CITY UNIVERSITY LONDON

Davide Lorenzoli, George Spanoudakis

EVEREST+

RUN-TIME SLA VIOLATION PREDICTION

SEVENTH FRAMEWORK PROGRAMME

SLA@SOI

MW4SOC, Bagalore, 29/11/2010

Outline

- Prediction approaches limitations
- Our vision
- EVEREST+
- Experimental results
- Conclusions

Prediction approaches limitations

- They tend to focus on system infrastructure properties rather than service level application based properties.
- They tend to focus on the prediction of specific types of properties without providing a more generic framework for building predictors.
- They are not integrated with environments for monitoring SLAs for service-based systems.

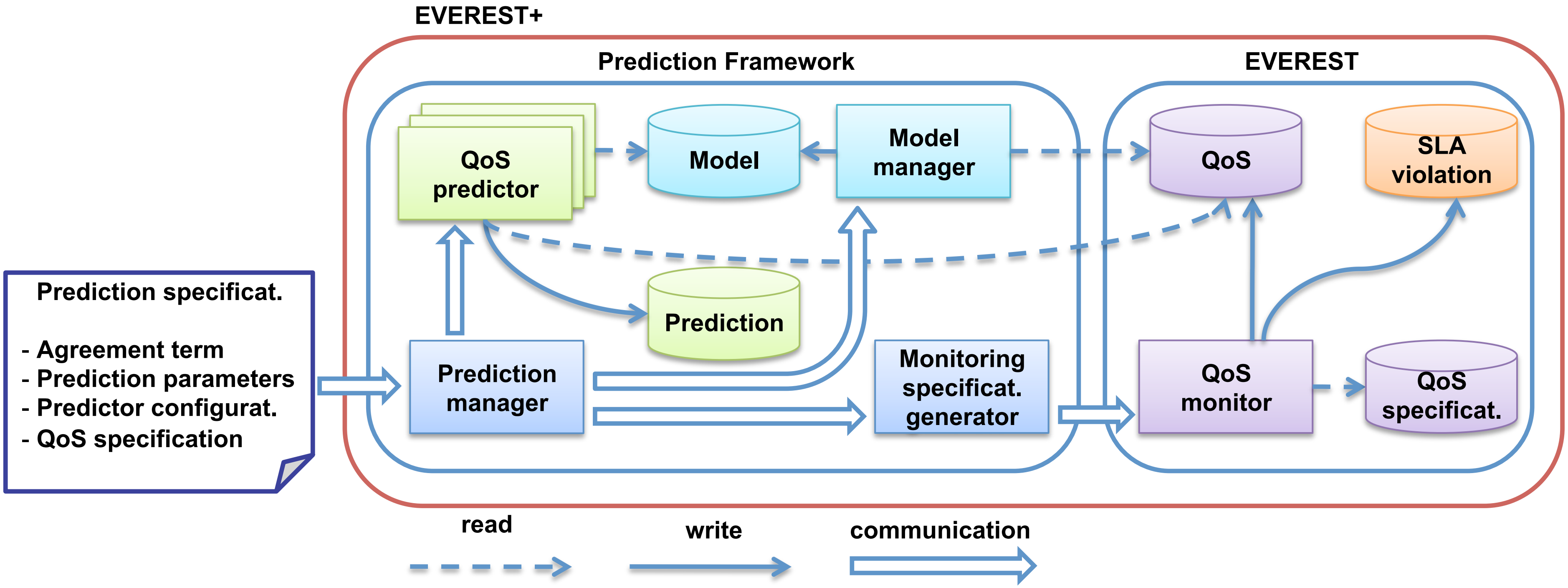
Our vision

- To focus on system infrastructure properties and service level application based properties.
- To provide a more generic framework for building predictors that can cover a wide or even the whole spectrum of service properties that can be part of an SLA
- To integrate with environments for monitoring SLAs for service-based systems

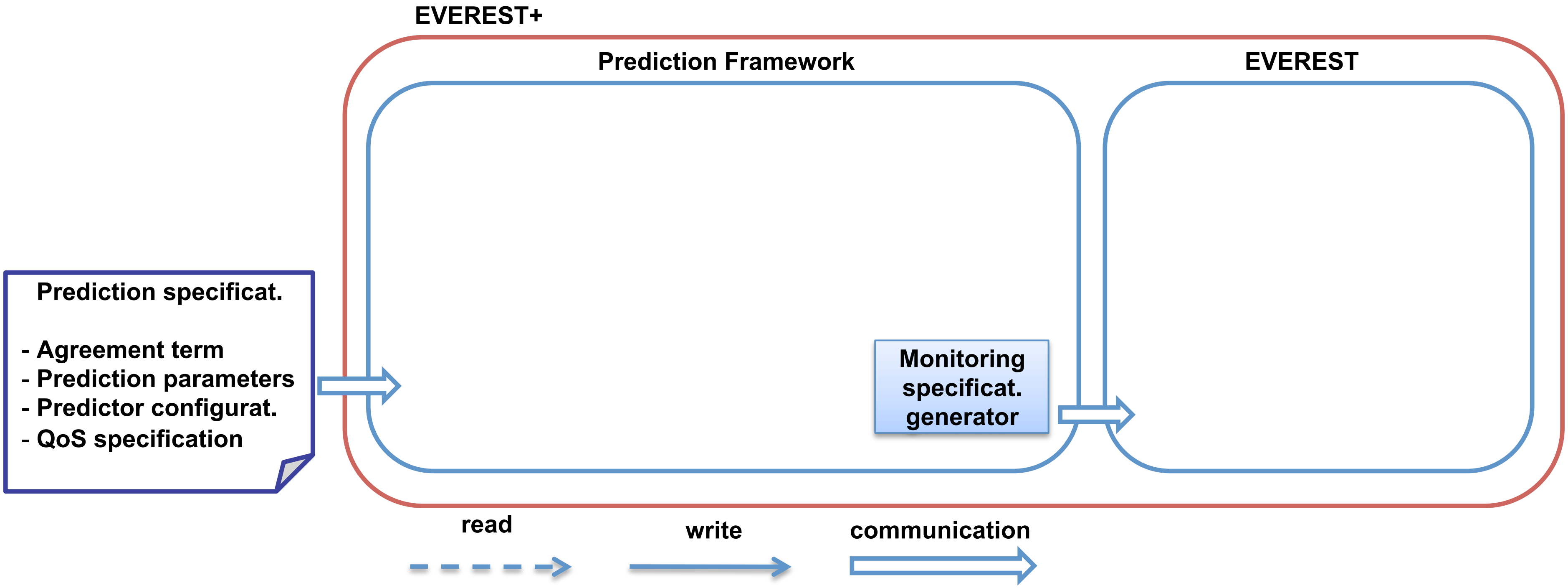
EVEREST+

- EVEREST+ is a framework for integrated monitoring and prediction
- EVEREST+ uses prediction specifications to setup both the monitoring and the prediction framework
- EVEREST+ provides the means for developing new predictors

