COMP6452 Lecture 9: Model-Driven Engineering

Ingo Weber | Principal Research Scientist & Team Leader
Architecture & Analytics Platforms (AAP) team
ingo.weber@data61.csiro.au

Conj. Assoc. Professor, UNSW Australia | Adj. Assoc. Professor, Swinburne University

www.data61.csiro.au

Agenda:

- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

CSIRO DATA 61

Agenda:

- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

CSIRO DATA 61

Motivation for MDE

- Blockchain is still a relatively new technology with limited tooling and documentation, skill set is rare
- According to a survey by Gartner, “23% of [relevant surveyed] CIOs said that blockchain requires the newest skills for implementation at various application areas, while 18% said that blockchain technique itself is one of the most difficult.”[1]
- Mistakes in smart contracts have led to massive economic loss, such as the DAO exploit.
- According to an ACS / Data61 report from 2019: 14 jobs per blockchain developer[2]

[1] Gartner Press Release. Gartner Survey Reveals the Scarcity of Current Blockchain Deployments. (3 May 2018). https://www.gartner.com/newsroom/id/3873790

[2] Bratanova, A., Devaraj, D., Horton, J., Naughtin, C., Kloester, B., Trinh, K., Weber, I., Dawson, D. (2019) Blockchain 2030: A Look at the Future of Blockchain in Australia. CSIRO Data61: Brisbane, Australia. https://www.researchgate.net/publication/332298704_Blockchain_2030_A_Look_at_the_Future_of_Blockchain_in_Australia

Model-driven Engineering

- A methodology for using models at various levels of abstraction and for different purposes during software development
  - Often concern-specific or domain-specific modelling languages
  - “Architecture-centric MDE”: architecture is the domain
  - Textual or graphical modelling notations can be used (i.e., not *necessarily* graphical)
  - Can cover static structures (like data models) or dynamic behaviour (like activity sequences)
- Requires you to formalize the architecture, can be an “architectural catalyst”
  - Helps to get a usable blueprint of the system
  - Makes architecture models immediately useful & motivates up-to-dateness
  - Architecture diagrams become **technical artefacts** of the system
- MDE can increase code quality and, even when starting from scratch, pay off within weeks

Model-driven Engineering – modelling levels

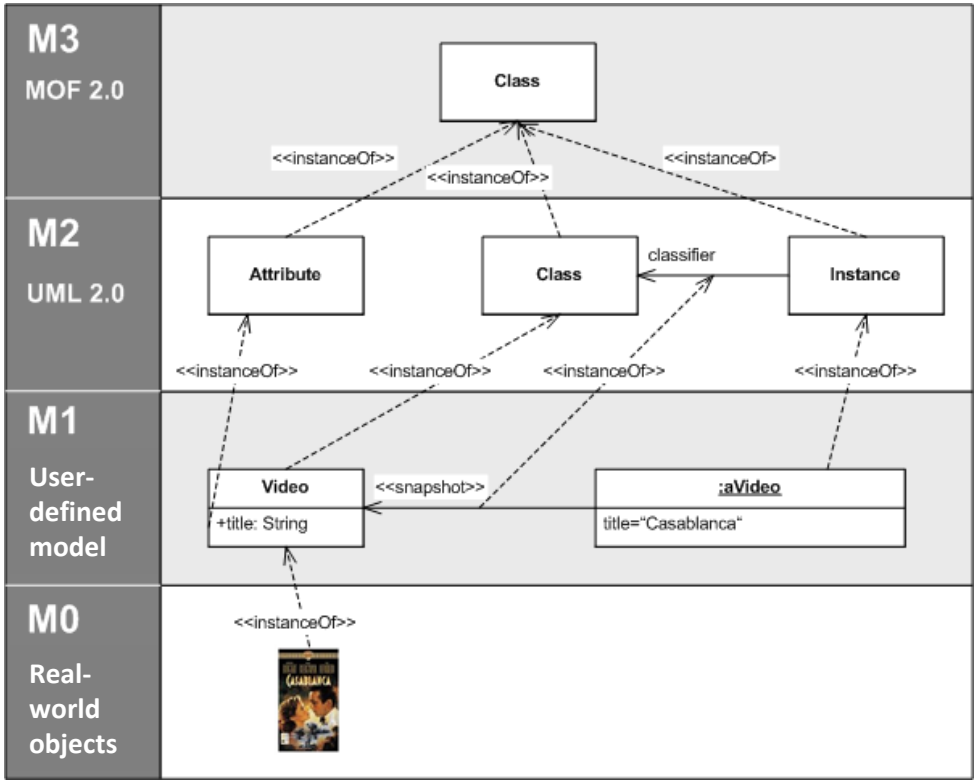
- Models are abstractions; what to model and what to abstract depends on the purpose
  - "[A] model which took account of all the variegation of reality would be of no more use than a map at the scale of one to one." Joan Robinson, 1962
- Low-level models: production code can be directly derived from the models
- High-level models: means of communication between business owners and developers implementing a system
- Intermediate levels can support model-based system analysis or system management tools
- Any level: can generate some of the code:
  - Code skeleton or early version of the code
  - Glue code and boilerplate code

MDE vs. MDD vs. MDA

- Model-driven development (MDD) vs. MDE:
  - The parts of MDE that focus on code generation
  - Think of software engineering vs. software development

- Model-driven architecture (MDA) vs. MDD/MDE:
  - OMG's standard for MDE, using OMG modelling languages – primarily Unified Modeling Language (UML)
  - Does not cover all aspects of an architecture, but architecture-centric models
  - Foundation standard: Meta-Object Facility (MOF) – see next slide
  - Other prominent concepts:
    - Platform-Independent Model (PIM)
    - Platform-Specific Model (PSM)
    - Transformations: model-to-model or model-to-code

Meta-Object Facility (MOF)



Adapted from https://en.wikipedia.org/wiki/Meta-Object_Facility

Advantages of MDE in the blockchain context

- Code generation can implement best practices and well-tested building blocks
  - Code can adhere to blockchain “standards” (like ERC-20, ERC-721, …)
- Improved productivity, especially for novices in a particular technology
  - However: someone should understand the code and review it
- Models can be independent of specific blockchain technologies or platforms
  - Avoid lock-in to specific blockchain technologies
- Models are often easier to understand than code – particularly useful in communicating with business partners about smart contracts
  - Facilitates building trust
  - Domain experts can understand how their ideas are represented in the system

Agenda:

- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

CSIRO DATA 61

Fungible and Non-fungible Tokens

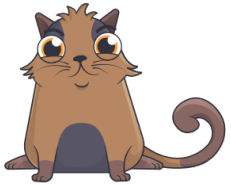
Recap from Lecture 1

可替代token 不可替代token

- Fungible tokens: interchangeable
  - E.g., $2 coin, $10 note
  - Main concern: how many?
  - Ethereum: ERC20 standard
    - https://theethereum.wiki/w/index.php/ERC20_Token_Standard
    - Example: OmiseGO (OMG).

      “The OmiseGO blockchain comprises a decentralized exchange, liquidity provider mechanism, clearinghouse messaging network, and asset-backed blockchain gateway. … It uses the mechanism of a protocol token to create a proof-of-stake blockchain to enable enforcement of market activity amongst participants. Owning OMG tokens buys the right to validate this blockchain, within its consensus rules.”

- Non-fungible tokens
  - E.g., houses, cars, patents
  - Main concern: which ones?
  - Ethereum: ERC721
    - https://github.com/ethereum/EIPs/issues/721
    - Example: cryptokitties https://www.cryptokitties.co/



    - Kitties are non-fungible, individual, and their appearance depends on the individual features

MDE for data structures and tokens

- Approach:
  - Enter high-level information
  - Model data structure (variables, types) – not for fungible tokens
  - Model relationships to other types / tokens
  - Select features

  → Code is generated directly – deploy or customize
- Feature examples:
  - Fungible tokens:
    - Can be minted? Burnt? By whom?
  - Non-fungible tokens
    - Include standard method(s) for sale
    - One contract for all tokens or one per token?
- Code generated is compliant with standards

