Software Engineering Research for Blockchain-Based Systems

Dr Mark Staples
Research Group Leader, Software Systems
Conjoint Associate Professor, School of Computer Science & Engineering, UNSW

www.csiro.au

Data61 – Australia Digital Innovation

1100+ people

415+ research students

38 university partners

31 government partners

91 corporate partners

- Data capture
- Communications & networks
- Hardware & software
- Cybersecurity
- Data, statistics, ML
- Decision sciences, AI
- Behavioral economics & cogsci

50 Years of Software Engineering

1968 NATO Conference


SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

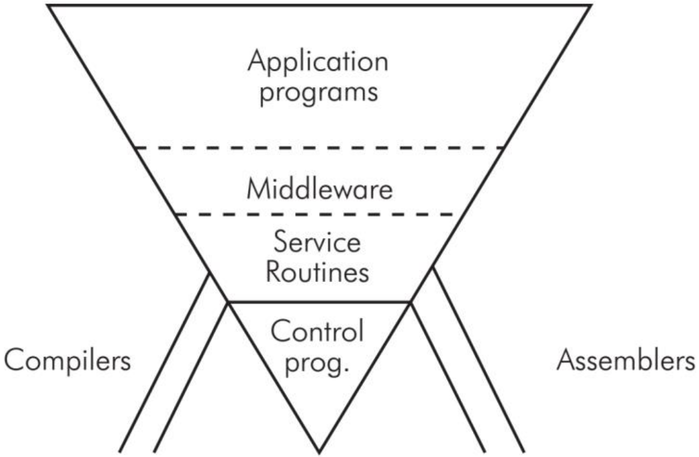
January 1969

- “The phrase ‘software engineering’ was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.”
- “…discussions […] under the three main headings: Design of Software, Production of Software, and Service of Software.”

homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/

Design, Production, Service

- Design Criteria
  - Use requirements
  - Reliability
  - Logical completeness
- Design Strategies
  - Sequencing the process
  - Structuring the design
  - Feedback with monitoring & simulation
  - High-level languages
- Communication & Management



Design, Production, Service

- Problems of scale
- Problems of reliability
- Management
  - Production planning
  - Personnel factors
  - Production control
  - Internal communication
- Technical aspects
  - Tools
  - Concepts
  - Performance monitoring

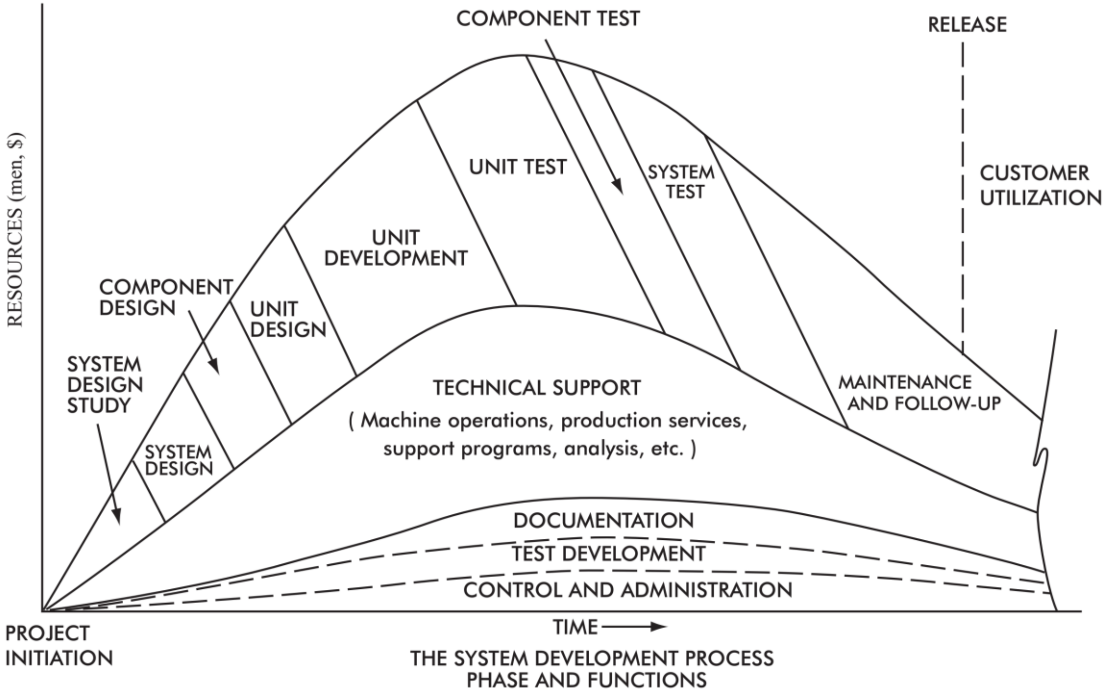
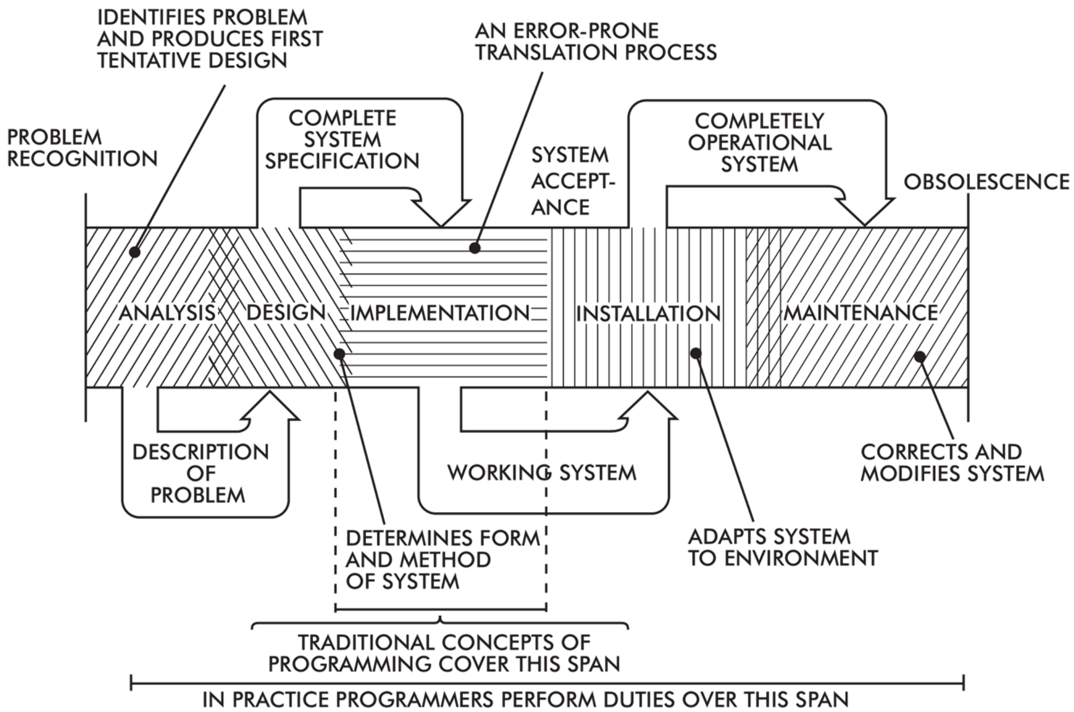


Figure 1. From Nash: Some problems in the production of large-scale software systems.

Design, Production, Service

- Service
  - Service goals
  - Initial release
  - Frequency of release
- Replication, Distribution, Maintenance
- Evaluation
  - Acceptance testing
  - Performance monitoring
  - Feedback from users
  - User documentation
  - Reprogramming



SCHEMATIC FOR THE TOTAL PROGRAMMING (SOFTWARE SYSTEM-BUILDING) PROCESS

Special Topics, Working Papers

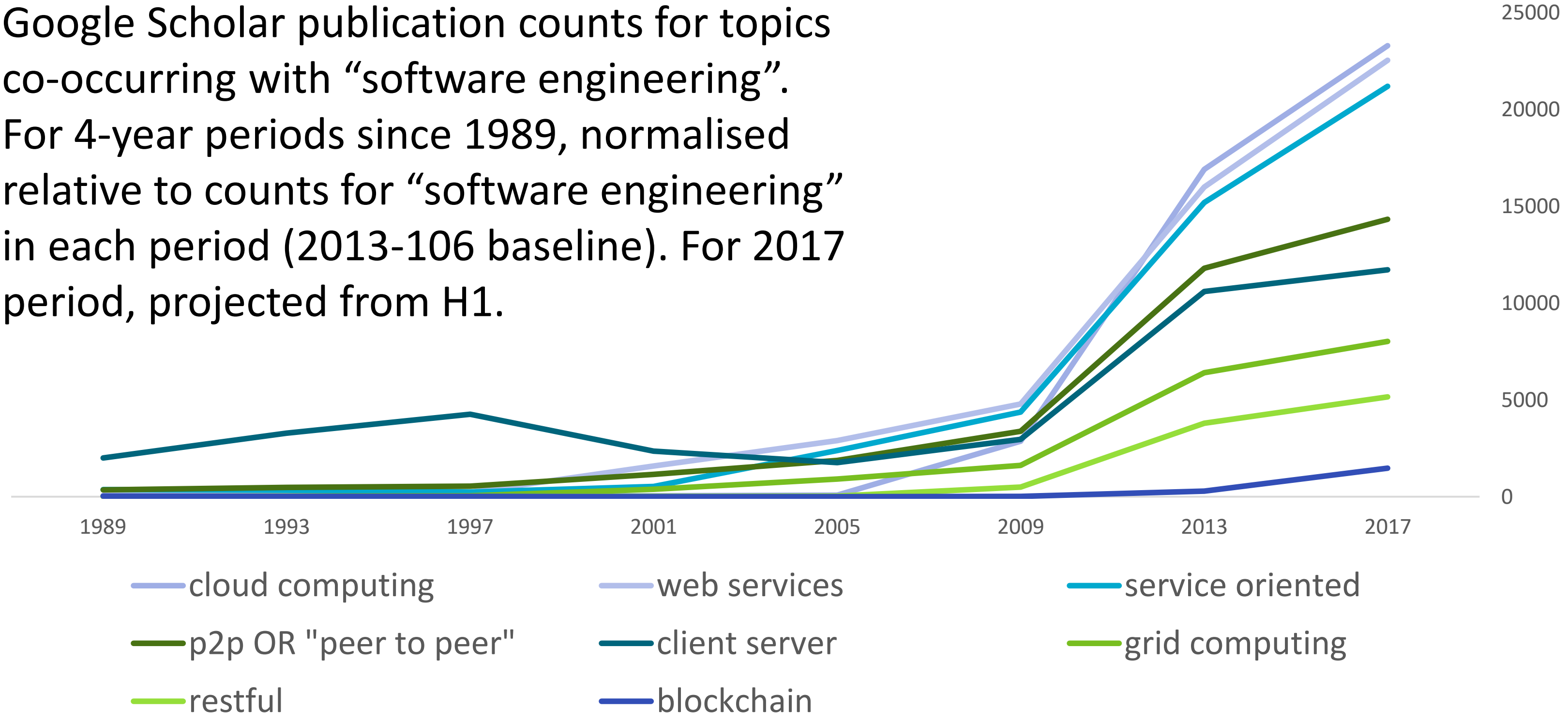
- Software crisis
- Education
- Software pricing

- Mass-produced components

- Checklist for planning software system production
- Complexity controlled by hierarchical ordering of function and variability
- Thoughts on the sequence of writing software
- The testing of computer software
- Performance monitoring and systems evaluation
- Towards a methodology of computing systems design

Software System Fashions

Google Scholar publication counts for topics co-occurring with “software engineering”. For 4-year periods since 1989, normalised relative to counts for “software engineering” in each period (2013-106 baseline). For 2017 period, projected from H1.



