# Service-Level Agreements for Service-Oriented Computing

Stephen Gilmore
LFCS, Edinburgh

Joint work with Allan Clark and Mirco Tribastone

Bertinoro 2009

Information Society Technologies

# What is a service-level agreement?



- A contract between service provider and client.

# What is a service-level agreement?



- A contract between service provider and client.
- May involve availability:
  - *Service has* $> 99\%$ *availability.*

# What is a service-level agreement?



- A contract between service provider and client.
- May involve availability:
  - *Service has* $>$ 99% *availability.*
- May involve response time:
  - *97% of requests receive a response within 3 seconds.*

# What is a service-level agreement?



- A contract between service provider and client.
- May involve availability:
  - *Service has* $>$ 99% *availability.*
- May involve response time:
  - *97% of requests receive a response within 3 seconds.*
- May be a combination of several statements such as these.

# What is service-oriented computing?



- A modern approach to distributed computing.

# What is service-oriented computing?



- A modern approach to distributed computing.
- Applications are built by composing services.

# What is service-oriented computing?



- A modern approach to distributed computing.
- Applications are built by composing services.
- Services are replicated across a number of servers.

# What is service-oriented computing?



- A modern approach to distributed computing.
- Applications are built by composing services.
- Services are replicated across a number of servers.
- Providers *publish* services in registries.

Information Society Technologies

# What is service-oriented computing?



- A modern approach to distributed computing.
- Applications are built by composing services.
- Services are replicated across a number of servers.
- Providers *publish* services in registries.
- Users *discover* services and *bind* to them.

# Outline



1. Analysing service-oriented computing
2. Stochastic process algebras
   - Continuous-time Markov Chains
   - Transient analysis
   - Passage-time computation
3. Example: Virtual University
   - Analysis results

# Outline

www.sensoria-ist.eu

1. Analysing service-oriented computing
2. Stochastic process algebras
   - Continuous-time Markov Chains
   - Transient analysis
   - Passage-time computation
3. Example: Virtual University
   - Analysis results

Information Society Technologies

# Uncertainties in service-oriented computing



- We do not know which service instances will be used.
- The service instances have different performance characteristics.
- The service instances may have different functionality.
- Plus all of the usual problems of distributed systems...

# Uncertainties in service-oriented computing



- We do not know which service instances will be used.
- The service instances have different performance characteristics.
- The service instances may have different functionality.
- Plus all of the usual problems of distributed systems. . .

. . . lots of difficulties for modellers!

# Analysis of service-oriented computing



- Put all possible descriptions of service behaviours together in one big model.

# Analysis of service-oriented computing



- Put all possible descriptions of service behaviours together in one big model.
  - Hope your favourite large state-space method can cope ...

# Statespace




Information Society Technologies

# Analysis of service-oriented computing



- Separate out service bindings into different cases. Analyse cases separately. Re-combine results.

# Analysis of service-oriented computing



- Separate out service bindings into different cases. Analyse cases separately. Re-combine results.
  - ... Scalable analysis of scalable systems.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
    - Model using a process calculus.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.
  - $r = \{0.1, 0.3, 0.5, 0.7\}$

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.
  - $r = \{0.1, 0.3, 0.5, 0.7\}$
    - Perform parameter sweep across all values.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.
  - $r = \{0.1, 0.3, 0.5, 0.7\}$
    - Perform parameter sweep across all values.
- A way of specifying uncertainty about bindings.

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.
  - $r = \{0.1, 0.3, 0.5, 0.7\}$
    - Perform parameter sweep across all values.
- A way of specifying uncertainty about bindings.
  - $\mathit{Server} = \{\mathit{UEDIN::Server}, \mathit{UNIPI::Server}\}$

# What we need, and what we need to do



- A way of specifying the behaviour of interest.
  - Model using a process calculus.
    - Generate the underlying labelled transition system.
- A way of specifying uncertainty about durations.
  - Use exponential distributions.
    - Numerically evaluate a Markov chain.
- A way of specifying uncertainty about rate parameters.
  - $r = \{0.1, 0.3, 0.5, 0.7\}$
    - Perform parameter sweep across all values.
- A way of specifying uncertainty about bindings.
  - $Server = \{UEDIN::Server, UNIPI::Server\}$
    - Analyse all possible configurations by cases.

# Languages and tools



- SRMC (Sensoria Reference Markovian Calculus)

# Languages and tools



- SRMC (Sensoria Reference Markovian Calculus)
  - Use `srmc` (Sensoria Reference Markovian Compiler) to compile to PEPA

# Languages and tools

- SRMC (Sensoria Reference Markovian Calculus)
  - Use `srmc` (Sensoria Reference Markovian Compiler) to compile to PEPA
- PEPA (Performance Evaluation Process Algebra)

# Languages and tools

- SRMC (Sensoria Reference Markovian Calculus)
  - Use `srmc` (Sensoria Reference Markovian Compiler) to compile to PEPA
- PEPA (Performance Evaluation Process Algebra)
  - Use `ipc` (International PEPA Compiler) to compile to Hydra

# Languages and tools



- SRMC (Sensoria Reference Markovian Calculus)
  - Use `srmc` (Sensoria Reference Markovian Compiler) to compile to PEPA
- PEPA (Performance Evaluation Process Algebra)
  - Use `ipc` (International PEPA Compiler) to compile to Hydra
- Hydra (Markovian response-time analyser)

# Languages and tools



- SRMC (Sensoria Reference Markovian Calculus)
  - Use `srmc` (Sensoria Reference Markovian Compiler) to compile to PEPA
- PEPA (Performance Evaluation Process Algebra)
  - Use `ipc` (International PEPA Compiler) to compile to Hydra
- Hydra (Markovian response-time analyser)
  - Use `hydra` to compute response-time quantiles

