Lecture 15: Outline

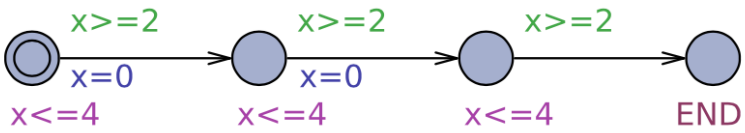
- Statistical model checking (reminder)
- A case study with Uppaal-SMC
- Course wrap-up. Exam information

The Uppaal SMC tool (reminder)

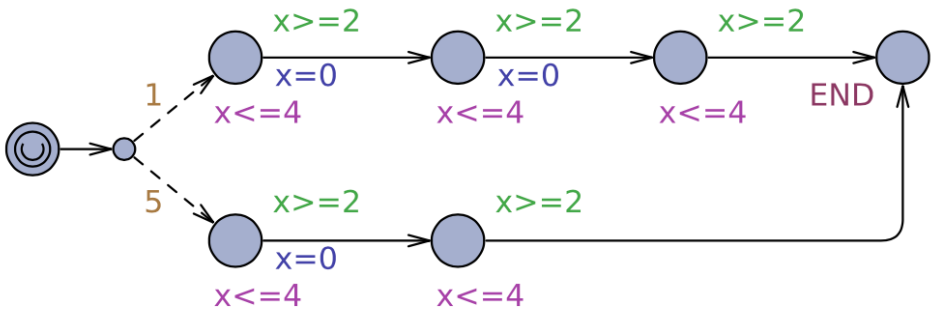
- Recently, the Uppaal tool was extended to support statistical model checking (SMC), where properties are checked with some statistical confidence (probabilities)
- The Uppaal language for modelling real-time systems has also been extended with probabilistic transition branching and clock rates (probabilistic distribution)
- The exhaustive model checking of the system state space is replaced with simulating the system behaviour for some number of "system runs" and accumulating the statistical results
- Timed automata are thus augmented to become stochastic timed automata

The Uppaal SMC tool (cont.)

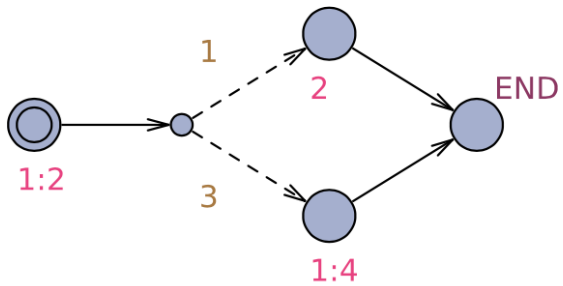
The language is extended with probabilistic transition branching (in 2nd and 3rd diagrams) and clock rates (in 3rd diagram)



(a) A_1.

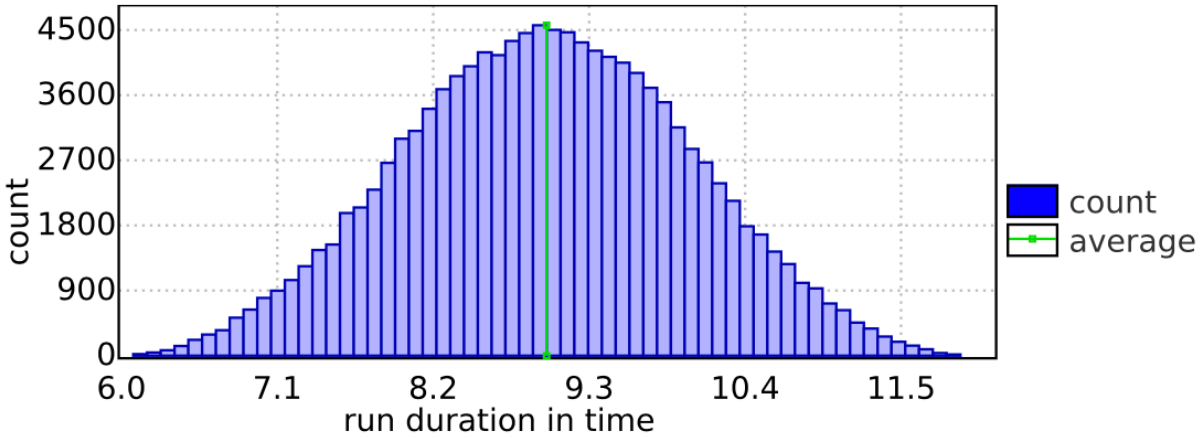


(b) A_2.

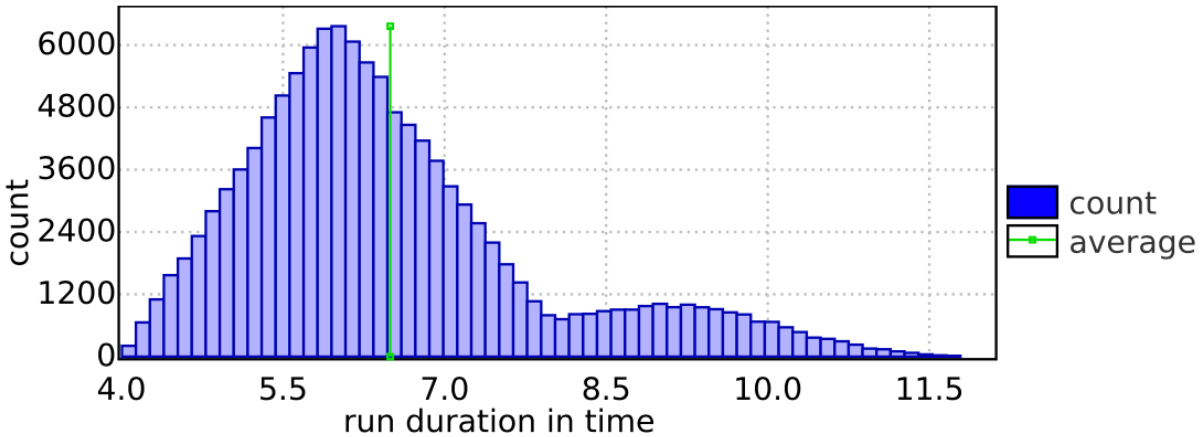


The Uppaal SMC tool (cont.)