EVEREST+: architecture



EVEREST+: architecture



EVEREST+

Prediction Framework

EVEREST

Prediction specificat.

- Agreement term
- Prediction parameters
- Predictor configurat.
- QoS specification

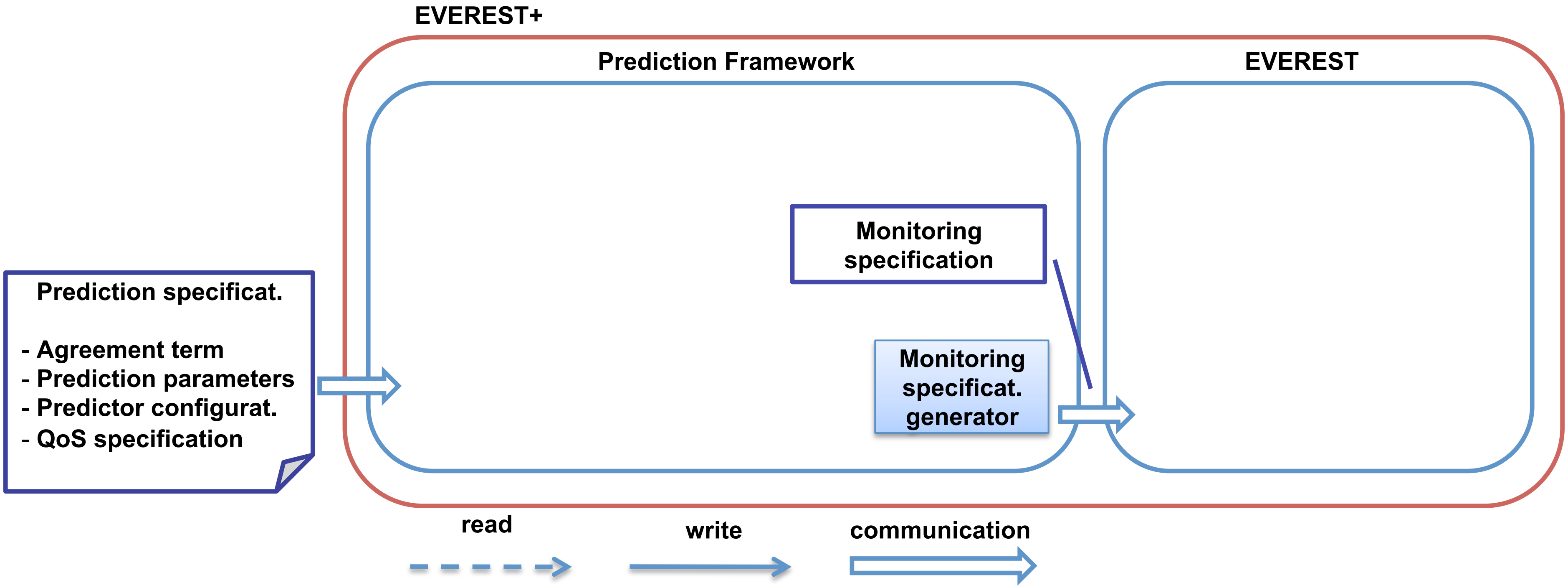
Monitoring specificat. generator

read

write

communication

EVEREST+: architecture

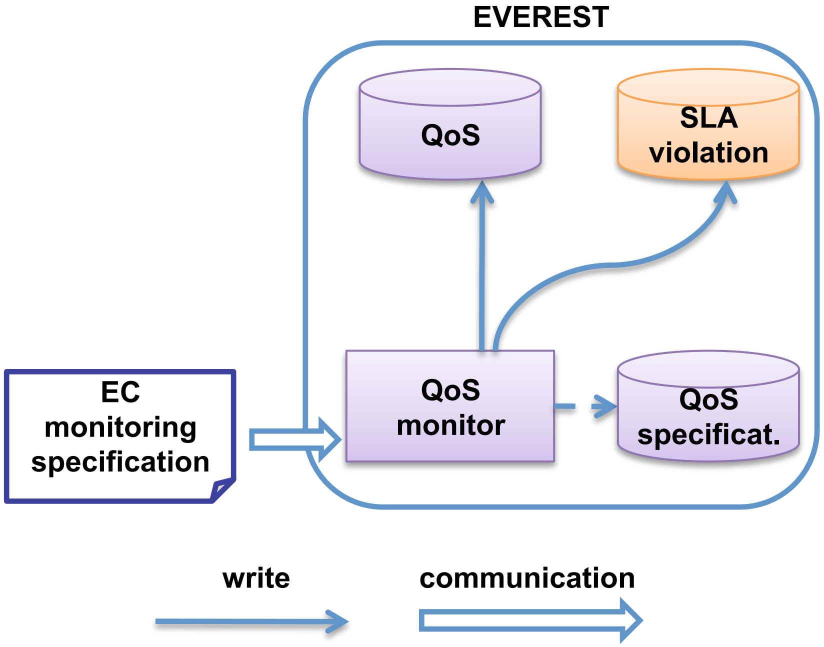


Prediction specificat.

- Agreement term
- Prediction parameters
- Predictor configurat.
- QoS specification

EVERST+: monitoring framework EVERST

- Generic
- Based on Event Calculus (EC)
- Rules:
  - `body ⇒ head`
- Predicates:
  - *Happens(e,t,R(lb,ub))*
  - *HoldsAt(f,t)*
  - *Initiates(e,f,t)*
  - *Terminates(e,f,t)*
  - *Initially(f)*



Example

EVERST+: monitoring framework EVERST

- Mean Time To Repair (MTTR) QoS: the formula checks whether the MTTR of service _*Srv* is always below a given threshold K, i.e., MTTR<K.