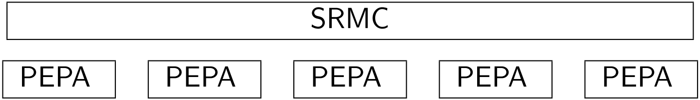
# Evaluation model



| SRMC |
| --- |

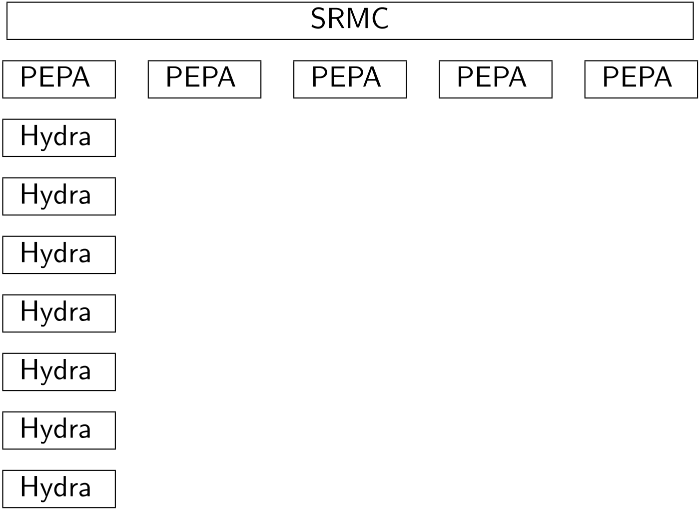
# Evaluation model





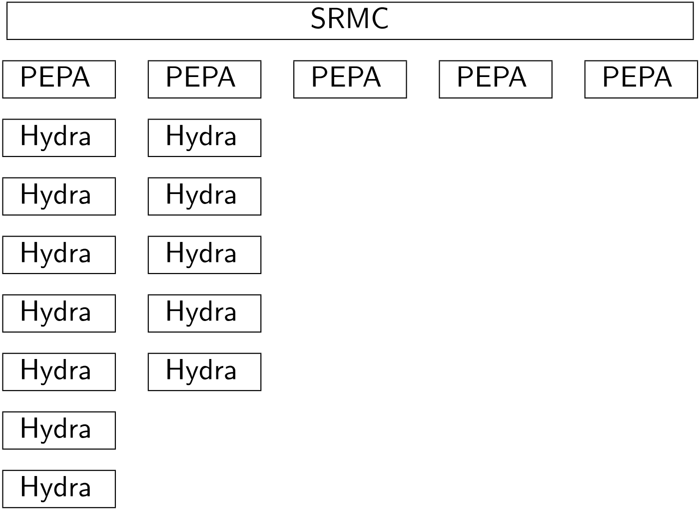
# Evaluation model





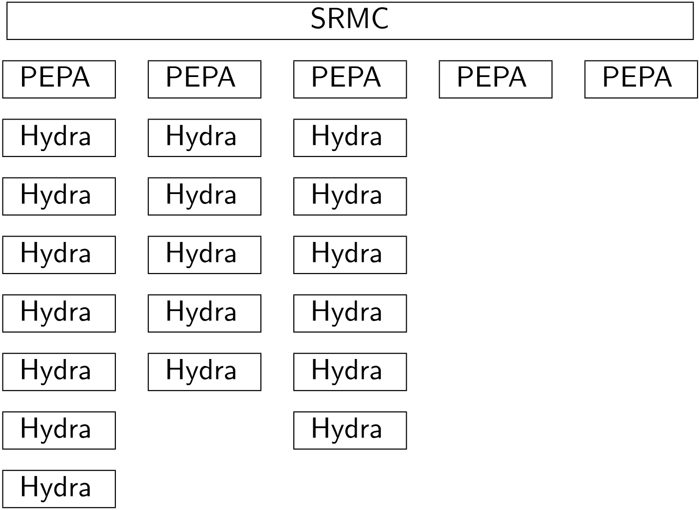
# Evaluation model





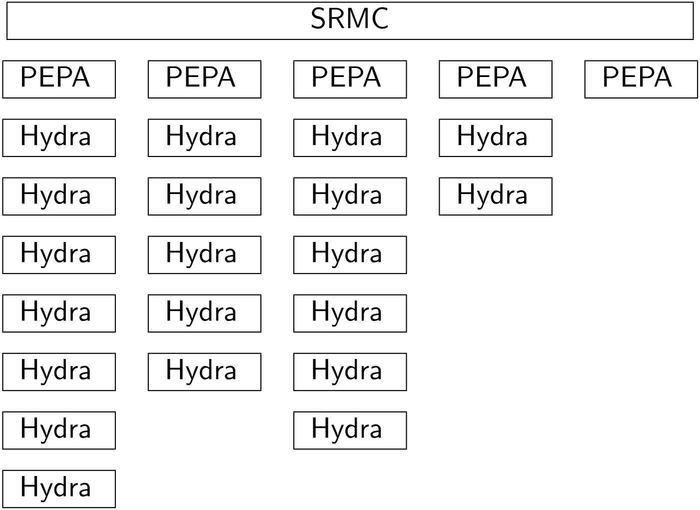
# Evaluation model





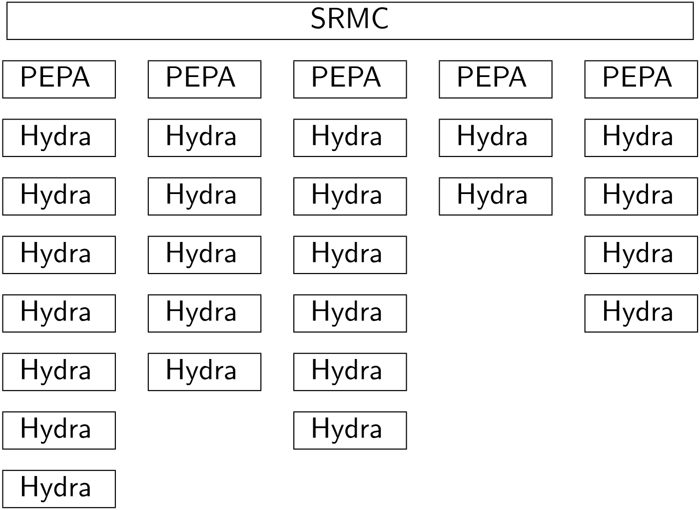
# Evaluation model





# Evaluation model

www.sensoria-ist.eu




Information Society Technologies

# Outline



1. Analysing service-oriented computing
2. Stochastic process algebras
   - Continuous-time Markov Chains
   - Transient analysis
   - Passage-time computation
3. Example: Virtual University
   - Analysis results

# Continuous-Time Markov Chains



We work with a stochastic process algebra (PEPA) which has Continuous-Time Markov Chains (CTMCs) as the underlying mathematical model.

## Markov chains

Markov chains are finite state stochastic processes. The transition system of a Markov chain can be stored as a generator matrix, $Q$, constructed such that when we find a transition from state $i$ to state $j$ at rate $r$ we add $r$ to the current value of $Q_{ij}$.

Information Society Technologies

# Steady-state analysis



Investigation of SLAs may require steady-state analysis of a CTMC.

## Steady-state analysis

We are concerned with finding the state probability row vector $\pi = [\pi_1, \ldots, \pi_n]$ where $\pi_i$ denotes the stationary probability that the CTMC is in state $i$.

# Computing the stationary distribution



The stationary distribution can be computed using procedures of numerical linear algebra.

# Computing the stationary distribution



The stationary distribution can be computed using procedures of numerical linear algebra.

**Global balance equation**

$\pi Q = 0$

# Computing the stationary distribution



The stationary distribution can be computed using procedures of numerical linear algebra.

### Global balance equation

$\pi Q = 0$

### Normalisation condition

$\sum \pi = 1$

# Modelling with quantified process algebras



## Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

This example defines a system with nine reachable states:

1. $\mathtt{P1} \parallel \mathtt{P1}$
2. $\mathtt{P1} \parallel \mathtt{P2}$
3. $\mathtt{P1} \parallel \mathtt{P3}$
4. $\mathtt{P2} \parallel \mathtt{P1}$
5. $\mathtt{P2} \parallel \mathtt{P2}$
6. $\mathtt{P2} \parallel \mathtt{P3}$
7. $\mathtt{P3} \parallel \mathtt{P1}$
8. $\mathtt{P3} \parallel \mathtt{P2}$
9. $\mathtt{P3} \parallel \mathtt{P3}$

The global balance equations and the normalisation condition ensure there is a unique stationary distribution over these states.

## Tiny example

$\texttt{P1} = (\texttt{start}, \texttt{r}).\texttt{P2} \qquad \texttt{P2} = (\texttt{run}, \texttt{r}).\texttt{P3} \qquad \texttt{P3} = (\texttt{stop}, \texttt{r}).\texttt{P1}$
$\texttt{System} = (\texttt{P1} \parallel \texttt{P1})$

The stationary distribution over the nine reachable states is:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1111
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

(Each state has two outgoing transitions with rate $r$ so none of them is more likely than the others.)

# Transient analysis



Investigation of SLAs often requires the transient analysis of a CTMC.

### Transient analysis

We are concerned with finding the transient state probability row vector $\pi(t) = [\pi_1(t), \ldots, \pi_n(t)]$ where $\pi_i(t)$ denotes the probability that the CTMC is in state $i$ at time $t$.

### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \quad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \quad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 0$:

1. 1.0000
2. 0.0000
3. 0.0000
4. 0.0000
5. 0.0000
6. 0.0000
7. 0.0000
8. 0.0000
9. 0.0000

# Analysis based on Continuous-time Markov Chains



## Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 1$:

1. 0.1642
2. 0.1567
3. 0.0842
4. 0.1567
5. 0.1496
6. 0.0804
7. 0.0842
8. 0.0804
9. 0.0432

### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 2$:

1. 0.1056
2. 0.1159
3. 0.1034
4. 0.1159
5. 0.1272
6. 0.1135
7. 0.1034
8. 0.1135
9. 0.1012

# Analysis based on Continuous-time Markov Chains



## Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 3$:

1. 0.1082
2. 0.1106
3. 0.1100
4. 0.1106
5. 0.1132
6. 0.1125
7. 0.1100
8. 0.1125
9. 0.1119

### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$

$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 4$:

1. 0.1106
2. 0.1108
3. 0.1111
4. 0.1108
5. 0.1110
6. 0.1113
7. 0.1111
8. 0.1113
9. 0.1116

# Analysis based on Continuous-time Markov Chains



**Tiny example**

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$
$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 5$:

1. 0.1111
2. 0.1110
3. 0.1111
4. 0.1110
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

# Analysis based on Continuous-time Markov Chains



### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$
$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 6$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1110
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

### Tiny example

$\mathtt{P1} = (\mathtt{start}, \mathtt{r}).\mathtt{P2} \qquad \mathtt{P2} = (\mathtt{run}, \mathtt{r}).\mathtt{P3} \qquad \mathtt{P3} = (\mathtt{stop}, \mathtt{r}).\mathtt{P1}$
$\mathtt{System} = (\mathtt{P1} \parallel \mathtt{P1})$

Using transient analysis we can evaluate the probability of each state with respect to time. For $t = 7$:

1. 0.1111
2. 0.1111
3. 0.1111
4. 0.1111
5. 0.1111
6. 0.1111
7. 0.1111
8. 0.1111
9. 0.1111

# Uniformisation



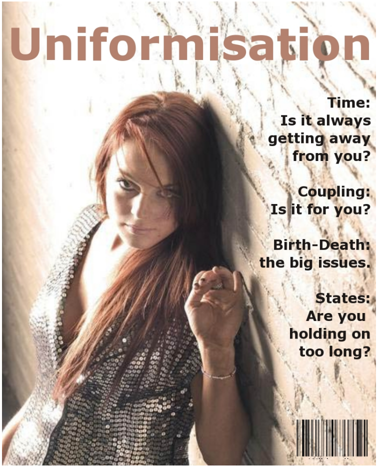
Transient and passage-time analysis of CTMCs proceeds by a numerical procedure called *uniformisation*.

## Uniformisation

The generator matrix, $Q$, is "uniformised" with:

$$P = Q/q + I$$

where $q > \max_i |Q_{ii}|$. This process transforms a CTMC into one in which all states have the same mean holding time $1/q$.




Information Society
Technologies

# Passage-time computation



Passage-time computation is concerned with knowing the probability of reaching a designated target state from a designated source state. It rests on two key sub-computations.

1. Finding the time to complete $n$ hops $(n = 1, 2, 3, \ldots)$, which is an Erlang distribution with parameters $n$ and $q$.
2. Finding the probability that the transition between source and target states occurs in exactly $n$ hops.

# Outline

# A Virtual University



The Sensoria Virtual University (SVU) is a (fictitious) virtual organisation formed by bringing together the resources of the universities at Edinburgh (UEDIN), Munich (LMU), Bologna (UNIBO), Pisa (UNIPI) and others.

The SVU federates the teaching and assessment capabilities of the universities allowing students to enrol in courses irrespective of where they are delivered geographically.

# The Sensoria Virtual University

SENSORIA
www.sensoria-ist.eu


Information Society Technologies

# The Sensoria Virtual University



# Content upload and download



Students download *learning objects* from the content download portals of the universities involved and upload archives of their project work for assessment. By agreement within the SVU, students may download from (or upload to) the portals at any of the SVU sites, not just the one which is geographically closest.

# What we leave out



It is likely that the students make a conscious decision about which portal to bind to. However, we do not anticipate having any data about how the students make their choice so we will not include in our model any representation of the reasoning process leading to selection of one portal or another.