Blockchain

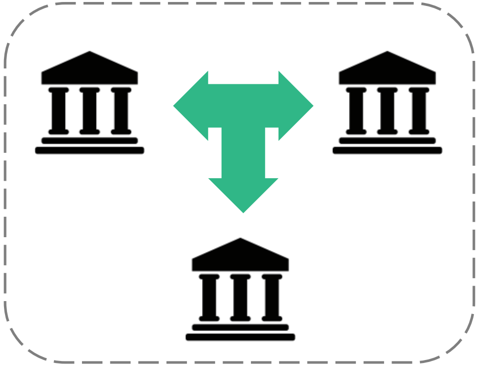
First, What Is A Blockchain?

(demo)

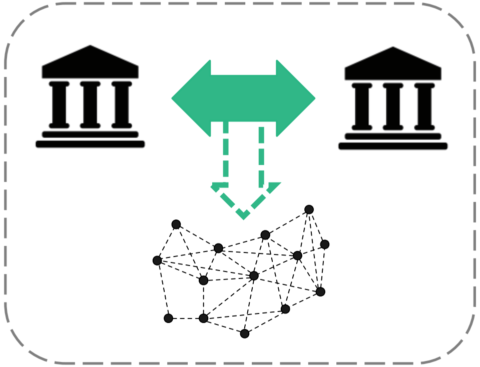
What Does a Blockchain Do?

- Functionally, blockchains are…
- A database (ledger)
  - – Record of transactions
- A compute platform
  - – “Smart contracts”
- Distributed, and no central owner

Centralised Trust using a Third-Party



Distributed Trust using a Blockchain



What's New About Blockchain?

- What does blockchain bring that is new, and makes a difference to software engineering research?

- What are the new software-engineering research challenges for blockchain-based systems?
  - Some threads of research at Data61
    - Treasury projects
    - Architecting application on blockchain
    - Using smart contracts for business process execution
    - Formal specification and verification of smart contracts
  - Some other research areas

Today’s Focus: Not Computer Science

- Not looking at topics on new/ improved/ analyses of:
  - Consensus algorithms
  - Hash functions
  - Crypto schemes
  - Zero-knowledge schemes
  - P2P networking
  - Smart contract virtual machines

Treasury projects

"Risks and Opportunities for Systems Using Blockchain and Smart Contracts", Staples, M., Chen, S., Falamaki, S., Ponomarev, A., Rimba, P., Tran, A. B., Weber, I., Xu, X., Zhu, J., 2017

"Distributed Ledgers: Scenarios for the Australian Economy Over the Coming Decades", Hanson, R. T., Reeson, A., Staples, M., 2017

Technical Perspectives

Software Architecture

Non-Functional Properties
- *Security, Performance, …*

Dependable Software Systems

Trusted and Trustworthy Systems
- *Risk, Evidence, Assurance, …*

Three Illustrative Use Cases

Supply Chain

Registries

Remittances

Non-Functional Properties

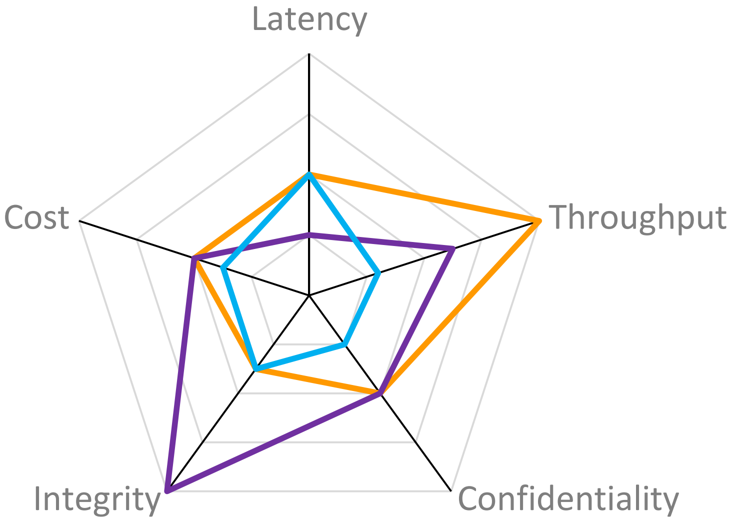
There are two kinds of requirements:

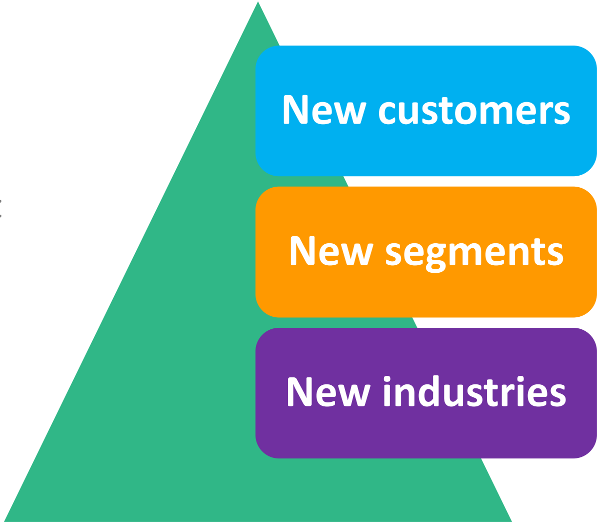
1. Functional Requirements (i.e. inputs vs. outputs)

2. Non-Functional Requirements (a.k.a. “Qualities”, or “-illities”)
   - e.g. Performance
     - latency, throughput, scalability (of latency, throughput, …), timeliness, …
   - e.g. Security
     - confidentiality, integrity, availability, non-repudiation, privacy, …
   - e.g. Usability, Reliability, Modifiability, …

Why Non-Functional Properties Matter

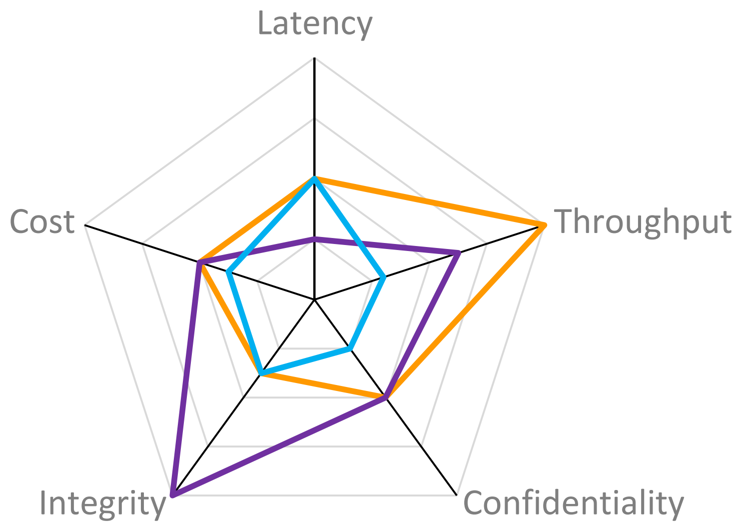
Regulatory Requirements ↔ System Performance ↔ Market Opportunities

