34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

Integrating Arcadia and Capella with SysML v2

Ed Seidewitz, Model Driven Solutions
Tony Komar, Siemens
Karen Ryan, Siemens

Outline

- SysML v1
  - Background
  - Distiller Example
- Arcadia/Capella
  - Background
  - SysML v1 to Capella
- SysML v2
  - Background
  - SysML v1 to SysML v2
  - Capella to SysML v2
- Conclusion

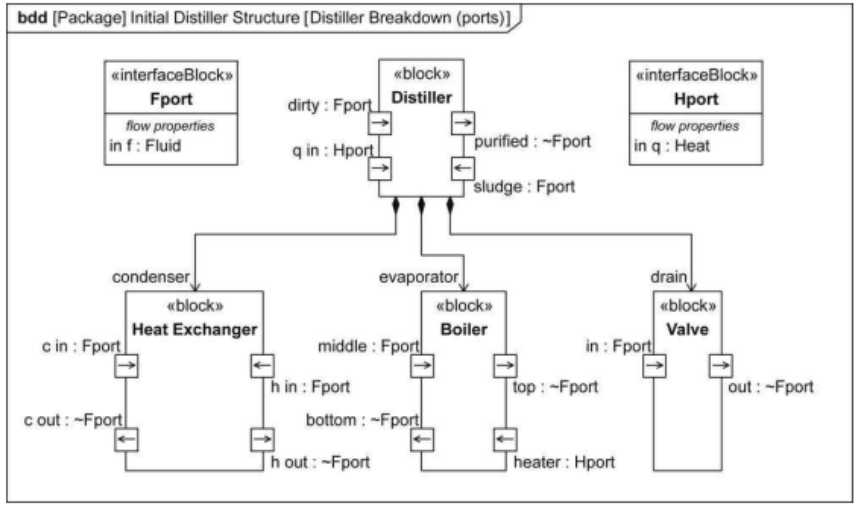
SysML v1

SysML v1: Background

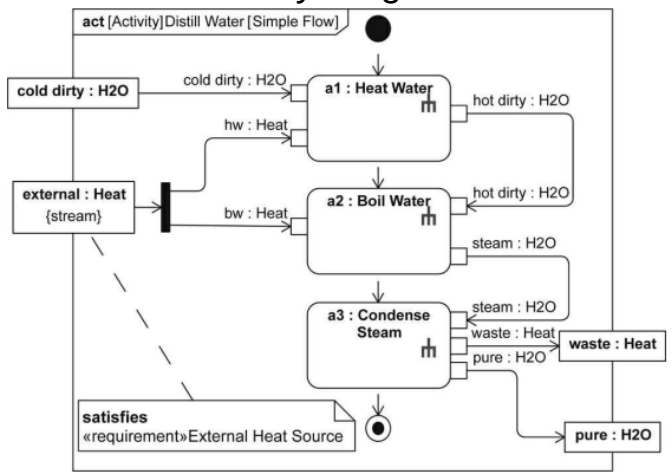
- Systems Modeling Language (SysML) “supports the specification, analysis, design, and verification and validation of complex systems that may include hardware, software, information, processes, personnel, and facilities”
- Designed as a profile/extension of the Unified Modeling Language (UML)
- Standardized by the Object Management Group (OMG)
  - Version 1.0 adopted in 2006
  - Version 1.7 adopted in 2022