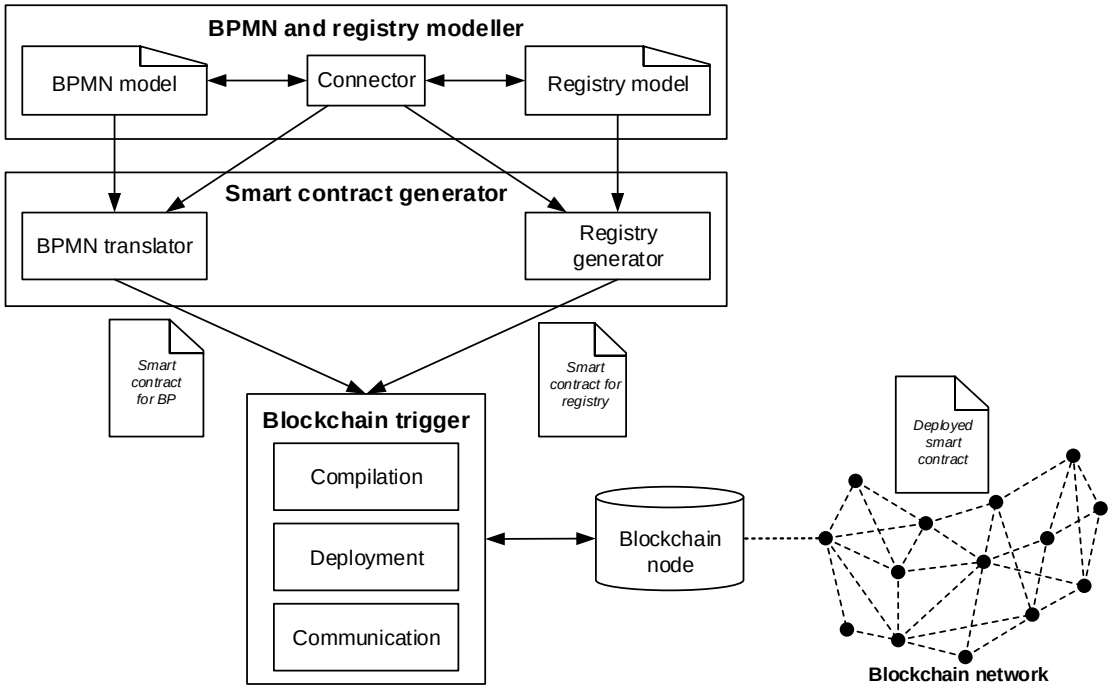
  → interface syntax and semantics

Task

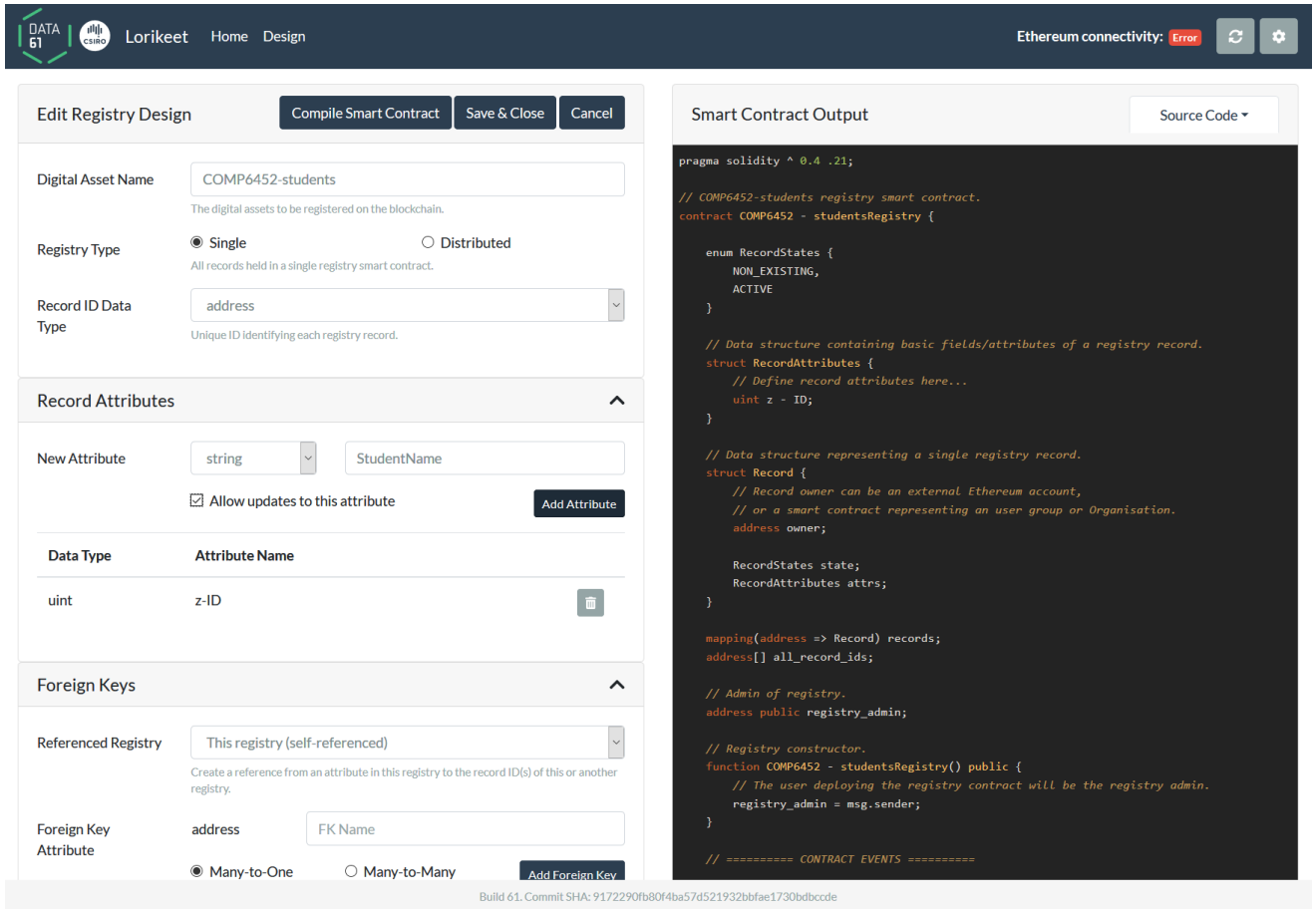
- In the next 3 minutes:
  - Pick a neighbour for later discussion
  - Open a new page / entry in your note pad / note-taking app
  - Think about what data to model for students of this class
  - Write down your top 4-10 data fields
  - Discuss with your neighbour
  - Think about any class-specific functions that the tool won't support & write it down

Data61 tool: Lorikeet

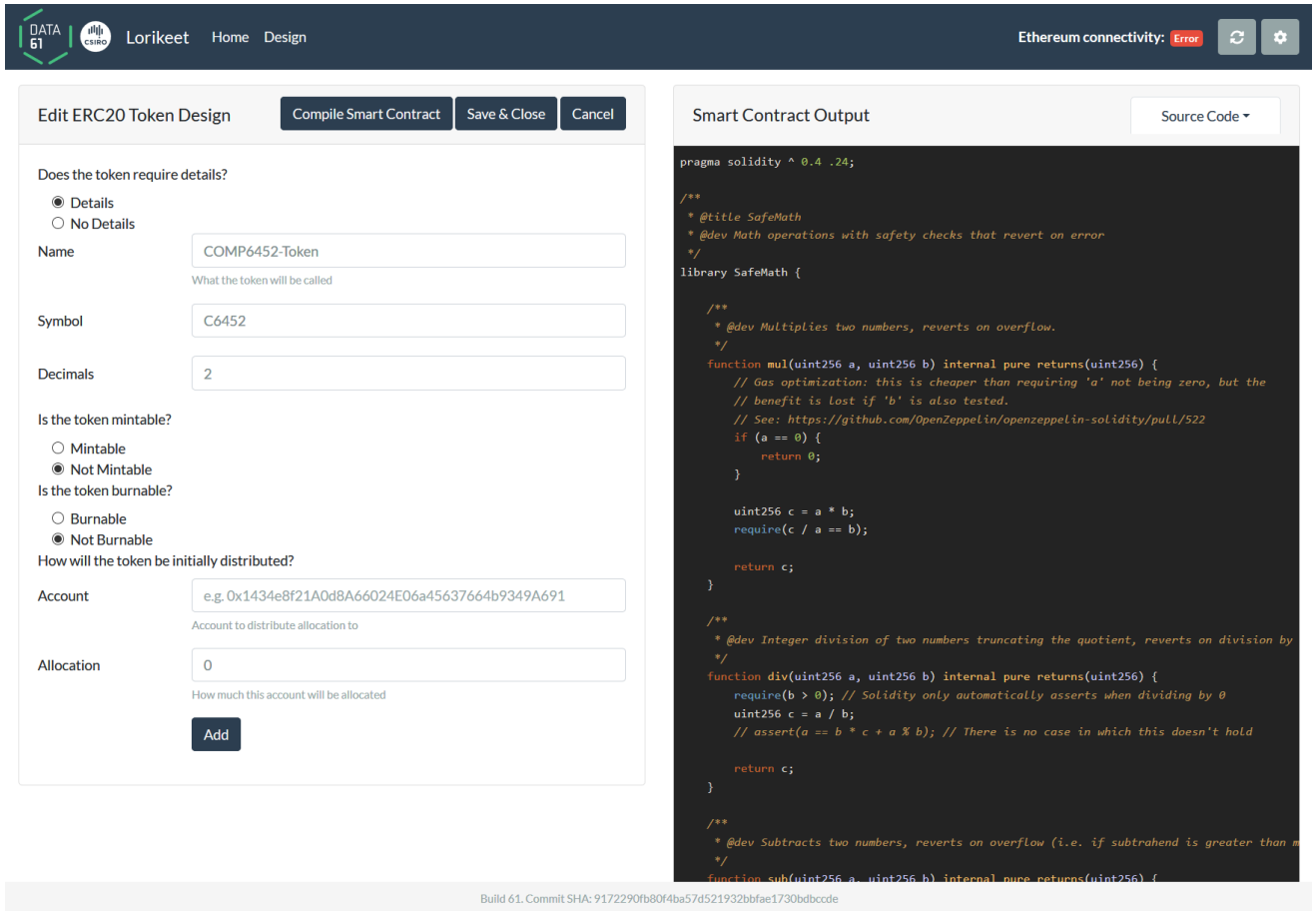
- Lorikeet: automatic generate smart contracts from BPMN models/registry data schema