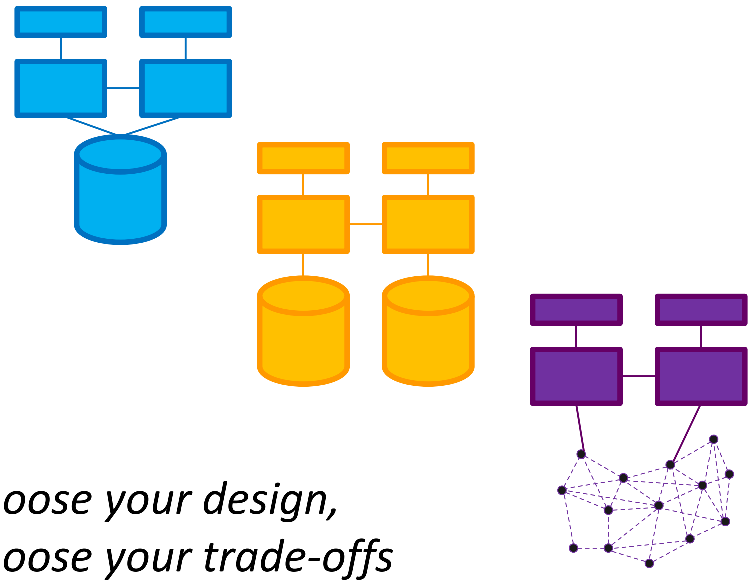




Software Architecture

Non-Functional Properties arise from Architectural Design Choices





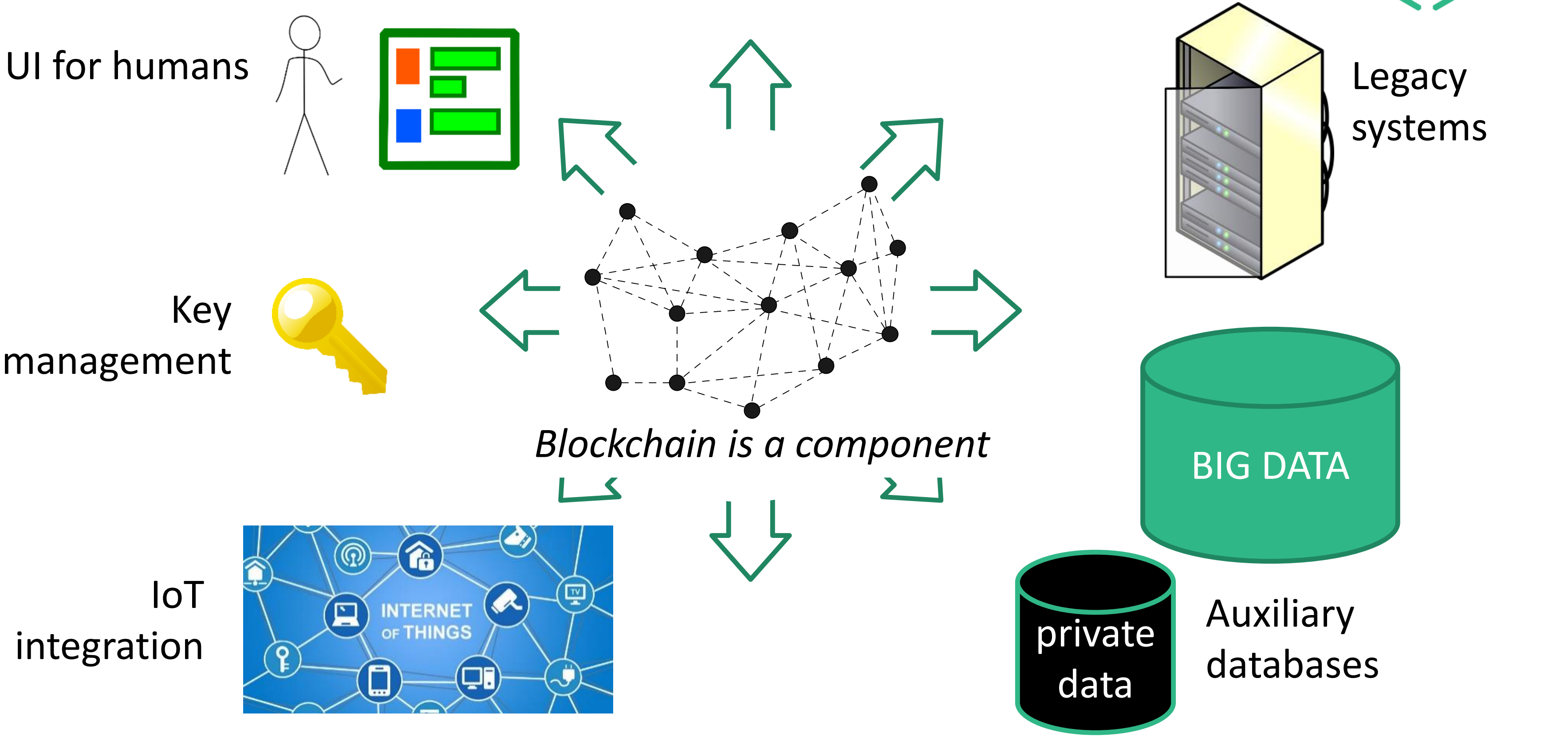
Choose your design,
Choose your trade-offs

Blockchain vs Conventional Databases

- Logically centralised;
  Physically and organisationally decentralised

- Trade-offs for Various Non-Functional Properties
  - (+) Integrity, Non-repudiation
  - (-) Confidentiality, Privacy
  - (-) Modifiability
  - (-) Throughput/ Scalability/ Big Data
  - (+ read/ - write) Availability/ Latency

Blockchains are Not Stand-Alone Systems



We (Will) Rely on Blockchain-Based Systems

- The DAO failure hurt
  - Cost USD$60M? + fork

- Huge future economic value (or else there's no point!)
  - e.g. supply chain, asset registries, settlement, …

- Security-critical and Safety-critical use cases
  - e.g. e-health records, pharma supply chain, IoT management, …

What is "Trust"?

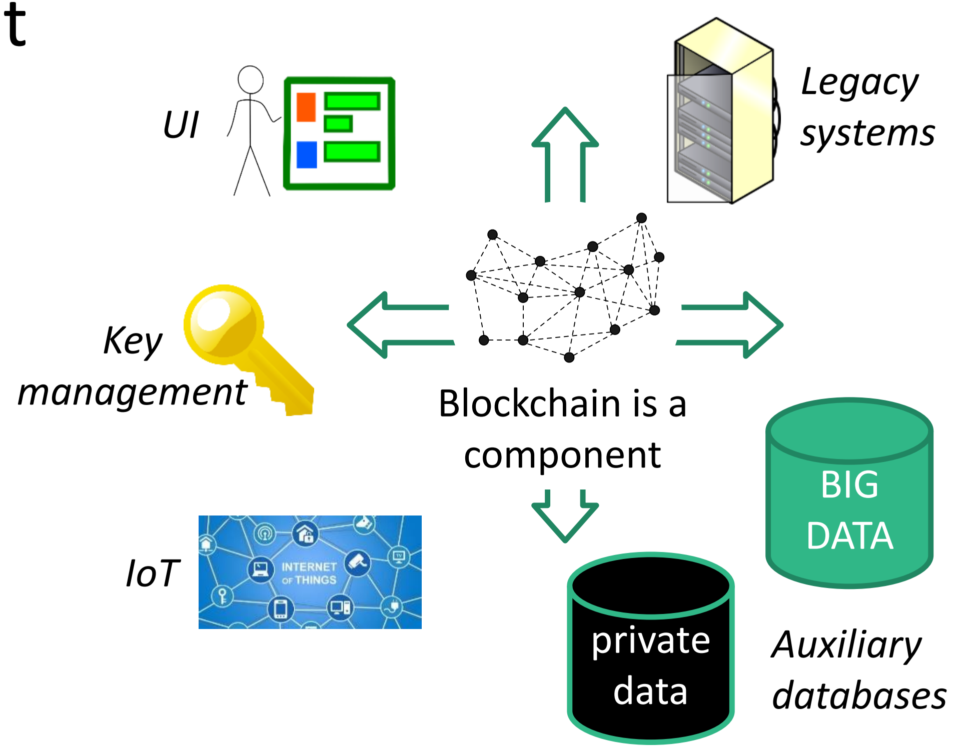
Dependable Software Systems says…

- Trusted System
  - A system you have chosen to rely on to fulfil a goal
    - When it fails, you suffer harm or loss
- Trustworthy System
  - A system where you have evidence it will not fail

"Trust" means accepting exposure to risk

Trustworthy Blockchain-Based Systems?

- What is good evidence that blockchain-based systems will do what we need?
  - Functional correctness
  - Non-Functional properties
- How do we get regulatory acceptance?



Assurance: Evidence & Acceptance

- Test blockchains in the rain

- Technologically-neutral regulation and policy
- But look carefully at blockchain-specific risks
- Need indicative guidance on regulatory acceptance of blockchain-based systems

- There are open questions about blockchain governance
- Increase R&D on trustworthy blockchains!

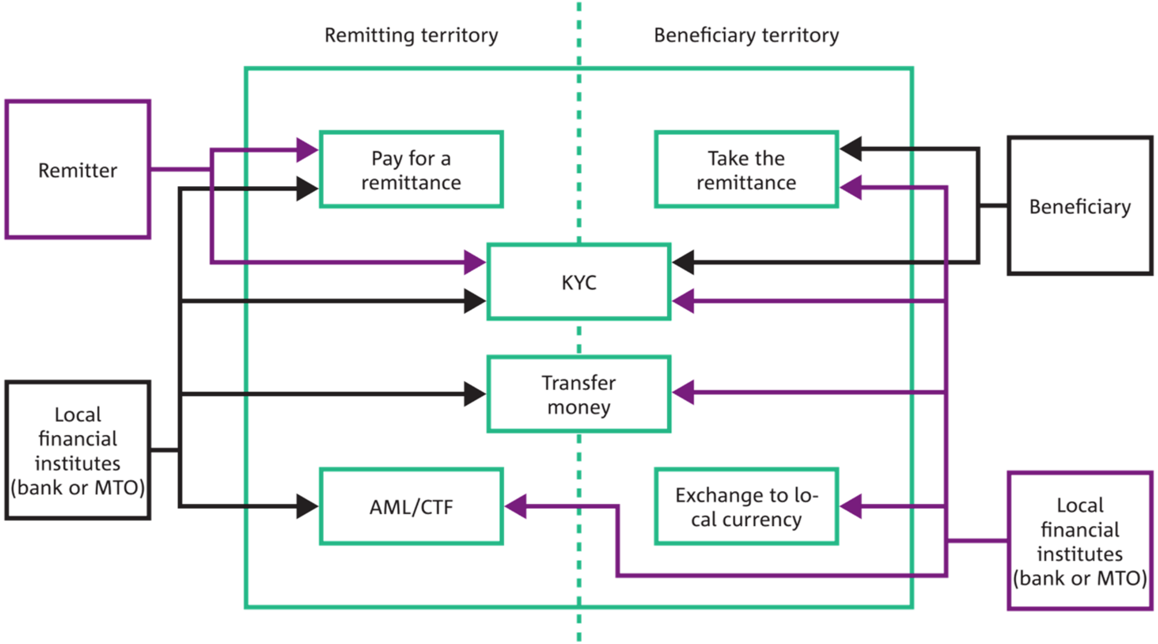
Potential Use Cases

- Financial Services
  - Digital currency
  - (International) payments
  - Reconciliation
  - Settlement
  - Markets
  - Trade finance
- Government Services
  - Registry & Identity
  - Grants & Social Security
  - Quota management
  - Taxation
- Enterprise and Industry
  - Supply chain
  - IoT
  - Metered access
  - Digital rights 7 IP
  - Data management
  - Attestation
  - Inter-divisional accounting
  - Corporate Affairs
- Three Illustrative Cases Selected
  1. Agricultural supply chain
  2. Open data registry
  3. Remittance payments

Remittances

- Blockchains may help reduce cost and time of remittances, but challenges remain for solutions to KYC