SysML v1: Distiller Example

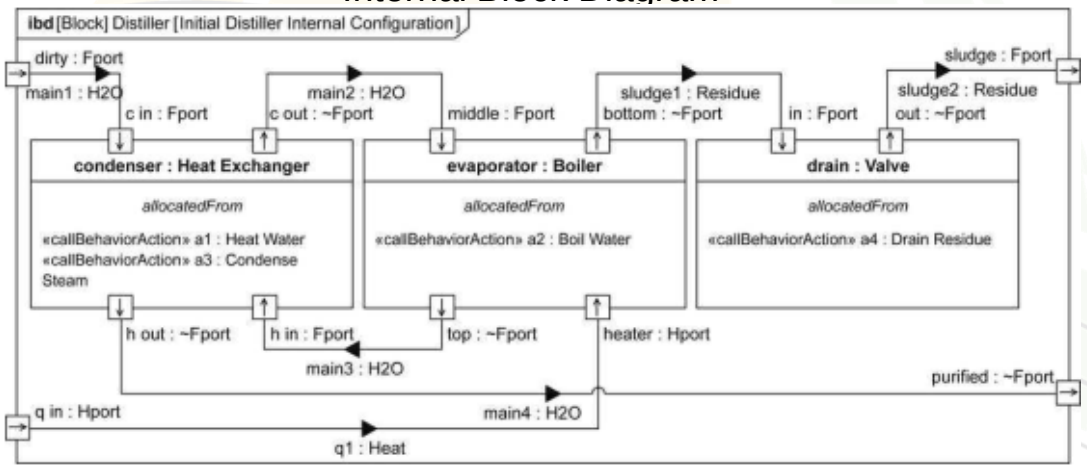
Block Definition Diagram



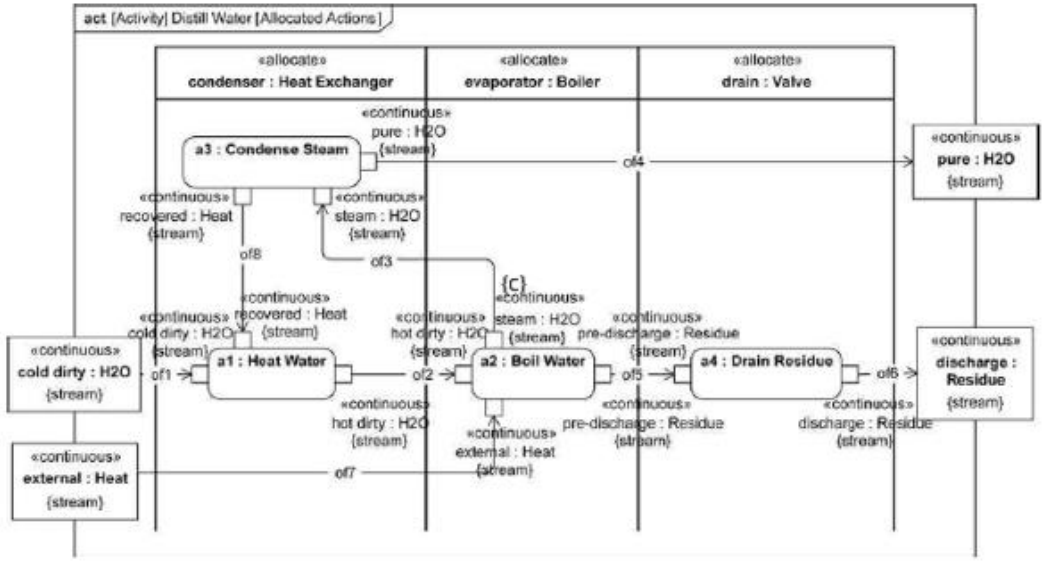
Activity Diagram



Internal Block Diagram



… with Swim Lanes / Allocations

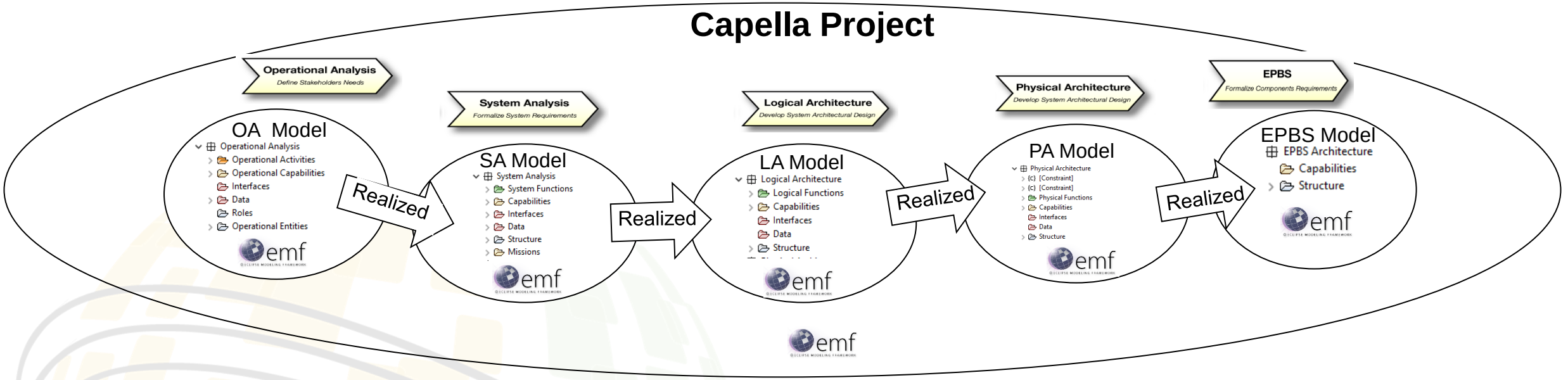


Arcadia and Capella

Arcadia: Background

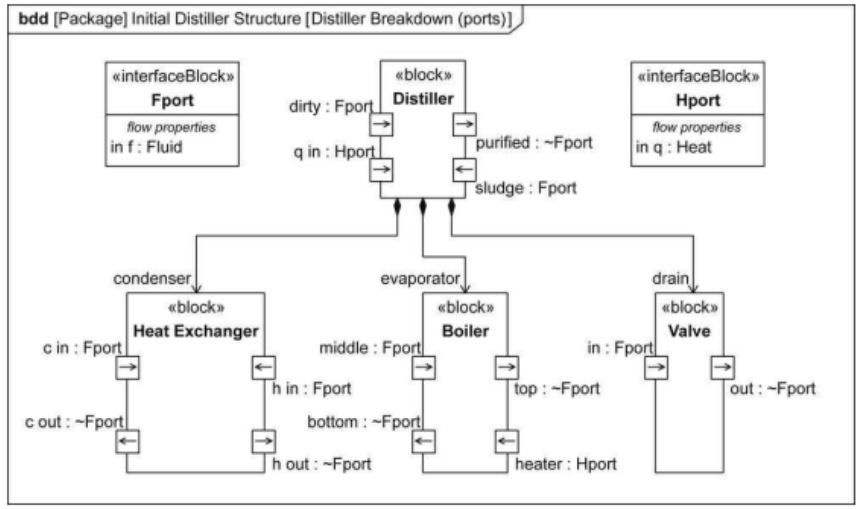
- Arcadia (Architecture Analysis and Design Integrated Approach) was originally developed by Thales from 2001 to 2009
  - Standardized by Association Française de Normalisation (AFNOR) in 2018
- Arcadia is a method, not just a modeling language
  - Conforms to ISO/IEC/IEEE 24641:2023 Systems and Software Engineering: Methods and tools for model-based systems and software engineering

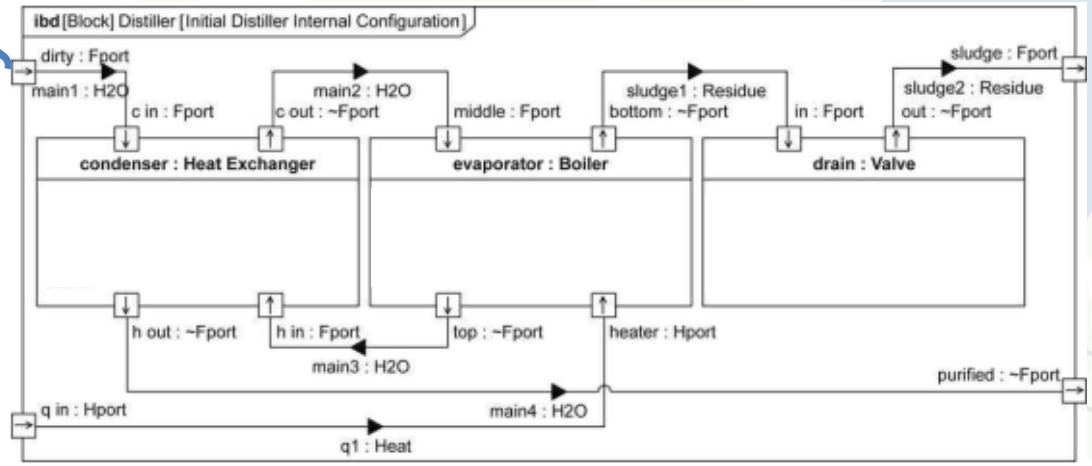
Capella: The tool for the Arcadia method



