Adaptive Tracing

Dorsal Lab- Polytechnique Montreal

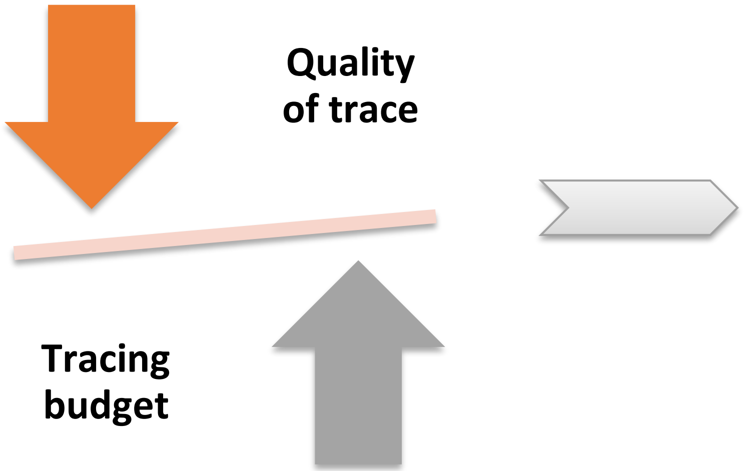
Student: Masoumeh Nourollahi

Advisor: Michel Dagenais

Co-Advisor: Naser Ezzati

Adaptive tracing

- Problem
  - Fixed tracing without considering execution
  - Overhead of tracing in collection and aggregation phase
  - Resource constraints
- Solution: Adaptive tracing
  - Automatic runtime instrumentation enable/disable
  - Hybrid runtime sampling rate changes