Design time for data / non-fungible tokens



Design time for fungible tokens

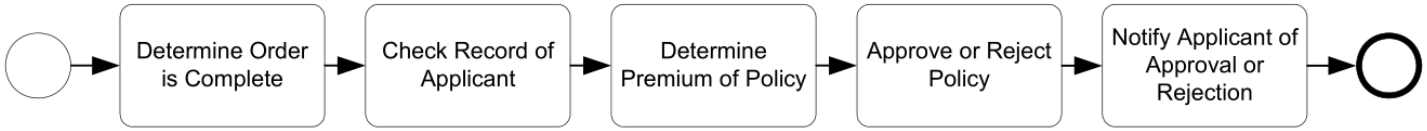


Agenda:

- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

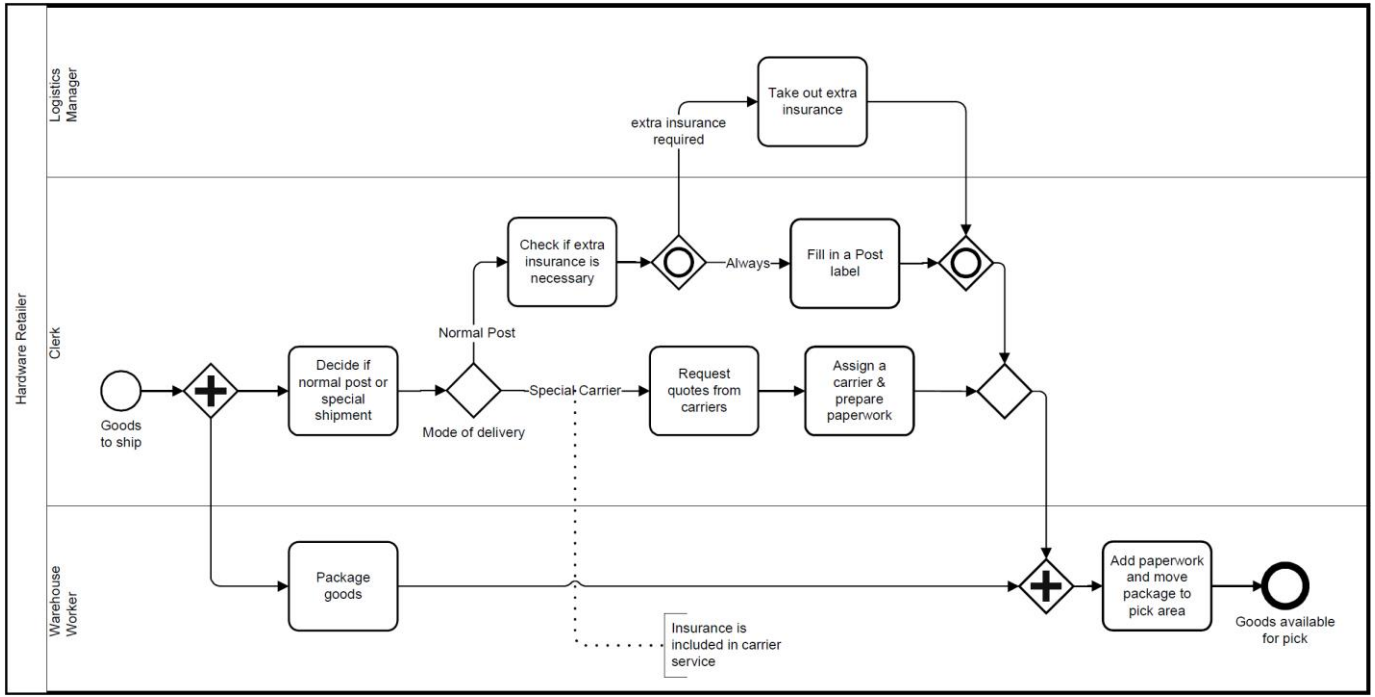
Business Processing Modelling: basic terminology

- BPMN example: insurance application processing



- Tasks: an activity that cannot be subdivided (unit of work)
- Sequence: some tasks need to be performed in a strict order
- Choice: certain tasks do not always need to be carried out
- Parallel: some tasks can be performed in parallel
- Synchronisation: some tasks need to wait for the result of previous tasks
- Iteration: some tasks need to be repeated

BPMN example: shipment process of a hardware retailer

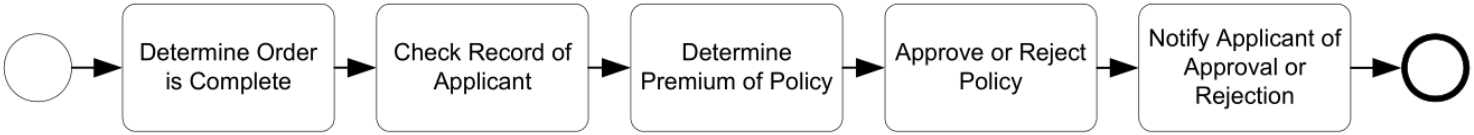


Source: BPMN 2.0 by Example, OMG Document Number dtc/2010-06-02

Business Process Model and Notation (BPMN): Motivation

- Focus of earlier workflow languages: clear execution semantics (e.g., PetriNets), expressivity (e.g., YAWL), interoperability (e.g., BPEL)
- Separate set of languages: conceptual process modelling notations without clear execution semantics, etc.
- Before BPMN, no notation existed that appealed to business users and has (somewhat) clear execution semantics

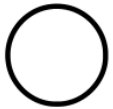
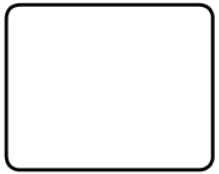
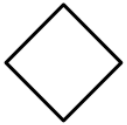
  → BPMN addresses this "niche"
- OMG Spec 2.0.2 in 2013: http://www.omg.org/spec/BPMN/2.0.2/PDF
  - From p.56 on, the elements are explained
  - (v1.0 released in 2006, v2.0 in 2011)
- BPMN example: private process



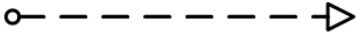
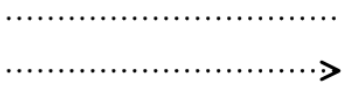
BPMN diagram elements

- Flow Objects
  - Events
  - Activities
  - Gateways
- Connecting Objects
  - Sequence Flow
  - Message Flow
  - Association
- Swimlanes
  - Pools
  - Lanes
- Artefacts
  - Data Object
  - Group
  - Annotation

Core flow objects (from BPMN Spec)

Element	Description	Notation
Event	An Event is something that "happens" during the course of a Process (see page 235) or a Choreography (see page 339). These Events affect the flow of the model and usually have a cause (*trigger*) or an impact (*result*). Events are circles with open centers to allow internal markers to differentiate different *triggers* or *results*. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.	
Activity	An Activity is a generic term for work that company performs (see page 149) in a Process. An Activity can be atomic or non-atomic (compound). The types of Activities that are a part of a Process Model are: Sub-Process and Task, which are rounded rectangles. Activities are used in both standard Processes and in Choreographies.	
Gateway	A Gateway is used to control the divergence and convergence of Sequence Flows in a Process (see page 147) and in a Choreography (see page 335). Thus, it will determine branching, forking, merging, and joining of paths. Internal markers will indicate the type of behavior control.	