Visualisation of distribution of reachability time



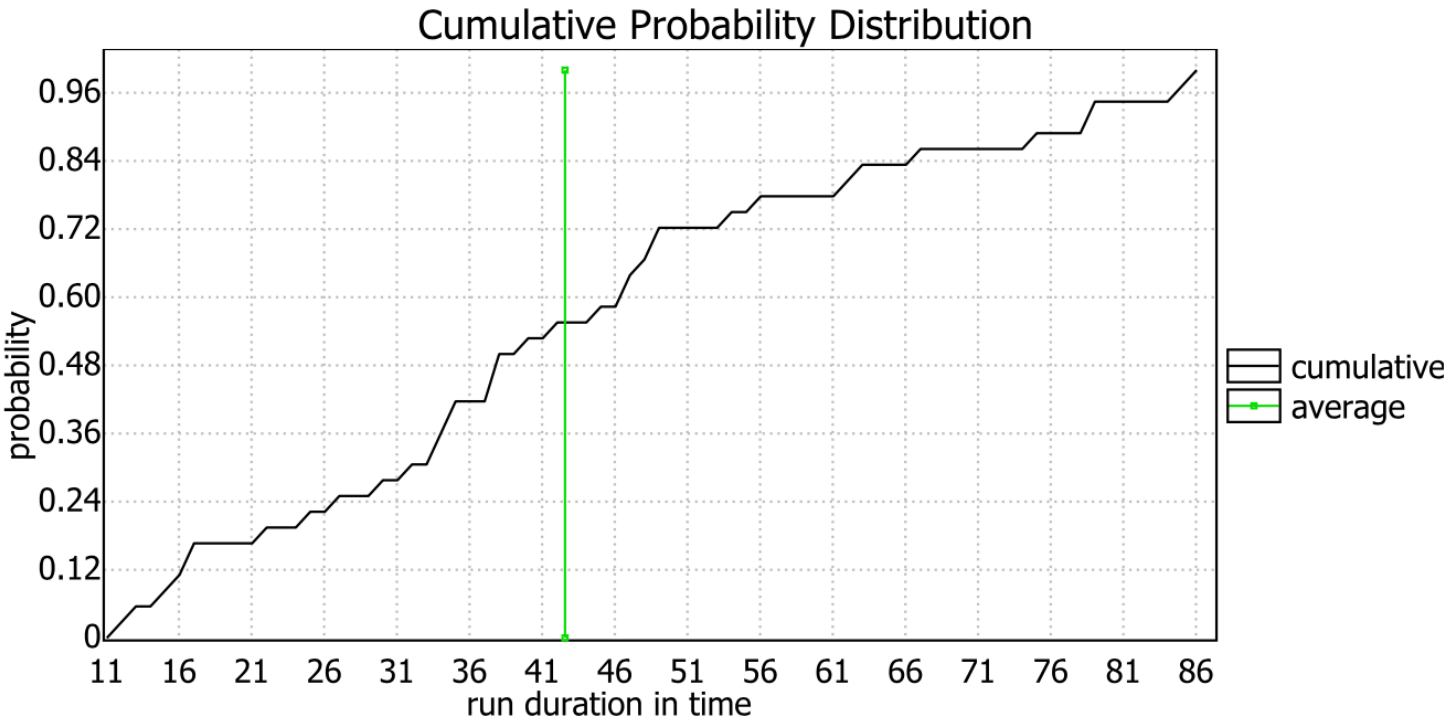
(a) A_1 arrival to END.



(b) A_2 arrival to END.

The Uppaal SMC tool (cont.)

The cumulative probability distribution of reaching a desired state within the given time



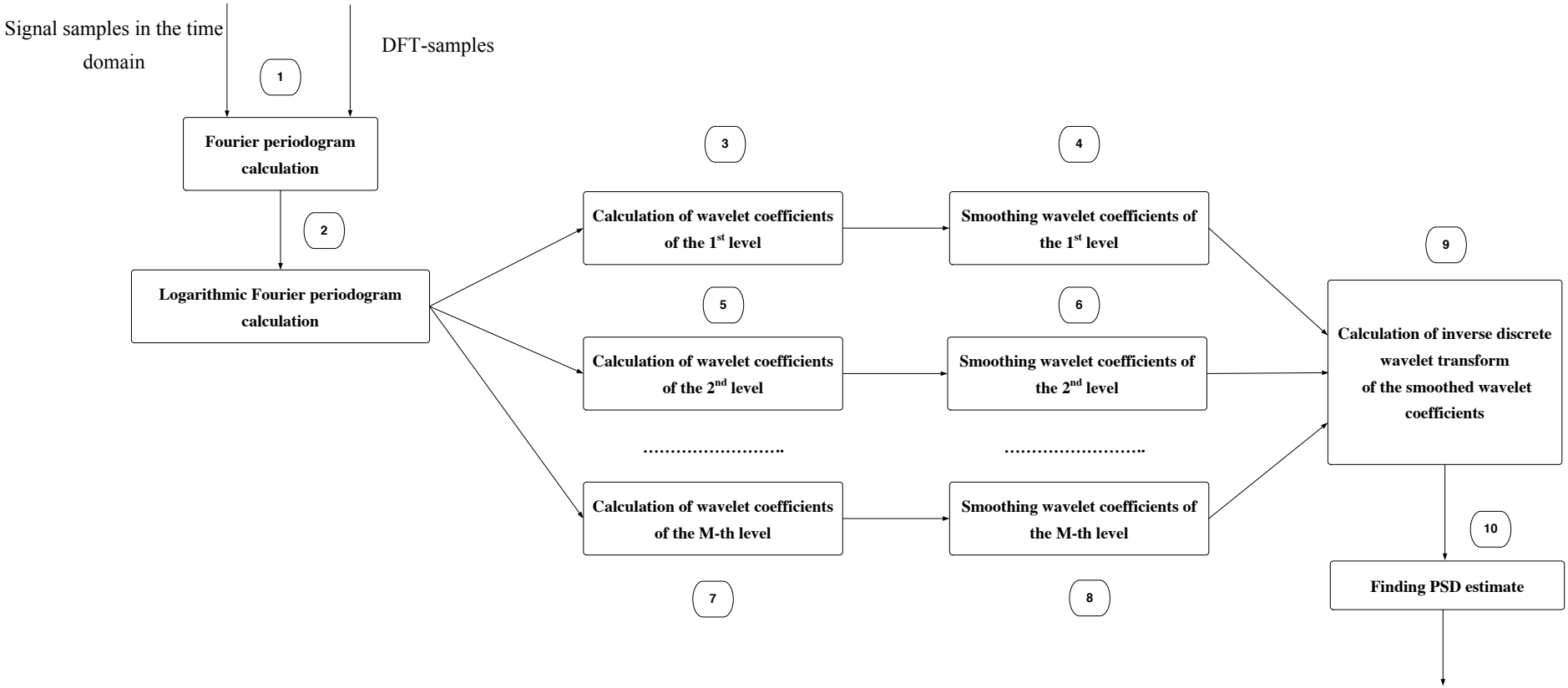
Parameters: α=0.05, ε=0.05, bucket width=1, bucket count=75
Runs: 36 in total, 36 (100%) displayed, 0 (0%) remaining
Span of displayed sample: [11.0447, 85.2899]
Mean of displayed sample: 42.5735 ± 7.10607 (95% CI)