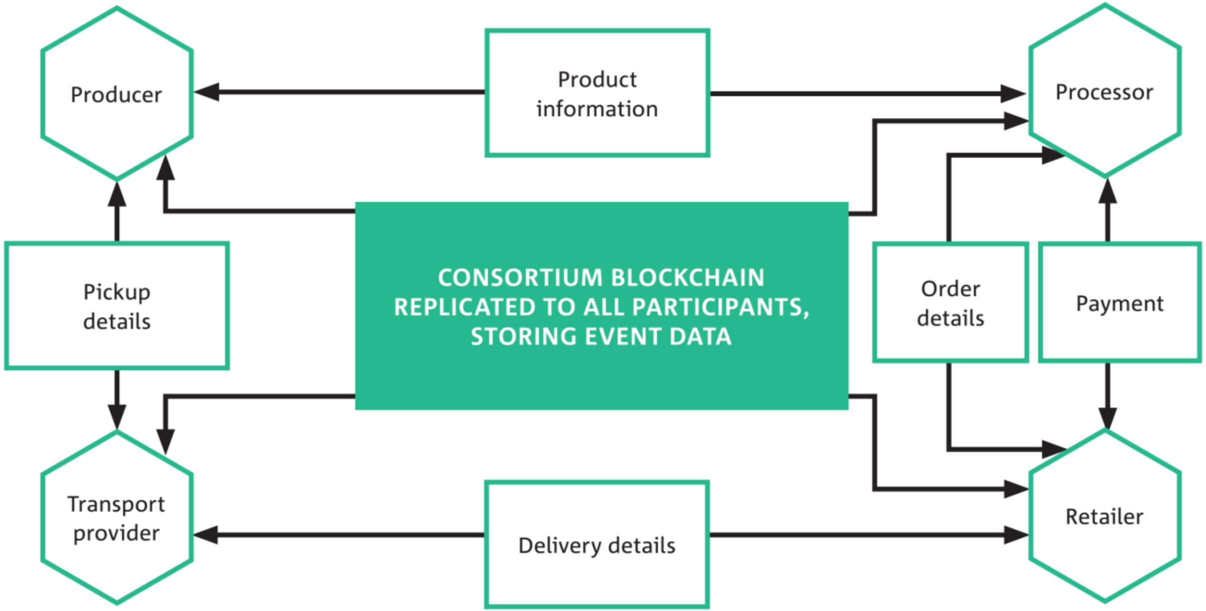
- Blockchains and smart contracts may make it possible to create 'programmable money'



Supply Chain

- Supply chains are a highly promising domain for blockchains

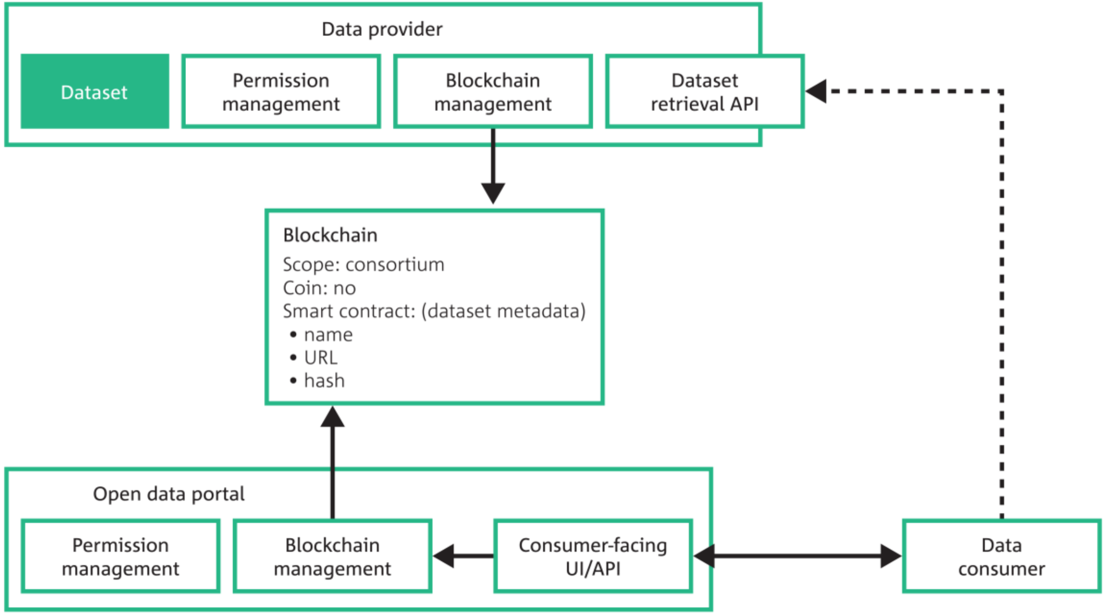
- May enable significant opportunities for trade finance and insurance



Registries

- Blockchains can help to federate registries

- Sometimes too much integrity causes problems



Other Findings

- Blockchains have a different cost model
- Private blockchains are often not private enough
- Public blockchains might be OK for some purposes, even in regulated industries
- Blockchains have limitations – sometimes that doesn’t matter!

Busting Blockchain Myths

Myth	Reality
Solves Every Problem	A kind of database
Trustless	Can shift trust and spread trust
Secure	Focus is Integrity, not Confidentiality
Smart contracts are legal contracts	May help execute parts of some legal contracts
Immutable	Many only offer probabilistic immutability
Need to waste electricity	Emerging blockchains are more efficient
Are inherently unscalable	Emerging blockchains are more scalable
If beneficial, will be adopted	Adoption can be hampered by FUD

Architecting Applications on Blockchain

“A taxonomy of blockchain-based systems for architecture design”, X. Xu, I. Weber, M. Staples et al., ICSA2017.

“The blockchain as a software connector”, X. Xu, C. Pautasso, L. Zhu et al., WICSA2016.

“Comparing blockchain and cloud services for business process execution”, P. Rimba, A. B. Tran, I. Weber et al., ICSA2017

“Predicting latency of blockchain-based systems using architectural modelling and simulation”, R. Yasaweerasinghelage, M. Staples and I. Weber, ICSA2017.

When to Use Blockchain? Trade-offs?

- Architecting applications on Blockchain:
  - Taxonomy and design process [1]
  - "Cost of Distrust": how much more expensive is blockchain? [2]
  - Latency: simulation under changes [3]

- Model-driven development of smart contracts
  - Business process execution [4,5]
  - Model-based generation of registries and UIs: "Regerator" [6]

Taxonomy

Blockchain-related design decisions regarding (de)centralisation, with an indication of their relative impact on quality properties

Legend: ⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favourable

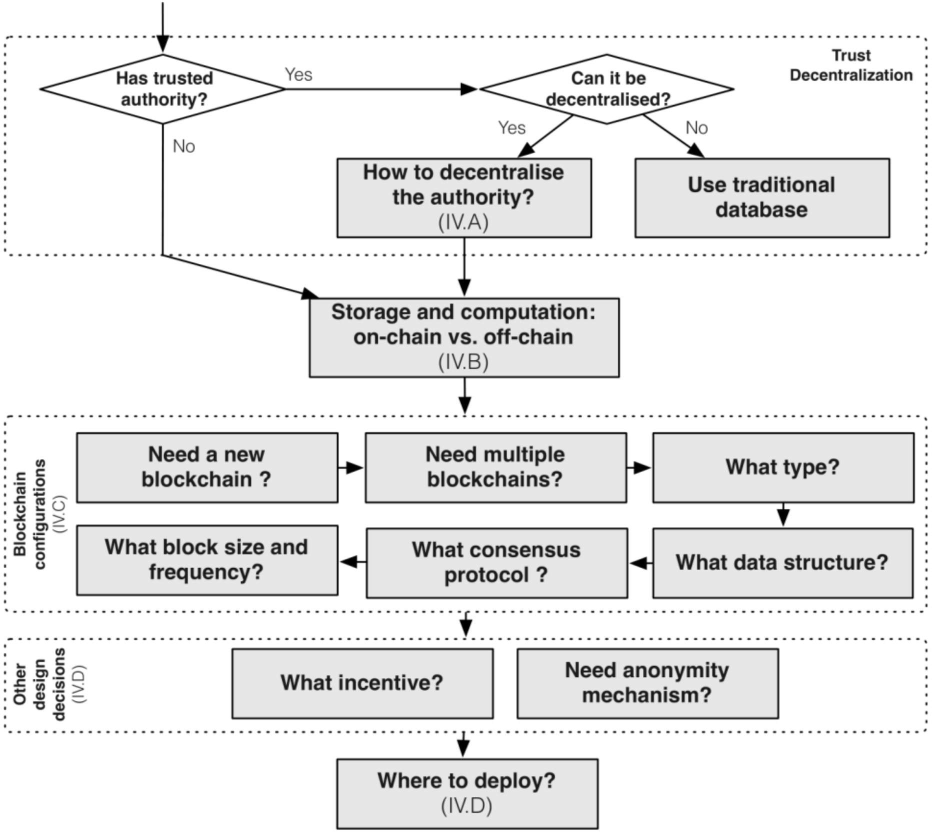
Design Decision	Option	Impact: Fundamental properties	Cost efficiency	Performance	#Failure points
Fully Centralised	Services with a single provider (*e.g.*, governments, courts)	⊕	⊕⊕⊕	⊕⊕⊕	1
	Services with alternative providers (*e.g.*, banking, online payments, cloud services)				
Partially Centralised & Partially Decentralised	Permissioned blockchain with permissions for fine-grained operations on the transaction level (*e.g.*, permission to create assets)	⊕⊕	⊕⊕	⊕⊕	*
	Permissioned blockchain with permissioned miners (write), but permission-less normal nodes (read)				
Fully Decentralised	Permission-less blockchain	⊕⊕⊕	⊕	⊕	Majority (nodes, power, stake)
		Fundamental properties	**Cost efficiency**	**Performance**	**#Failure points**
Verifier	Single verifier trusted by the network (external verifier signs valid transactions; internal verifier uses previously-injected external state)	⊕⊕	⊕⊕	⊕⊕	1
	M-of-N verifier trusted by the network	⊕⊕⊕	⊕	⊕	M
	Ad hoc verifier trusted by the participants involved	⊕	⊕⊕⊕	⊕⊕	1 (per ad hoc choice)

Taxonomy

Blockchain-related design decisions regarding storage and computation, with an indication of their relative impact on quality properties
Legend: ⊕: Less favourable, ⊕⊕: Neutral, ⊕⊕⊕: More favourable

Design Decision		Option	Fundamental properties	Impact: Cost efficiency	Impact: Performance	Impact: Flexibility
Item data	On-chain	Embedded in transaction (Bitcoin)	⊕⊕⊕⊕	⊕	⊕	⊕⊕
		Embedded in transaction (Public Ethereum)		⊕⊕⊕⊕	⊕	⊕⊕⊕
		Smart contract variable (Public Ethereum)		⊕⊕	⊕⊕⊕	⊕
		Smart contract log event (Public Ethereum)		⊕⊕⊕	⊕⊕	⊕⊕
	Off-chain	Private / Third party cloud	⊕	~KB Negligible	⊕⊕⊕⊕	⊕⊕⊕⊕
		Peer-to-Peer system		⊕⊕⊕⊕	⊕⊕⊕	⊕⊕⊕
Item collection	On-chain	Smart contract	⊕⊕⊕⊕	⊕⊕⊕⊕ (public)	⊕⊕⊕⊕	⊕
		Separate chain		⊕ (public)	⊕	⊕⊕⊕⊕
Computation	On-chain	Transaction constraints	⊕⊕⊕⊕	⊕	⊕	⊕
		Smart contract				
	Off-chain	Private / Third party cloud	⊕	⊕⊕⊕⊕	⊕⊕⊕⊕	⊕⊕⊕⊕