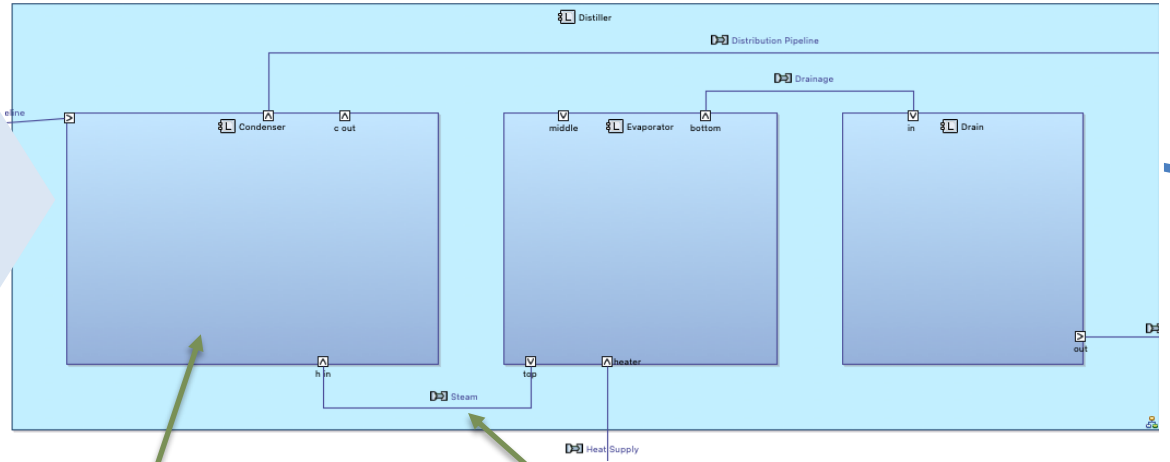
SysML v1 to Capella: Components

SysML v1





Instance-based component modeling



Capella

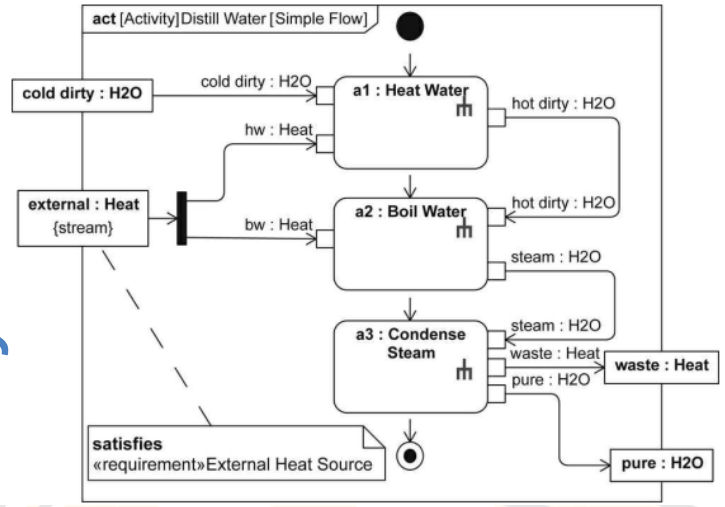
Logical Components

Component Exchanges

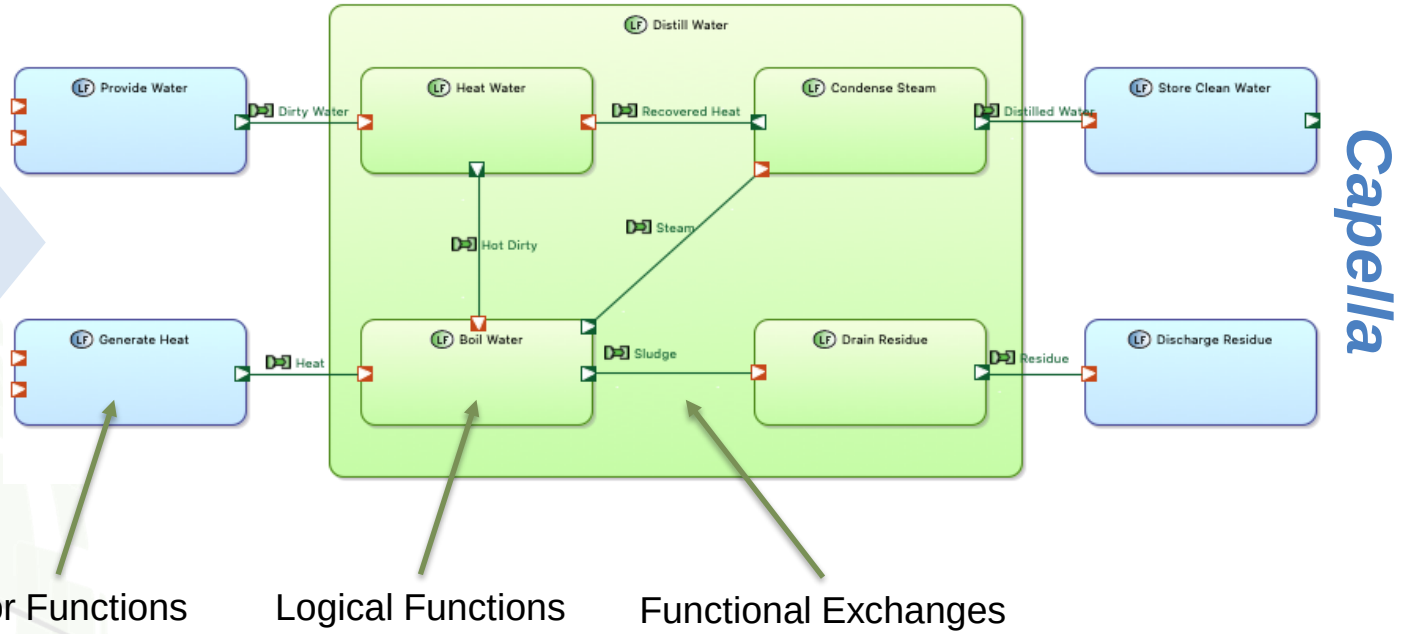
SysML v1 to Capella: Functions

Nested connection modeling

SysML v1

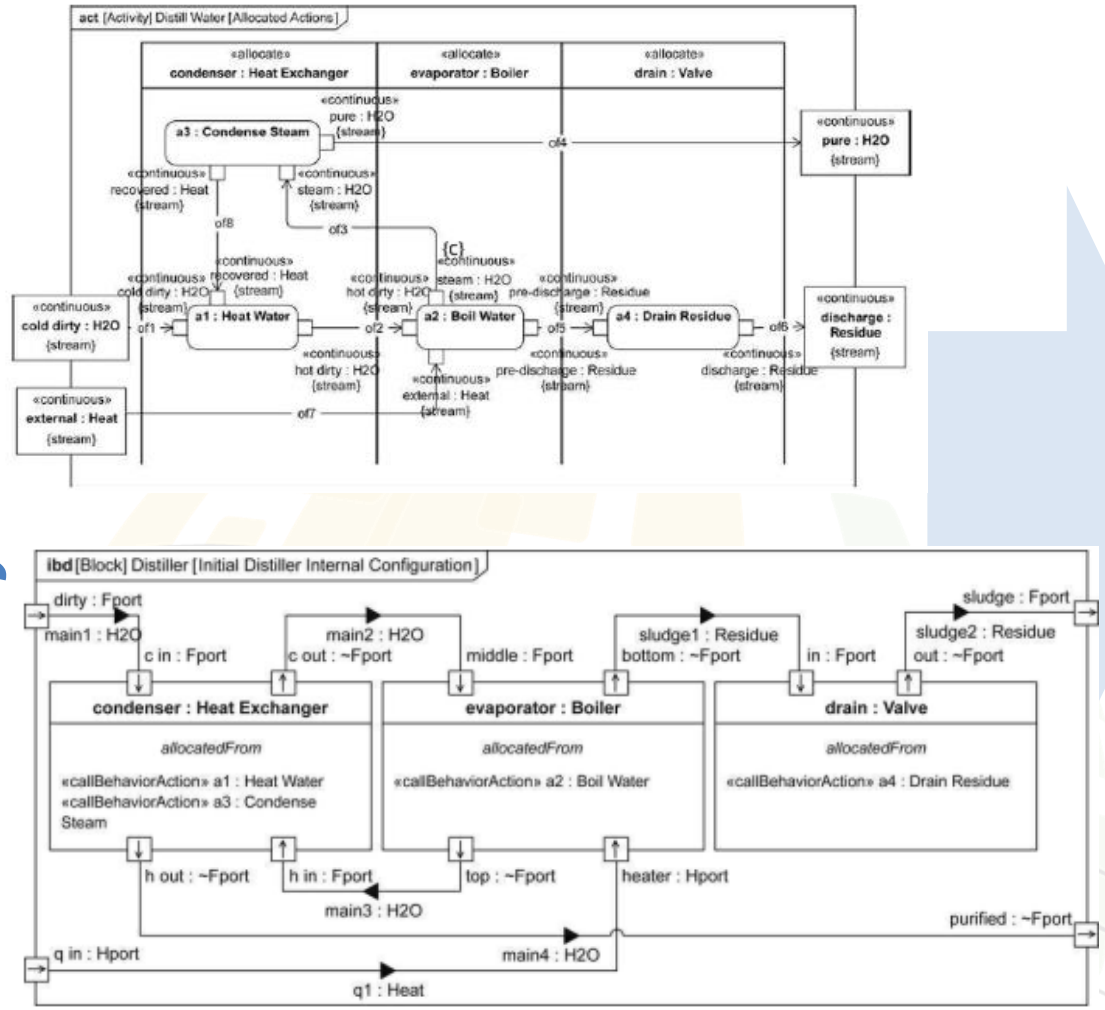


Capella



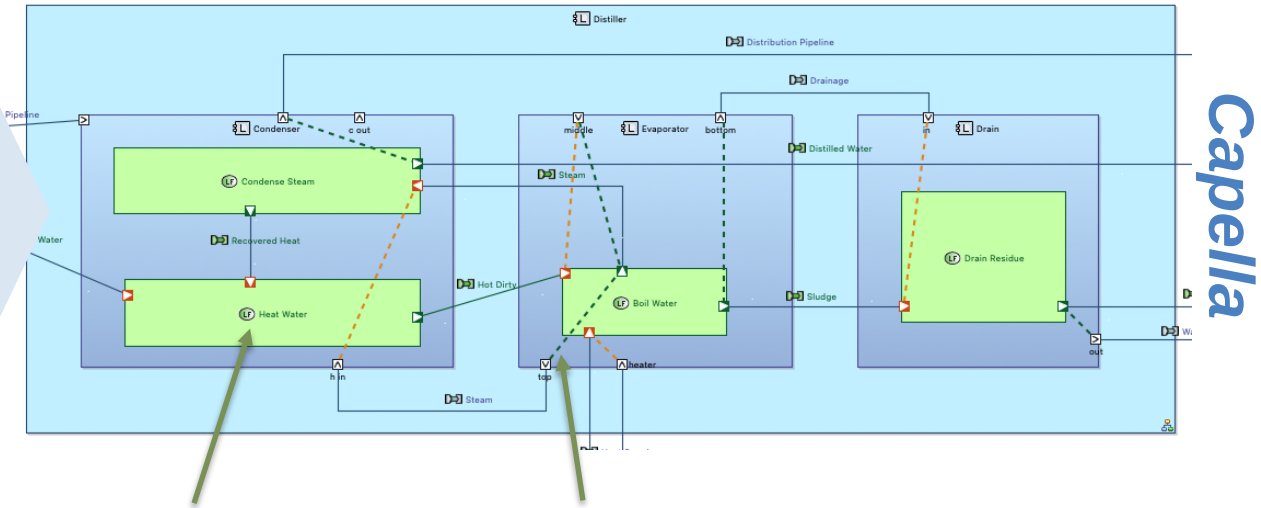
SysML v1 to Capella: Allocation

SysML v1



Natural functional allocation to components

Capella



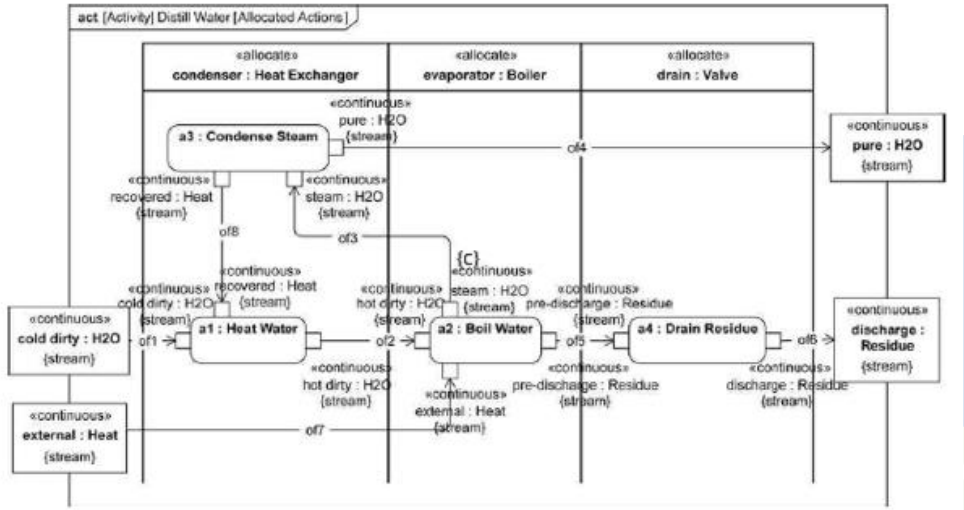
Component Functional Allocation

Port Allocation

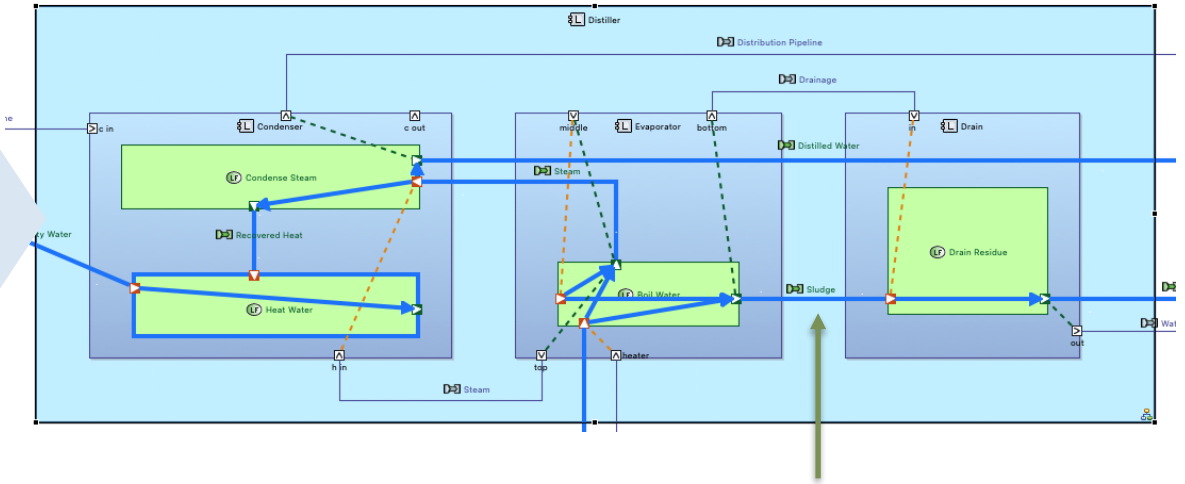
SysML v1 to Capella: Functional Chains