# SRMC description of the UEDIN server



```
UEDIN::{
   lambda = 1.65;  mu = 0.0275;  gamma = 0.125;  delta = 3.215;
   avail = { 0.6, 0.7, 0.8, 0.9, 1.0 };

   UploadPortal::{
      Idle = (upload, avail * lambda).Idle + (fail, mu).Down;
      Down = (repair, gamma).Idle;
   }

   DownloadPortal::{
      Idle = (download, avail * delta).Idle + (fail, mu).Down;
      Down = (repair, gamma).Idle;
   }
}
```

# SRMC description of the LMU server



```
LMU::{
   lambda = 0.965;   delta = 2.576;
   avail = { 0.5, 0.6, 0.7, 0.8, 0.9 };

   UploadPortal::{
      Idle = (upload, avail * lambda).Idle;
   }

   DownloadPortal::{
      Idle = (download, avail * delta).Idle;
   }
}
```

# SRMC description of the UNIBO server



```
UNIBO::{
   lambda = 1.65;  mu = 0.0275;  gamma = 0.125;  delta = 3.215;
   slambda = 1.25; sdelta = 2.255;  avail = { 0.8, 0.9, 1.0 };

   UploadPortal::{
      Idle = (upload, avail * lambda).Idle + (fail, mu).Down
           + (supload, avail * slambda).Idle;
      Down = (repair, gamma).Idle;
   }

   DownloadPortal::{
      Idle = (download, avail * delta).Idle  + (fail, mu).Down
           + (sdownload, avail * sdelta).Idle;
      Down = (repair, gamma).Idle;
   }
}
```

# SRMC description of a cautious client



```
Sally::{
   Idle = (start, 1.0).Download;
   Download = (download, _).(download, _).(download, _).Upload
       + (sdownload, _).(sdownload, _).(sdownload, _).Upload;
   Upload = (upload, _).(upload, _).Disconnect
       + (supload, _).(supload, _).Disconnect;
   Disconnect = (finish, 1.0).Idle;
}
```

# SRMC description of an incautious client



```
Harry::{
   Idle = (start, 1.0).Download;
   Download = (download, _).(download, _).(download, _).Upload;

   Upload = (upload, _).(upload, _).Disconnect;

   Disconnect = (finish, 1.0).Idle;
}
```

# Client, Upload and Download portals



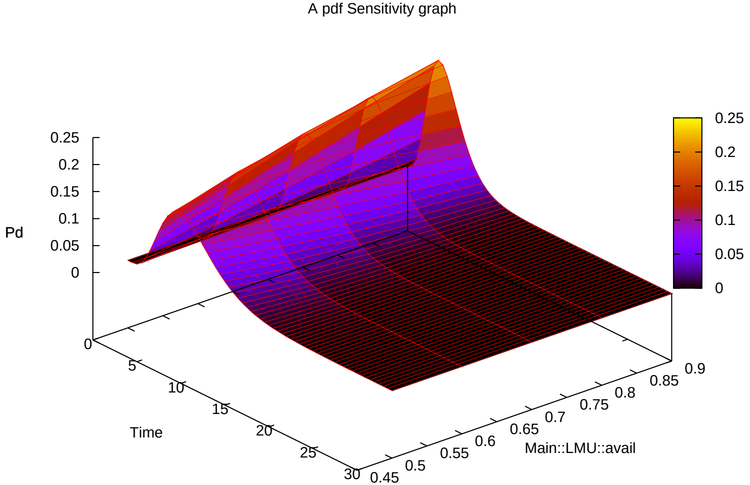
```
SVU::Client =
   { Harry::Idle, Sally::Idle };

SVU::UploadPortal =
   { UEDIN::UploadPortal::Idle, LMU::UploadPortal::Idle,
     UNIBO::UploadPortal::Idle, UNIPI::UploadPortal::Idle };

SVU::DownloadPortal =
   { UEDIN::DownloadPortal::Idle, LMU::DownloadPortal::Idle,
     UNIBO::DownloadPortal::Idle, UNIPI::DownloadPortal::Idle };
```

# Probability Distribution Function [Harry, LMU, LMU]

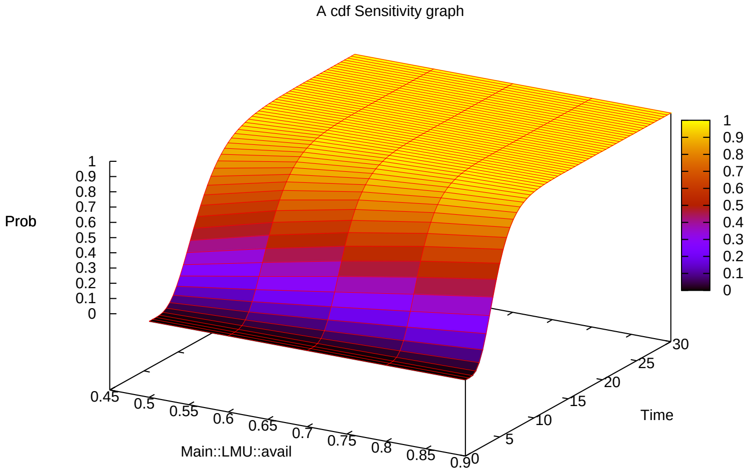





Information Society Technologies

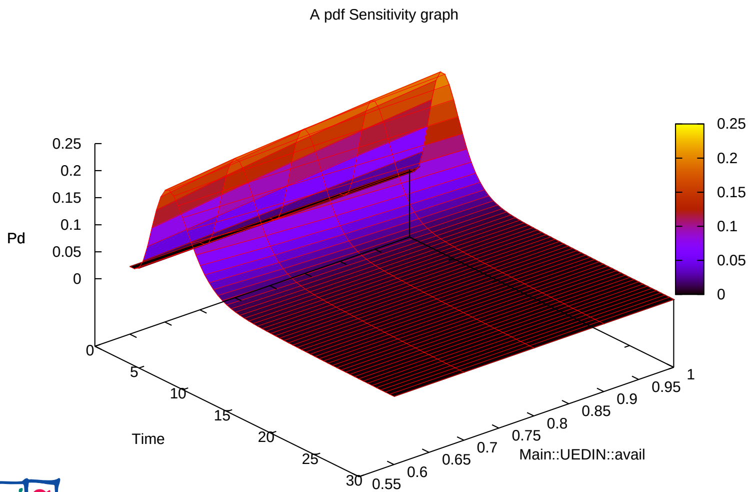
# Cumulative Distribution Function [Harry, LMU, LMU]





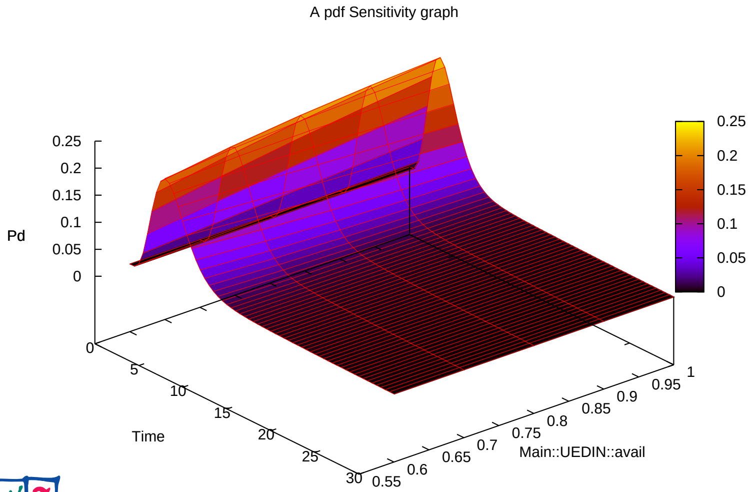
# PDF [Harry, UEDIN, LMU], (LMU::avail=0.5)

www.sensoria-ist.eu



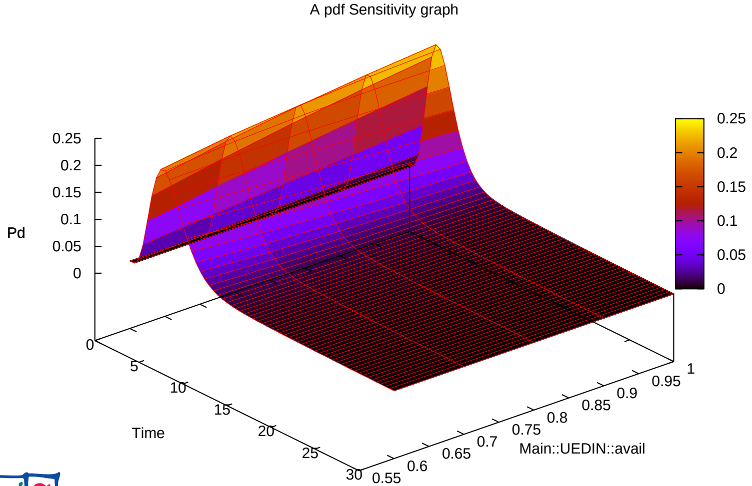

Information Society Technologies

# PDF [Harry, UEDIN, LMU], (LMU::avail=0.6)



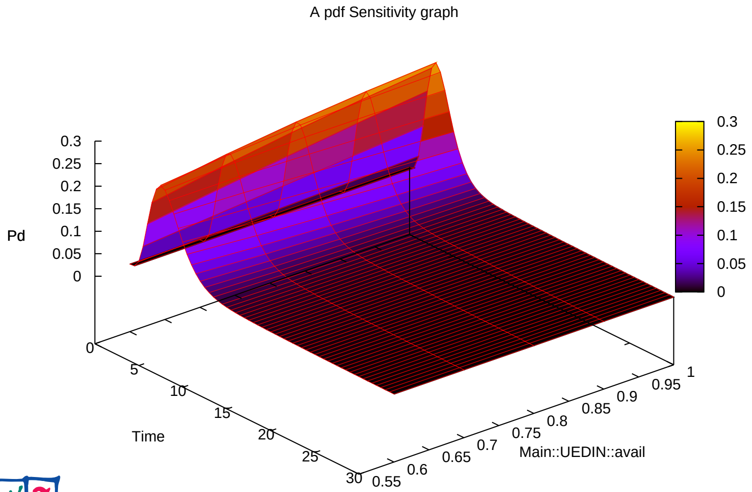
# PDF [Harry, UEDIN, LMU], (LMU::avail=0.7)