✔ Desired Observability Level

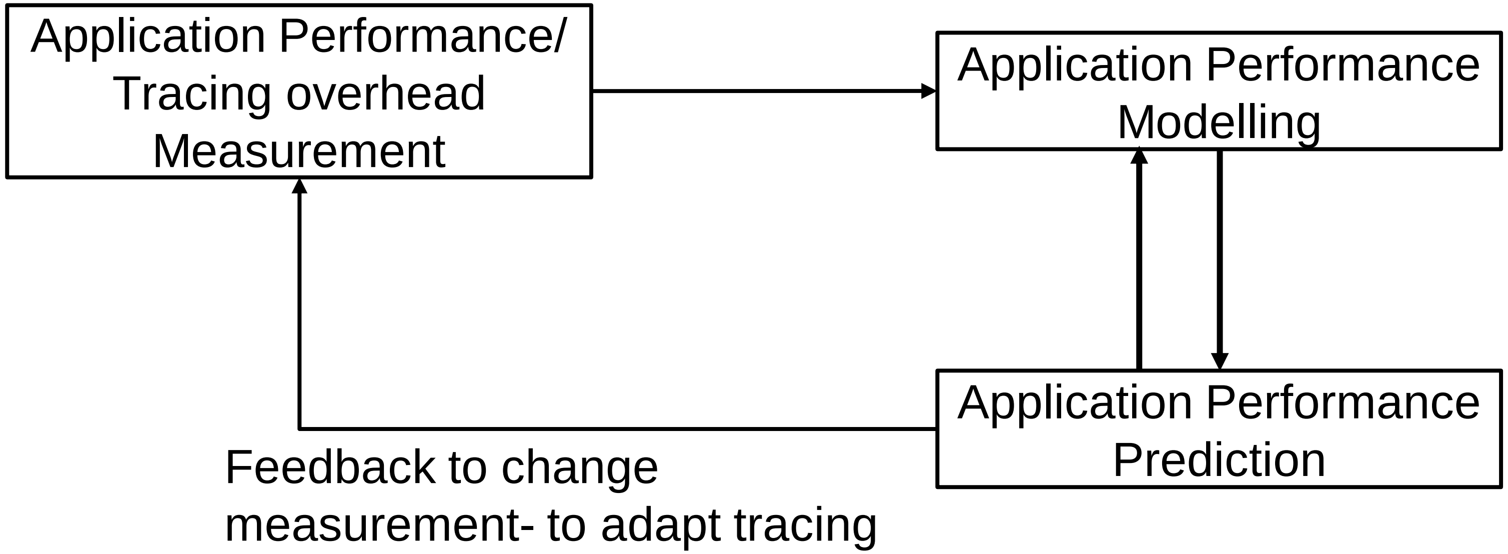
✔ Specified Tracing Overhead

Related work

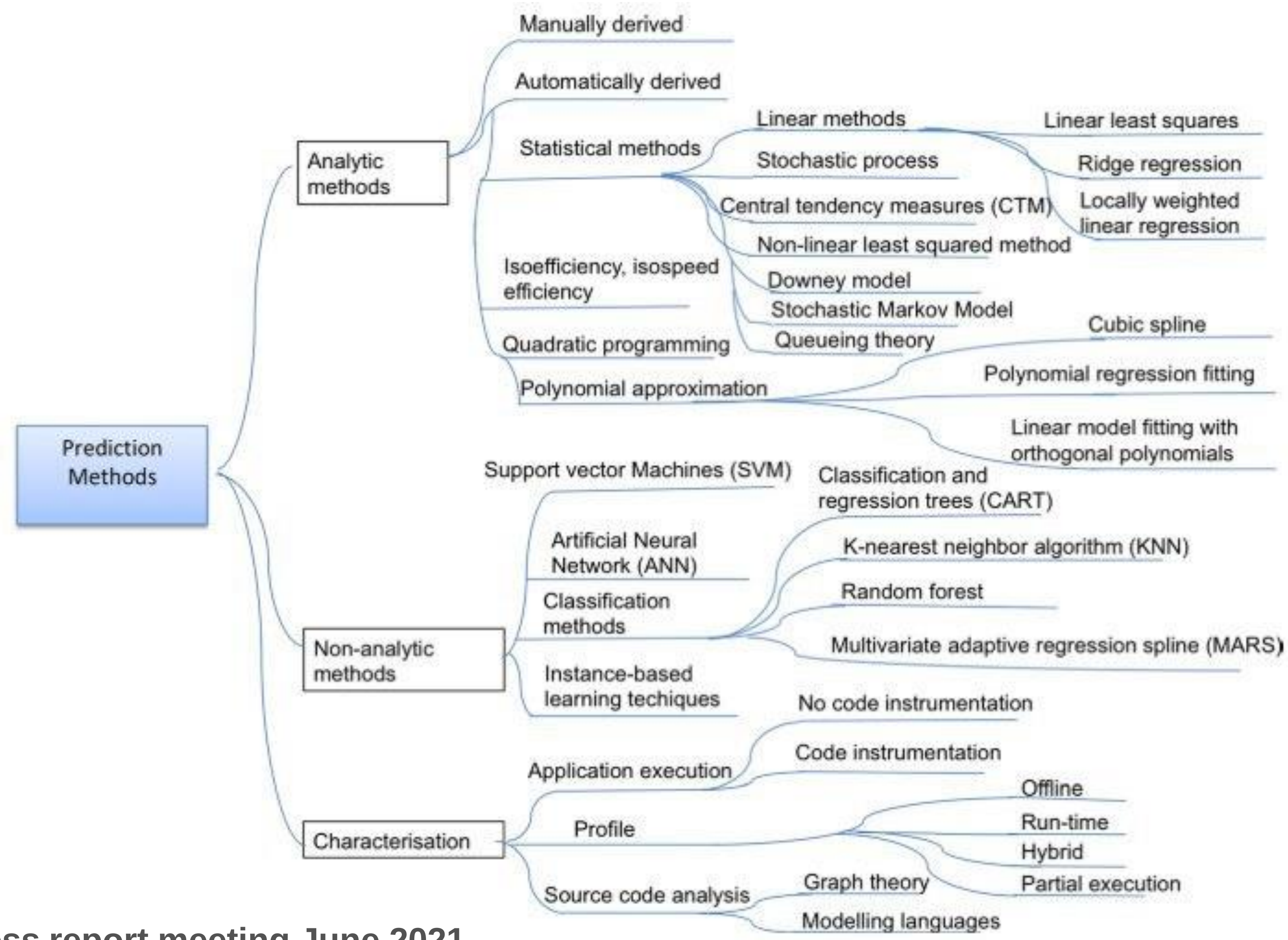
- Measurement
  - Using Performance Variation for Instrumentation Placement in Distributed Systems- 2019
  - Runtime latency detection and analysis- 2016
- Modeling
  - Tracey - Distributed Trace Comparison and Aggregation using NLP techniques- 2019
  - Diagnostic Framework for Distributed Application Performance Anomaly Based on Adaptive Instrumentation- 2020
  - Automated Analysis of Distributed Tracing: Challenges and Research Directions- 2021
- Simulation