Proposed design process (using Taxonomy)



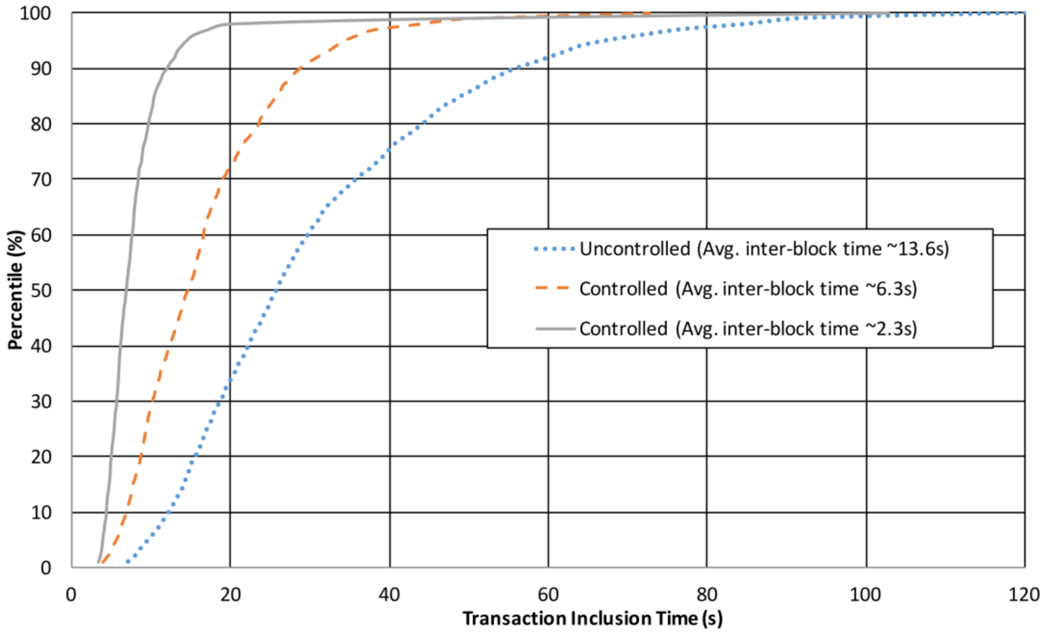
Cost of Distrust

- RQ: How much more expensive is blockchain over Cloud services?
  - Lens: business process execution
  - AWS SWF vs. Ethereum public blockchain
    - In both cases: pay per instruction
  - Experiments on two use cases:
    - Incident management (literature)
      - 32 instances on public Ethereum vs. 1000 runs on SWF
    - Invoicing (industry, 5316 log traces, 65K events)
      - Full log replayed on SWF and private Ethereum
- Result:
  - 2 orders of magnitude more expensive to use blockchain
  - ~US$ 0.35 per process instance on public blockchain
    - outweighed by cost of escrow (if needed) for about US$ 10 of value

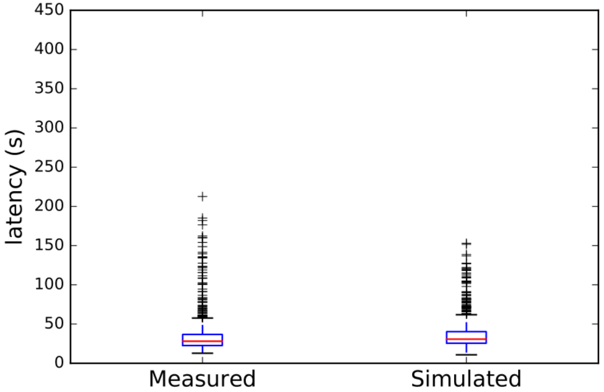
Latency simulation

- Goal: predict latency for blockchain-based applications
- Approach: Architecture performance modeling
  - Paladio Component Models with individual latency distributions + connections + probability of branching
  - Allows changing the models for *What-If analysis*
  - For instance: change inter-block time on private blockchain – what does that mean for overall application latency?

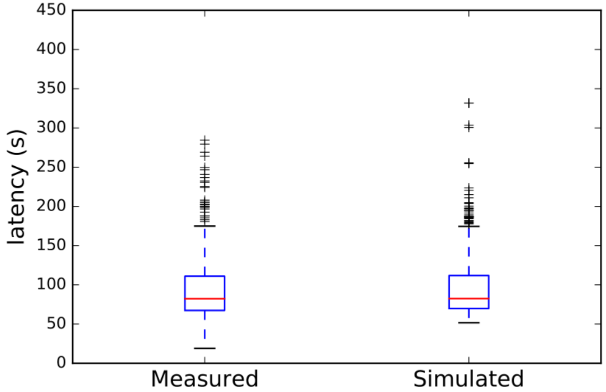
Latency simulation



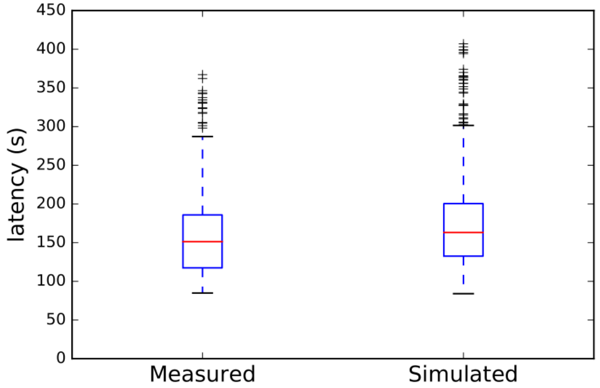
Latency: what if we change required number of confirmation blocks?



1 block



6 blocks



12 blocks

Using Smart Contracts for Business Process Monitoring and Execution

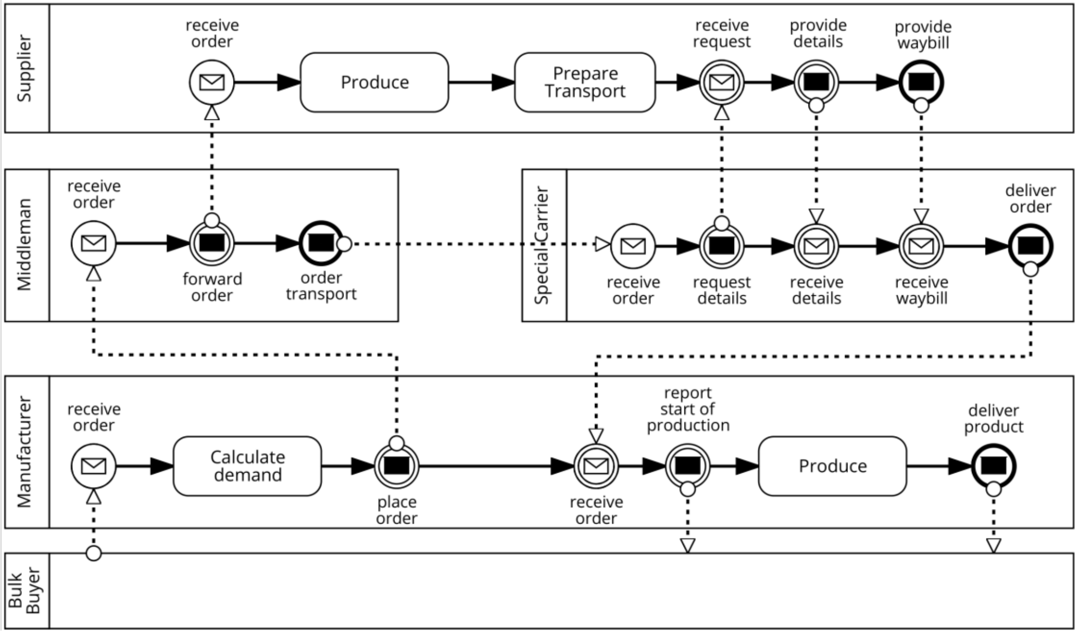
"Untrusted business process monitoring and execution using blockchain", I. Weber, X. Xu, R. Riveret et al., BPM2016
"Optimized Execution of Business Processes on Blockchain", L. García-Bañuelos, A. Ponomarev, M. Dumas, Ingo Weber, BPM2017

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

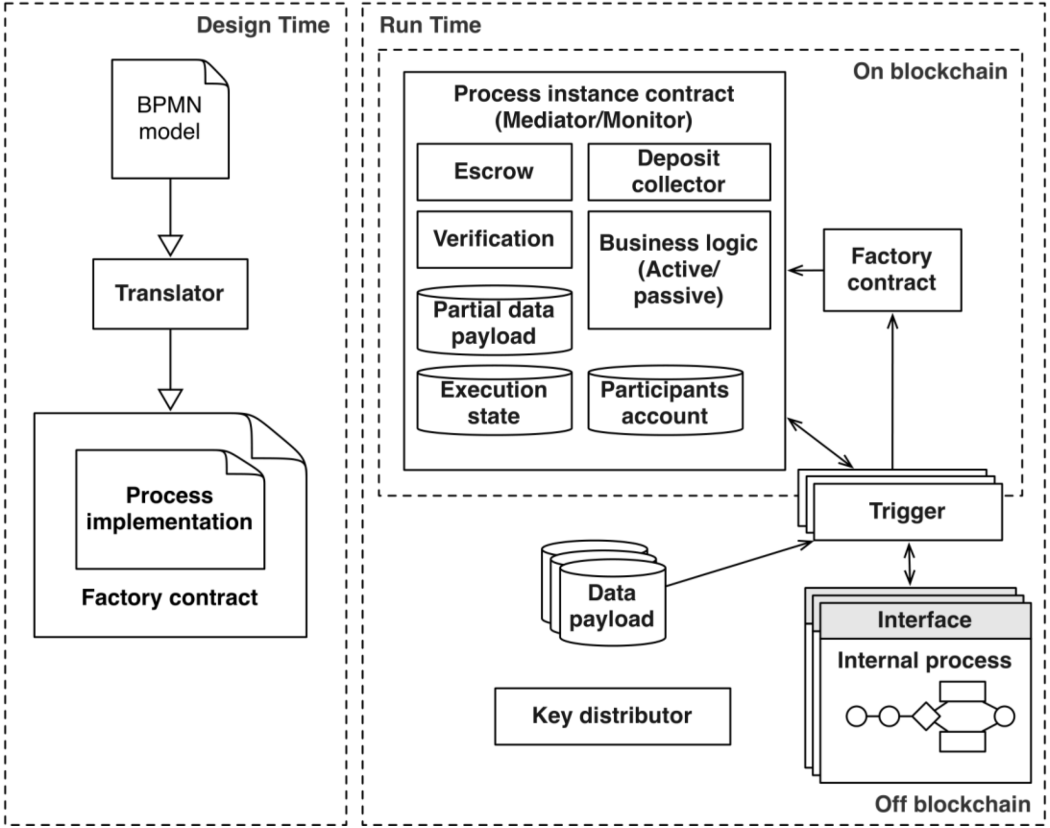
→ ~~Trusted 3rd party~~ ?