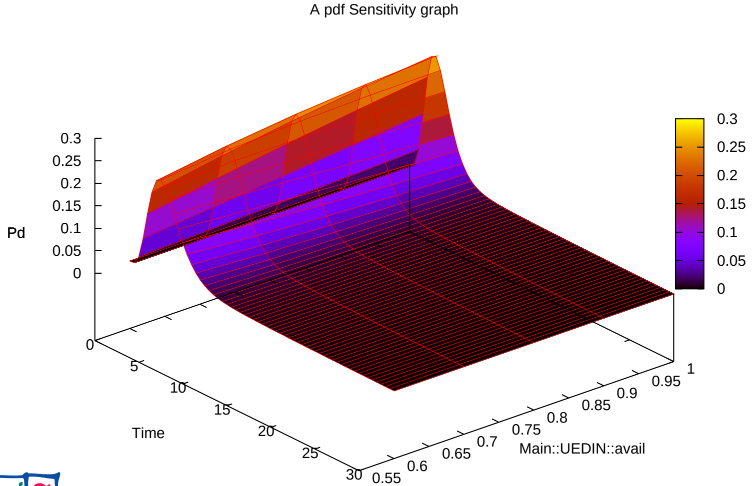
# PDF [Harry, UEDIN, LMU], (LMU::avail=0.8)

www.sensoria-ist.eu

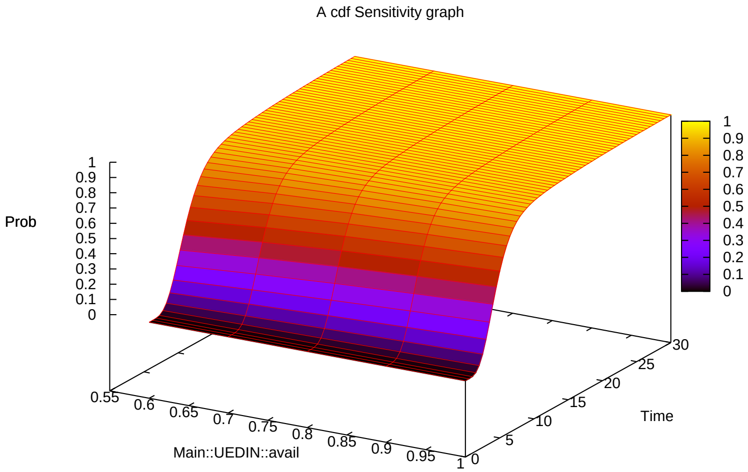



Information Society Technologies

# PDF [Harry, UEDIN, LMU], (LMU::avail=0.9)

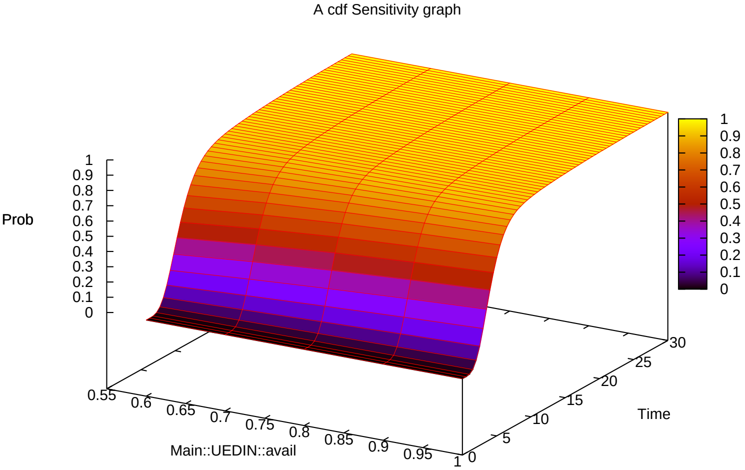


# CDF [Harry, UEDIN, LMU], (LMU::avail=0.5)



# CDF [Harry, UEDIN, LMU], (LMU::avail=0.6)

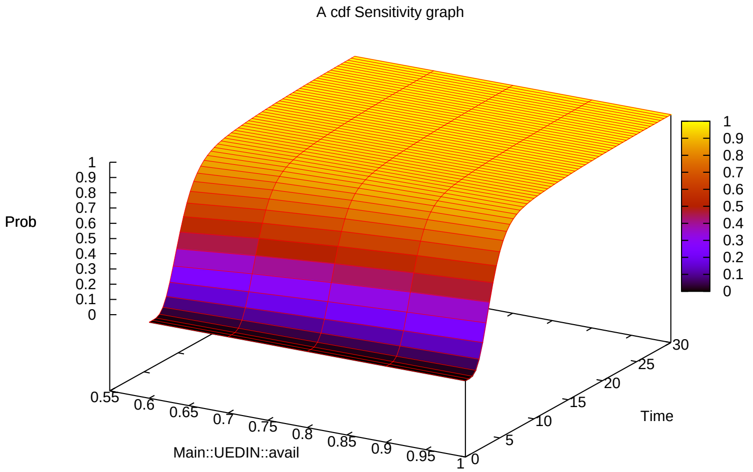





Information Society Technologies

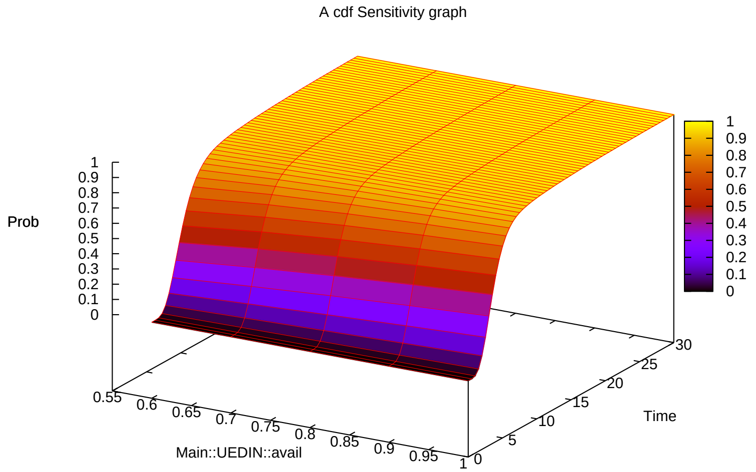
# CDF [Harry, UEDIN, LMU], (LMU::avail=0.7)





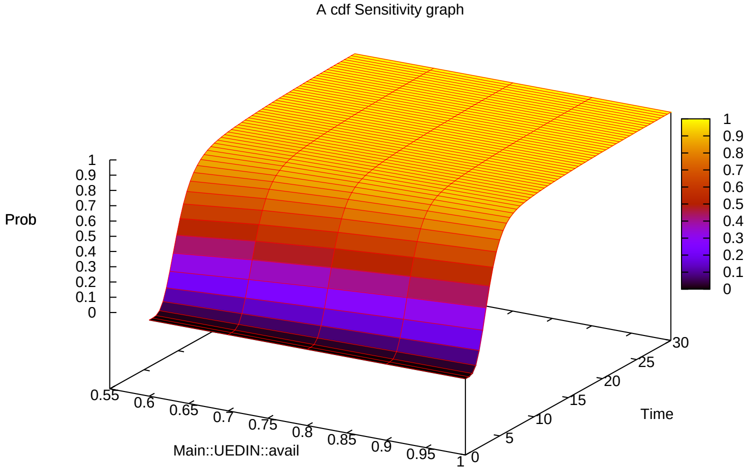
# CDF [Harry, UEDIN, LMU], (LMU::avail=0.8)

www.sensoria-ist.eu




Information Society Technologies

# CDF [Harry, UEDIN, LMU], (LMU::avail=0.9)



# Service-level agreements for SOC



The type of service-level argeement which we would attempt to state for service-oriented computing systems would include a *confidence interval*, a *path* through the system, a *time bound* and lower and upper *probability bounds*.

# Service-level agreements for SOC

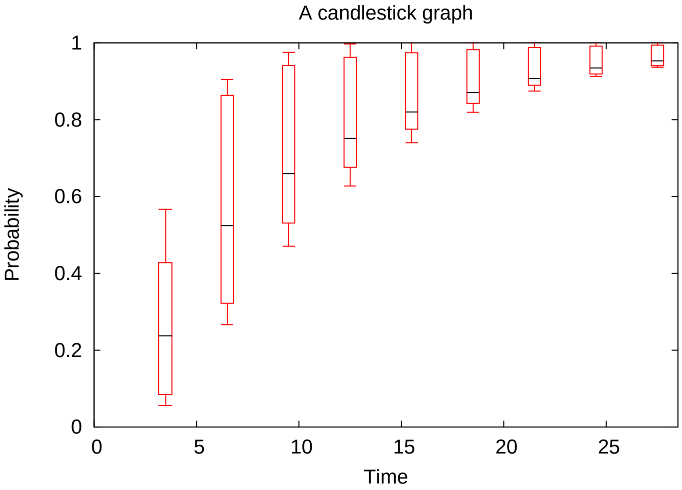


The type of service-level argeement which we would attempt to state for service-oriented computing systems would include a *confidence interval*, a *path* through the system, a *time bound* and lower and upper *probability bounds*.

For example: "Ninety percent of sessions will complete within 29 minutes with probability between 93.9% and 99.3%".

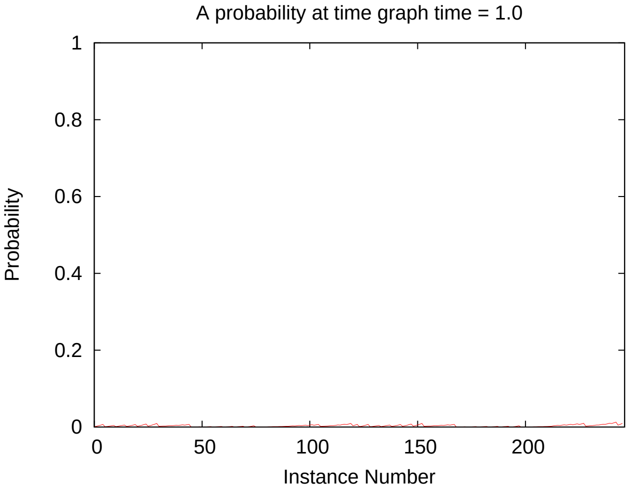
# Candlestick 10% to 90%





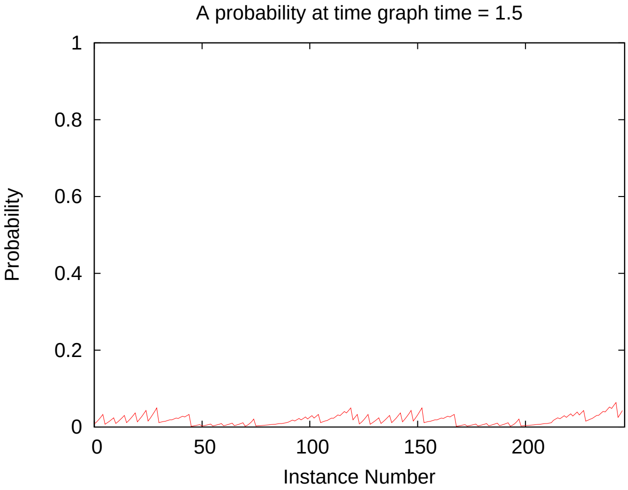
# Probability of completion at $t = 1.0$