Another case study: signal processing in a distributed system

- Signal processing of acoustic signals in a dynamic distributed system with a number of data processing workers (servers), based on the pre-defined calculation algorithm
- Signal preprocessing from multi-channel observations (recorded sea sounds from reception hydrophones), filtering noises and other interferences
- Purpose of the project: to formally develop and quantitatively evaluate software architectures that most suitable for effective application of the proposed signal processing algorithm
- Again, both Event-B (the Rodin platform) and Uppaal (statistical version this time) were used for that

Signal processing

Algorithm structure:



Signal processing: system architecture

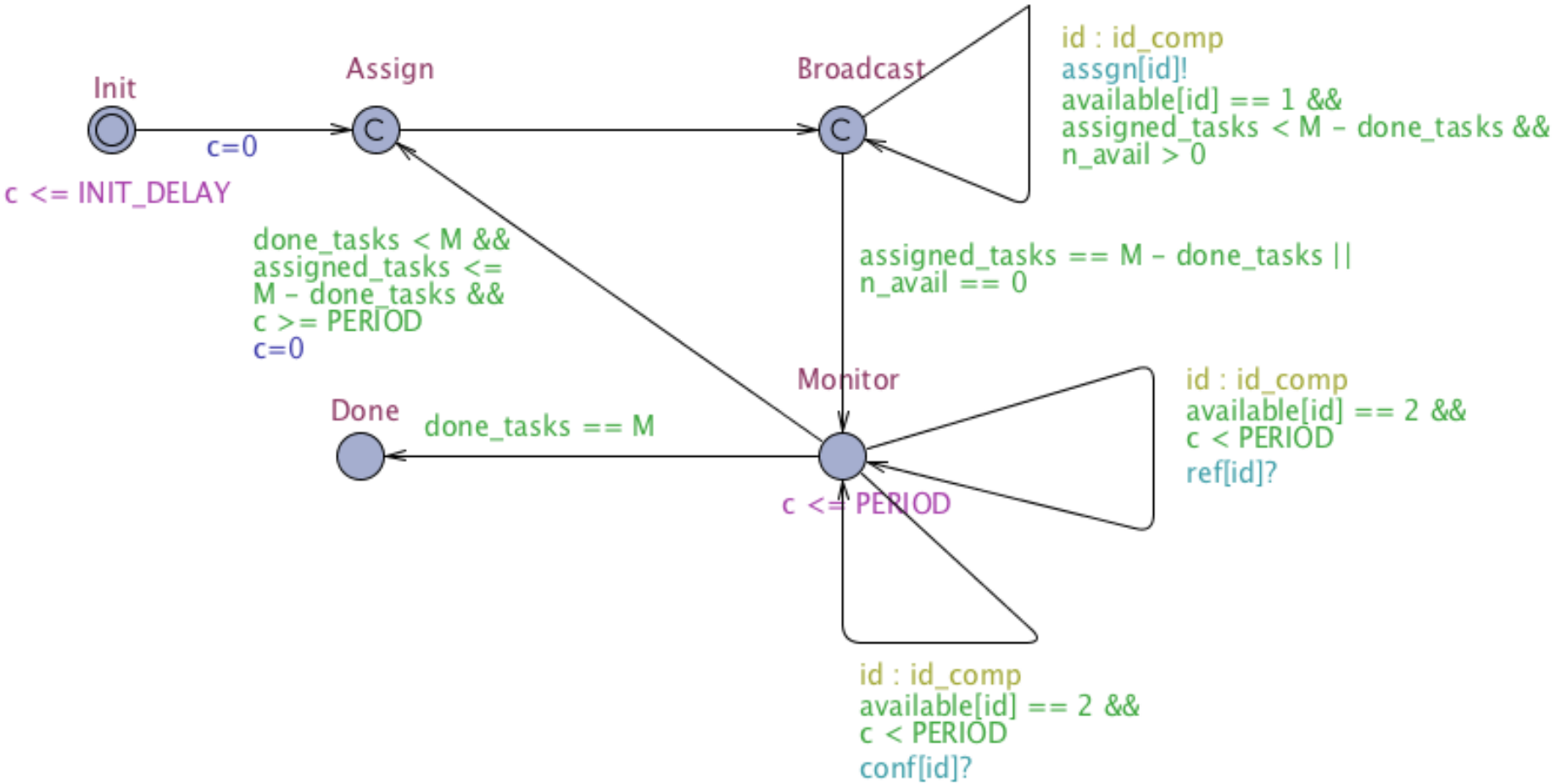
- One master component coordinating the overall execution of the algorithm
- Many worker components that are capable of executing specific calculations assigned by the master component
- Some algorithm phases involve (if possible) parallel computations by several workers. The optimal number of parallel computations that a specific phase can be split into is known beforehand
- The availability of workers is dynamic (i.e., some components can be busy/unavailable, they also can dynamically fail or recover)

Signal processing: system parameters

System parameters as Uppaal textual declarations (in a separate file):