Rule R1:

Happens$(e(\_id1, \_Snd, \_Srv, Call(\_O), \_Srv), t_1, [t_1,t_1]) \wedge$

Happens$(e(\_id2, \_Srv, \_Snd, Response(\_O), \_Srv), t_2, [t_1,t_1+d]) \wedge$

$\exists$ *_PN, _STime, _MTTR:* ***HoldsAt***$(Unavailable(\_PN, \_Srv, \_STime), t_1)) \wedge$

HoldsAt$(MTTR(\_Srv, \_PN, \_MTTR), t_1))$

$\Rightarrow \_MTTR < K$

CITY UNIVERSITY
LONDON

Example

EVERST+: monitoring framework EVERST

- Mean Time To Repair (MTTR) QoS: the formula checks whether the MTTR of service *_Srv* is always below a given threshold K, i.e., MTTR<K.

Rule R1:

$\boxed{\textbf{\textit{Happens}}(e(\_id1, \_Snd, \_Srv, Call(\_O), \_Srv), t_1, [t_1,t_1]) \wedge}$

$\textbf{\textit{Happens}}(e(\_id2, \_Srv, \_Snd, Response(\_O), \_Srv), t_2, [t_1,t_1+d]) \wedge$

$\exists \_PN, \_STime, \_MTTR\text{: } \textbf{\textit{HoldsAt}}(Unavailable(\_PN, \_Srv, \_STime), t_1)) \wedge$

$\textbf{\textit{HoldsAt}}(MTTR(\_Srv, \_PN, \_MTTR), t_1))$

$\Rightarrow \_MTTR < K$

A call to operation *_O* of service *_Srv* is performed at time point t_1

CITY UNIVERSITY
LONDON

Example

EVERST+: monitoring framework EVERST

- Mean Time To Repair (MTTR) QoS: the formula checks whether the MTTR of service _*Srv* is always below a given threshold K, i.e., MTTR<K.

Rule R1:

$$\textbf{Happens}(e(\_id1, \_Snd, \_Srv, Call(\_O), \_Srv), t_1, [t_1,t_1]) \wedge$$

$$\boxed{\textbf{Happens}(e(\_id2, \_Srv, \_Snd, Response(\_O), \_Srv), t_2, [t_1,t_1+d]) \wedge}$$

$$\exists \_PN, \_STime, \_MTTR: \textbf{HoldsAt}(Unavailable(\_PN, \_Srv, \_STime), t_1)) \wedge$$

$$\textbf{HoldsAt}(MTTR(\_Srv, \_PN, \_MTTR), t_1))$$

$$\Rightarrow \_MTTR < K$$

A response from service _*Srv* is received at time point t_1+d

Example

EVERST+: monitoring framework EVERST

- Mean Time To Repair (MTTR) QoS: the formula checks whether the MTTR of service _*Srv* is always below a given threshold K, i.e., MTTR<K.

Rule R1:

Happens(e(_id1, _Snd, _Srv, Call(_O), _Srv), t_1

Happens(e(_id2, _Srv, _Snd, Response(_O), _Srv), t_2, $[t_1, t_1+d]$) ∧

∃ _PN, _STime, _MTTR: ***HoldsAt***(Unavailable(_PN, _Srv, _STime), t_1)) ∧

HoldsAt(MTTR(_Srv, _PN, _MTTR), t_1))

⇒ _MTTR < K

Checks whether an operation call happened at the the time when the service has been unavailable

Example

EVERST+: monitoring framework EVERST

- Mean Time To Repair (MTTR) QoS: the formula checks whether the MTTR of service _*Srv* is always below a given threshold K, i.e., MTTR<K.

Rule R1:

Happens*(e(_id1, _Snd, _Srv, Call(_O), _Sr*

Happens*(e(_id2, _Srv, _Snd, Response(_O*

∃ *_PN, _STime, _MTTR:* ***HoldsAt****(Unavailable(_PN, _Srv, _STime),* t_1*))*

HoldsAt*(MTTR(_Srv, _PN, _MTTR),* t_1*))*

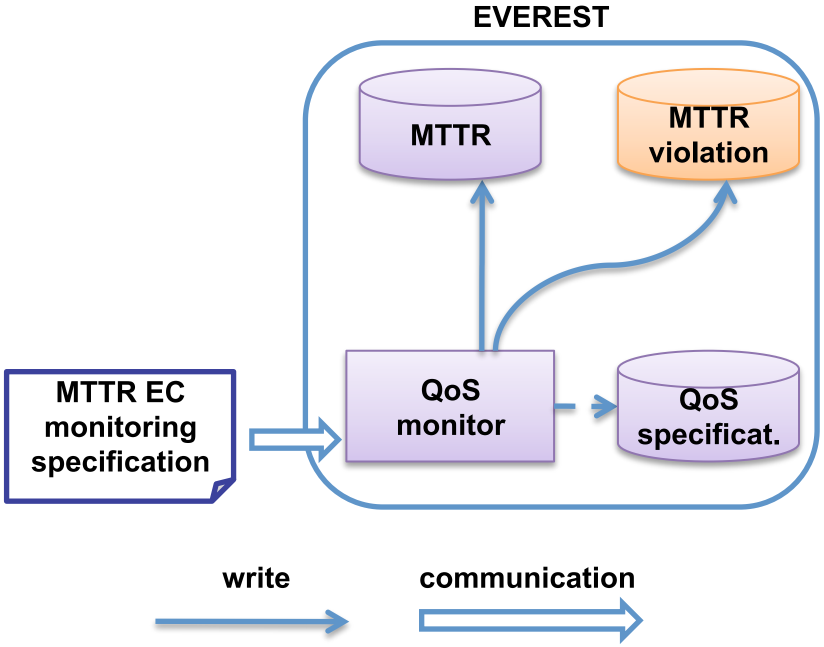
⇒ *_MTTR < K*

Checks for MTTR violations (*MTTR≥K*) when a call to an operation _*O* of the service _*Srv* is served after a period of unavailability