Adaptive tracing pipeline

With specified tracing budget



Performance Prediction Methods



Performance Modeling

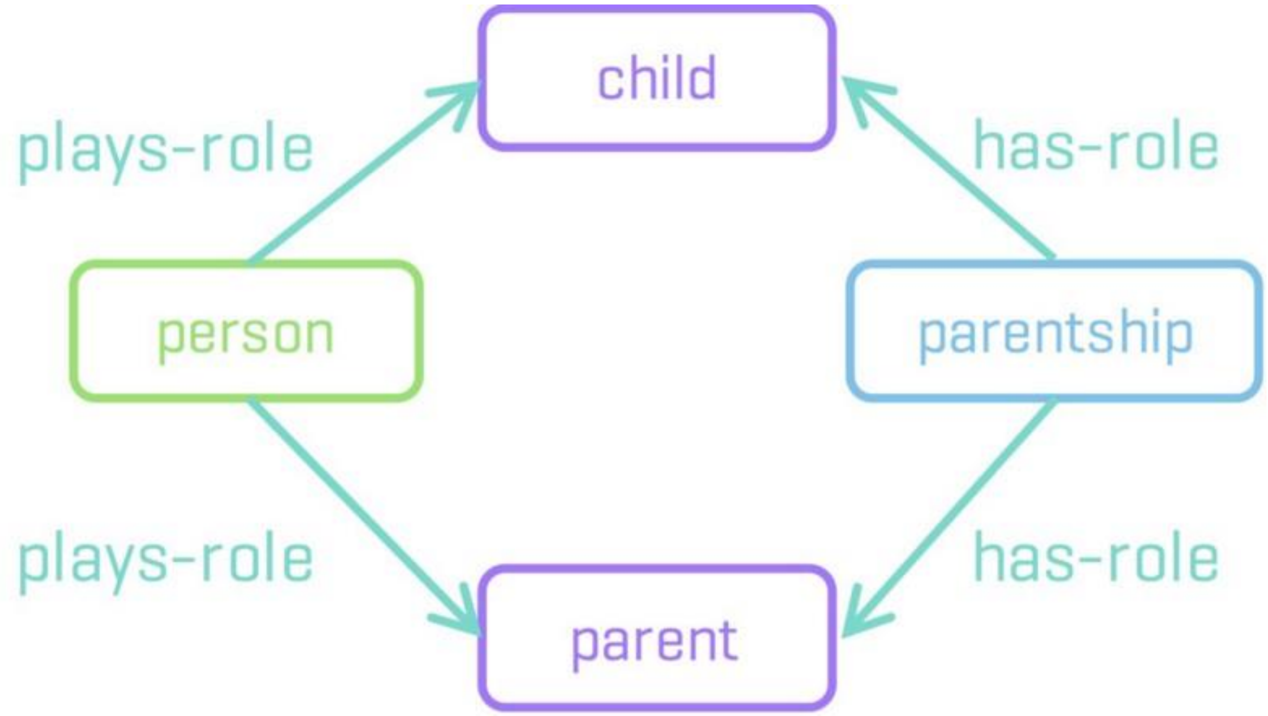
- Methods
  - Simulation
  - Analytic
    - Queuing models
    - Execution flow models
    - Models of communicating objects

- Goals:

1. Modeling expected behavior of the system according to SLAs provided or Performance prediction of the system
2. Modeling search space for the tracer to pinpoint the area in system performance that requires more attention

Knowledge-based models

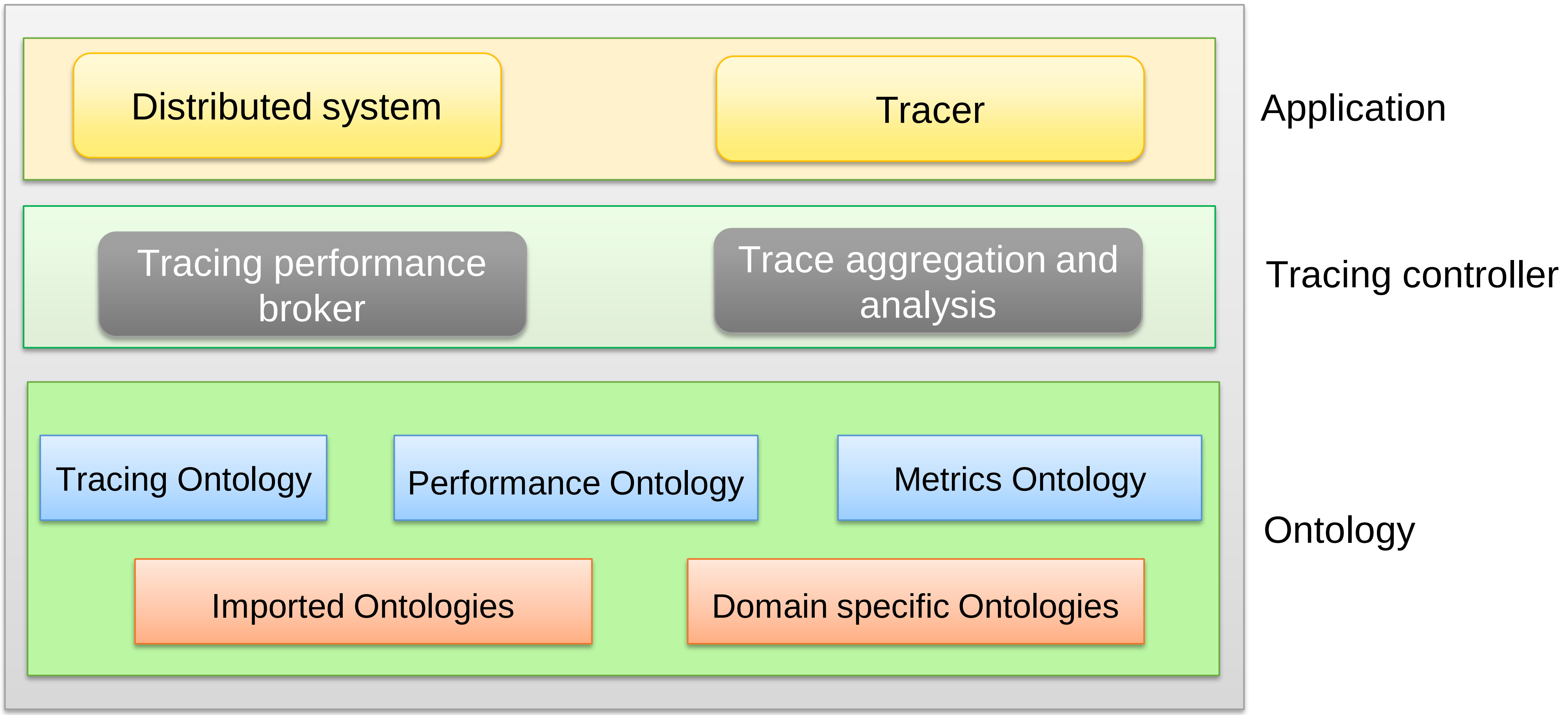
- Ontology is a model for describing the world that consists of a set of entities, properties, and relationship types.