Functional chain modeling

SysML v1



Capella



Functional Chain

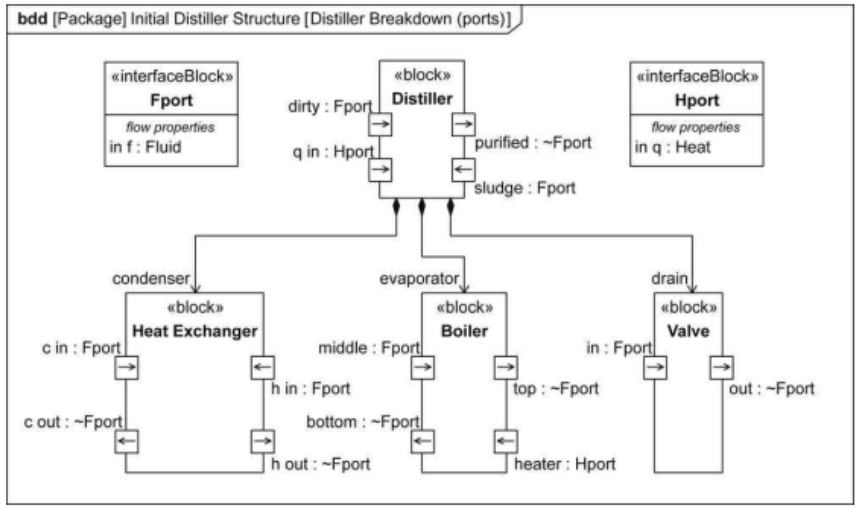
SysML v2

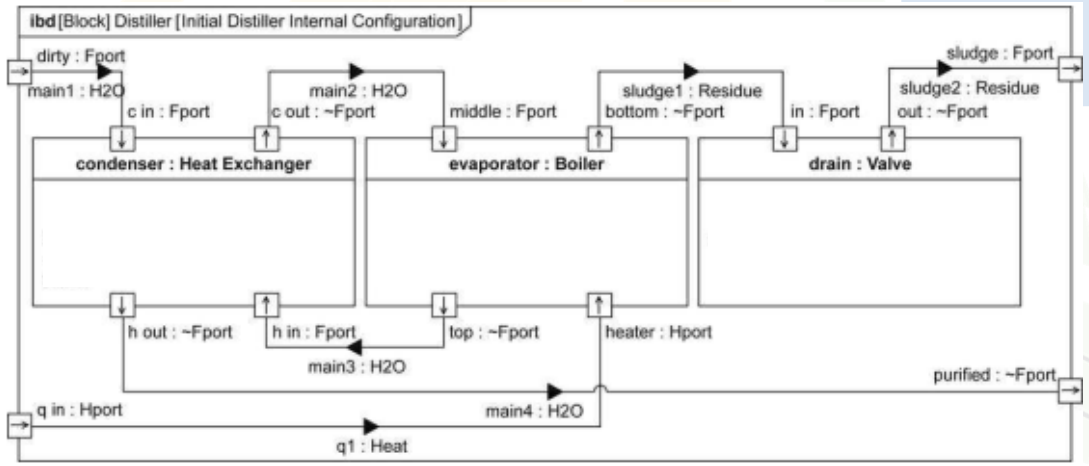
SysML v2: Background

- SysML v2 is the next generation systems modeling language intended to address limitations of SysML v1
- Based on a new Kernel Modeling Language (KerML), rather than UML
- Has a textual as well as a graphical notation
- Version 2.0 beta adopted by OMG in 2023
  - Currently planned to complete “finalization” in 2024

SysML v1 to v2: Components

SysML v1






```
part Distiller {
    port dirty : Fport;
    port 'q in' : Hport;
    port purified : ~Fport;
    port sludge : ~Fport;

    part Condenser {
        port 'c in' : Fport;
        port 'c out' : ~Fport;
        port 'h in' : Fport;
        port 'h out' : ~Fport;
    }
    part Evaporator {
        port top : ~Fport;
        port middle : Fport;
        port bottom : ~Fport;
        port heater : Hport;
    }
    part Drain {
        port 'in' : Fport;
        port 'out' : ~Fport;
    }

    interface dirty to Condenser.'c in';
    interface 'q in' to Evaporator.heater;
    interface Condenser.'c out' to Evaporator.middle;
    interface Evaporator.top to Condenser.'h in';
    interface Evaporator.bottom to Drain.'in';
    interface Condenser.'h out' to purified;
    interface Drain.'out' to sludge;
}
```