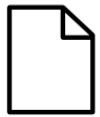
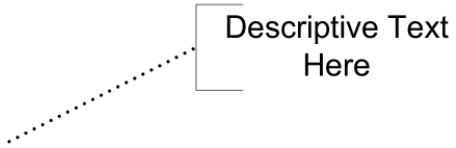
Connecting objects (from BPMN Spec)

Sequence Flow	A Sequence Flow is used to show the order that Activities will be performed in a Process (see page 95) and in a Choreography (see page 320).	
Message Flow	A Message Flow is used to show the flow of Messages between two *Participants* that are prepared to send and receive them (see page 113). In BPMN, two separate Pools in a Collaboration Diagram will represent the two *Participants* (e.g., PartnerEntities and/or PartnerRoles).	
Association	An Association is used to link information and Artifacts with BPMN graphical elements (see page 65). Text Annotations (see page 69) and other Artifacts (see page 64) can be Associated with the graphical elements. An arrowhead on the Association indicates a direction of flow (e.g., data), when appropriate.	

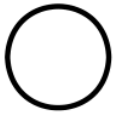
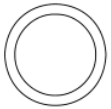
Swimlanes (from BPMN Spec)

Pool	A Pool is the graphical representation of a *Participant* in a Collaboration (see page 113). It also acts as a "swimlane" and a graphical container for partitioning a set of Activities from other Pools, usually in the context of B2B situations. A Pool MAY have internal details, in the form of the Process that will be executed. Or a Pool MAY have no internal details, i.e., it can be a "black box."	
Lane	A Lane is a sub-partition within a Process, sometimes within a Pool, and will extend the entire length of the Process, either vertically or horizontally (see on page 304). Lanes are used to organize and categorize Activities.	

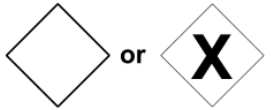
Artefacts (from BPMN Spec)

Data Object	Data Objects provide information about what Activities require to be performed and/or what they produce (see page 204), Data Objects can represent a singular object or a collection of objects. Data Input and Data Output provide the same information for Processes.	
Message	A Message is used to depict the contents of a communication between two *Participants* (as defined by a business PartnerRole or a business PartnerEntity—see on page 91).	
Group (a box around a group of objects within the same category)	A Group is a grouping of graphical elements that are within the same Category (see page 68). This type of grouping does not affect the Sequence Flows within the Group. The Category name appears on the diagram as the group label. Categories can be used for documentation or analysis purposes. Groups are one way in which Categories of objects can be visually displayed on the diagram.	
Text Annotation (attached with an Association)	Text Annotations are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram (see page 69).	

Start, intermediate, end events (from BPMN Spec)

Start	As the name implies, the Start Event indicates where a particular Process (see page 235) or Choreography (see page 339) will start.	Start 
Intermediate	Intermediate Events occur between a Start Event and an End Event. They will affect the flow of the Process (see page 237) or Choreography (see page 340), but will not start or (directly) terminate the Process.	Intermediate 
End	As the name implies, the End Event indicates where a Process (see page 245) or Choreography (see page 343) will end.	End

Gateways (from BPMN Spec)

Gateway Control Types	Icons within the diamond shape of the Gateway will indicate the type of flow control behavior. The types of control include: • Exclusive decision and merging. Both Exclusive (see page 286) and Event-Based (see page 296) perform exclusive decisions and merging Exclusive can be shown with or without the “X” marker. • Event-Based and Parallel Event-based gateways can start a new instance of the Process. • Inclusive Gateway decision and merging (see page 291). • Complex Gateway -- complex conditions and situations (e.g., 3 out of 5; page 294). • Parallel Gateway forking and joining (see page 292). Each type of control affects both the incoming and outgoing flow.	**Exclusive**  or **Event-Based** **Parallel Event-Based** **Inclusive** **Complex** **Parallel**

Task

- In the next 5 minutes, model the process of taking this class:
  - What are the activities?
  - In which order do they occur?
  - What are decision points in the process, what can happen?
  - Draw a model & discuss with your neighbour

Agenda:

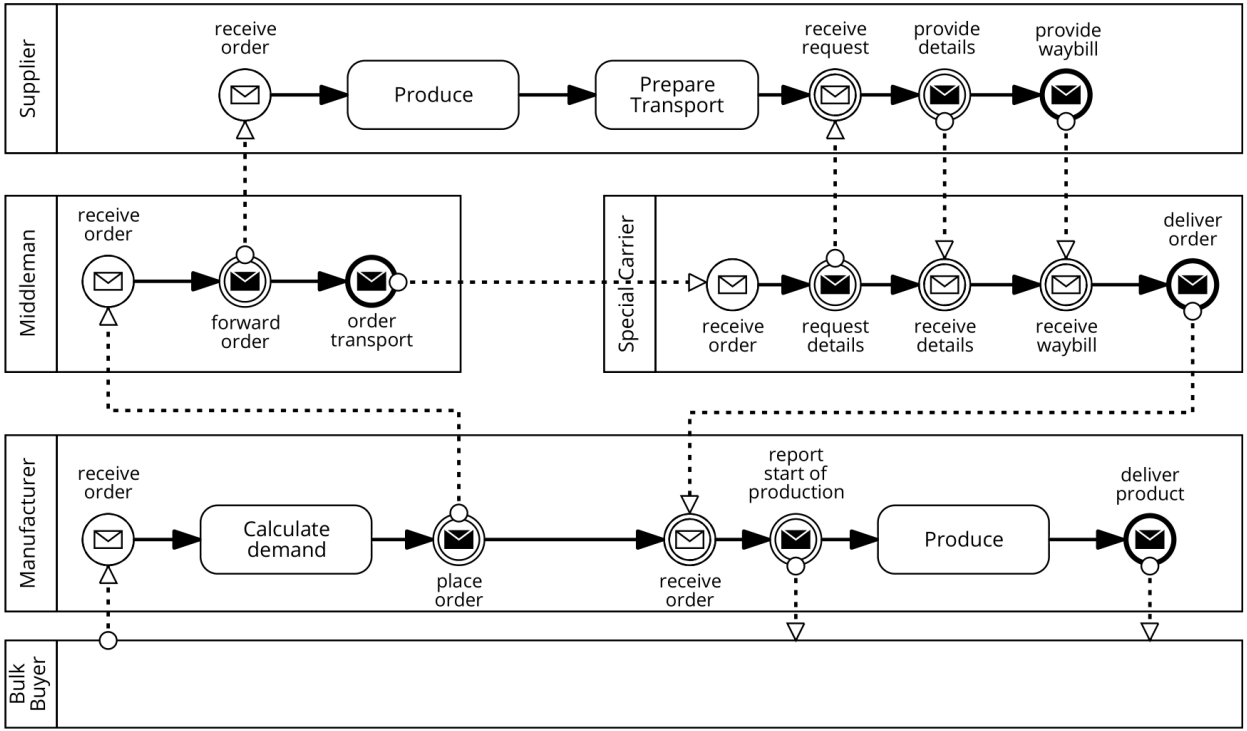
- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

CSIRO DATA 61

MDE for Processes – Motivation

- Integration of business processes across organizations: a key driver of productivity gains
- Collaborative process execution
  - Doable when there is trust – supply chains can be tightly integrated
  - Problematic when involved organizations have a **lack of trust** in each other
    → if 3+ parties should collaborate, where to execute the process that ties them together?
  - Common situation in “coopetition”

Motivation: example



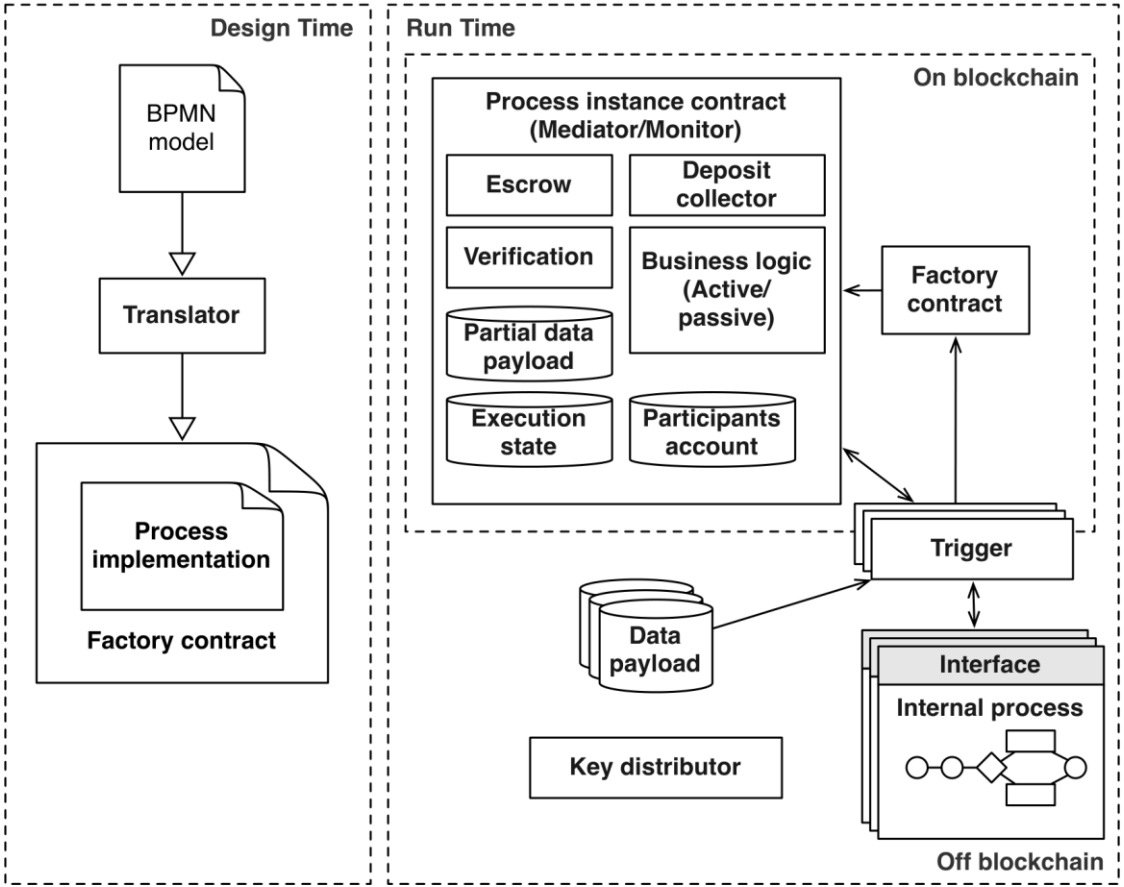
Issues:

- Knowing the status, tracking correct execution
- Handling payments
- Resolving conflicts

Approach in a nutshell

- Goal: implement collaborative business processes as smart contracts
  - Translate (enriched) BPMN to smart contract code
  - Triggers act as bridge between Enterprise world and blockchain
  - Smart contract provides:
    - Independent, global process monitoring
    - Conformance checking and process enforcement: only expected messages are accepted, only from the respective role
    - Automatic payments & escrow
    - Data transformation
    - Encryption
- Processes can make use of data / token contracts
  - Process activity to hand over title to a car / shipment / grain / ..., e.g., in exchange for fungible tokens

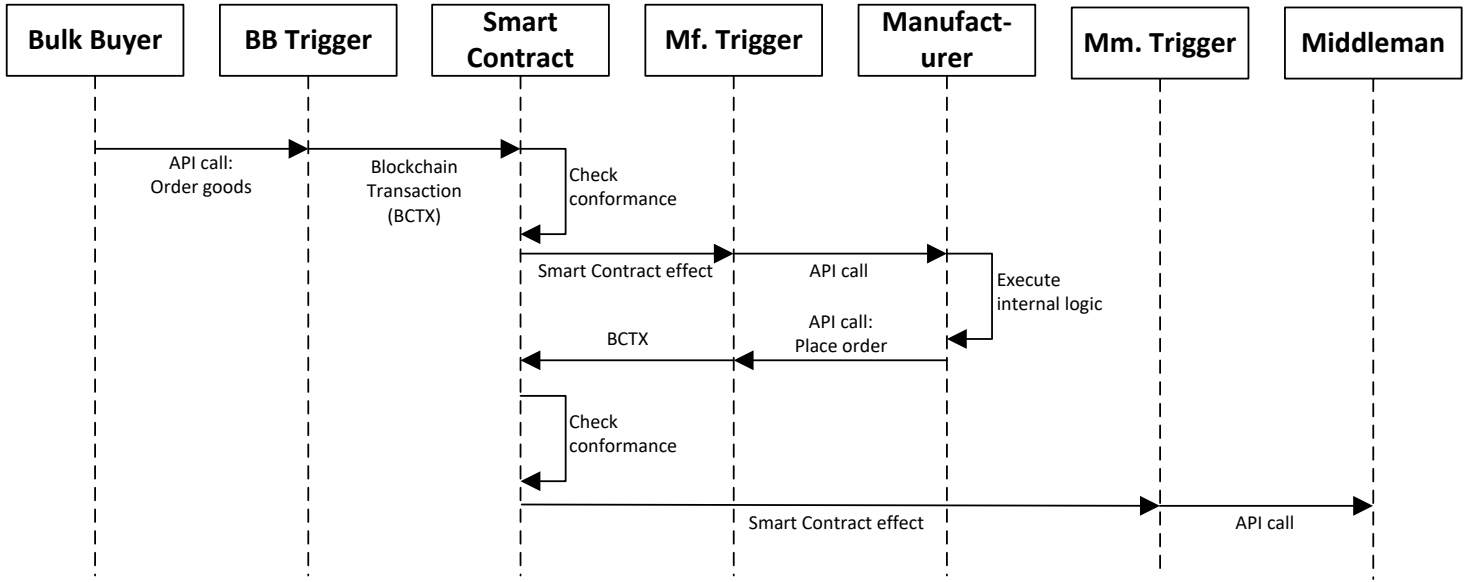
Architecture overview



Runtime

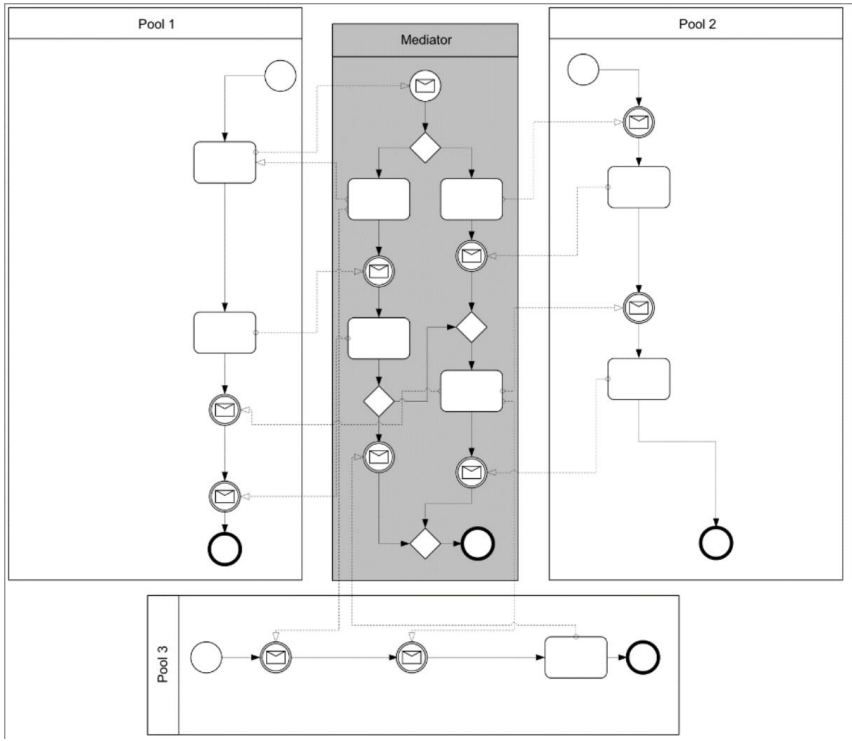
- Instantiation:
  - New *instance contract* per process instance
  - Assign accounts to roles during initialization
  - Exchange keys and create secret key for the instance
- Messaging:
  - Instead of sending direct WS calls: send through triggers & smart contract
  - Instance contract handles:
    - Global monitoring
    - Conformance checking
    - Automated payments*
    - Data transformation*

Runtime – UML Sequence Diagram

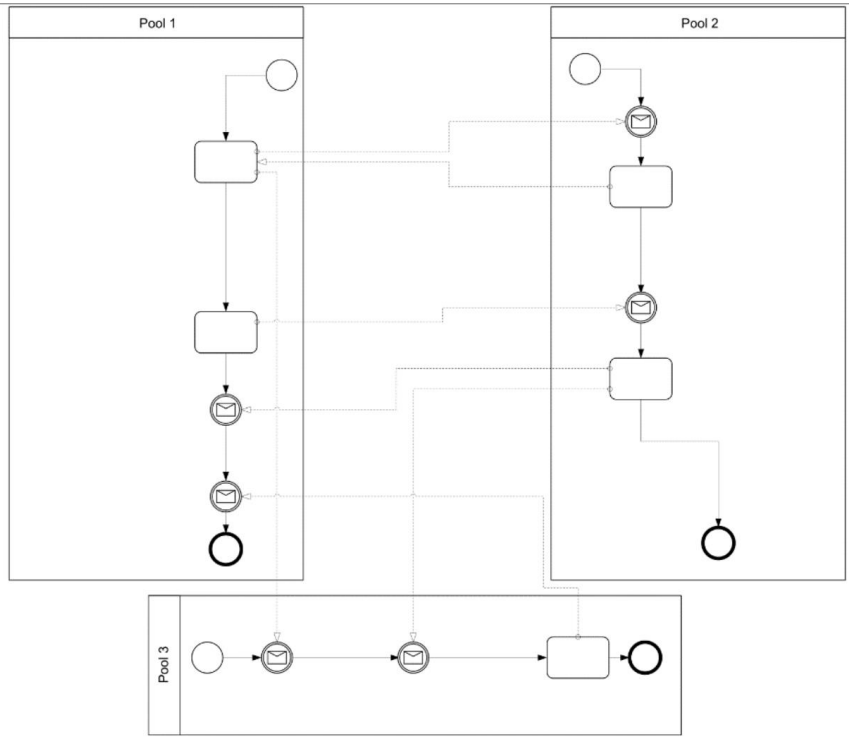


Collaborative processes: variants

Mediator (orchestration)