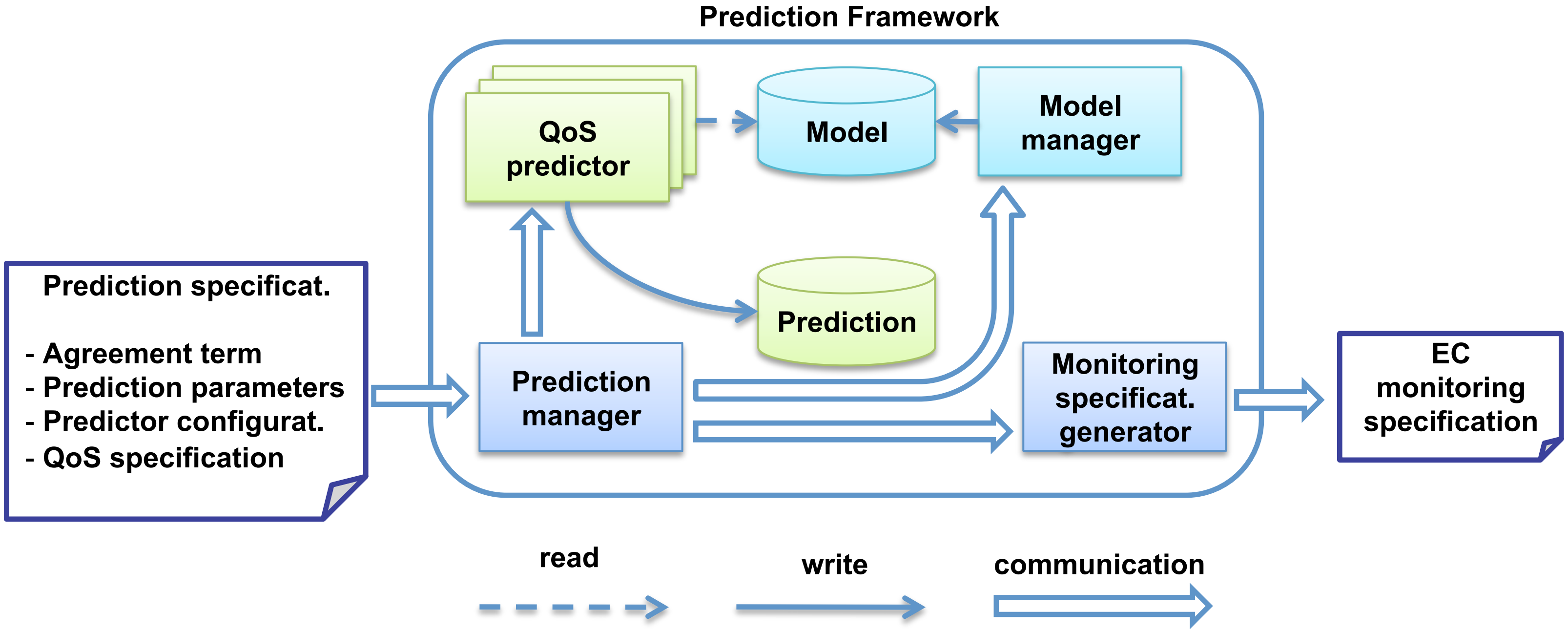
Example

EVERST+: monitoring framework EVERST

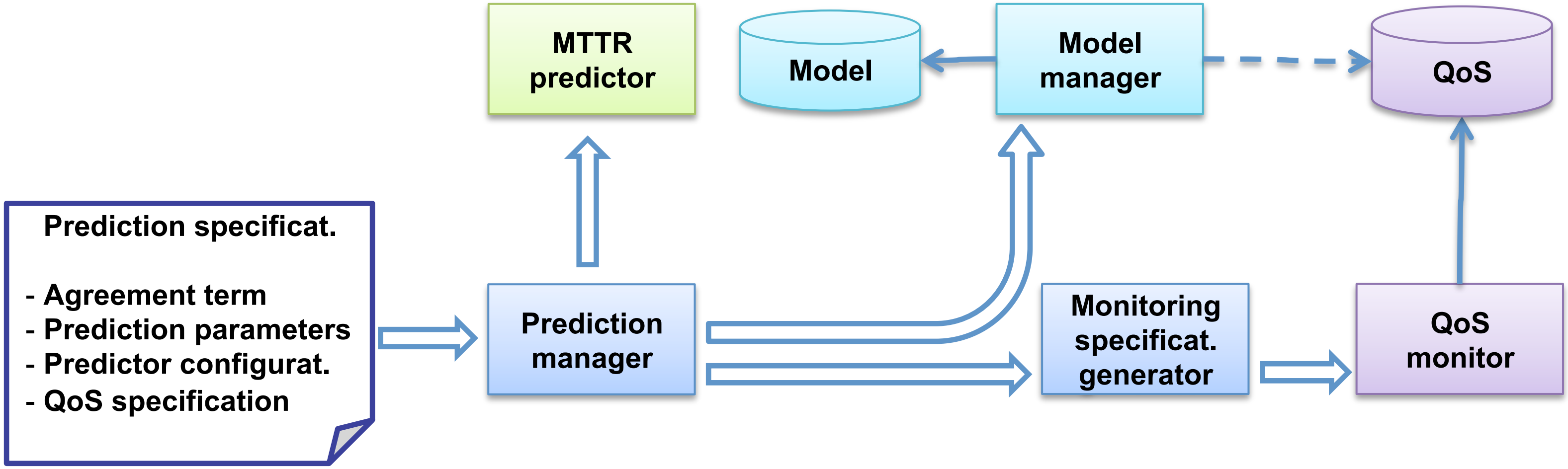
- After receiving a monitoring specification
  - computes/store MTTR values
  - checks for MTTR violations