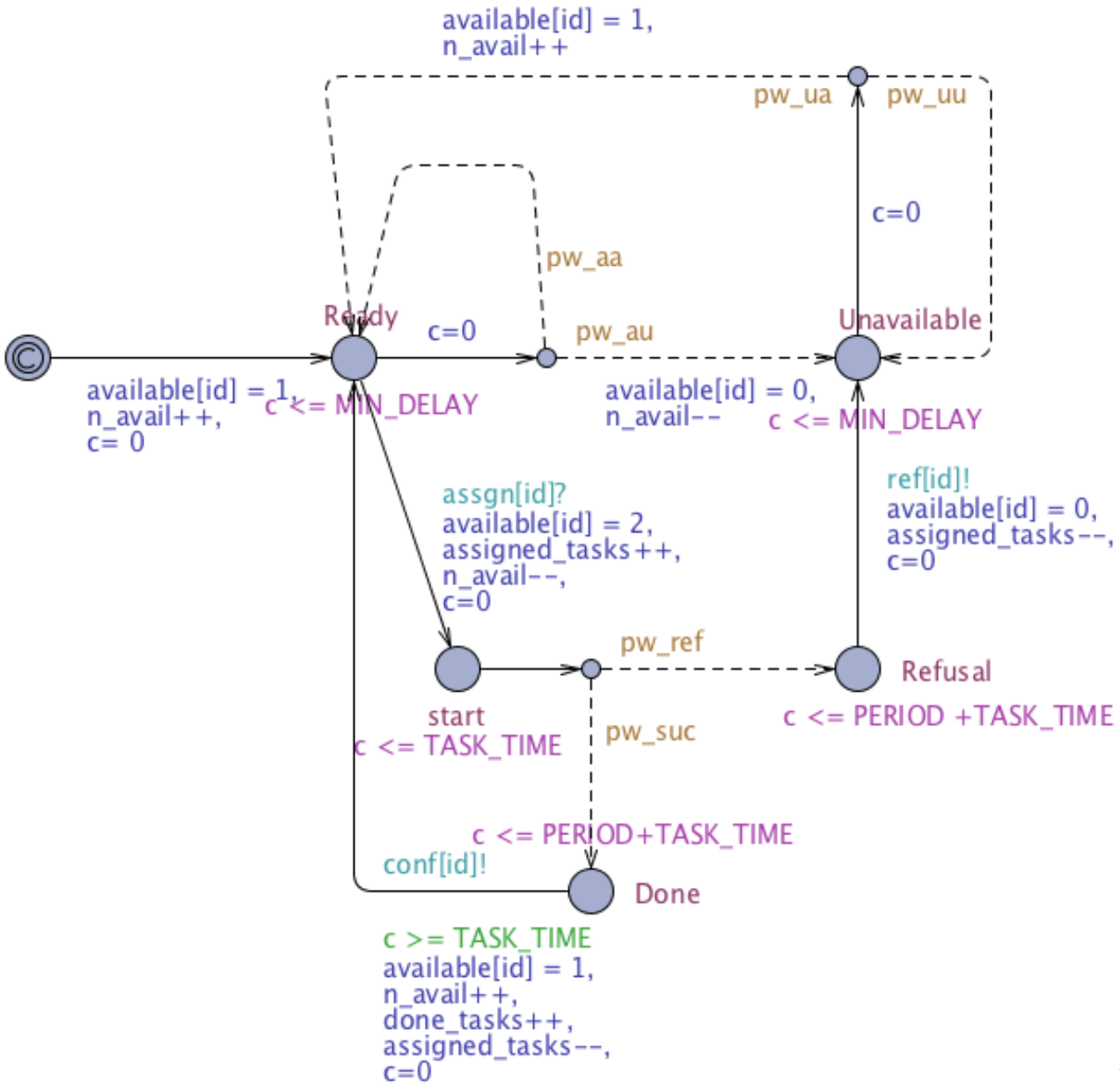
```
// Place global declarations here.

const int N = 10;                  // number of components
typedef int[0,N-1] id_comp;

const int M = 16;                   // number of possible parallel computations
typedef int [0,M-1] id_tasks;

const int TASK_TIME = 34; // task calculation time
const int INIT_DELAY = 2; // initial delay for receiving data
const int PERIOD = 5;     // monitoring period
const int MIN_DELAY = 1;  // minimal communication delay

const int pw_aa = 95;     // probab. weight of component staying available
const int pw_au = 5;      // probab. weight of component becoming unavailable
const int pw_ua = 15;     // probab. weight of component becoming available
const int pw_uu = 85;     // probab. weight of component staying unavailable
const int pw_suc = 95;    // probab. weight of component finishing a given task
const int pw_ref = 5;     // probab. weight of component failing a given task

broadcast chan assgn[N];  // channel for broadcasting task assignments
broadcast chan conf[N];   // channel for confirming task completion
broadcast chan ref[N];    // channel for refusing task execution

int[0,M] done_tasks = 0;       // number of completed tasks
int[0,M] assigned_tasks = 0;   // number of assigned tasks
int[0,N] n_avail = 0;          // number of components available for task execution

int[0,2] available[N] = {0,0,0,0,0,0,0,0,0,0}; // component availability status
```

Signal processing: the Master component



Here `id : id_comp` is a component parameter (like `ANY` clause in Event-B)