→ Blockchain!

Approach in a nutshell

- Goal: execute collaborative business processes as smart contracts
  - Translate (enriched) BPMN to smart contract code
  - Triggers act as bridge between Enterprise world and blockchain
  - Smart contract does:
    - Independent, global process monitoring
    - Conformance checking: only expected messages are accepted, only from the respective role
    - Automatic payments & escrow
    - Data transformation
    - Encryption

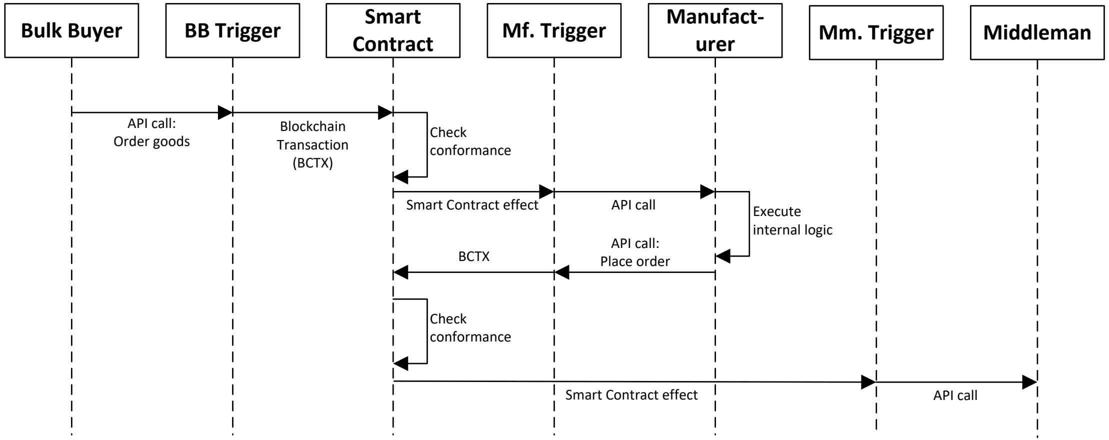
Architecture



Runtime

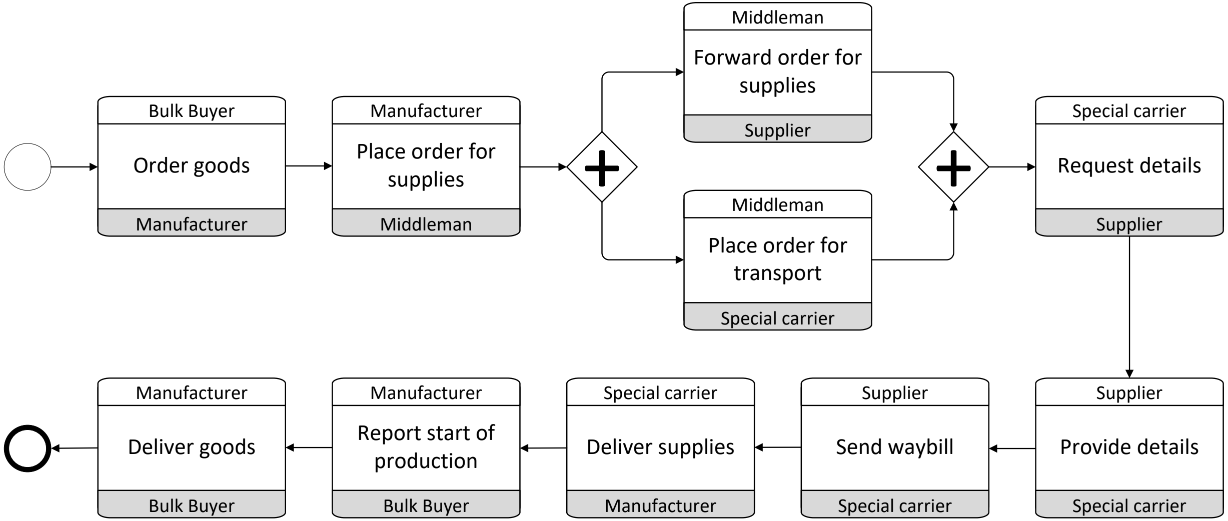
- Instantiation:
  - New *instance contract* per process instance
  - Assign accounts to roles during initialization
  - Exchange keys and create secret key for the instance

- Messaging:
  - Instead of direct WS calls, send through triggers & smart contract
  - Instance contract handles:
    - Global monitoring
    - Conformance checking
    - Automated payments*
    - Data transformation*