EVEREST+: prediction framework



EVEREST+: specification driven

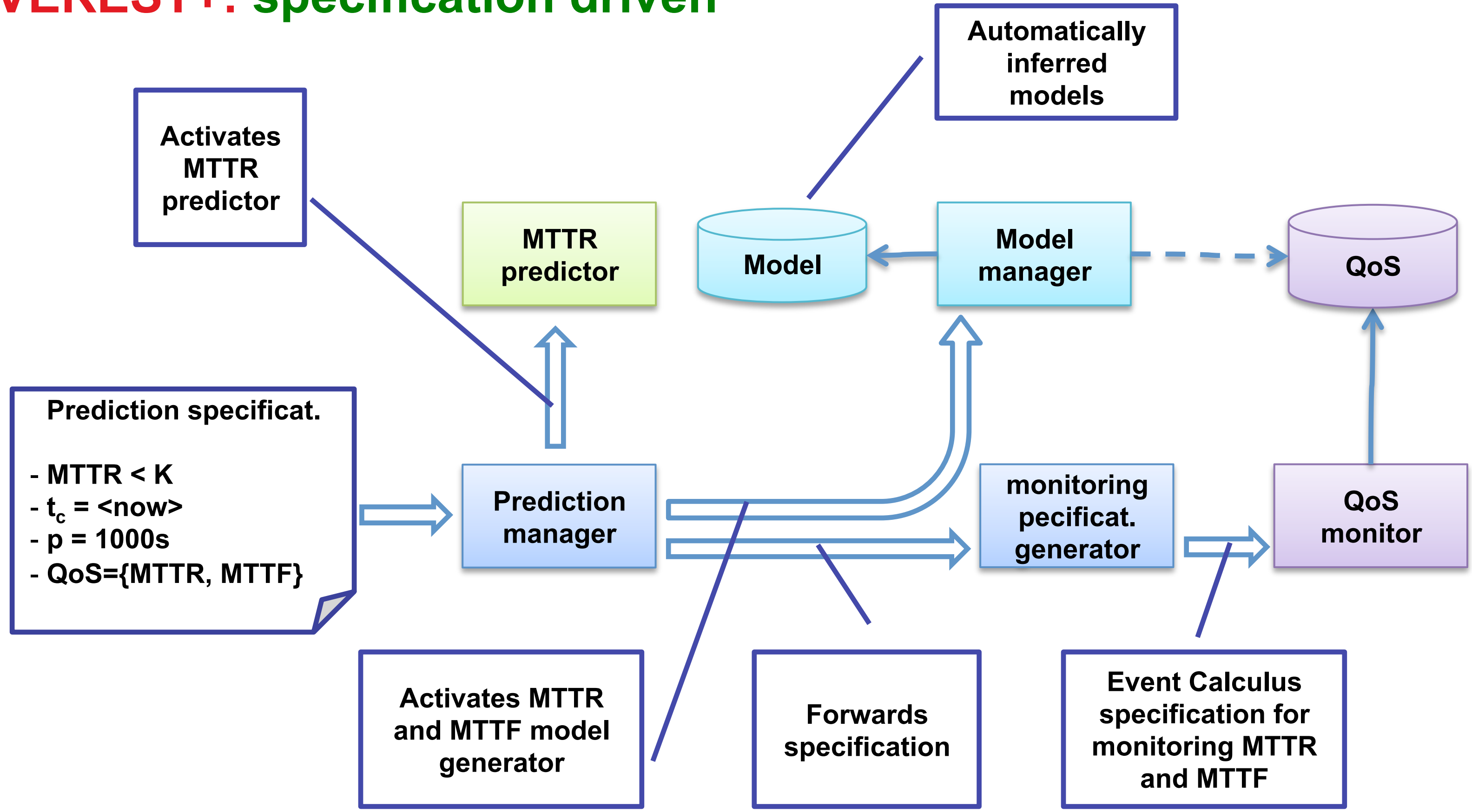


Prediction specificat.

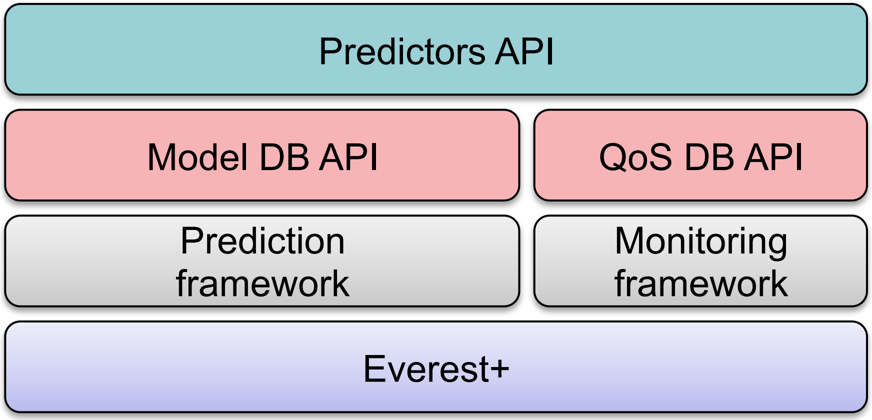
- Agreement term
- Prediction parameters
- Predictor configurat.
- QoS specification