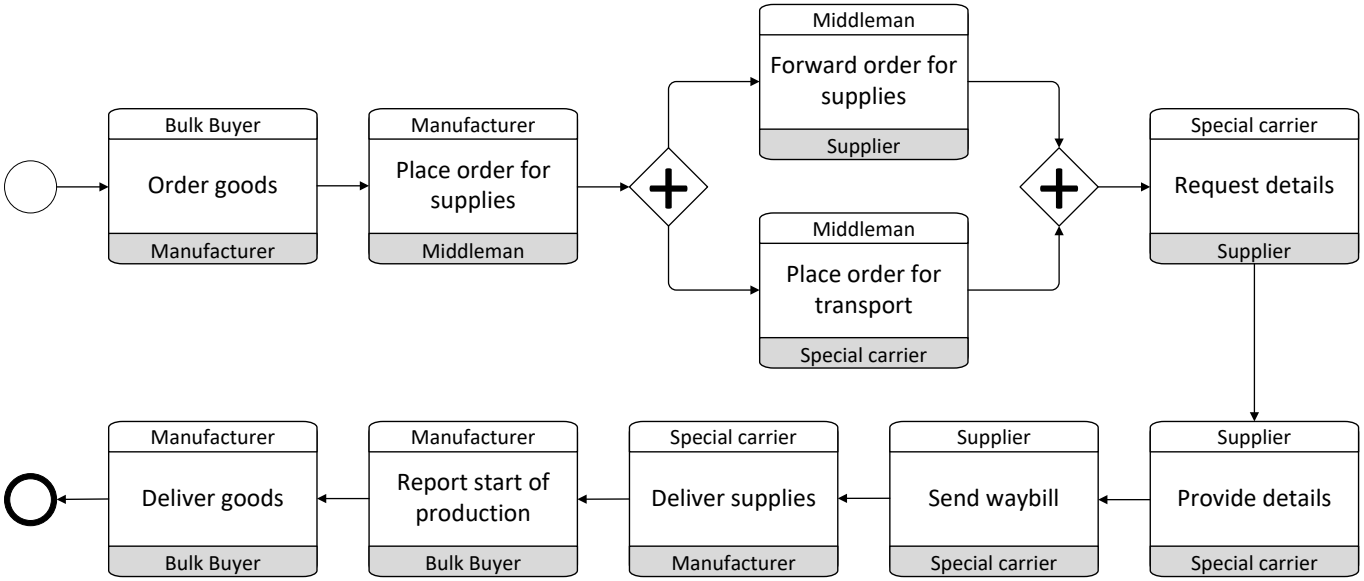


Choreography → C-Monitor

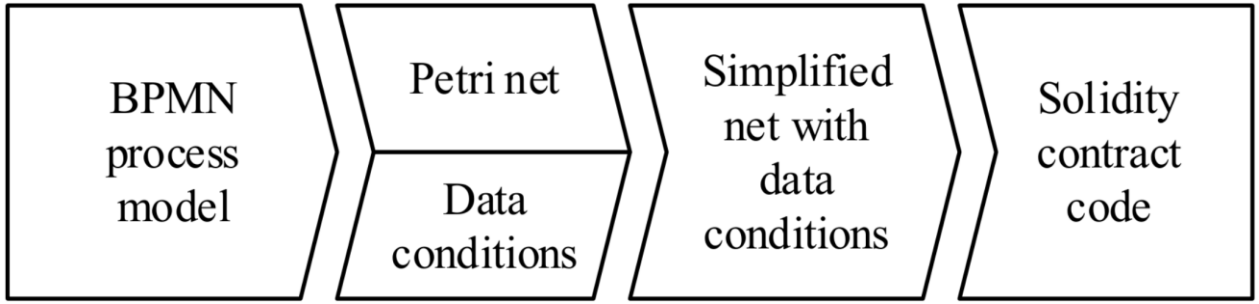


Translator

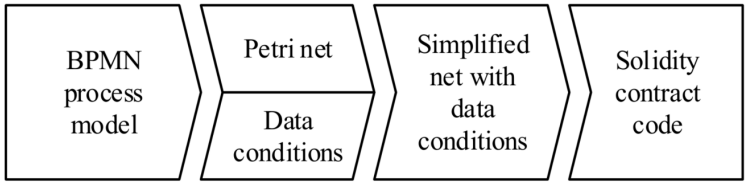
- Translate subset of BPMN elements to *Solidity*
  - BPMN Choreography diagrams or regular BPMN models with pools
  - BPMN Choreography diagram example:



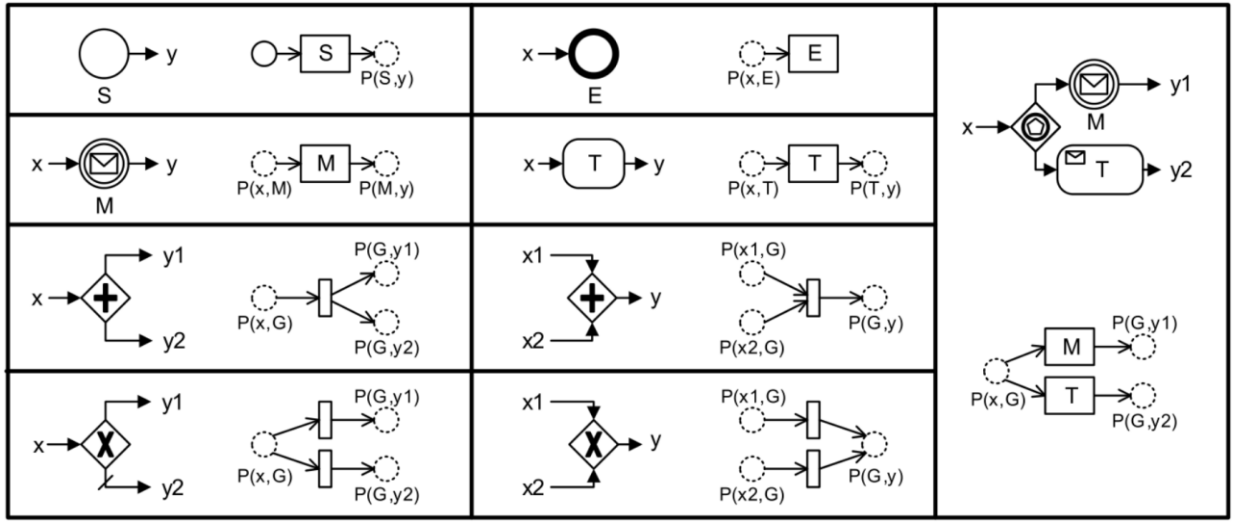
Translator


BPMN process model
Petri net
Data conditions
Simplified net with data conditions
Solidity contract code