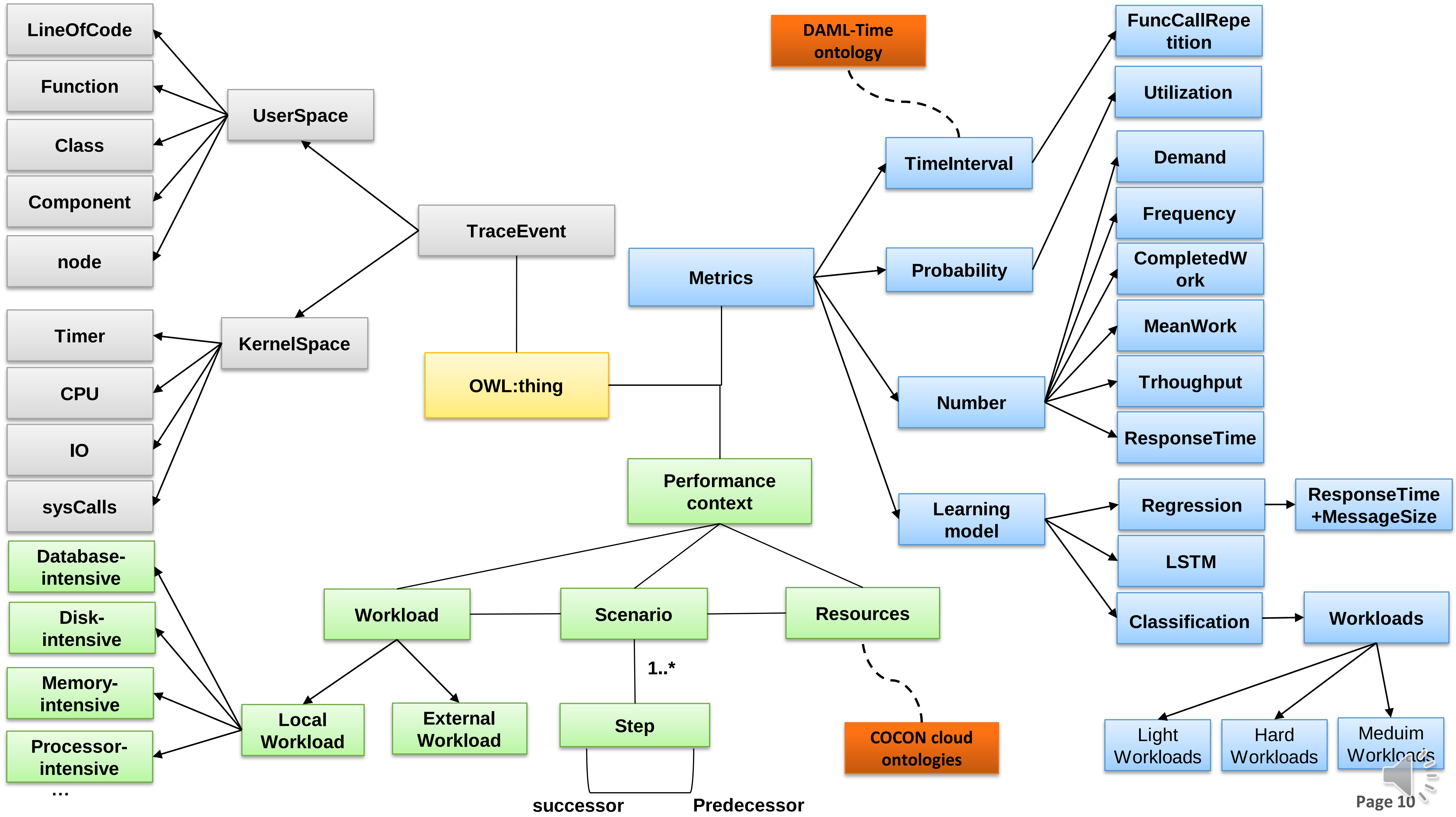


Why ontologies?