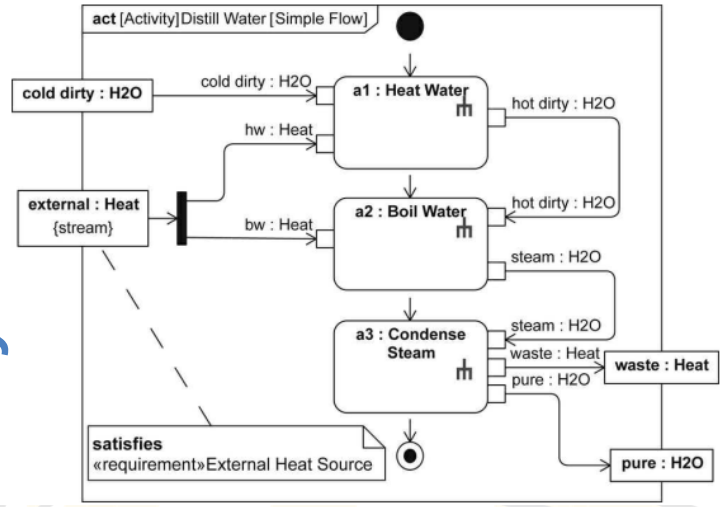
Usage-focused part decomposition

Connections between ports

SysML v2

SysML v1 to v2: Functions

SysML v1



Usage-focused action decomposition

Flows between parameters

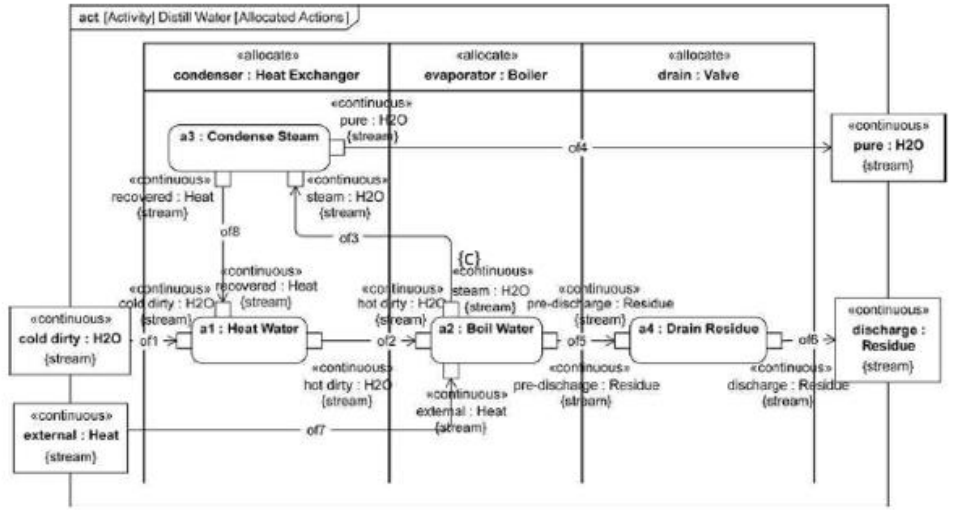
Action parameters

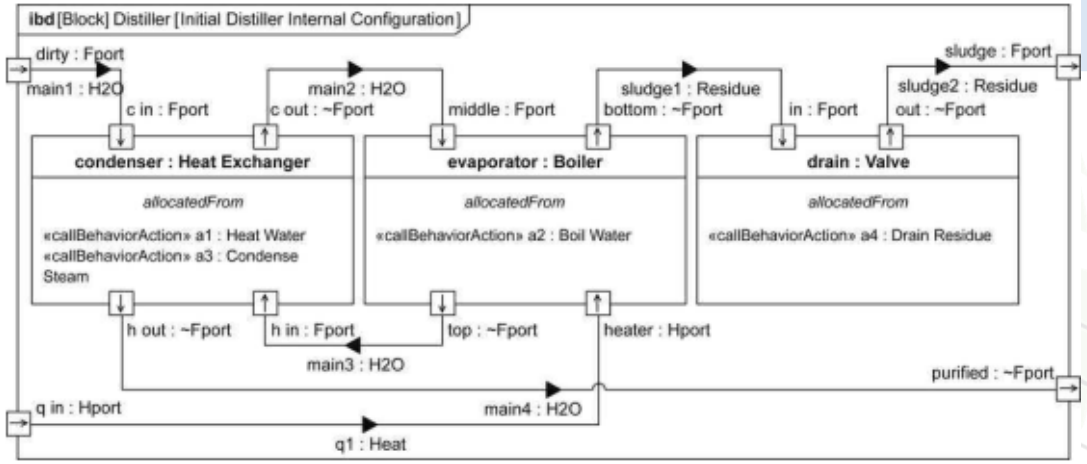
```
action 'Distill Water' {
    in 'cold dirty' : H2O; in external : Heat;

    flow main1 of H2O from 'cold dirty' to 'Heat Water'.'cold dirty';
    flow q0 of Heat from external to 'Heat Water'.hw;
    action 'Heat Water' {
        in hw : Heat; in 'cold dirty' : H2O;
        out 'hot dirty' : H2O;
    }

    flow q1 of Heat from external to 'Boil Water'.bw;
    flow main2 of H2O from 'Heat Water'.'hot dirty' to 'Boil Water'.'hot dirty';
    action 'Boil Water' {
        in bw : Heat; in 'hot dirty' : H2O;
      out steam : H2O; out sludge : Residue;
    }

    flow main3 of H2O from 'Boil Water'.steam to 'Condense Steam'.steam;
    action 'Condense Steam' {
        in steam : H2O;
        out pure : H2O; out waste : Heat;
    }
    flow main4 of H2O from 'Condense Steam'.pure to pure;
    flow main5 of H2O from 'Condense Steam'.waste to waste;

    flow sludge1 of Residue from 'Boil Water'.sludge to 'Drain Residue'.'in';
    action 'Drain Residue' {
        in 'in' : Residue;
        out 'out' : Residue;
    }
    flow sludge2 of Residue from 'Drain Residue'.'out' to sludge;

    out pure : H2O; out waste : Heat; out sludge : Residue;
}
```

SysML v2

SysML v1 to v2: Allocation

SysML v1





Parts perform actions

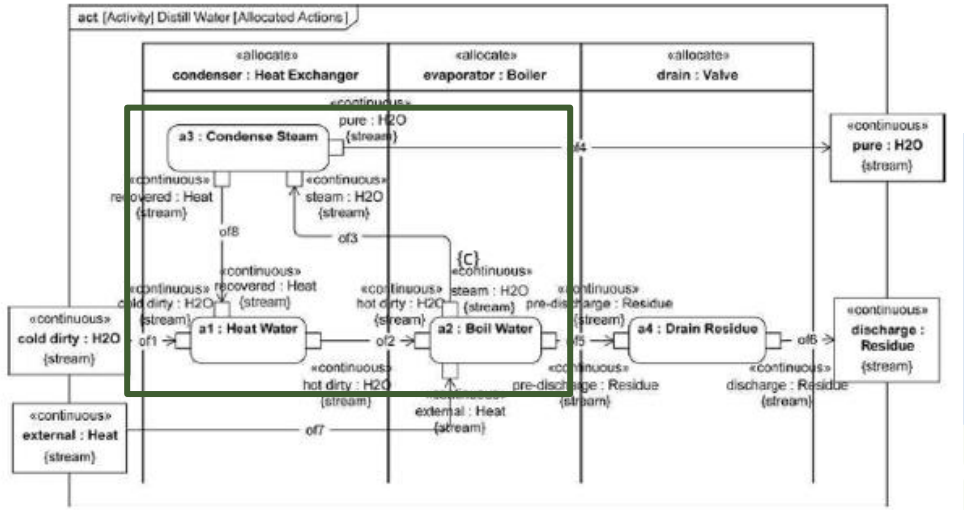
Interfaces perform flows

```
part Distiller { …
    part Condenser { …
        perform 'Distill Water'.'Heat Water';
        perform 'Distill Water'.'Condense Steam';
    }
    part Evaporator { …
        perform 'Distill Water'.'Boil Water';
    }
    part Drain { …
        perform 'Distill Water'.'Drain Residue';
    }

    interface 'q in'to Evaporator.heater {
        perform 'Distill Water'.q1;
    }
    interface dirty to Condenser.'c in'{
        perform 'Distill Water'.main1;
    }
    interface Condenser.'c out' to Evaporator.middle {
        perform 'Distill Water'.main2;
    }
    interface Evaporator.top to Condenser.'h in' {
        perform 'Distill Water'.main3;
    }
    interface Condenser.'h out'to purified {
        perform 'Distill Water'.main4;
    }
    interface Evaporator.bottom to Drain.'in' {
        perform 'Distill Water'.sludge1;
    }
    interface Drain.'out' to sludge {
        perform 'Distill Water'.sludge2;
    }
}
```

SysML v2

SysML v1 to v2: Functional Chains

SysML v1



Functional chain represented as an action…

```
action 'Boil and Condense Water' {
        perform Distiller.Condenser.'Heat Water';
        perform 'Distill Water'.'Boil Water';
        perform 'Distill Water'.'Condense Steam’;

        perform 'Distill Water'.main2;
        perform 'Distill Water'.main3;
}
```