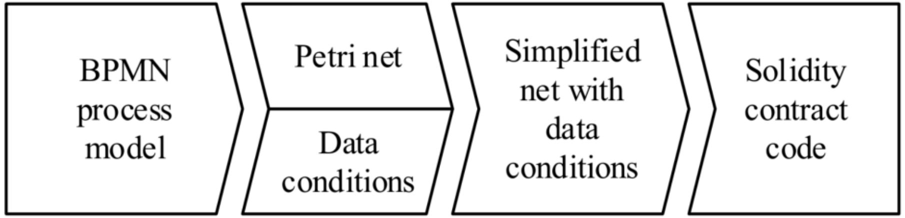


Translator

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



Translator



1. Translate BPMN control flow to Wfnet
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

Bit vector check: does pos 1 have value “1”?

```
contract BPMNContract {
  uint marking = 1;
  uint predicates = 0;

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

Bit vector update: set pos 1 to “0”

set pos 2 and 3 to “1”

Payments, escrow, data handling

- Payment / escrow using crypto-coins:
  - Instance contract has an account
  - Only the contract code governs what gets paid out, but anyone can pay in
  - Contract can base validity of transactions on associated payments
  - Anyone can see balance of the contract – enables trust:
    - All parties know when the money is there
    - The contract code specifies who gets paid how much, and when

- Data:
  - Status update data has to readable for the contract, most other data can be encrypted
    - Data used in conditions has to be readable
  - Sending data over blockchain is costly – can split on-chain vs. off-chain
    - On-chain: URI to the data + hash
    - Off-chain: reachable and addressable data store (IPFS, S3, ...)

Evaluation

- 4 use case processes, 1 from industry with 5316 traces and 65K events
- Executed ~150K transactions on private and 256 TXs on public Ethereum blockchain
- Look at Correctness, Cost, Latency, Throughput

Evaluation – Cost

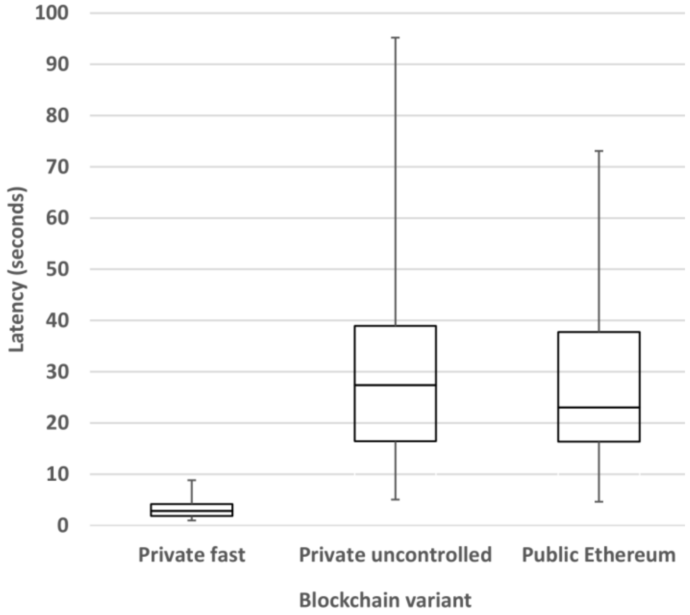
- Per process instance: cost on average of 0.0347 Ether (~$0.40) before optimization

- After optimization:
  - About 25% reduction for industrial process (Opt-CF)
  - Up to 75% reduction if smart contract can be reused (Opt-Full)

Process	Tested Traces	Variant	Avg. Cost Instant.	Avg. Cost Exec.	Savings (%)
Invoicing	5316	Default	1,089,000	33,619	–
		Opt-CF	807,123	26,093	-24.97
		Opt-Full	54,639	26,904	-75.46
Supply chain	62	Default	304,084	25,564	–
		Opt-CF	298,564	24,744	-2.48
		Opt-Full	54,248	25,409	-42.98
Incident mgmt.	124	Default	365,207	26,961	–
		Opt-CF	345,743	24,153	-7.04
		Opt-Full	54,499	25,711	-57.96
Insurance claim	279	Default	439,143	27,310	–
		Opt-CF	391,510	25,453	-8.59
		Opt-Full	54,395	26,169	-41.14

Evaluation – Latency

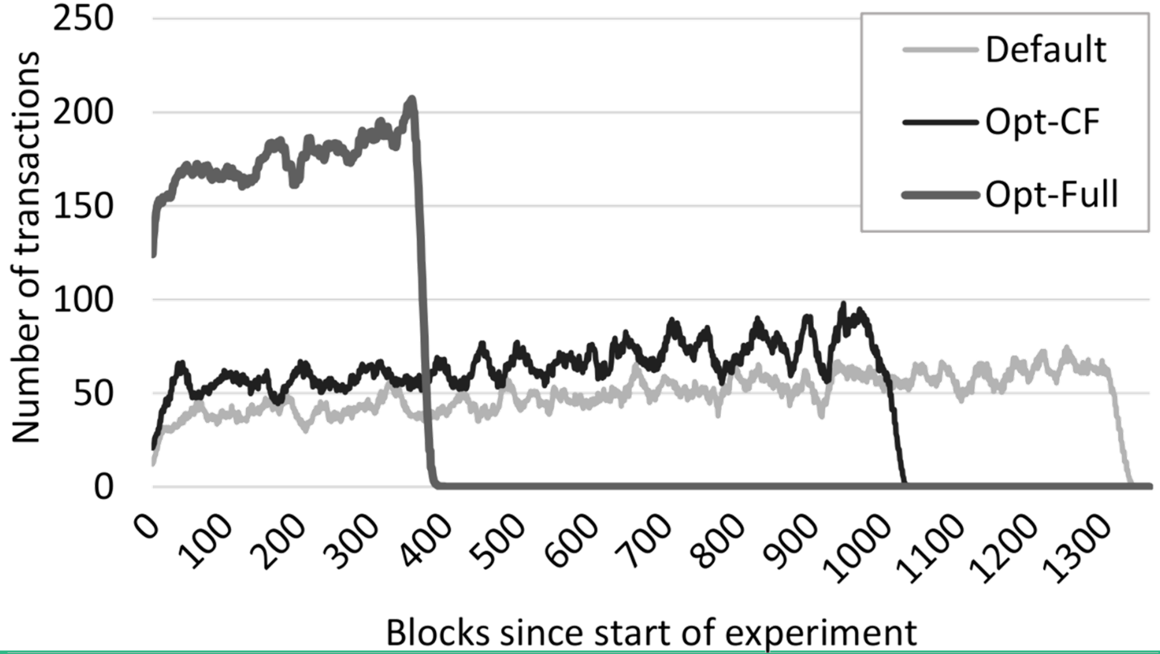
- Primary source of latency: mining time
  - Public Ethereum blockchain: median time between blocks set to 13-15s
  - Private blockchain: can control it



Evaluation – Throughput

Optimized implementation can fill up each block

- Limit: blockchain network-defined gas limit



- Unoptimized
- Control flow optimized
- Fully optimized

Formal Specification and Verification of Smart Contracts

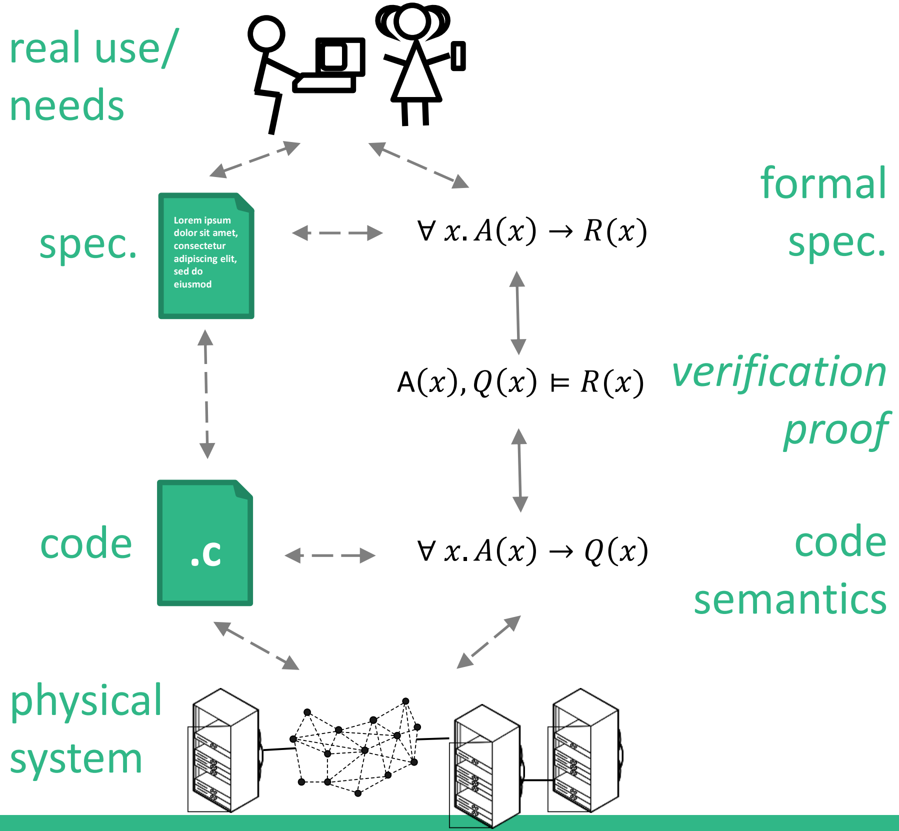
"Evaluation of Logic-Based Smart Contracts for Blockchain Systems", F. Idelberg, G. Governatori, R. Riveret et al., RuleML2016

Formal Specification & Verification

Program testing can be used to show the presence of bugs, but never to show their absence!

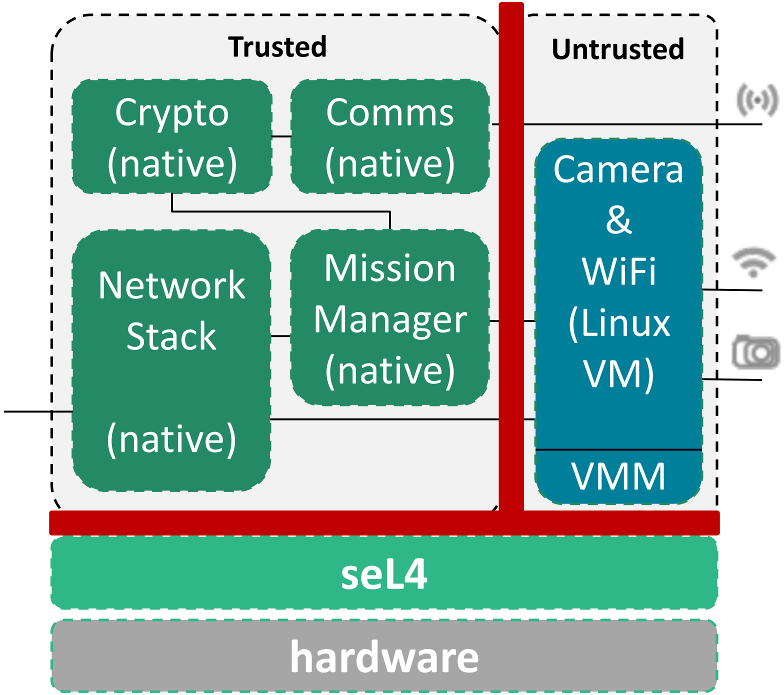
- Dijkstra, ’60s/’70s

- Formal methods is logical reasoning about all possible behaviours of software
- Is the strongest kind of evidence that software is correct



seL4 verified @ Data61

- seL4: embedded systems microkernel
  - high-performance, in C and ASM
  - capability-based security
  - 2009 formal proof of correctness
  - ongoing extensions and proofs
- Demonstrated in DARPA HACMS
  - unmanned helicopter
  - red team couldn't hack



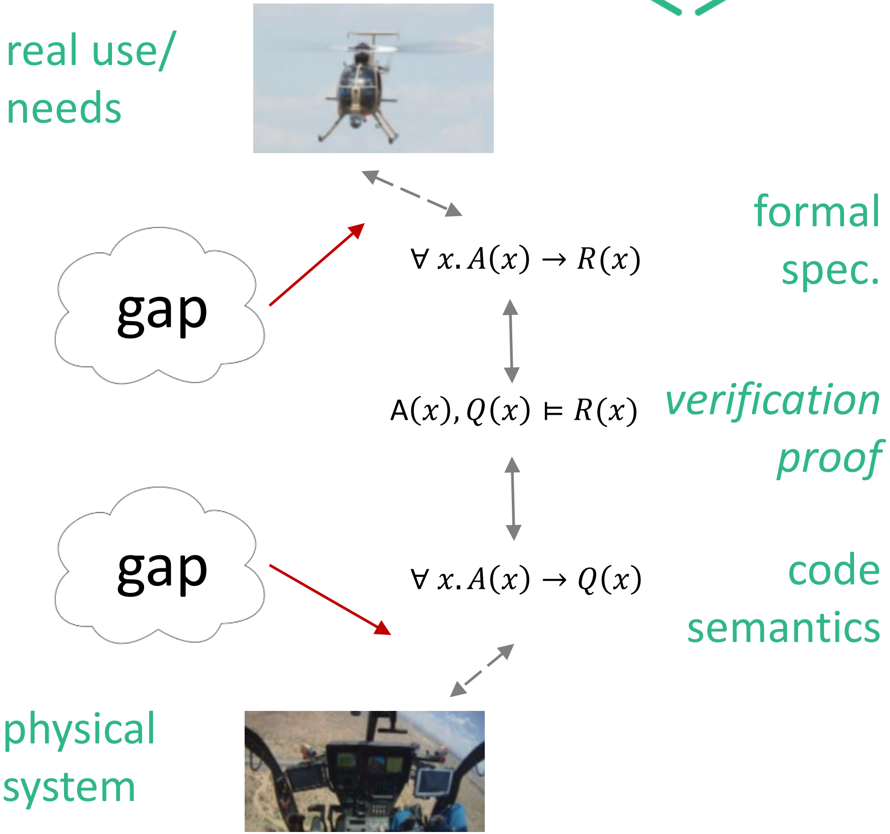
Rockwell Collins | galois | DATA 61 CSIRO | University of Minnesota | BOEING

Limits of Formal Methods

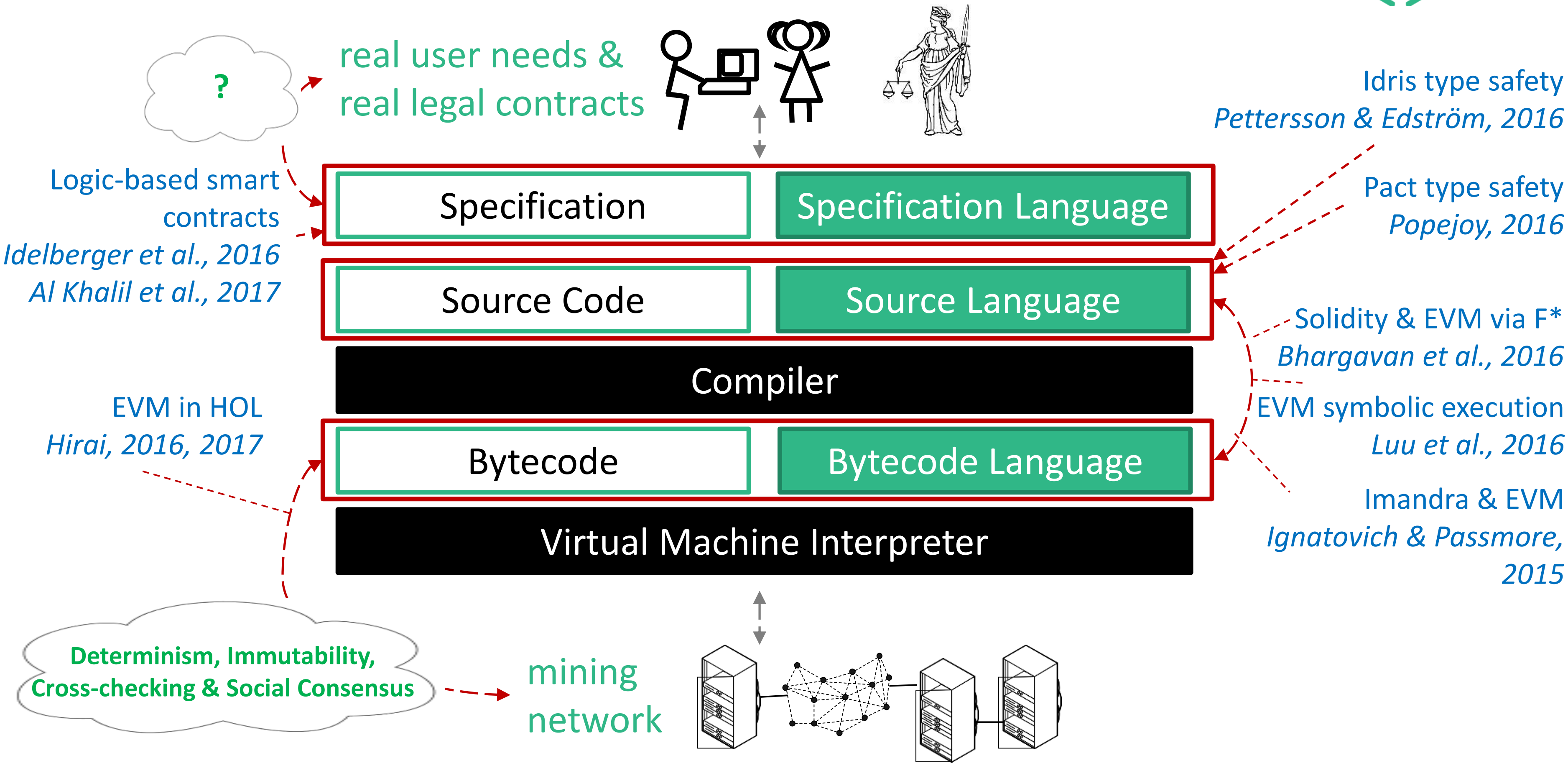
- seL4 is "bug free"*
  - No null pointer dereferences
  - No undefined behaviour
  - All behaviours conform to spec
  - Security policies are enforced
  - …

(*) conditions apply:

- Correct hardware
- Correct boot code
- DMA off or trusted
- …



Formalising Smart Contracts

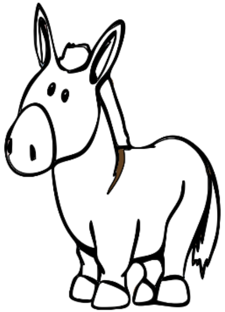


Smart Contract Verification

- Data61 is now contributing to Yoichi Hirai's formalization of EVM
  - Within the Data61 group that verified and maintains seL4
  - https://github.com/pirapira/eth-isabelle
- "Proof engineering": cheaper & more scalable formal verification
  - Verified code frameworks for compositional code & compositional proof
  - Formal decompilation from bytecode to higher-level languages
  - Metric-driven proof improvement and estimation