Example

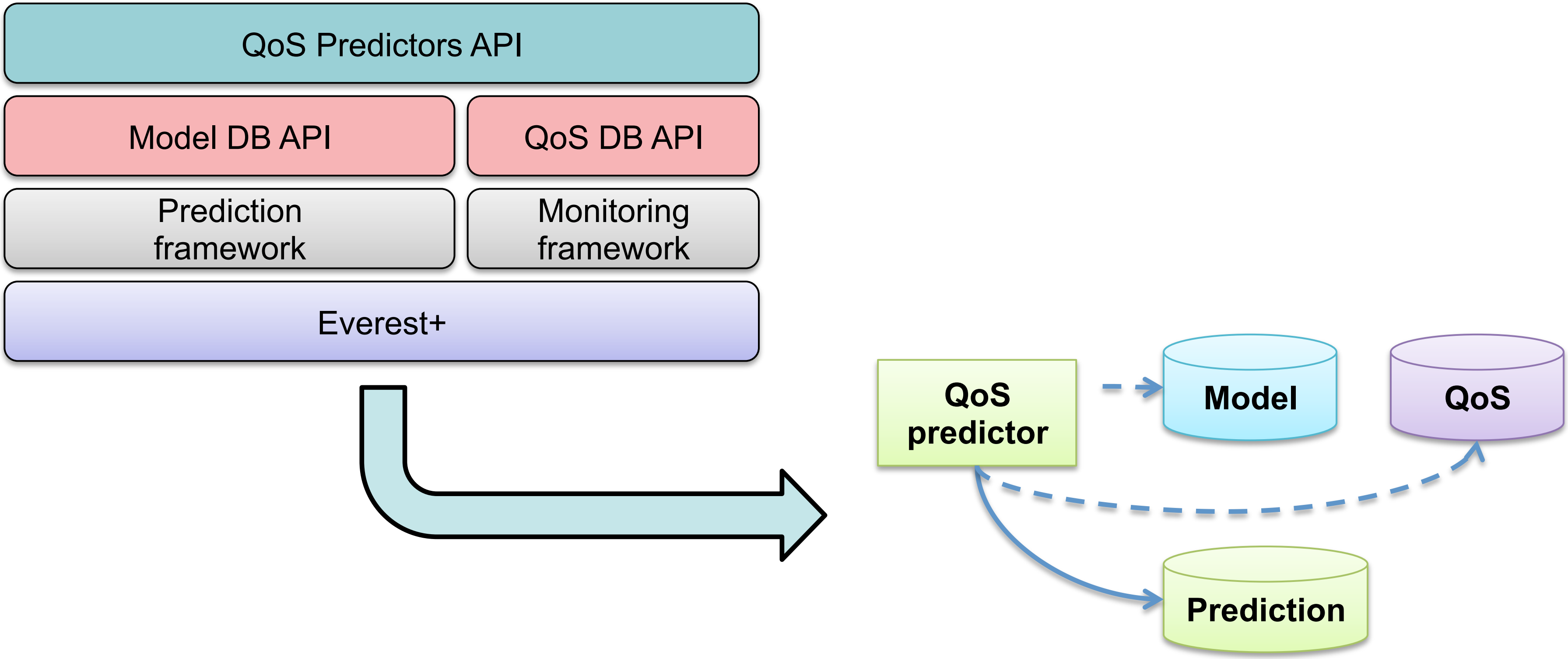
EVEREST+: specification driven



EVEREST+: generic framework for building predictors



EVEREST+: generic framework for building predictors



QoS Predictors API

Model DB API

QoS DB API

Prediction framework

Monitoring framework

Everest+

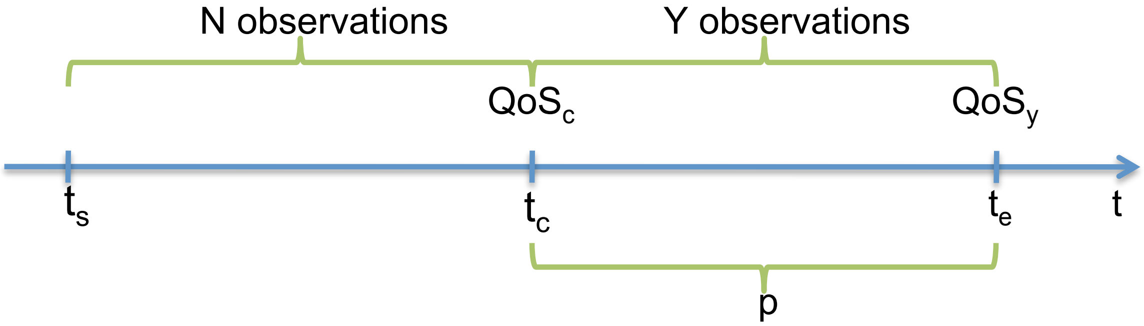
QoS predictor

Model

QoS

Prediction

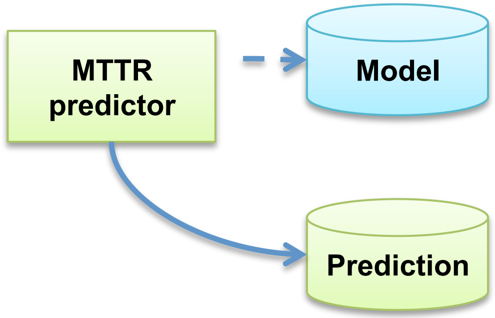
Prediction problem



Pr(QoS,K, t_e): Given a request for predicting whether a QoS property will violate a given constraint *K* set for it at some future time point t_e that is received at a time point t_c, prediction is the computation of the probability that the QoS property will violate the constraint at t_e

Example

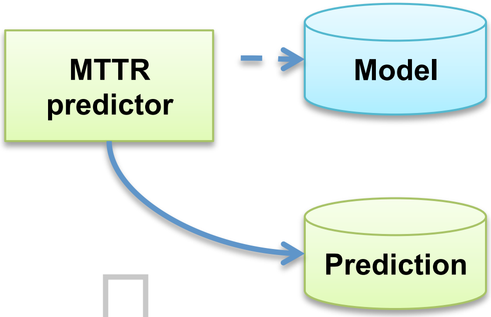
EVEREST+: generic framework for building predictors



CITY UNIVERSITY
LONDON

Example

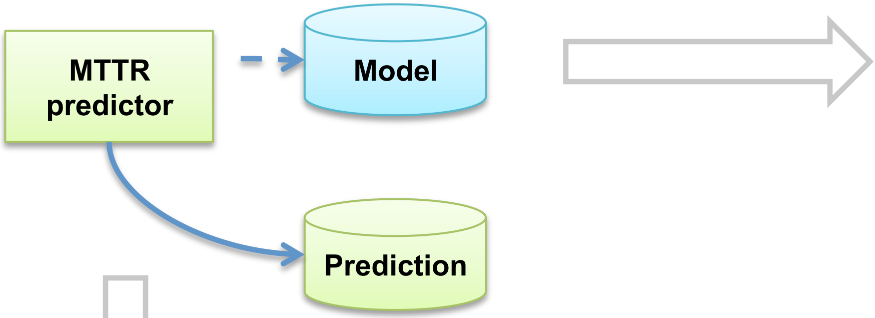
EVEREST+: generic framework for building predictors