Signal processing: the Worker component

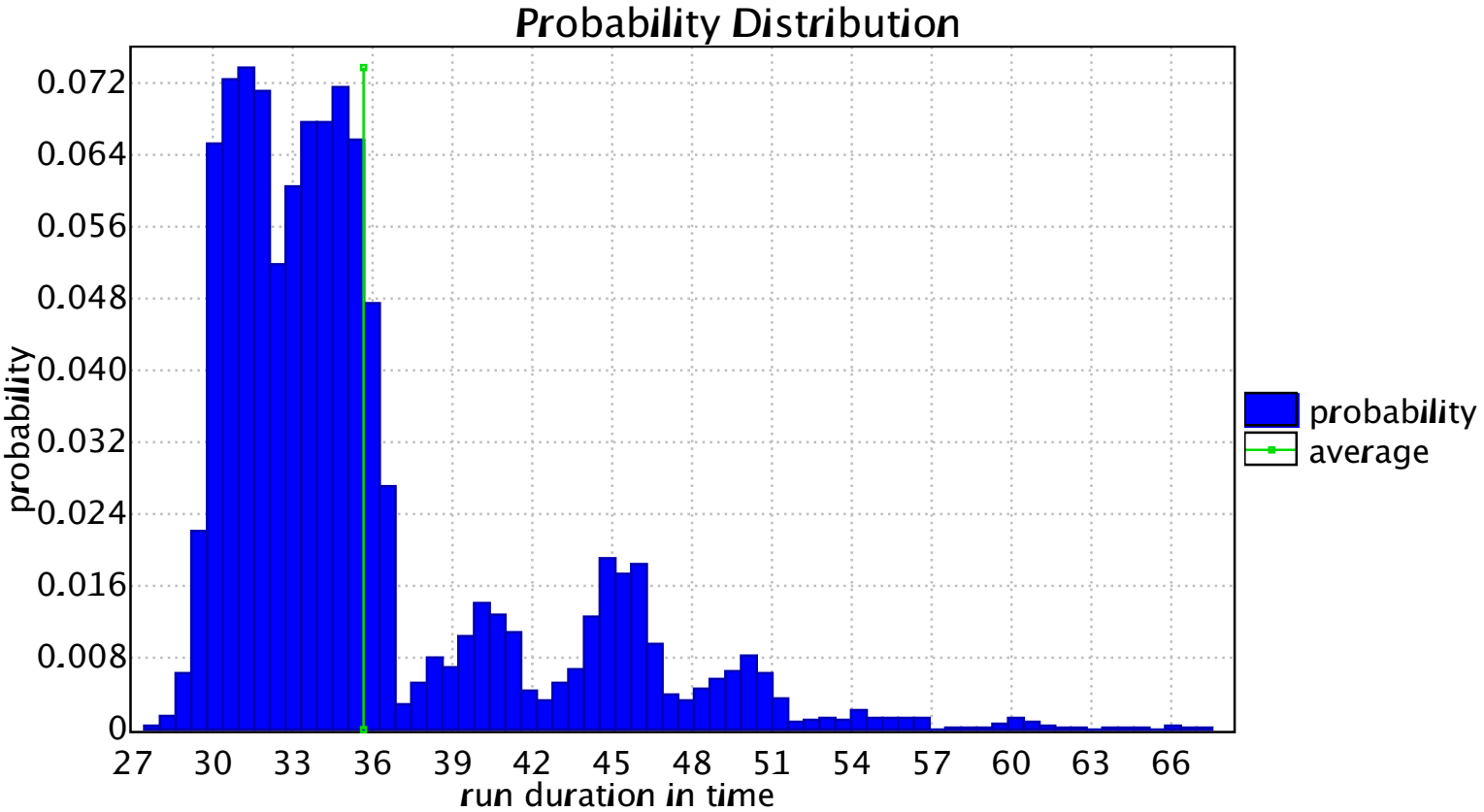


Signalling: quantitative verification examples

- A number of required time reachability properties, considering different value combinations for system parameters. All the verified properties are of the form:
  `Pr [<= time_bound](<> Master.Done)`
- The result is the probability that the Master component eventually reaches the state Master.Done (i.e., the state where all the parallel task calculations are successfully completed) within the given time bound
- Moreover, the obtained results can be graphically plotted to show probability distribution or cumulative probability distribution for different values of the clock

Signal processing

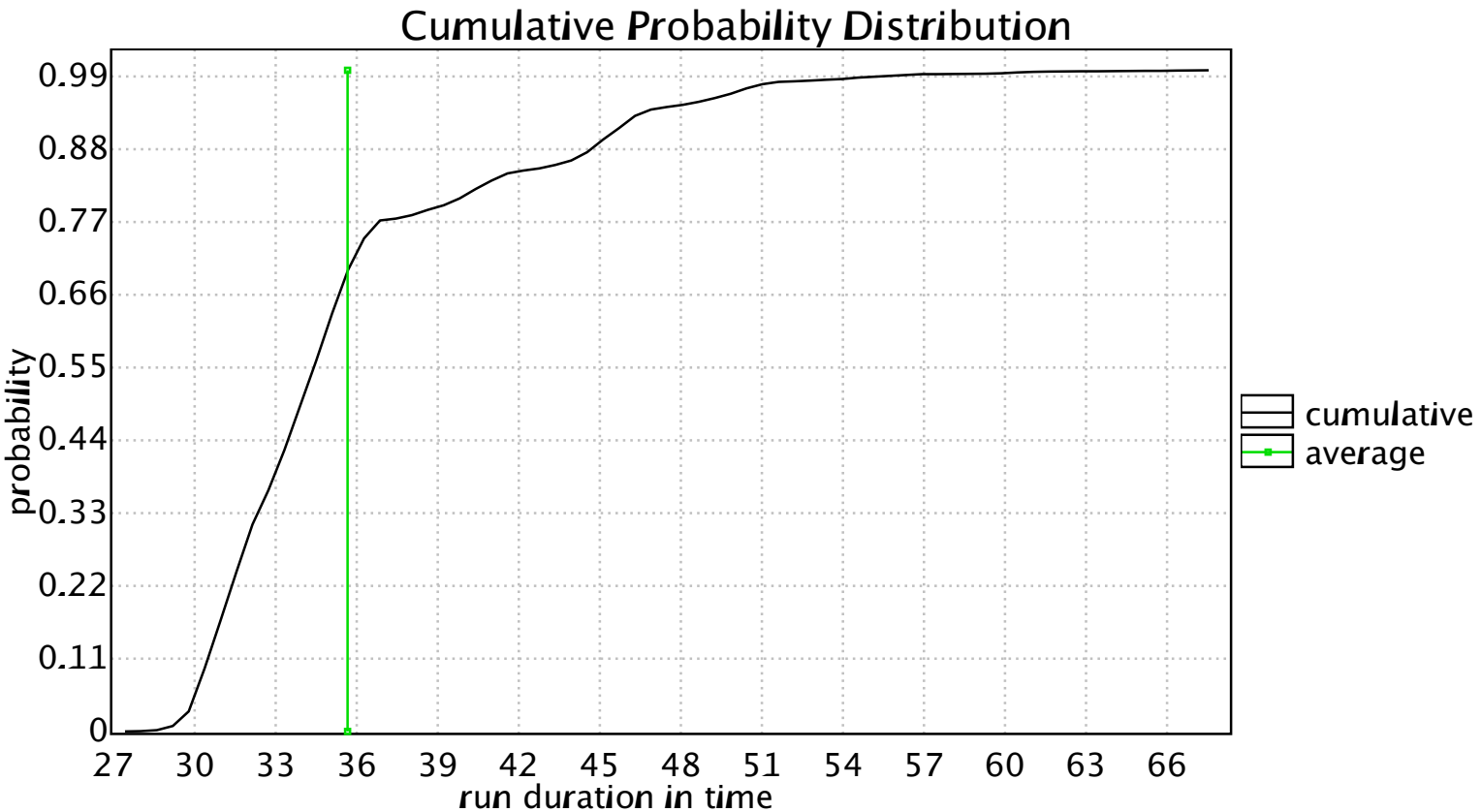
Probability distribution (for the data size 20 and 10 worker components):