- Linking to formal specification of smart contracts…

Code is Not a Good Specification Language

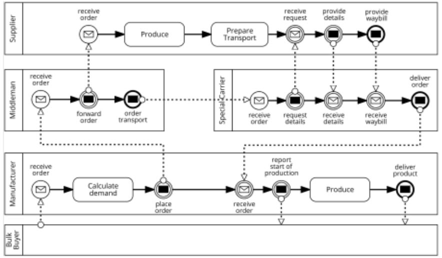
$$\frac{\text{"Code is Law"}_{\textit{Lessig}} \qquad \text{"The Law is an Ass"}_{\textit{trad.}}}{\text{"Code is an Ass"}}$$



Smart Contract Specification & Generation

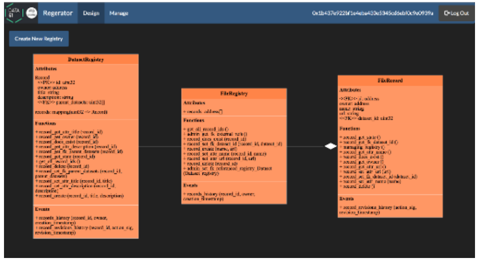
- Create and verify high-level models; transform to code & proofs
  - Domain-Specific Languages/ Model-Driven Architecture/ Formal Synthesis

- Process models, for inter-org. coordination



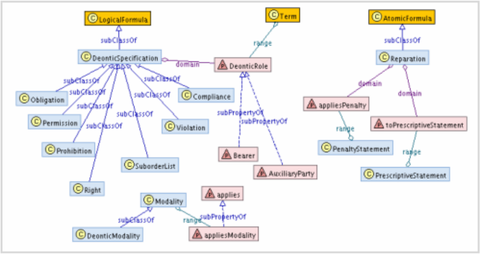
Weber et al., 2016
García-Bañuelos et al., 2017

- Schemas, for Registries



Tran et al., 2017

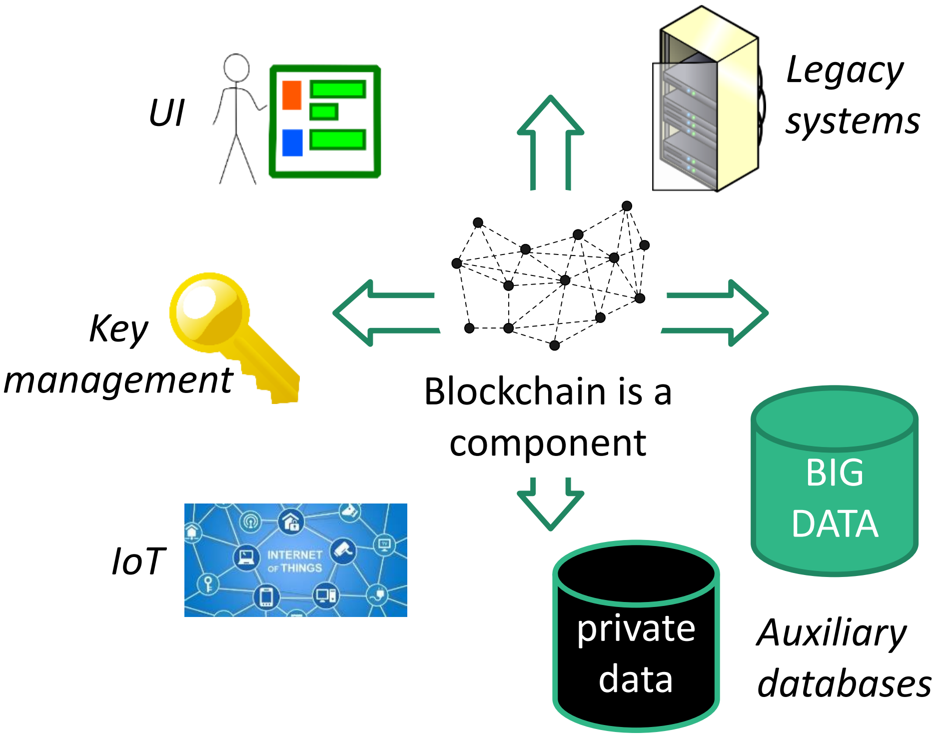
- Declarative logics, for Legal contracts



Idelberger et al., 2016

Trustworthy Blockchain-Based Systems?

- Smart Contracts are just part of the technology stack
- Functional correctness is just one aspect of fitness
- Need system-level assurance arguments about all qualities



Wrapping Up…

Other Research Areas? Porru et al. (2017)

- Blockchain challenges
  - New professional roles
  - Security and reliability
    - Smart contract testing
    - Blockchain transaction testing
  - Software architecture
  - Modeling languages
  - Metrics
- New research directions
  - Testing
  - Collaboration
  - Enhancement of testing & debugging
  - Creation of software tools for smart contract languages

“Blockchain-oriented Software Engineering: Challenges and Directions”, Porru, S., Pinna, A., Marchesi, M., Tonelli, R., ICSE2017

NATO 1968 for Blockchain?

<u>Design</u> • Model-driven blockchain-based systems • Smart contract specification • Blockchain architecture & design • Cost, performance prediction	<u>Service</u> • Release management • Blockchain monitoring • Inter-chain operations • Recovery from forks/uncles
<u>Production</u> • Testing for smart contracts & txns • Estimation & metrics for blockchain dev • Smart contract tooling • Performance and scaling	<u>Other topics</u> • Professional roles • Education • Handling blockchain software crises

Thanks!

Questions?

www.csiro.au

References

1. Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. *A taxonomy of blockchain-based systems for architecture design.* In ICSA'17: IEEE International Conference on Software Architecture, Gothenburg, Sweden, April 2017.
2. Paul Rimba, An Binh Tran, Ingo Weber, Mark Staples, Alexander Ponomarev, and Xiwei Xu. *Comparing blockchain and cloud services for business process execution.* In ICSA'17: IEEE International Conference on Software Architecture, short paper, Gothenburg, Sweden, April 2017.
3. Rajitha Yasaweerasinghelage, Mark Staples, and Ingo Weber. *Predicting latency of blockchain-based systems using architectural modelling and simulation.* In ICSA'17: IEEE International Conference on Software Architecture, short paper, Gothenburg, Sweden, April 2017.
4. Ingo Weber, Sherry Xu, Regis Riveret, Guido Governatori, Alexander Ponomarev and Jan Mendling. *Untrusted business process monitoring and execution using blockchain.* In BPM'16: International Conference on Business Process Management, Rio de Janeiro, Brazil , September, 2016
5. Luciano García-Bañuelos, Alexander Ponomarev, Marlon Dumas, and Ingo Weber. *Optimized Execution of Business Processes on Blockchain.* In BPM'17: International Conference on Business Process Management, Barcelona, Spain, September 2017
6. An Binh Tran, Xiwei Xu, Ingo Weber, Mark Staples, and Paul Rimba. *Regerator: a registry generator for blockchain.* In CAiSE'17: International Conference on Advanced Information Systems Engineering, Forum Track (demo), June 2017.
7. Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran and Shiping Chen. *The blockchain as a software connector.* In: WICSA, Venice, Italy, April, 2016
8. Luke Anderson, Ralph Holz, Alexander Ponomarev, Paul Rimba, Ingo Weber. *New kids on the block: an analysis of modern blockchains.* http://arxiv.org/abs/1606.06530
9. X. L. Yu, X. Xu, B. Liu, *EthDrive: A Peer-to-Peer Data Storage with Provenance*, In CAiSE'17: International Conference on Advanced Information Systems Engineering, June 2017.
10. C. Natoli, V. Gramoli, *The Blockchain Anomaly*, NCA2016
11. V. Gramoli, *On the Danger of Private Blockchains*, DCCL2016
12. T. Crain, V. Gramoli, M. Larrea, M. Raynal, *(Leader/Randomization/Signature)-free Byzantine Consensus for Consortium Blockchains*, http://arxiv.org/abs/arXiv:1702.03068, 2016