- Reusability
  - an ontology integrates in its definition all other ontologies of different knowledge
- Context-awareness
- Reasoning
  - Automatically check for unintended relationships between classes to discover inconsistencies and also automatically infers implicit information from data

Adaptive Tracing




LineOfCode
Function
Class
Component
node
UserSpace
TraceEvent
KernelSpace
Timer
CPU
IO
sysCalls
OWL:thing
DAML-Time ontology
TimeInterval
Probability
Metrics
Number
Learning model
FuncCallRepetition
Utilization
Demand
Frequency
CompletedWork
MeanWork
Trhoughput
ResponseTime
Regression
ResponseTime +MessageSize
LSTM
Classification
Workloads
Light Workloads
Hard Workloads
Meduim Workloads
Performance context
Workload
Scenario
Resources
Database-intensive
Disk-intensive
Memory-intensive
Processor-intensive
…
Local Workload
External Workload
Step
1..*
successor
Predecessor
COCON cloud ontologies

Sample use case-FuncCallRepetition metric

Load Generation

The HEAT CLINIC

An online shop application written with Java
Deployed on Apache tom-cat server
Database: mysql
Platform: ubuntu 18.04
Tracer: LTTng

Load generation: APACHE JMeter™

Workload:
1. Normal
2. View on demand
3. Shopping specific item on demand

Scenarios:
1. Browsing the main page of the shop
2. Browsing a specific product
3. Add to cart
4. Finalize purchase

Metric: FuncCallRepetition

Data gathering:
1. Instrument Heat-Clinic application by inserting probes on different levels (Loc, function, …)
2. Label each event by LOC+OriginFile+Priority
3. Run different workloads on the test application
4. Gather traces by activating full userspace tracing

Measurement method:
1. Find caller and callee of each event by building paths of length 3 of events
2. Count number of repetitions of each 3-event length path in the specified time-interval (eg. 10 seconds)

3 disjoint classes for call frequency of event X in path Y:
1. High-freq: In time-interval T, number of event X calls in path Y is greater than threshold1
2. Medium-freq: In time-interval T, number of event X calls in path Y is between threshold1 and threshold12
3. Low-freq: In time-interval T, number of event X calls in path Y is less than threshold2

2 disjoint classes for event priority:
1. High-priority: priority of event X calls in path is high
2. Low-priority: priority of event X calls in path is low

Tracing Adaptation

6 disjoint classes for event X in path Y:
Freq1: (priority high, freq greater than threshold1)
Freq2: (priority low, freq greater than threshold1)
Freq3: (priority high , freq between threshold 1,2)
Freq4: (priority low, freq between threshold 1,2)
Freq5: (priority high, freq less than threshold2)
Freq6: (priority low, freq less than threshold2)

Actions:
Freq1: sample 1% of event X calls
Freq2: sample event X 1 time every time-interval
Freq3: sample 1% of event X calls
Freq4: disable tracepoints for event X
Freq5: no action
Freq6: disable tracepoints for event X