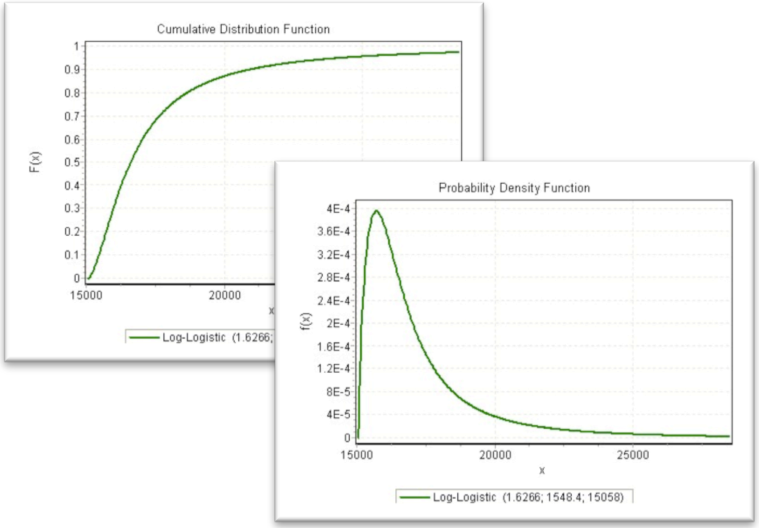


$$\Pr\left(\bigwedge_{y=1}^{M} E_y\right) = \begin{cases} 1 - \sum_{y=1}^{M}[\Pr(y) * \Pr(MTTR_{crit} > MTTR_y)], & MTTR_{crit} > K \\ \sum_{y=1}^{M}[\Pr(y) * \Pr(MTTR_{crit} \le MTTR_y)], & MTTR_{crit} \le K \end{cases}$$

Example

EVEREST+: generic framework for building predictors

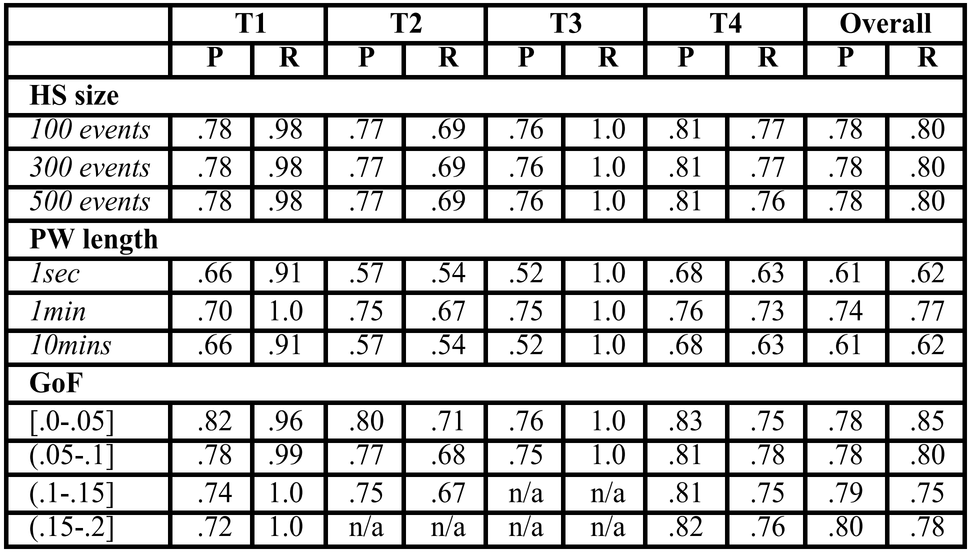




$$\Pr(\bigwedge_{y=1}^{M} E_y) = \begin{cases} 1 - \sum_{y=1}^{M}[\Pr(y) * \Pr(\mathrm{MTTR}_{crit} > \mathrm{MTTR}_y)], & \mathrm{MTTR}_{crit} > K \\ \sum_{y=1}^{M}[\Pr(y) * \Pr(\mathrm{MTTR}_{crit} \le \mathrm{MTTR}_y)], & \mathrm{MTTR}_{crit} \le K \end{cases}$$

Experimental results

- 4 MTTR Trends:
    - *T1: cyclic*
    - *T2: increasing*
    - *T3: decreasing*
    - *T4: random*
- 3 Variables:
    - History size
    - Prediction window
    - Goodness of fit
- 40 prediction measures

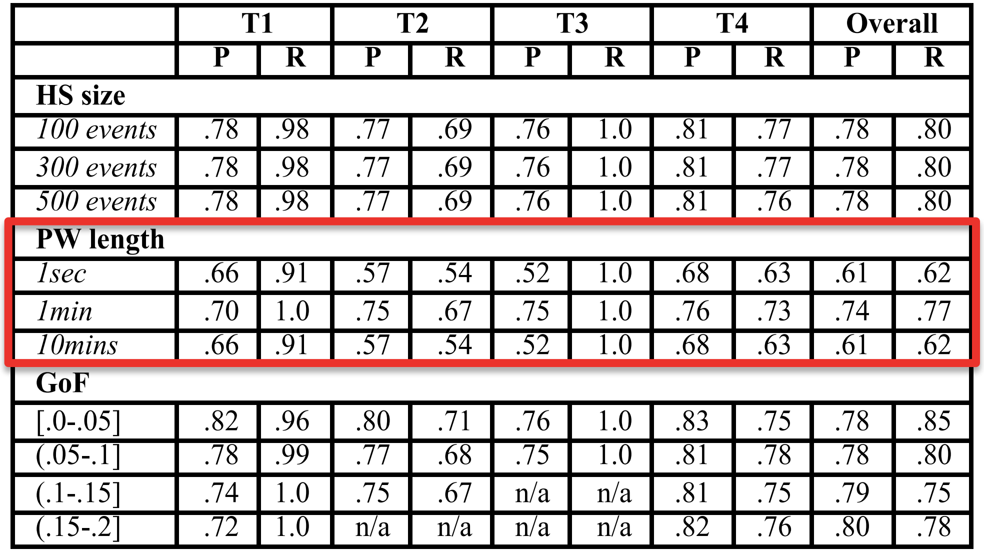


	T1		T2		T3		T4		Overall	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
HS size										
100 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
300 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
500 events	.78	.98	.77	.69	.76	1.0	.81	.76	.78	.80
PW length										
1sec	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
1min	.70	1.0	.75	.67	.75	1.0	.76	.73	.74	.77
10mins	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
GoF										
[.0-.05]	.82	.96	.80	.71	.76	1.0	.83	.75	.78	.85
(.05-.1]	.78	.99	.77	.68	.75	1.0	.81	.78	.78	.80
(.1-.15]	.74	1.0	.75	.67	n/a	n/a	.81	.75	.79	.75
(.15-.2]	.72	1.0	n/a	n/a	n/a	n/a	.82	.76	.80	.78