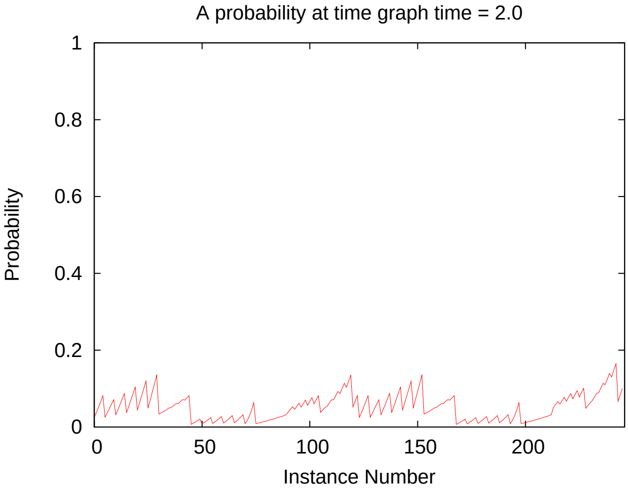
# Probability of completion at $t = 1.5$





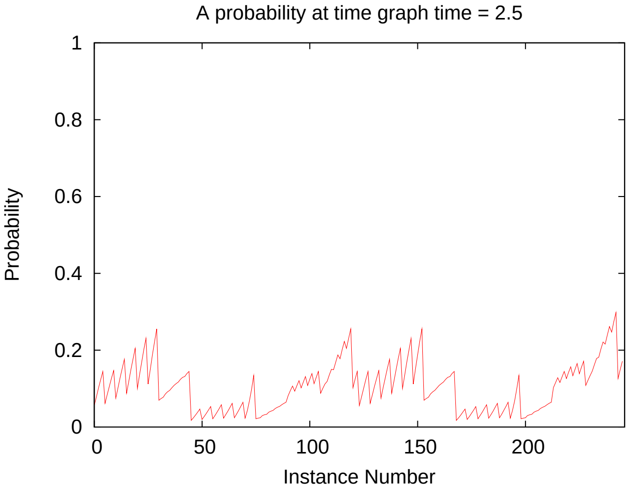
# Probability of completion at $t = 2.0$





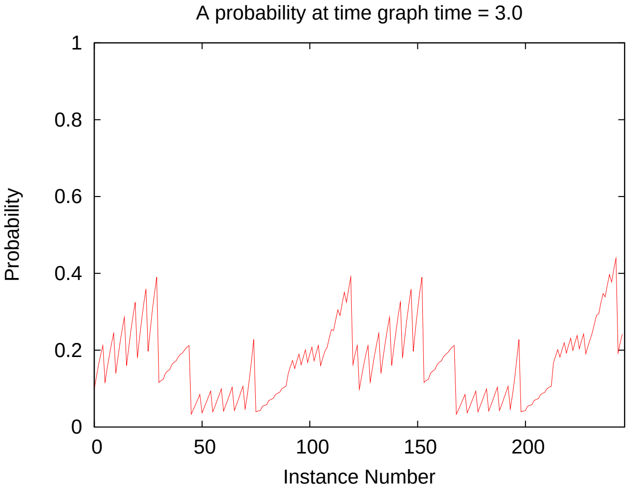
# Probability of completion at $t = 2.5$





# Probability of completion at $t = 3.0$

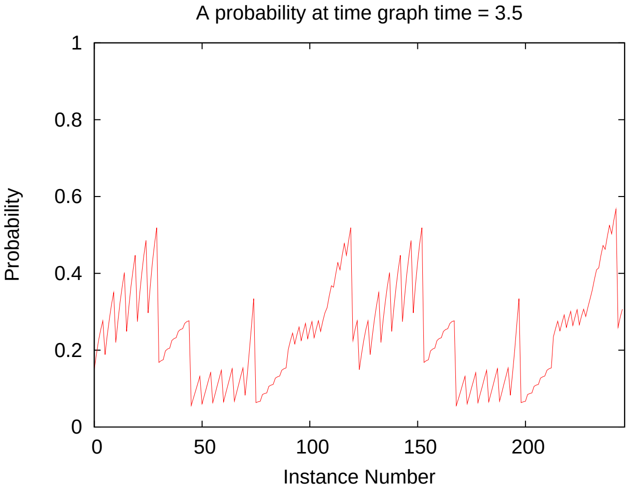
www.sensoria-ist.eu



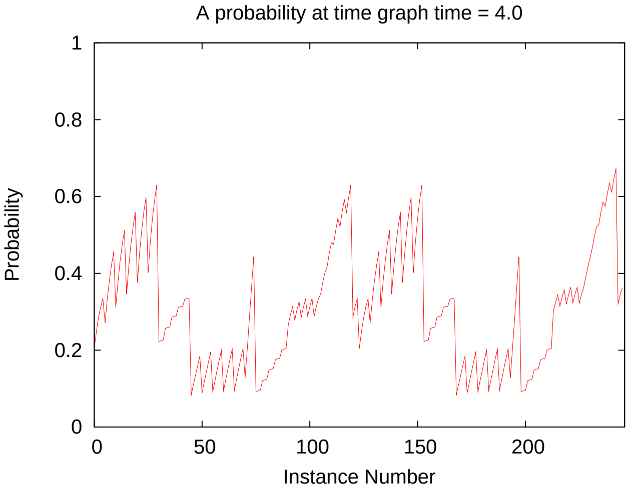

Information Society Technologies

# Probability of completion at $t = 3.5$



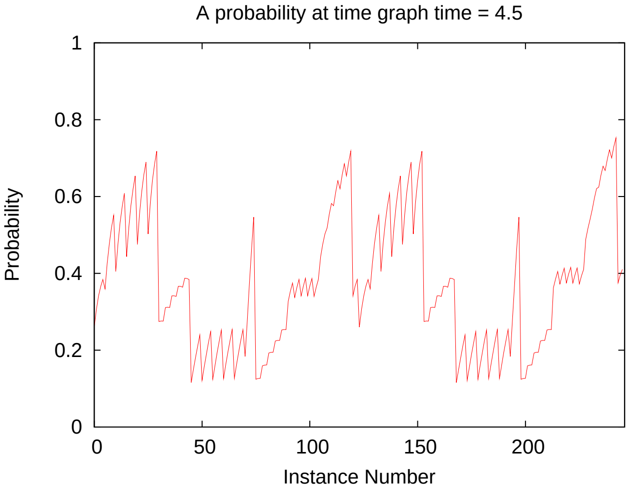


# Probability of completion at $t = 4.0$



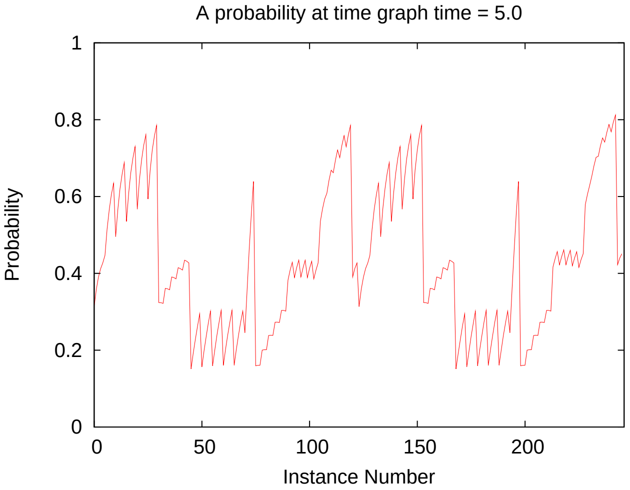
# Probability of completion at $t = 4.5$





# Probability of completion at $t = 5.0$

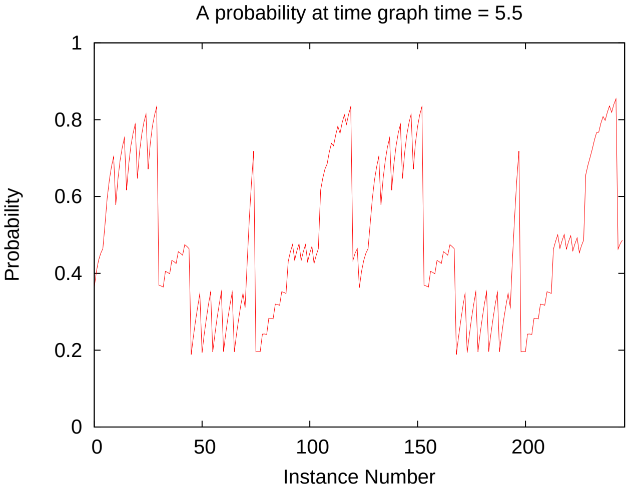
www.sensoria-ist.eu




Information Society Technologies

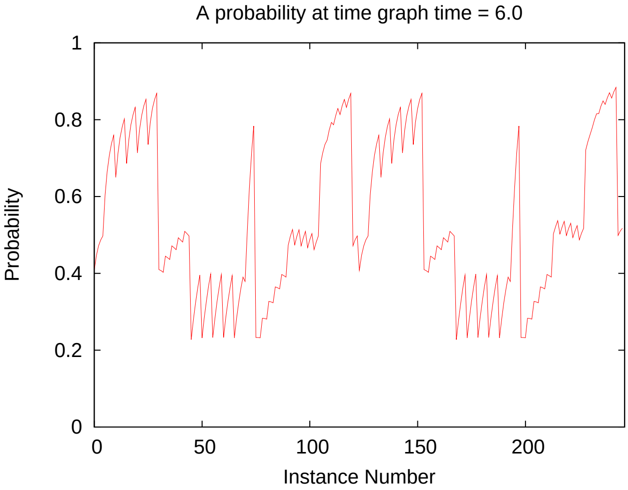
# Probability of completion at $t = 5.5$