…that performs selected system actions and flows

SysML v2

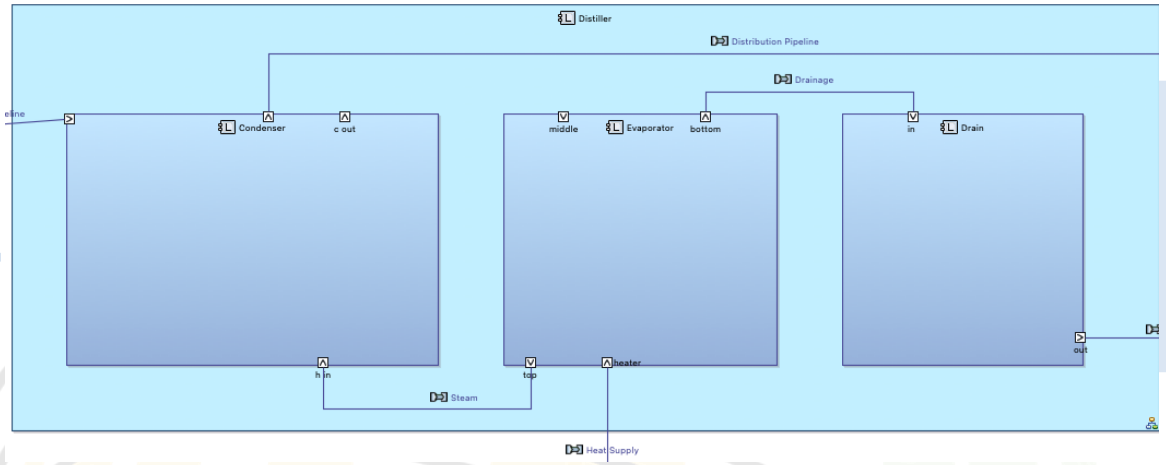
Capella and SysML v2

Capella to SysML v2: Components

```
part def LogicalComponent specializes Component {
    part ownedLogicalComponents[0..*] : LogicalComponent;
    ...
}
```

Library definitions represent Capella concepts

Capella




```
part Distiller : LogicalComponent {
    part Condenser subsets ownedLogicalComponents {
        in port 'h in' subsets ownedComponentPorts;
        out port 'c out' subsets ownedComponentPorts;
        out port 'CP 4' subsets ownedComponentPorts;
        in port 'c in' subsets ownedComponentPorts;
    }
    part Evaporator subsets ownedLogicalComponents {
        in port bottom subsets ownedComponentPorts;
        in top subsets ownedComponentPorts;
        out middle subsets ownedComponentPorts;
        out heater subsets ownedComponentPorts;
    }
    part Drain subsets ownedLogicalComponents {
        in port 'in' subsets ownedComponentPorts;
        out port 'out' subsets ownedComponentPorts;
    }

    interface Drainage : ComponentExchange
        connect Evaporator.bottom to Drain.'in';
    interface Steam : ComponentExchange
        connect Evaporator.top to Condenser.'h in';
    interface 'Hot Water 1' : ComponentExchange
        connect Condenser.'c out' to 'Diverter Assembly'.main2;
}
```

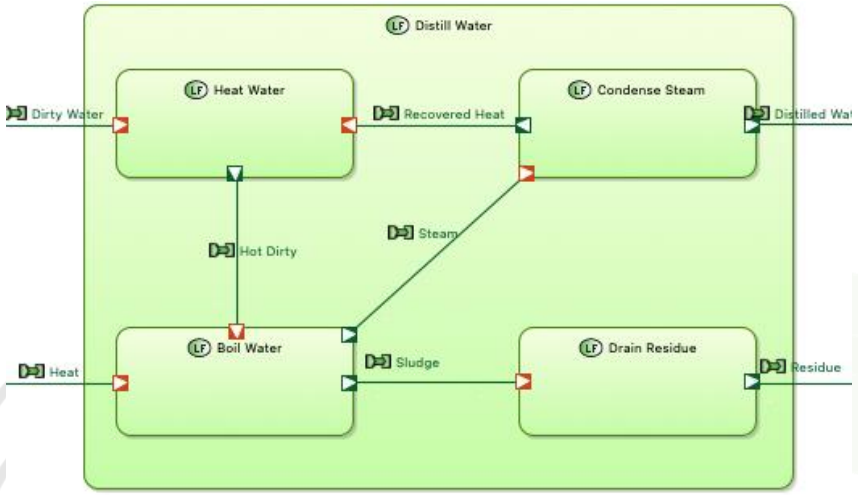
SysML v2

```
interface def ComponentExchange
        specializes AbstractComponentExchange {
    end port sourcePort : ComponentPort;
    end port targetPort : ComponentPort;
}
```

Capella to SysML v2: Functions

Capella

```
port def FunctionInputPort specializes FunctionPort {
      in item incomingExchangeItems[0..*] : ExchangeItem;
}
port def FunctionOutputPort specializes FunctionPort {
      out item outgoingExchangeItems[0..*] : ExchangeItem;
}
```



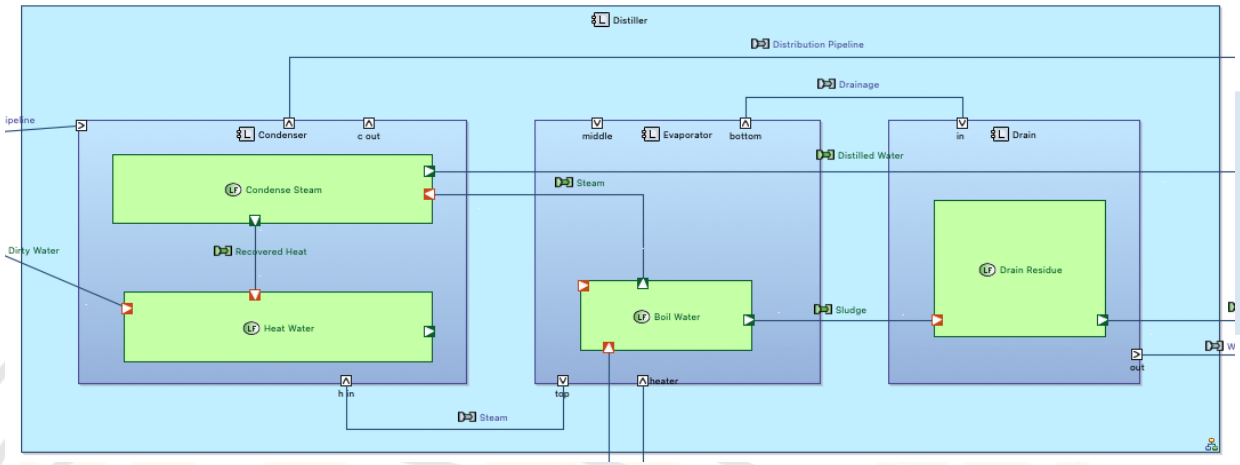

```
interface def FunctionalExchange {
      end port sourceFunctionOutputPort : FunctionOutputPort;
      end port targetFunctionInputPort : FunctionInputPort;
      message transfer of exchangedItems : ExchangeItem[0..*]
            from sourceFunctionOutputPort to targetFunctionInputPort;
}
```