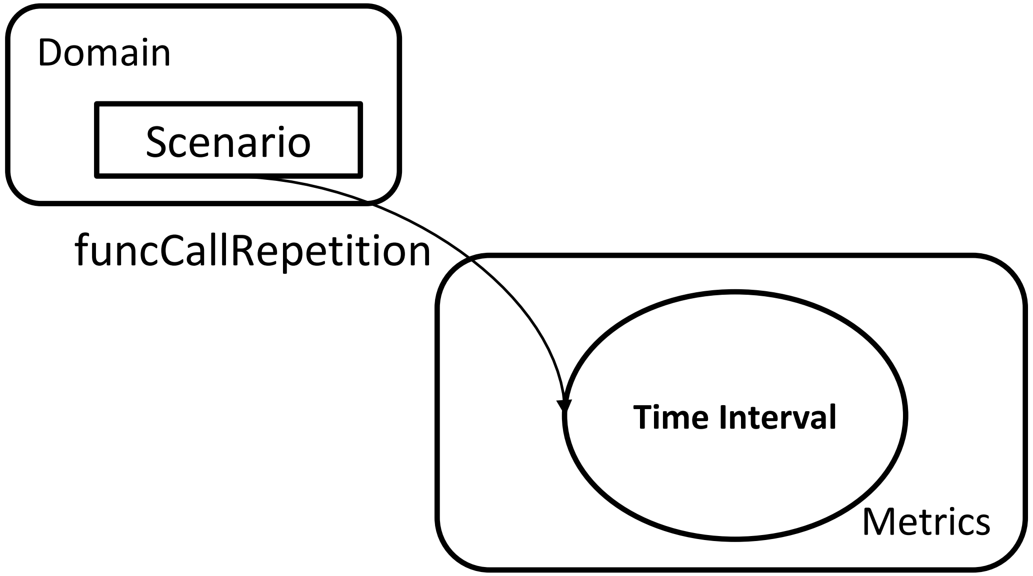
Example OWL Class Definition

Function Call Repetition metric description in OWL

```
<owl:FunctionalProperty rdf:ID=" funcCallRepetition ">
        <rdfs:range rdf:resource="# timeInterval "/>
        <rdf:type rdf:resource="&owl;ObjectProperty"/>
        <rdfs:domain rdf:resource="#Scenario"/>
</owl:FunctionalProperty>

<owl:FunctionalProperty rdf:ID="population">
        <rdfs:domain rdf:resource="#Frequncy"/>
        <rdfs:range rdf:resource="#Number"/>
        <rdf:type rdf:resource="&owl;ObjectProperty"/>
</owl:FunctionalProperty>

<owl:Class rdf:ID="Number">
        <rdfs:subClassOfrdf:resource="#Metric"/>
 </owl:Class>
```



Example OWL Metric Class Definition

Rule definition of High-freq path call

Event(?event) ∧
Scenario(?sce) ∧
frequency(?sce, ?interval) ∧
count(?sce, ?number) ∧
hasEventScenario(?event, ? count) ∧
swrlb:greaterThan(? number, ?threshold1)
→ High-freq(?sce)

Rule definition of Medium-freq path call

Event(?event) ∧
Scenario(?sce) ∧
frequency(?sce, ?interval) ∧
count(?sce, ?number) ∧
hasEventScenario(?event, ? count) ∧
swrlb:lessThan(? number, ?threshold1) ∧
swrlb: greaterThan(? number, ?threshold2)
→ Medium-freq(?sce)

Rule definition of Low-freq path call

Event(?event) ∧
Scenario(?sce) ∧
frequency(?sce, ?interval) ∧
count(?sce, ?number) ∧
hasEventScenario(?event, ? count) ∧
swrlb:lessThan(? number, ?threshold2)
→ Low-freq(?sce)

Rule definition LowPriorityEvent

Event(?event) ∧
Scenario(?sce) ∧
priority(?event, ?pr) ∧
hasPath(?event, ?sce)
→ LowPriorityEvent(?event)

Rule definition HighPriorityEvent

Event(?event) ∧
Scenario(?sce) ∧
priority(?event, ?pr) ∧
hasPath(?event, ?sce)
→ HighPriorityEvent(?event)

Example OWL FuncCallRepetition Metric Class and Action Definition

OWL definition of FuncCallRepetition

```
<owl:Class rdf:ID=" FuncCallRepetition">
        <rdfs:subClassOf rdf:resource="#Interval"/>
        <rdfs:subClassOf>
                <owl:Class>
                <owl:intersectionOf rdf:parseType="collection">
                        <owl:Class rdf:about="# eventFequency"/>
                        <owl:Class rdf:about="# eventPriority"/>
                </owl:intersectionOf>
                </owl:Class>
        </rdfs:subClassOf>
</owl:Class>
```

Freq1 Class Action-Rule definition

Scenario(?sce) ∧ Event(?event) ∧
isExecutedIn(?sce, ?event) ∧
hasFrequncyInTimeInterval(?sce, ?Value) ∧
hasPriority(?pr, ?value)
swrlb:greaterThan(?value, ?threshold1)
→Action:Sample 1% of event X calls (?event)

Conclusions and future work

- Modeling provides us an abstract view which facilitates observability goal-based tracing
- “Ontology reuse” makes this abstract view flexible to work with any tracing tool, domain and infrastructure, by considering their own specific knowledge models
- To demonstrate modelling benefits in use we plan to implement several use-cases to achieve observability goals like bottleneck identification