www.sensoria-ist.eu



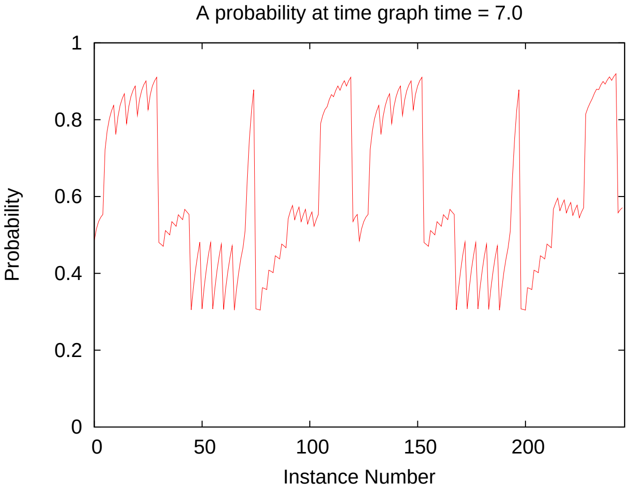

Information Society Technologies

# Probability of completion at $t = 6.0$

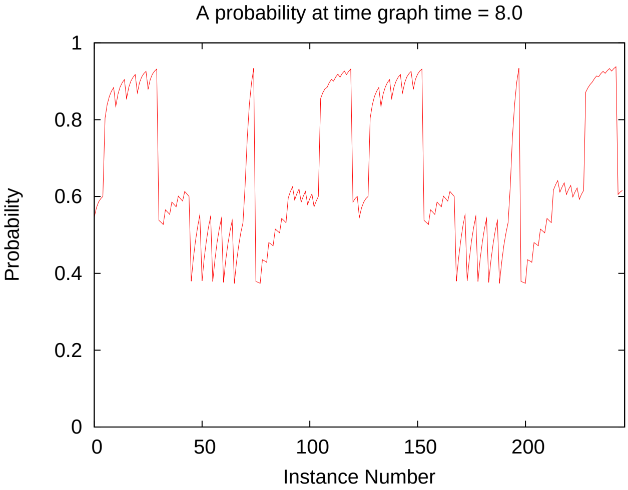


# Probability of completion at $t = 7.0$

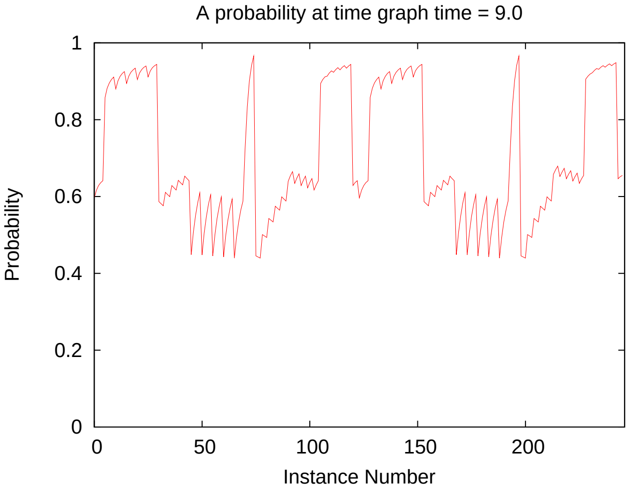




# Probability of completion at $t = 8.0$

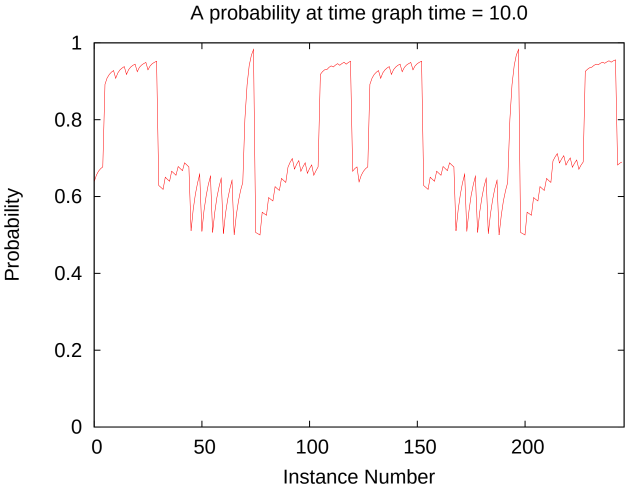


# Probability of completion at $t = 9.0$



# Probability of completion at $t = 10.0$

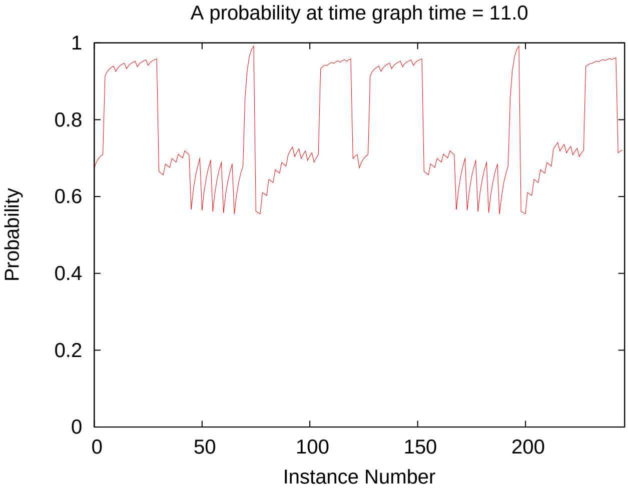
www.sensoria-ist.eu




Information Society Technologies

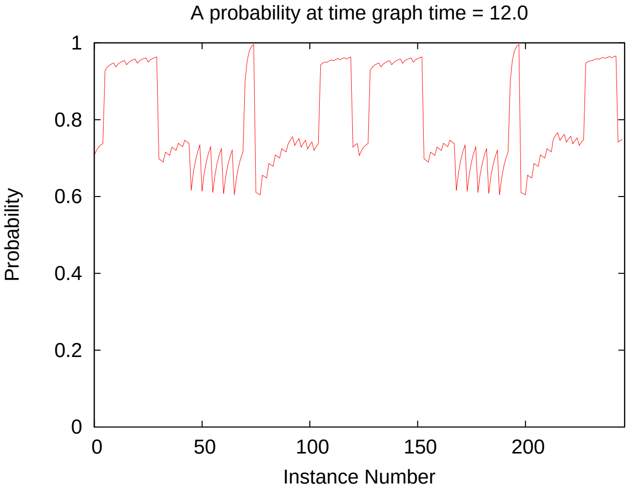
# Probability of completion at $t = 11.0$

www.sensoria-ist.eu

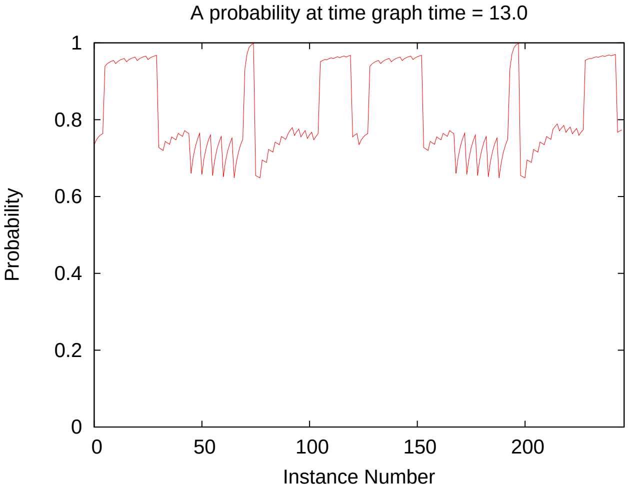



Information Society Technologies

# Probability of completion at $t = 12.0$

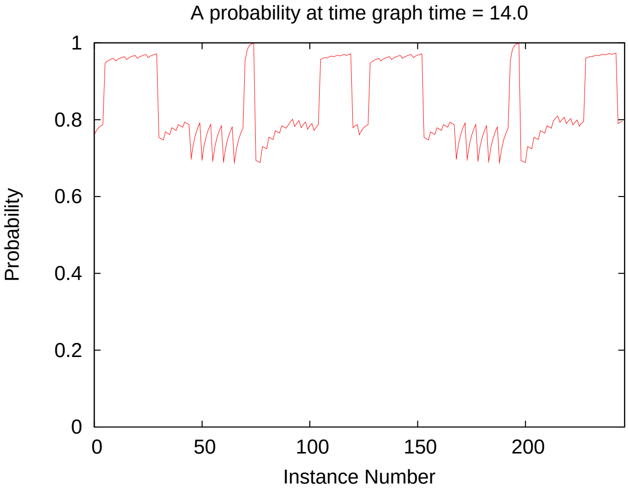


# Probability of completion at $t = 13.0$



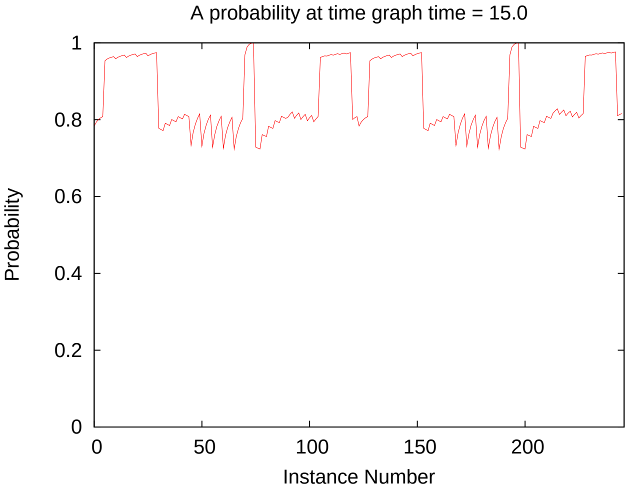
# Probability of completion at $t = 14.0$