SysML v2

```
action 'Distill Water' : LogicalFunction {
      action 'Heat Water' subsets ownedLogicalFunctions {
            port hw subsets inputs;
            port ‘cold dirty' subsets inputs;
            port ‘hot dirty' subsets outputs;
      }
      action 'Boil Water' subsets ownedLogicalFunctions {
            port bw subsets inputs;
            port ‘hot dirty' subsets inputs;
            port steam subsets outputs;
            port sludge subsets outputs;
      }
      action 'Condense Steam' subsets ownedLogicalFunctions {
            port steam subsets inputs;
            port pure subsets outputs;
            port waste subsets outputs;
      }
      action 'Drain Residue' subsets ownedLogicalFunctions {
            port ‘in' subsets inputs;
            port ‘out ' subsets outputs;
      }

      interface 'Hot Dirty' : FunctionalExchange
            connect 'Heat Water'.'hot dirty’ to 'Boil Water'.'hot dirty';
      interface 'Steam' : FunctionalExchange
            connect 'Boil Water'.steam to 'Condense Steam'.steam;
      interface 'Sludge' : FunctionalExchange
            connect 'Boil Water'.sludge to 'Drain Residue'.'in';
      interface 'Recovered Heat' : FunctionalExchange
            connect 'Condense Steam'.waste to 'Heat Water'.hw;
}
```

Capella to SysML v2: Allocation

```
abstract part def AbstractFunctionalBlock {
    ...
    perform action allocatedFunctions[0..*] : AbstractFunction;
}
```




```
allocation def ComponentFunctionalAllocation {
    end allocatedFunction : AbstractFunction
        subsets allocatingBlock.allocatedFunctions;
    end allocatingBlock : AbstractFunctionalBlock;
}
```

```
part Distiller : LogicalComponent {
    ...
    part Condenser subsets ownedLogicalComponents { ... }
    part Evaporator subsets ownedLogicalComponents { ... }
    part Drain subsets ownedLogicalComponents { ... }
    ...
}

action 'Distill Water' : LogicalFunction {
    action 'Heat Water'
        subsets ownedLogicalFunctions { ... }
    action 'Boil Water'
        subsets ownedLogicalFunctions { ... }
    action 'Condense Steam'
        subsets ownedLogicalFunctions { ... }
    action 'Drain Residue'
        subsets ownedLogicalFunctions { ... }
    ...
}

allocation : ComponentFunctionalAllocation
        allocate 'Distill Water'.'Condense Steam'
        to 'Logical System'.Condenser;
allocation : ComponentFunctionalAllocation
        allocate 'Distill Water'.'Heat Water'
        to 'Logical System'.Condenser;
allocation : ComponentFunctionalAllocation
        allocate 'Distill Water'.'Boil Water'
        to 'Logical System'.Evaporator;
allocation : ComponentFunctionalAllocation
        allocate 'Distill Water'.'Drain Residue'
        to 'Logical System'.Drain;
```

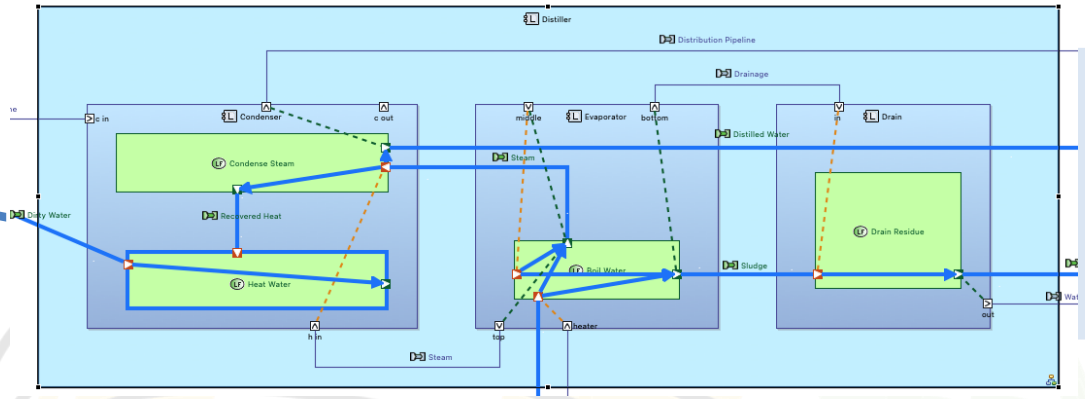
Capella

SysML v2

Capella to SysML v2: Functional Chains

```
action def FunctionalChain {
    …
    ref action involvedFunctions[0..*] : AbstractFunction;
    ref interface involvedFunctionalExchanges[0..*] : FunctionalExchange
        connect involvedFunctions.outputs[0..1]
        to involvedFunctions.inputs[0..1];
}
```

Capella




```
action ‘Chain - Distill and Store Water' : FunctionalChain {
    perform 'Distill Water'.'Heat Water’ subsets involvedFunctions;
    perform 'Distill Water'.'Boil Water’ subsets involvedFunctions;
    perform 'Distill Water'.'Condense Steam’ subsets involvedFunctions;
    perform 'Distill Water'.'Drain Residue’ subsets involvedFunctions;
    perform 'Distill Water'.'Heat Water’ subsets involvedFunctions;
    perform 'Store Clean Water' subsets involvedFunctions;
    perform 'Discharge Residue' subsets involvedFunctions;

    ref interface references 'Hot Water’
        subsets involvedFunctionalExchanges
        connect 'Heat Water'.'hot dirty’ to 'Boil Water'.'hot dirty';
    ref interface references 'Distill Water'.'Steam’
        subsets involvedFunctionalExchanges
        connect 'Boil Water'.steam to 'Condense Steam'.steam;
    ref interface references 'Distill Water'.'Recovered Heat’
        subsets involvedFunctionalExchanges
        connect 'Condense Steam'.waste to 'Heat Water'.hw;
    ref interface references 'Distilled Water’
        subsets involvedFunctionalExchanges
        connect 'Condense Steam'.pure to 'Store Clean Water’.pure;
    ref interface references 'Distill Water'.'Sludge’
        subsets involvedFunctionalExchanges
        connect 'Boil Water'.sludge to 'Drain Residue'.'in';
    ref interface references Residue
        subsets involvedFunctionalExchanges
        connect 'Drain Residue'.'out' to 'Discharge Residue'.'in';
}
```

SysML v2

Conclusion

- Arcadia/Capella and SysML v2 both address issues with SysML v1
  - Instance (usage) based modeling
  - Nested connection modeling
  - Function allocation and functional chains
- Capella models can also be represented in SysML v2
  - Using a SysML v2 model of syntactic and semantic concepts from Arcadia/Capella
- This demonstrates a general approach
  - Represent models from various tools/languages using SysML v2
  - To provide a syntactic and semantic basis for tool integration

34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024
#INCOSEIS