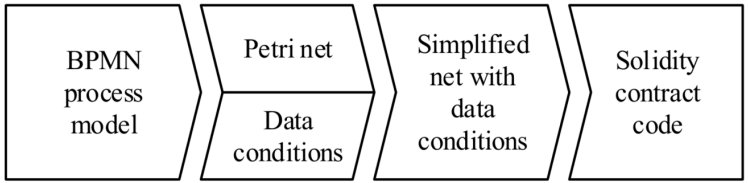
Translator



1. Translate BPMN control flow to WFnet (proven to be safe)

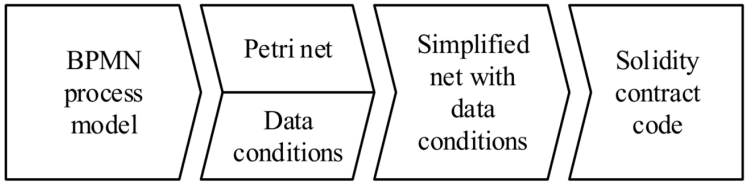


Translator

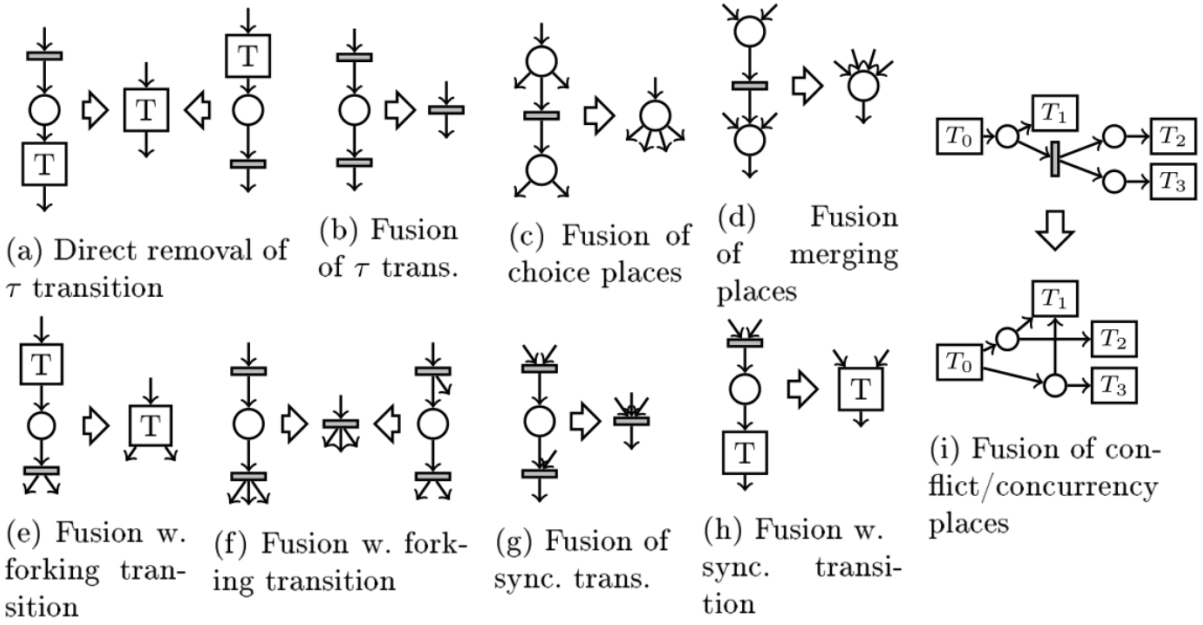


1. Translate BPMN control flow to WFnet (proven to be safe)
2. Capture dataflow and conditions

Translator



1. Translate BPMN control flow to WFnet (proven to be safe)
2. Capture dataflow and conditions
3. Reduce WFnet and annotate dataflow



(a) Direct removal of τ transition

(b) Fusion of τ trans.

(c) Fusion of choice places

(d) Fusion of merging places

(e) Fusion w. forking transition

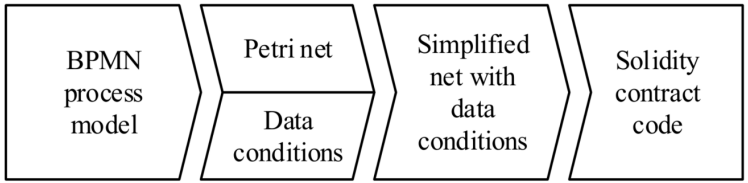
(f) Fusion w. forking transition

(g) Fusion of sync. trans.

(h) Fusion w. sync. transition

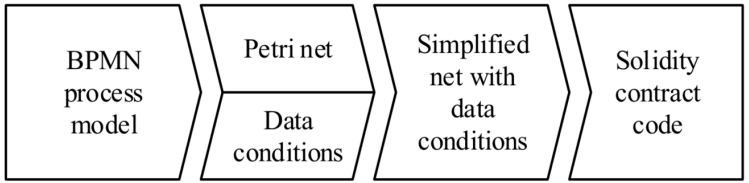
(i) Fusion of conflict/concurrency places

Translator



1. Translate BPMN control flow to WFnet (proven to be safe)
2. Capture dataflow and conditions
3. Reduce WFnet and annotate dataflow
4. Translate into Solidity code
   - Status of the process is kept in a bit vector
   - Updates are bit-wise operations

Translator



1. Translate BPMN control flow to WFnet (proven to be safe)
2. Capture dataf
3. Reduce WFne
4. Translate into
   - Status of the
   - Updates are b

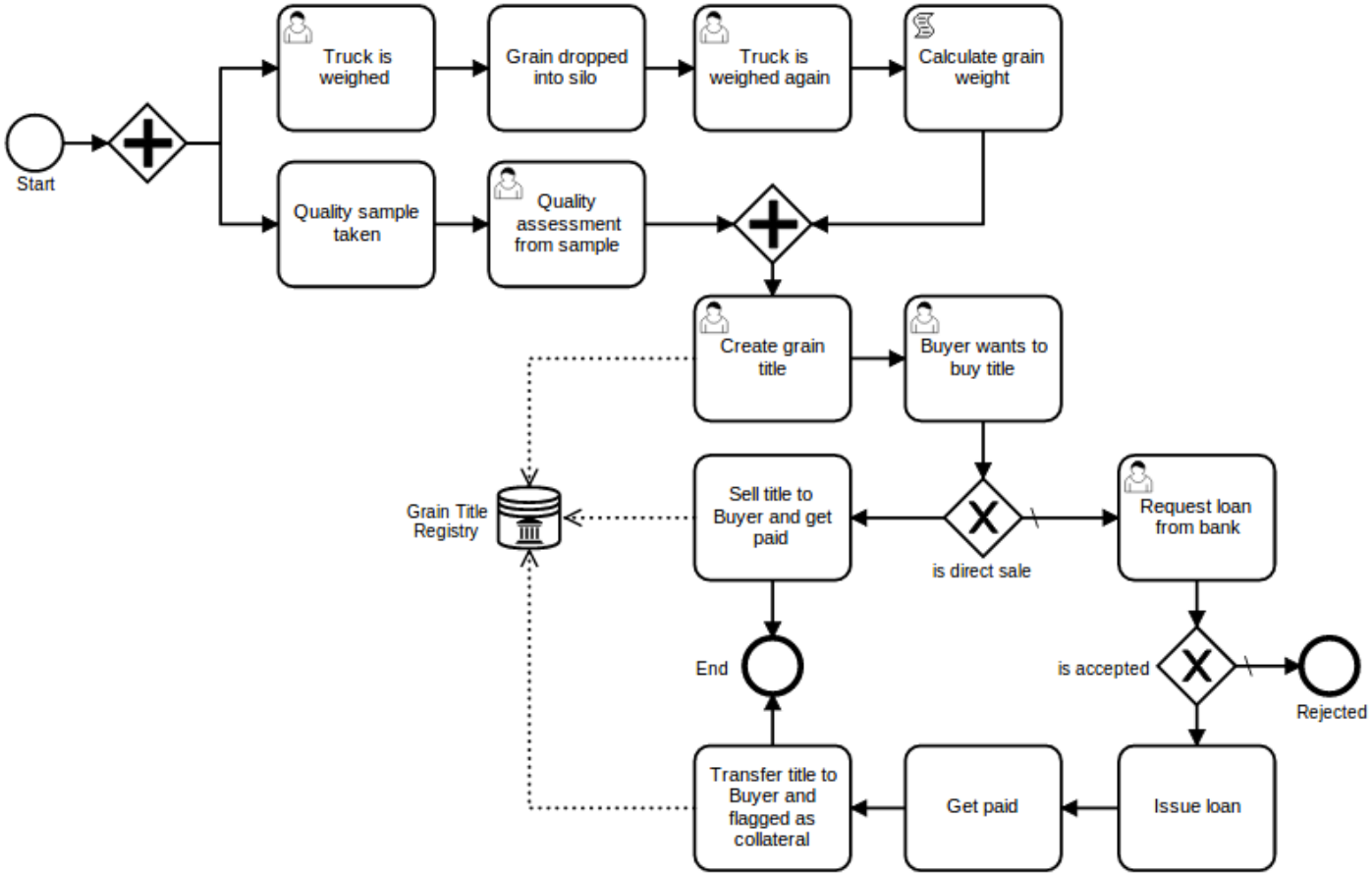
Bit vector check: does pos 1 have value "1"?

```
1  contract BPMNContract {
2    uint marking = 1;
3    uint predicates = 0;
4
5    function CheckApplication( -input params- ) returns (bool) {
6      if (marking & 2 == 2) { // is there a token in place p1?
7        // Task B's script goes here, e.g. copy value of input params to contract variables
8        uint tmpPreds = 0;
9        if ( -eval P- ) tmpPreds |= 1; // is loan application complete?
10       if ( -eval Q- ) tmpPreds |= 2; // is the property pledged?
11       step(
12         marking & uint(~2) | 12,        // New marking
13         predicates & uint(~3) | tmpPreds // New evaluation for "predicates"
14       );
15       return true;
16     }
17     return false;
18   }
```