Runs: 4612 in total, 4609 displayed, 3 remaining.
Probability sums: 0.99935 displayed, 0.000650477 remaining.
Minimum, maximum, average: 27.4211, 67.5414, 35.6614.

Signal processing

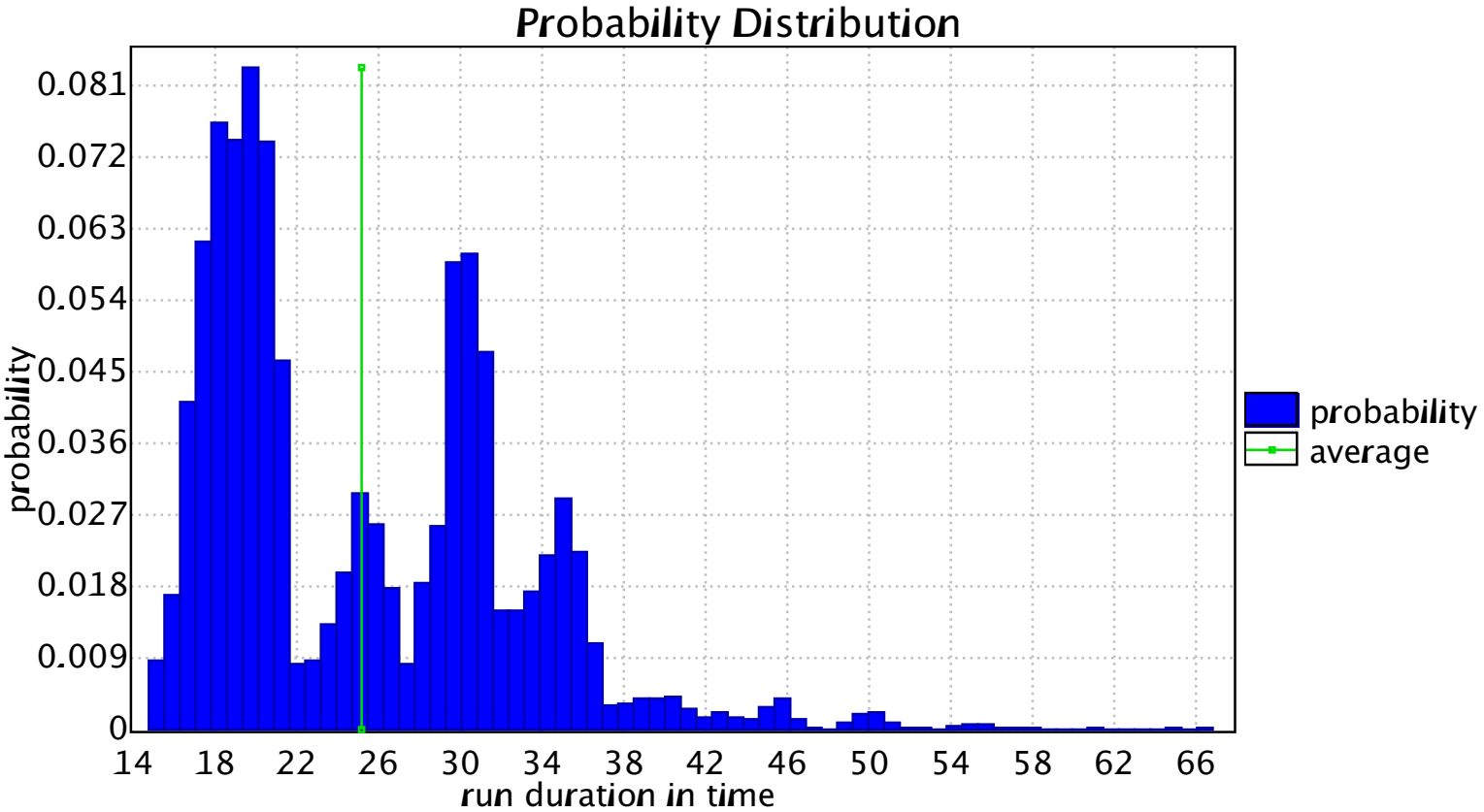
Cumulative probability distribution (for the data size 20 and 10 worker components)::



Runs: 4612 in total, 4609 displayed, 3 remaining.
Probability sums: 0.99935 displayed, 0.000650477 remaining.
Minimum, maximum, average: 27.4211, 67.5414, 35.6614.