CITY UNIVERSITY
LONDON

Experimental results

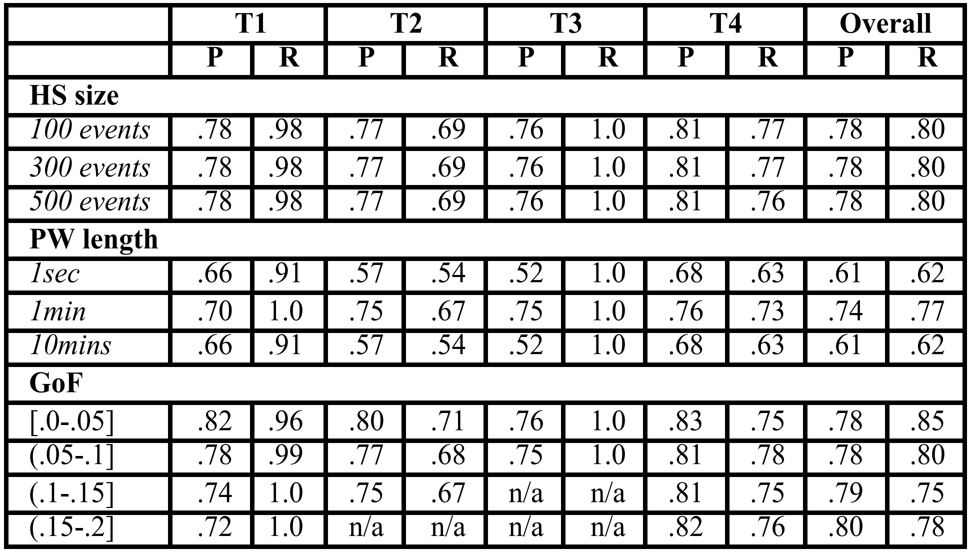
- 4 MTTR Trends:
  - *T1: cyclic*
  - *T2: increasing*
  - *T3: decreasing*
  - *T4: random*
- 3 Variables:
  - History size
  - Prediction window
  - Goodness of fit
- 40 prediction measures



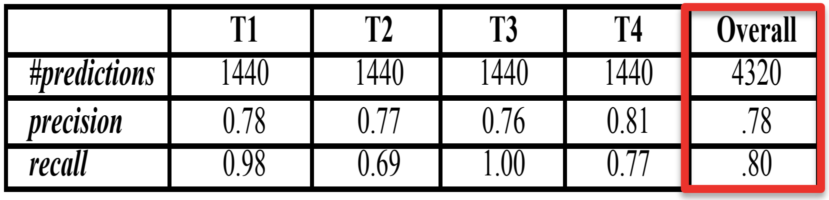
	T1		T2		T3		T4		Overall	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
HS size										
100 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
300 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
500 events	.78	.98	.77	.69	.76	1.0	.81	.76	.78	.80
PW length										
1sec	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
1min	.70	1.0	.75	.67	.75	1.0	.76	.73	.74	.77
10mins	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
GoF										
[.0-.05]	.82	.96	.80	.71	.76	1.0	.83	.75	.78	.85
(.05-.1]	.78	.99	.77	.68	.75	1.0	.81	.78	.78	.80
(.1-.15]	.74	1.0	.75	.67	n/a	n/a	.81	.75	.79	.75
(.15-.2]	.72	1.0	n/a	n/a	n/a	n/a	.82	.76	.80	.78

Experimental results

- 4 MTTR Trends:
  - *T1: cyclic*
  - *T2: increasing*
  - *T3: decreasing*
  - *T4: random*
- 3 Variables:
  - History size
  - Prediction window
  - Goodness of fit
- 40 prediction measures



	T1		T2		T3		T4		Overall	
	P	**R**	**P**	**R**	**P**	**R**	**P**	**R**	**P**	**R**
HS size										
100 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
300 events	.78	.98	.77	.69	.76	1.0	.81	.77	.78	.80
500 events	.78	.98	.77	.69	.76	1.0	.81	.76	.78	.80
PW length										
1sec	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
1min	.70	1.0	.75	.67	.75	1.0	.76	.73	.74	.77
10mins	.66	.91	.57	.54	.52	1.0	.68	.63	.61	.62
GoF										
[.0-.05]	.82	.96	.80	.71	.76	1.0	.83	.75	.78	.85
(.05-.1]	.78	.99	.77	.68	.75	1.0	.81	.78	.78	.80
(.1-.15]	.74	1.0	.75	.67	n/a	n/a	.81	.75	.79	.75
(.15-.2]	.72	1.0	n/a	n/a	n/a	n/a	.82	.76	.80	.78



	T1	T2	T3	T4	Overall
#predictions	1440	1440	1440	1440	4320
precision	0.78	0.77	0.76	0.81	.78
recall	0.98	0.69	1.00	0.77	.80

Conclusions & Future Work

- EVEREST+ is a framework for integrated monitoring and prediction
- EVEREST+ uses prediction specifications to setup both the monitoring and the prediction framework
- EVEREST+ provides the means for developing new predictors
- Testing existing predictors against data coming from different contexts
- Designing and implementation of a wider set of predictors
- Everest+ support for other monitoring frameworks

Bibliography

- Lorenzoli D., Spanoudakis G.: *EVEREST+: Runtime SLA Violations Prediction*, 5th Middleware for Service-oriented Computing Workshop - in conjunction with the 11th ACM/ IFIP/USENIX International Middleware Conference. Bangalore, India, 2010.

- Tsigritis T., Spanoudakis G.: *Diagnosing Runtime Violations of Security & Dependability Properties*, 20th International Conference on Software Engineering and Knowledge Engineering, 2008.

THANK YOU