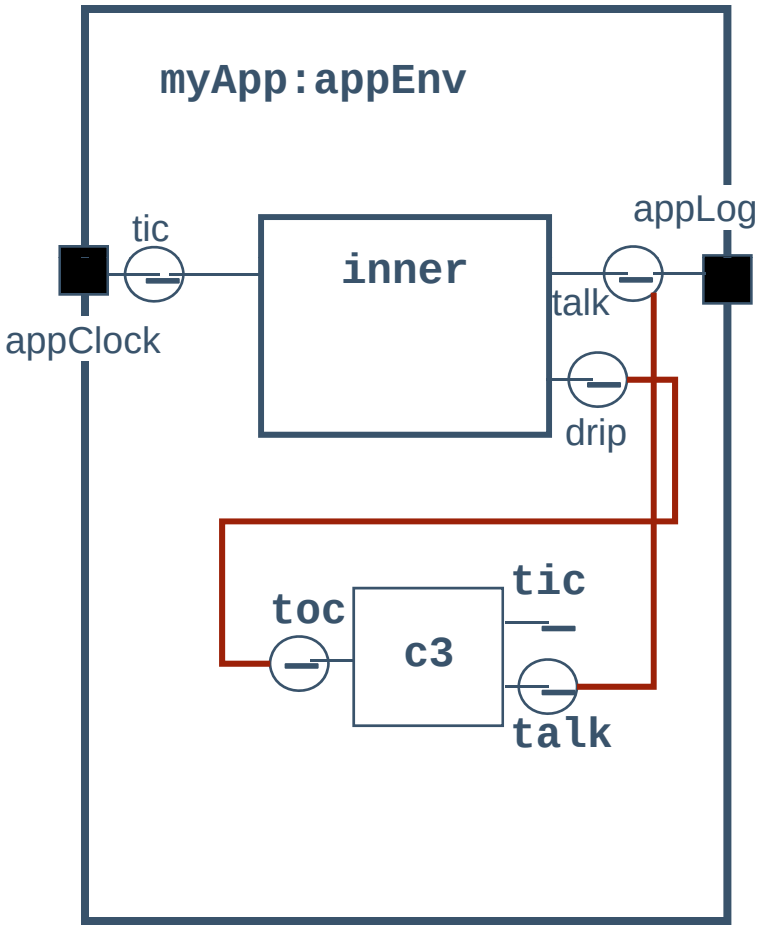
Example – Instantiate Application




```
// CCL
E appEnv() {
  E:Clock appClock();
  E:Log appLog();
  }
};

Outer myApp()(appEnv) {…}
```

deploy the assembly (instance) in an environment (instance)

Example – Instantiate Application




```
// CCL
E appEnv() {
  E:Clock appClock();
  E:Log appLog();
  }
};

Outer myApp()(appEnv) {
  Outer:outerClock = appEnv:appClock;
  Outer:outerLog = appEnv:appLog;
};
```

at elaboration satisfy the assumptions of the assembly type

Example Specification

```
// CCL specification of example 1
environment E() {
  service Clock() {
     source unicast tic();
     threaded react ticking(tic) {
        start->run{}
        run->run1{trigger after(10); action ^tic();}
        run1->run{trigger $tic();}
      }
   }

   service Log() {
     sink asynch listen();
     threaded react logging(listen) {
        start->run{}
        run->run{trigger ^listen(); action $listen();}
     }
   }
}

component C() {
  sink asynch toc();
  source unicast tic();
  source unicast talk();

  threaded react passIt(toc, tic, talk) {
        start->ready{}
        ready->work{trigger ^toc(); action ^tic();}
        work->log{trigger $tic(); action ^talk();}
        log->ready{trigger $talk(); action $toc();}
  }
}
```

1

```
assembly Inner ()(E) {
   assume {
      E:Clock innerClock();
      E:Log innerLog();
   }
   C c1(), c2();  // instantiation

   innerClock:tic ~> c1:toc;
   c1:tic ~> c2:toc;
   c1:talk ~> innerLog:listen;
   c2:talk ~> innerLog:listen;

   expose {c2:tic as drip}
}

assembly Outer ()(E) {
   assume {
      E:Clock outerClock();
      E:Log outerLog();
   }
   Inner inner(){
      Inner:innerClock = outerClock;
      Inner:innerLog = outerLog;
   };
   C c3();

   inner:drip ~> c3:toc;
   c3:talk ~> outerLog:listen;

   expose {}
}
E appEnv() { E:Clock appClock(); E:Log appLog(); };

Outer myApp()(appEnv) {
  Outer:outerClock = appEnv:appClock;
  Outer:outerLog = appEnv:appLog;
};
```

2