References

- Flores-Contreras, J., Duran-Limon, H.A., Chavoya, A. *et al.* Performance prediction of parallel applications: a systematic literature review. *J Supercomput* 77, 4014–4055 (2021).
- Al Haider N., Gaudin B., Murphy J. (2012) Execution Trace Exploration and Analysis Using Ontologies. In: Khurshid S., Sen K. (eds) Runtime Verification. RV 2011. Lecture Notes in Computer Science, vol 7186. Springer, Berlin, Heidelberg.
- Bento, A., Correia, J., Filipe, R. *et al.* Automated Analysis of Distributed Tracing: Challenges and Research Directions. *J Grid Computing* 19, 9 (2021).
- S. Zhang, D. Liu, L. Zhou, Z. Ren and Z. Wang, "Diagnostic Framework for Distributed Application Performance Anomaly Based on Adaptive Instrumentation," *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, 2020, pp. 164-169, doi: 10.1109/ICCCI49374.2020.9145997.
- Johng H., Kim D., Hill T., Chung L. (2018) Estimating the Performance of Cloud-Based Systems Using Benchmarking and Simulation in a Complementary Manner. In: Pahl C., Vukovic M., Yin J., Yu Q. (eds) Service-Oriented Computing. ICSOC 2018. Lecture Notes in Computer Science, vol 11236. Springer, Cham
- Sturmann, Lilian. 2019. Using Performance Variation for Instrumentation Placement in Distributed Systems. Master's thesis, Harvard Extension School
- Zhang Q., Haller A., Wang Q. (2019) CoCoOn: Cloud Computing Ontology for IaaS Price and Performance Comparison. In: Ghidini C. et al. (eds) The Semantic Web – ISWC 2019. ISWC 2019. Lecture Notes in Computer Science, vol 11779. Springer, Cham.
- Williams L.G., Smith C.U. (1995) Information requirements for software performance engineering. In: Beilner H., Bause F. (eds) Quantitative Evaluation of Computing and Communication Systems. TOOLS 1995. Lecture Notes in Computer Science, vol 977. Springer, Berlin, Heidelberg.
- Vittorio Cortellessa, Antinisca Di Marco, and Paola Inverardi. 2011. Model-Based Software Performance Analysis (1st. ed.). Springer Publishing Company, Incorporated.
- C. Guerrero, C. Juiz and R. Puigjaner, "Web Performance and Behavior Ontology," *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, 2008, pp. 219-225, doi: 10.1109/CISIS.2008.101.
- Lera, I., Sancho, P.P., Juiz, C. *et al.* Performance assessment of intelligent distributed systems through software performance ontology engineering (SPOE). *Software Qual J* 15, 53–67 (2007).
- Isaac Lera, Carlos Juiz, Ramon Puigjaner,"Performance-related ontologies and semantic web applications for on-line performance assessment of intelligent systems", Science of Computer Programming, Volume 61, Issue 1, 2006, Pages 27-37, ISSN 0167-6423,
- Lera I., Juiz C., Puigjaner R. (2005) Web Operational Analysis Through Performance-Related Ontologies in OWL for Intelligent Applications. In: Lowe D., Gaedke M. (eds) Web Engineering. ICWE 2005. Lecture Notes in Computer Science, vol 3579. Springer, Berlin, Heidelberg
- A. Desai, K. Rajan, K. Vaswani, "Critical Path based Performance Models for Distributed Queries",Microsoft Technical Report, MSR-TR-2012-121, 2012.
- Junior, Vanderlei Freitas et al. “ONTOLOGY FOR PERFORMANCE MEASUREMENT INDICATORS’ COMPARISON.” *International Journal of Digital Information and Wireless Communications* 6 (2016): 87-96.
- Soergel D, Helfer O. A Metrics Ontology. An intellectual infrastructure for defining, managing, and applying metrics. *Knowl Organ Sustain World Chall Perspect Cult Sci Technol Shar Connect Soc (2016)*. 2016;15:333-341.