Signal processing

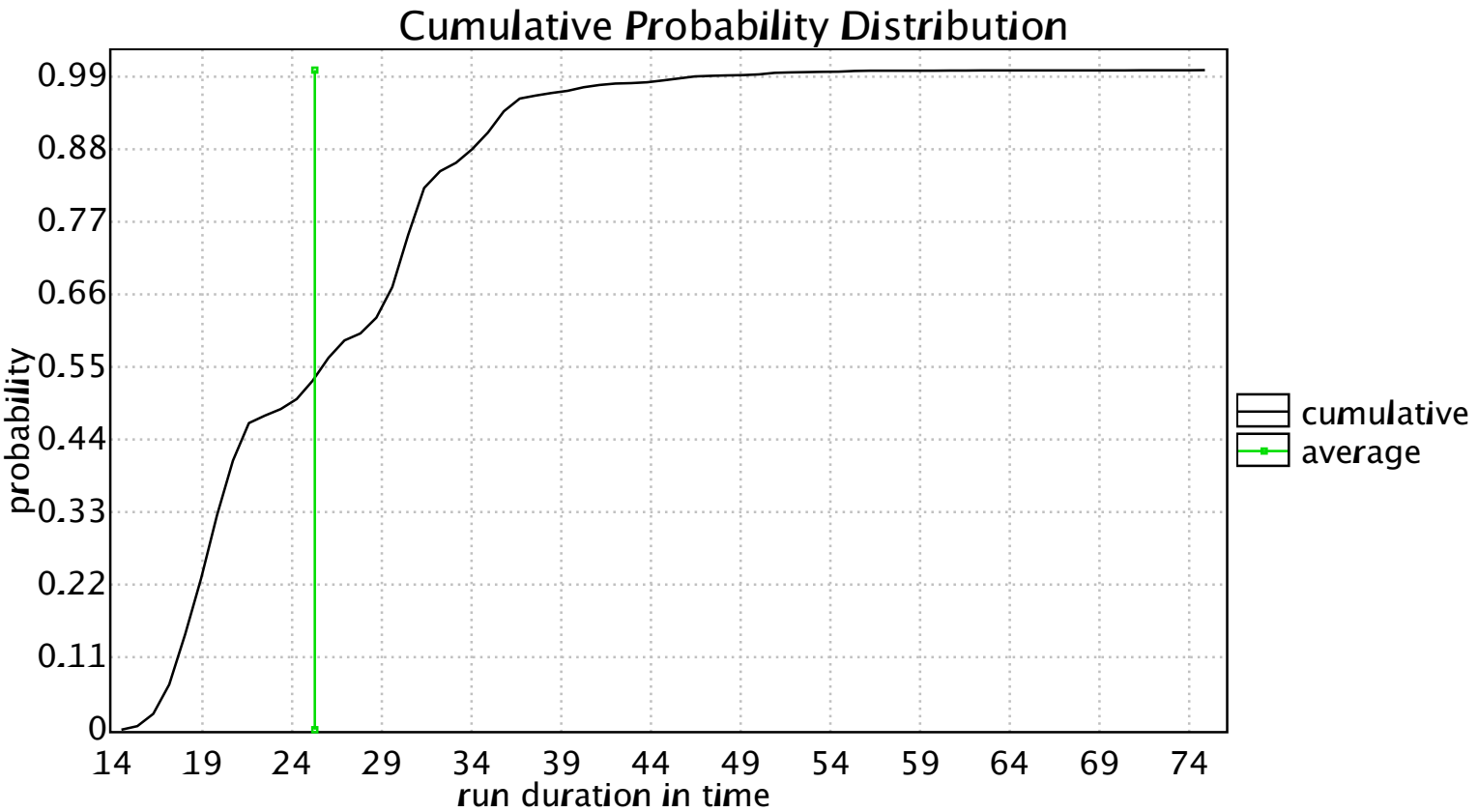
Probability distribution (for the data size 20 and 16 worker components)::



Runs: 4612 in total, 4612 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 14.7551, 66.8134, 25.1675.

Signal processing

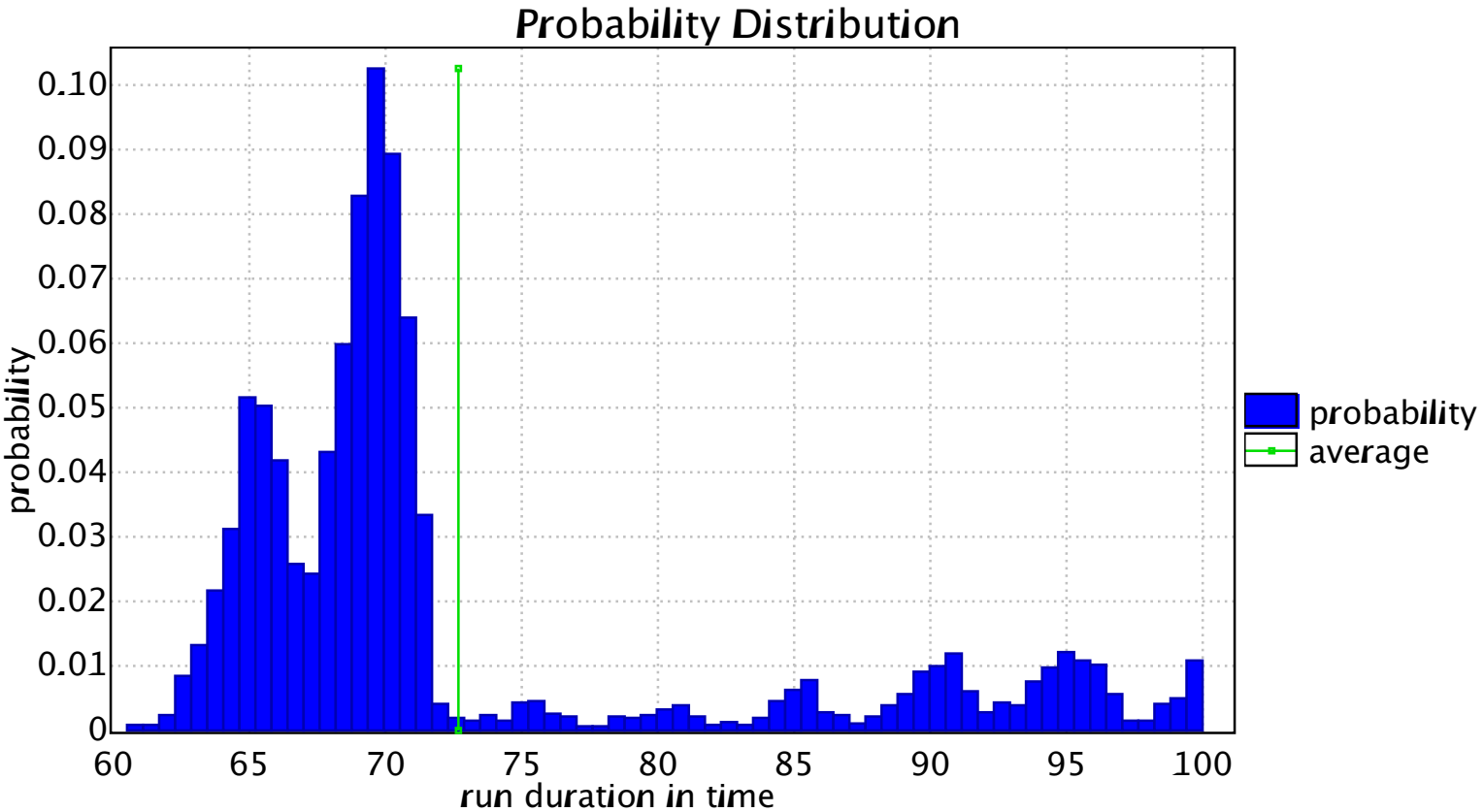
Cumulative probability distribution (for the data size 20 and 16 worker components)::