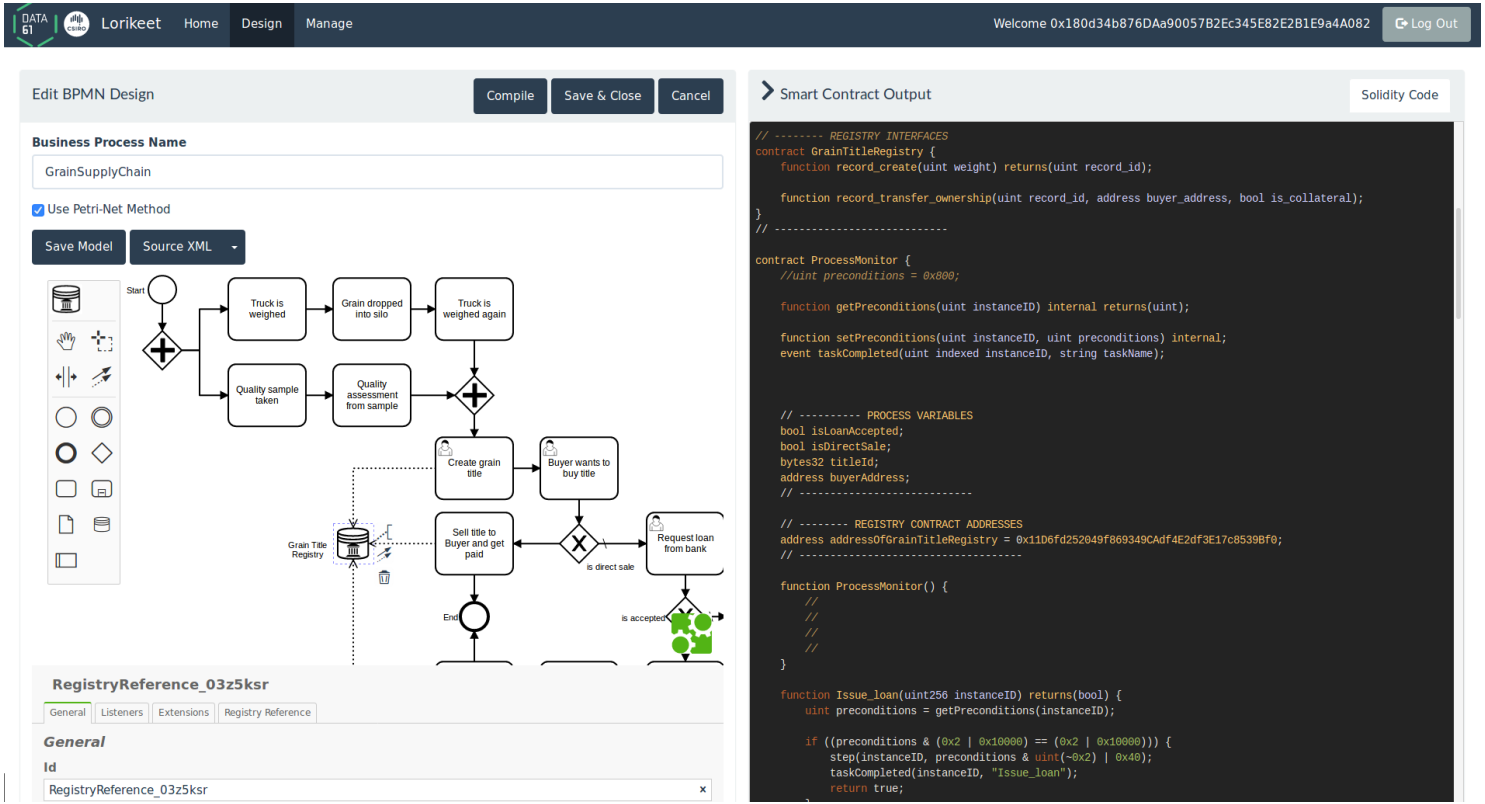
Bit vector update: set pos 1 to "0"

set pos 2 and 3 to "1"

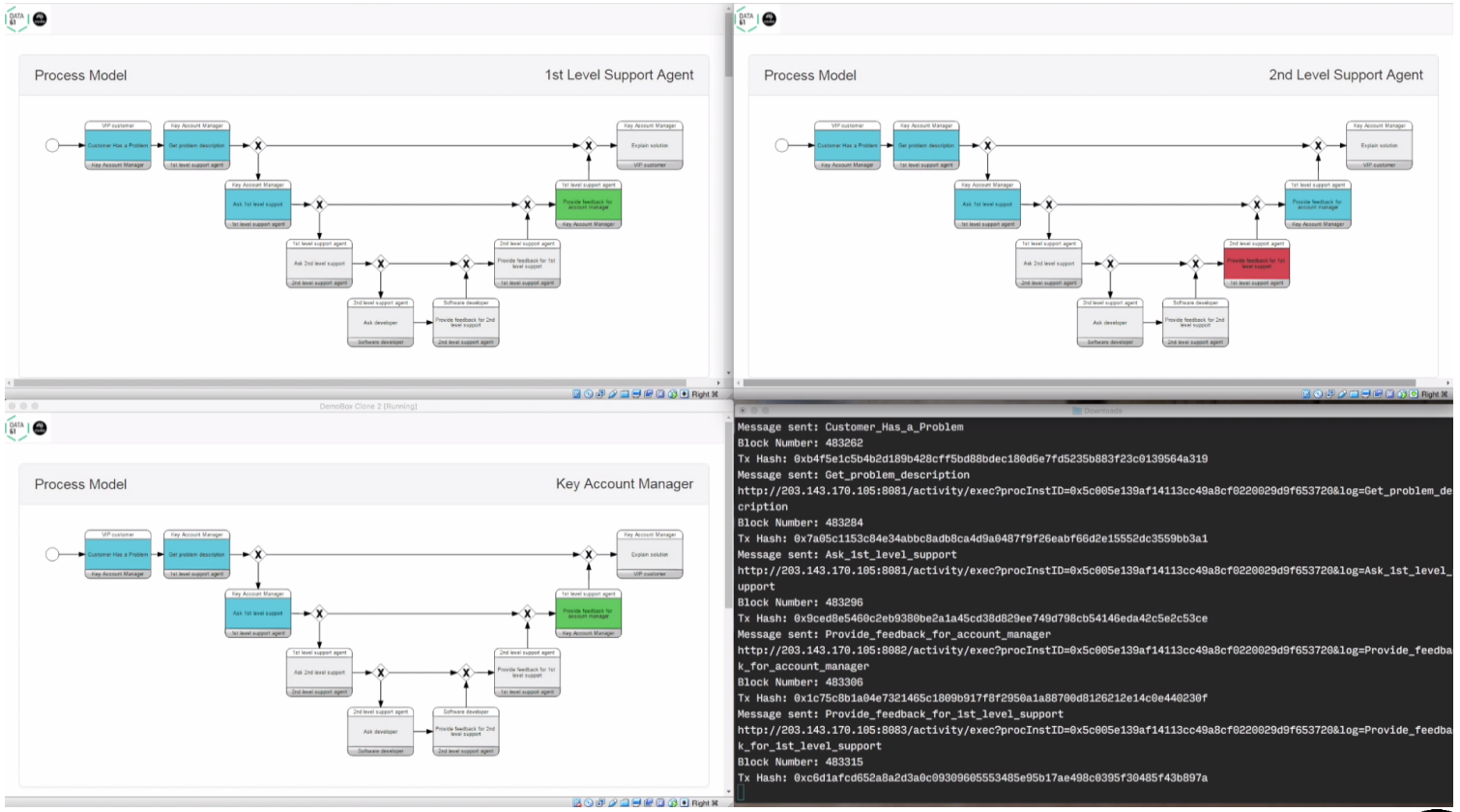
Combining process and data/token models



Tooling: Design time



Tooling: Runtime



Summary:

- Model-Driven Engineering basics
- MDE for data and token model
- Business Process Model and Notation (BPMN)
- MDE for process models

CSIRO DATA 61

Course Outline – next two weeks

Week	Date	Lecturer	Lecture Topic	Relevant Book Chapters	Notes
10th	22 Apr			Easter Holiday	
11th	29 Apr	Guest Lecturer + Mark Staples	Guest Lecture and Summary	Case study chapters Foreword, Epilogue	

End of Lecture / Consultation

Ingo Weber | Principal Research Scientist & Team Leader

Architecture & Analytics Platforms (AAP) team

ingo.weber@data61.csiro.au

Conj. Assoc. Professor, UNSW Australia | Adj. Assoc. Professor, Swinburne University

www.data61.csiro.au