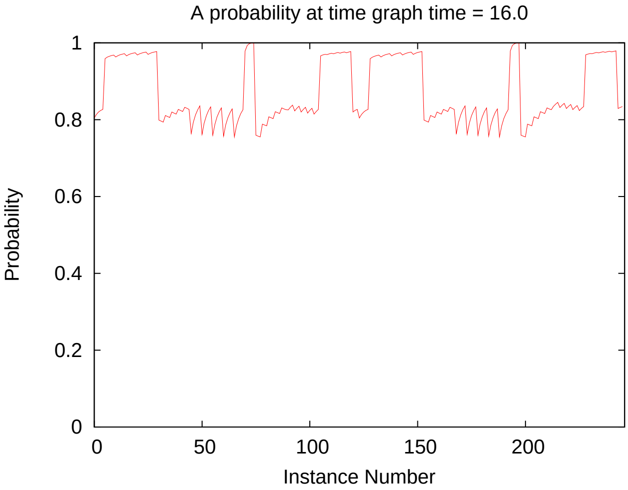

Information Society Technologies

# Probability of completion at $t = 15.0$



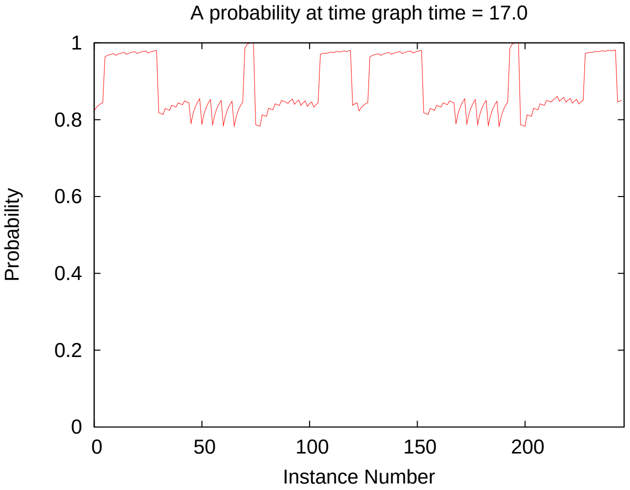
# Probability of completion at $t = 16.0$





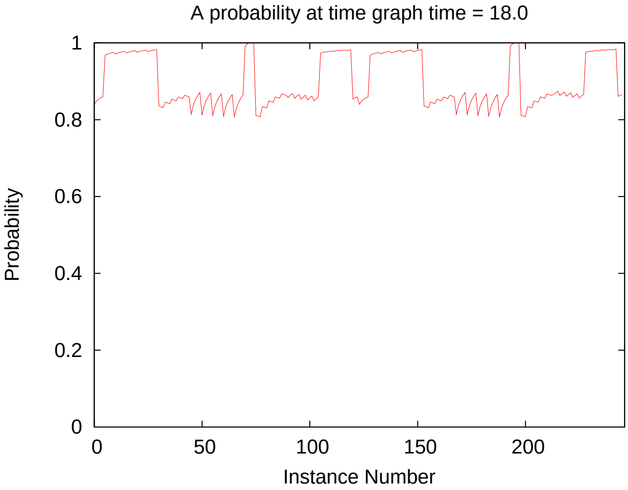

Information Society Technologies

# Probability of completion at $t = 17.0$



# Probability of completion at $t = 18.0$

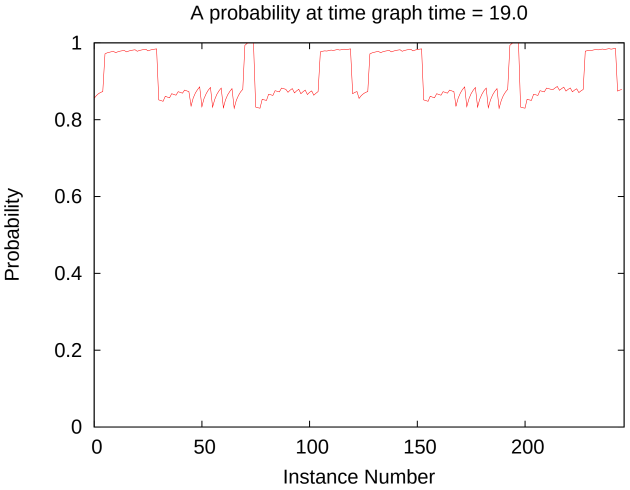
www.sensoria-ist.eu




Information Society Technologies

# Probability of completion at $t = 19.0$

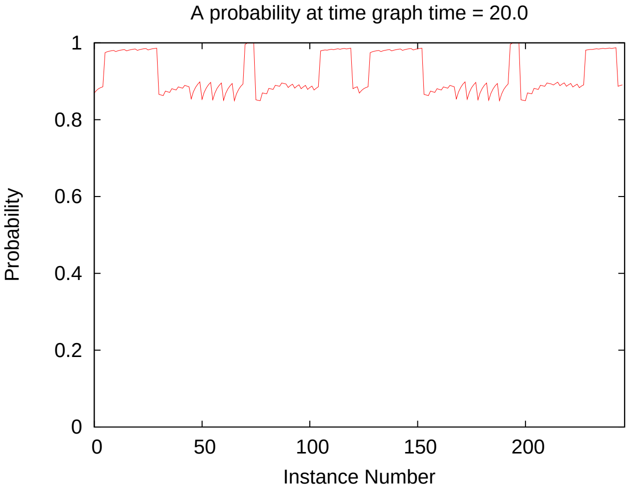
SENSORIA
www.sensoria-ist.eu




Information Society Technologies

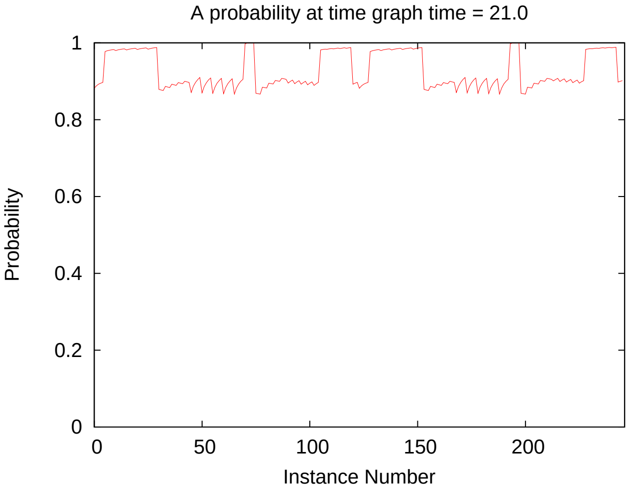
# Probability of completion at $t = 20.0$





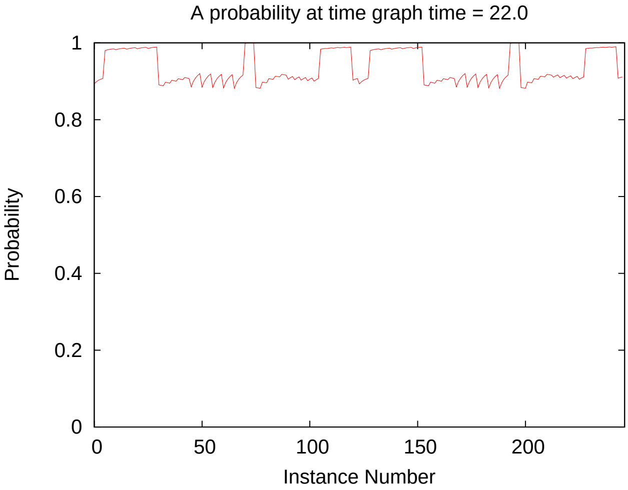
# Probability of completion at $t = 21.0$





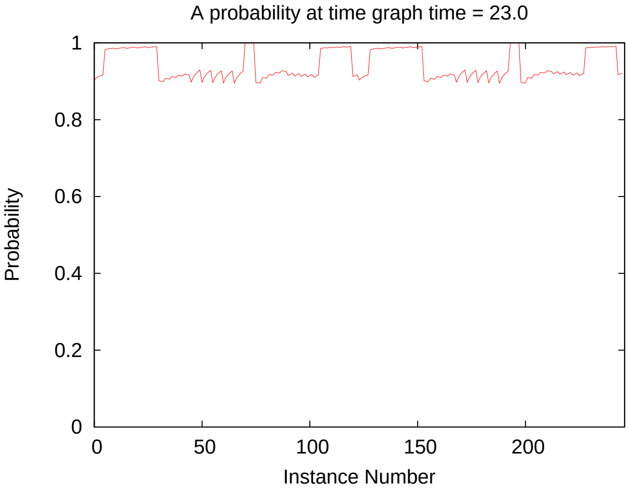
# Probability of completion at $t = 22.0$





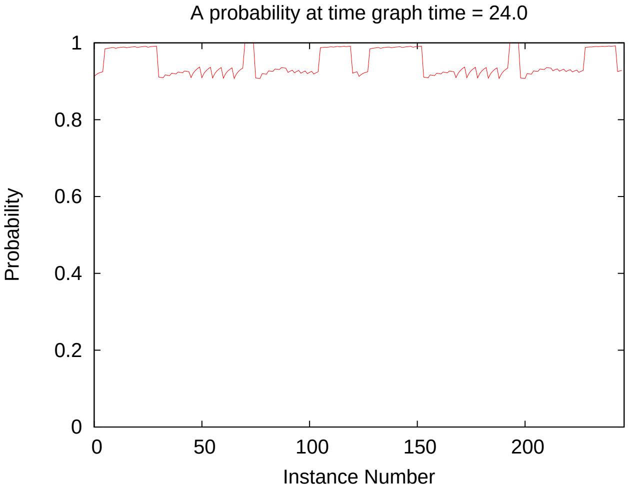
# Probability of completion at $t = 23.0$