Runs: 4612 in total, 4612 displayed, 0 remaining.
Probability sums: 1 displayed, 0 remaining.
Minimum, maximum, average: 14.484, 74.8756, 25.2627.

Signal processing

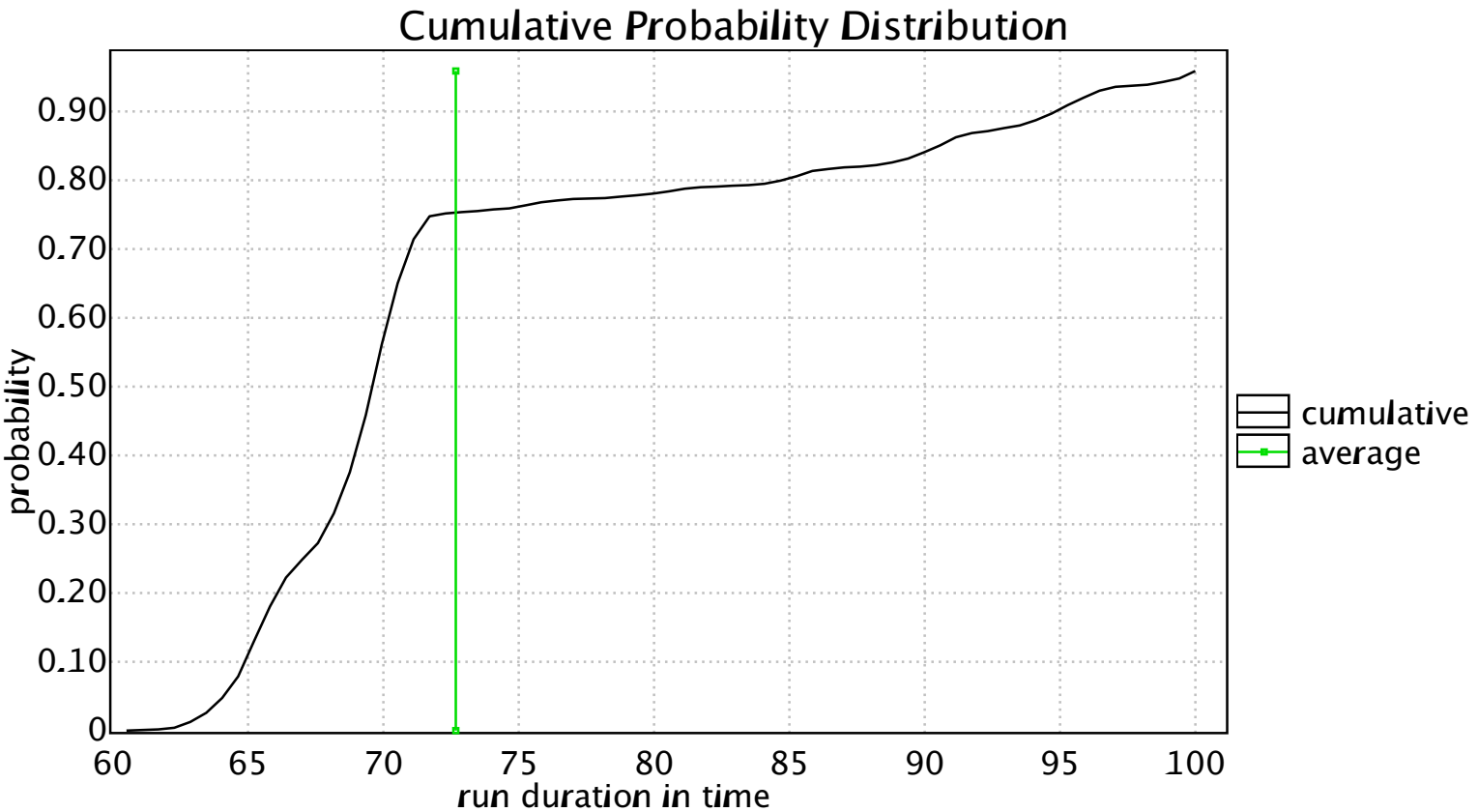
Probability distribution (for the data size 40 and 10 worker components)::



Runs: 4612 in total, 4422 displayed, 190 remaining.
Probability sums: 0.958803 displayed, 0.0411969 remaining.
Minimum, maximum, average: 60.5077, 99.9952, 72.6749.

Signal processing

Cumulative probability distribution (for the data size 40 and 10 worker components)::



Runs: 4612 in total, 4422 displayed, 190 remaining.
Probability sums: 0.958803 displayed, 0.0411969 remaining.
Minimum, maximum, average: 60.5077, 99.9952, 72.6749.

Signal processing: analysis of the optimal system architecture

- Further human expertise needed to choose the best architecture from these generated basic analysis data
- Possible "rules of thumb" (heuristics): probability distribution should be similar to the normal probabilistic one, with average time being the only main peak. Or, cumulative distribution should reach 0.99 and plateau from there as soon as possible.
- Different artificial intelligence and machine learning techniques are also applicable here

Wrapping-up of the course

- Extra time for exercise presentations: next Wednesday, June 5th, 18.00 – 21.00, 421 Didlaukio (Scheduler slot reservation in Moodle)
- Exam: 07.06.2018, 16.00 – 18.00, 103 Didlaukio
- No more than two tasks: defining data in a context or writing an invariant or an operation (event) for a given partial model, creating an Event-B model from system requirements
- Using printed lecture slides or any written notes is allowed