# Probability of completion at $t = 24.0$

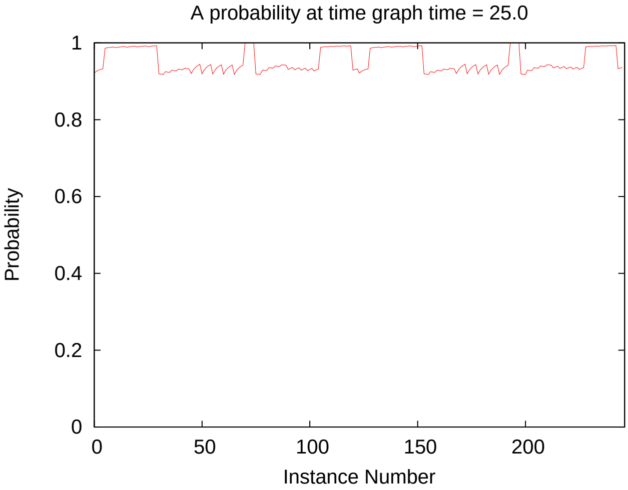





Information Society Technologies

# Probability of completion at $t = 25.0$

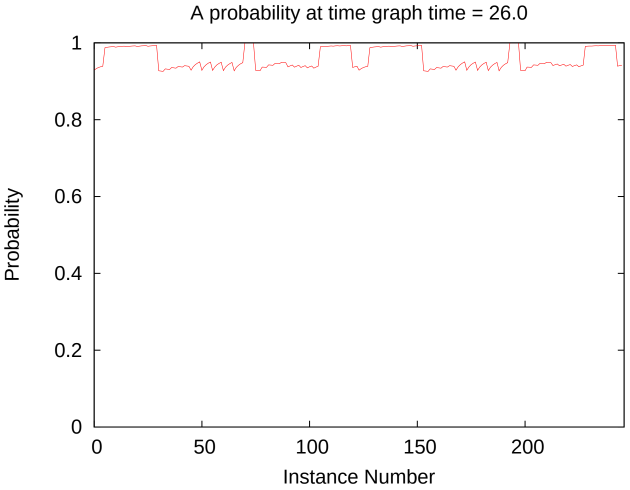
www.sensoria-ist.eu




Information Society Technologies

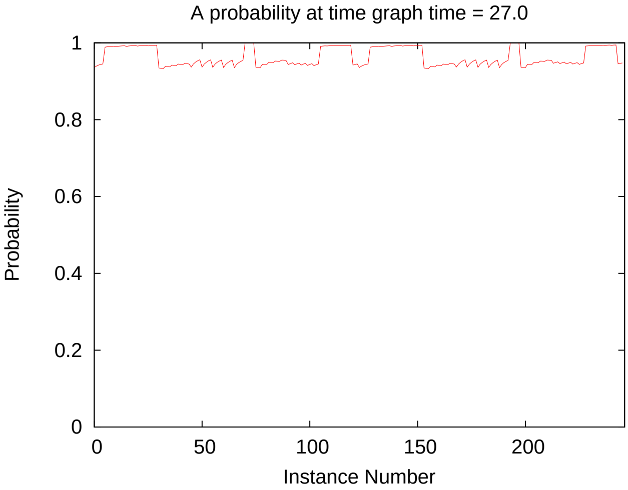
# Probability of completion at $t = 26.0$





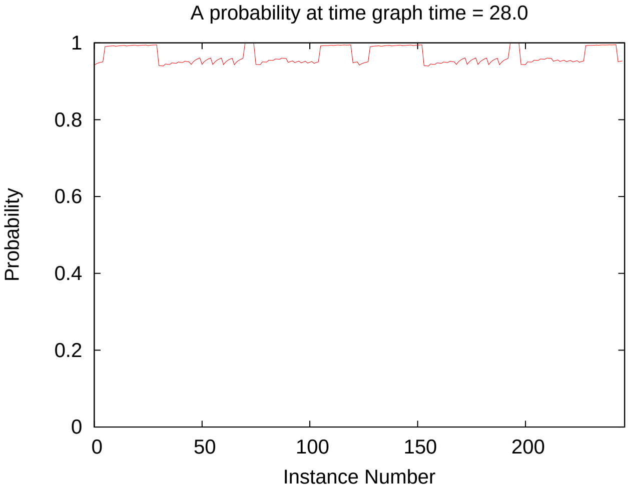
# Probability of completion at $t = 27.0$



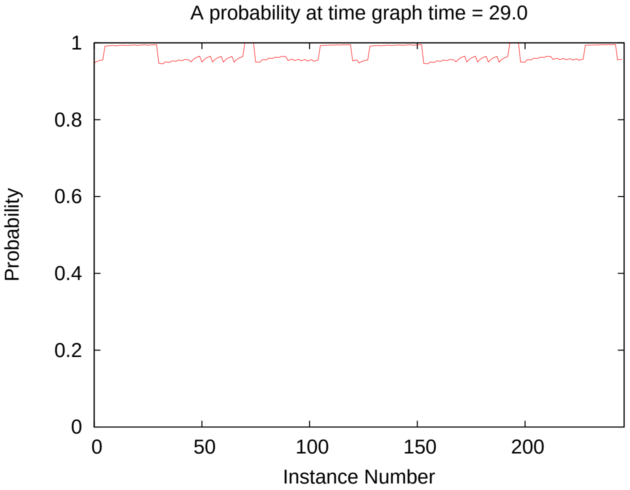


# Probability of completion at $t = 28.0$



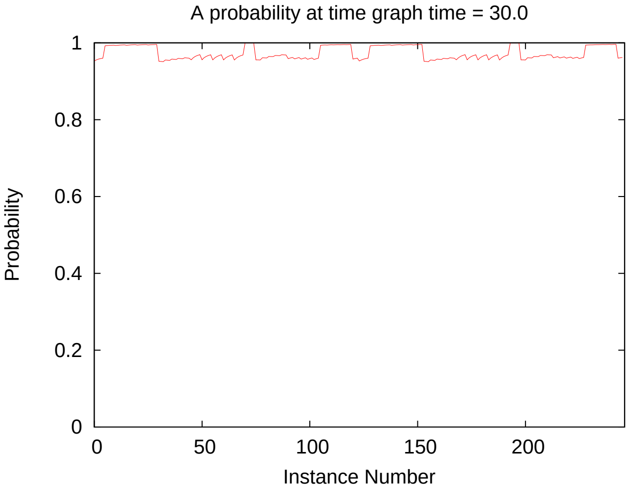


# Probability of completion at $t = 29.0$



# Probability of completion at $t = 30.0$





# Discussion



- We might wonder if our infrastructure is necessary. Do we need to generate separate configurations and recombine the results?
- To test this we compared our SRMC model against a plain CTMC model of the system generated from a Generalised Stochastic Petri Net with vanishing markings.

# Experiments with increasing model size



| # | Client | Upload Portal | Download Portal |
|---|---|---|---|
| 1 | {Harry} | {UEDIN} | {UEDIN} |
| 2 | {Harry} | {UEDIN, LMU} | {UEDIN} |
| 3 | {Harry} | {UEDIN, LMU} | {UEDIN, LMU} |
| 4 | {Harry} | {UEDIN, LMU, UNIBO} | {UEDIN, LMU} |
| 5 | {Harry} | {UEDIN, LMU, UNIBO} | {UEDIN, LMU, UNIBO} |
| 6 | {Harry} | {UEDIN, LMU, UNIBO, UNIPI} | {UEDIN, LMU, UNIBO} |
| 7 | {Harry} | {UEDIN, LMU, UNIBO, UNIPI} | {UEDIN, LMU, UNIBO, UNIPI} |
| 8 | {Harry, Sally} | {UEDIN, LMU, UNIBO, UNIPI} | {UEDIN, LMU, UNIBO, UNIPI} |

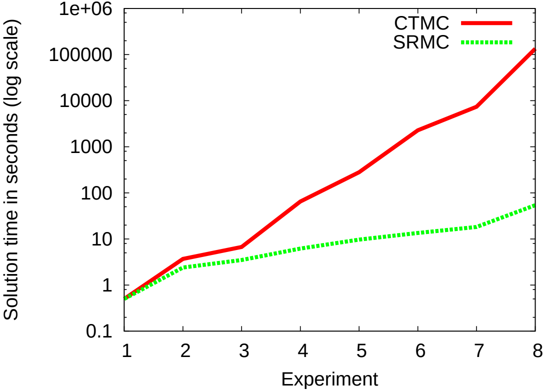
# Experiments with increasing model size



| | CTMC | | | | SRMC | | | |
|---|---|---|---|---|---|---|---|---|
| # | Num. states | Num. config | Num. runs | time (secs) | Num. states | Num. config | Num. runs | time (secs) |
| 1 | 32 | 1 | 5 | 0.5 | 32 | 1 | 5 | 0.5 |
| 2 | 48 | 1 | 25 | 3.7 | 32 | 2 | 30 | 2.4 |
| 3 | 72 | 1 | 25 | 6.7 | 32 | 4 | 60 | 3.5 |
| 4 | 120 | 1 | 75 | 65.0 | 32 | 6 | 90 | 6.2 |
| 5 | 200 | 1 | 75 | 280.0 | 32 | 9 | 123 | 9.7 |
| 6 | 280 | 1 | 225 | 2280.0 | 32 | 12 | 162 | 13.5 |
| 7 | 392 | 1 | 225 | 7390.0 | 32 | 16 | 204 | 18.2 |
| 8 | 784 | 1 | 225 | $\sim$ 134100.0 | 32 | 32 | 408 | 54.0 |

# Experiments with increasing model size

www.sensoria-ist.eu




Information Society Technologies

# Conclusions



- We addressed the inherent uncertainty in service-oriented computing by analysing by cases. We perform parameter sweep for each case. We can evaluate these in parallel (using Condor).
- The analysis methods scale well with increasing problem size.
- We build on tried and trusted compilers and analysers.
- Hopefully a “real world” approach to a “real world” problem.

# Thank you!


Statespace
Vanishing states: declutter your life
Are BDDs making you fat?
Modelling detail: Do you love it too much?
Deadlock detection: Are you in a dead-end relationship?
Uniformisation
Time: Is it always getting away from you?
Coupling: Is it for you?
Birth-Death: the big issues.
States: Are you holding on too long?
Statespace
Vanishing states: declutter your life
Are BDDs making you fat?
Modelling detail: Do you love it too much?
Deadlock detection: Are you in a dead-end relationship?

# Conclusions

SENSORIA
www.sensoria-ist.eu

- We addressed the inherent uncertainty in service-oriented computing by analysing by cases. We perform parameter sweep for each case. We can evaluate these in parallel (using Condor).
- The analysis methods scale well with increasing problem size.
- We build on tried and trusted compilers and analysers.
- Hopefully a “real world” approach to a “real world” problem.

Information Society Technologies