Scalability Analysis of the RADAR Decision Support Tool

Saheed A. Busari, Emmanuel Letier
Department of Computer Science
University College London
London, United Kingdom
{saheed.busari.13, e.letier}@ucl.ac.uk

This reports presents a theoretical complexity analysis and empirical scalability analysis of the Requirements and Architecture Decision Analyser (RADAR) described in a paper presented at ICSE'17 [1]. The report assumes familiarity with the ICSE'17 paper. Further information about the tool and its applications can be found in the ICSE paper and the tool website (http://www0.cs.ucl.ac.uk/staff/S.Busari/RADAR/). Send us an email if you have any question.

I. Complexity Analysis

The analysis of a RADAR model involves four main steps:

1) Generating the design space
2) Simulating all solutions in the design space
3) Shortlisting the Pareto-optimal solutions
4) Computing expected information value over the shortlisted solutions

The time and space complexities of these steps are summarised in Table I The first step, generating the design space, involves a single recursive traversal of the model's abstract syntax tree (AST). The time and space complexity of this step is thus $\mathcal{O}(m)$ where m is the model length measured as number of nodes in the model's AST. The second step is the most computationally expensive. It generates a matrix *SimResult* of dimension $|DS| \times |Obj|$ where DS is the model's design space and Obj the model's objectives; $SimResult[s, obj]$ denotes the simulated value of objective obj for solution s. Each cell in the matrix is computed by generating N simulations

Analysis Step	Time Complexity	Space Complexity
Generating the design space	$\mathcal{O}(m)$	$\mathcal{O}(m)$
Simulating all solutions in the design space	$\mathcal{O}(\lvert Obj\rvert \times \lvert DS\rvert \times N \times m)$	$\mathcal{O}(\lvert Obj\rvert \times \lvert DS\rvert \times N \times m)$
Shortlisting the Pareto-optimal solutions	$\mathcal{O}(\lvert DS\rvert^2)$	$\mathcal{O}(\lvert DS\rvert^2)$
Computing expected information value over the shortlisted solutions	$\mathcal{O}(N \times \lvert S\rvert)$	$\mathcal{O}(N \times \lvert S\rvert)$

TABLE I: Time and Space Complexity of RADAR analysis algorithms. m is the number of nodes in the model's AST, Obj is the model objectives, DS is the Design Space, N is the number of simulations and S is the shortlisted solutions.

of the objective's random variable. Each simulation involves a single recursive traversal of the model's AST and is thus $\mathcal{O}(m)$. Generating N simulations for all objectives and all solutions thus has a time and space complexity of $\mathcal{O}(|Obj| \times |DS| \times N \times m)$. Our implementation does not store the objectives' random variables simulations. Doing so would involve maintaining a three dimensional matrix of size $|Obj| \times |DS| \times N$ which would be too big for most computer's memory for decision problems such as the Emergency Response System considered in the ICSE paper where the design space size is 6912 and number of simulations N is 10^4. The third step involves finding the Pareto-optimal solutions in the *SimResults* matrix generated in the second step. Our implementation, that involves comparing pairs of solutions, has a worst case complexity of $\mathcal{O}(|DS|^2)$. However, an optimised algorithm exist with a complexity of $\mathcal{O}(|DS|log|DS|)$ when $|Obj| \leq 3$ and $\mathcal{O}(|DS|(log|DS|)^{|Obj|-2})$ when $|Obj| \geq 4$ [2]. The fourth step involves computing the expected value of total perfect information ($EVTPI$) and the expected value of partial perfect information ($EVPPI$). $EVTPI$ and $EVPPI$ are always evaluated with respect to a given objective, noted $\mathbf{Max}\ \mathbf{EV}(NB)$, and for a given set S of alternative solutions. Our implementation takes S to be the set of Pareto-optimal solutions shortlisted in step 3. We estimate $EVTPI$ using the classic formula:

$$EVTPI = \underset{i:1..N}{\text{mean}} \max_{j:1..M} \overline{\overline{NB}}[i,j] - \max_{j:1..M} \underset{i:1..N}{\text{mean}} \overline{\overline{NB}}[i,j].$$

where $\overline{\overline{NB}}$ is the simulation matrix that contains simulations of NB for each solution in S [3]. The time complexity of such operation is $\mathcal{O}(N \times |S|)$. We estimate $EVPPI$ using a recent efficient algorithm that estimates $EVPPI$ for a parameter x from $\overline{\overline{NB}}$ and and the vector $\overline{x}$ containing the simulations of parameter x [4]. The complexity of this algorithm is also $\mathcal{O}(N \times |S|)$.

II. Empirical Scalability Analysis

In the ICSE paper [1], we report the application and running time of RADAR analysis on four real-world examples. The largest model, the analysis of architecure decisions for an emergency response system, has a design space of 6912 solutions and takes 111 seconds to analyse (less than 2 minutes). In this section, we further evaluate RADAR's scalability by measuring its running time on larger synthetic models. We perform experiments to answer the following research questions:

- **RQ1:** What is RADAR's Scalability with respect to the number of simulations?
- **RQ2:** What is RADAR's Scalability with respect to the size of the design space?
- **RQ3:** What is RADAR's Scalability with respect to the number of objectives?
- **RQ4:** What is the time spent and memory consumed by each analysis step?

To perform experiments answering these questions, we have implemented a synthetic model generator that generates random syntactically valid RADAR models with a given number of objectives, decisions, number of options per decisions and minimum number of model variables. The model generator can produce RADAR models with or without decision dependencies. The model generator constructs a synthetic RADAR model following the RADAR syntax defined in the ICSE'17 paper [1]: a model's objective is declared as either a maximisation or a minimisation problem, whose definition could be an expectation or a probability defined over a random variable. For each objective, the random variable that defines it is refined into three child variables which are related by randomly chosen arithmetic operators (e.g. "+", "–", "/" and "*"). The three child variables are typically of the form *ANDRefinement*, *ParameterEstimation* and *OrRefinement*, respectively. Following the same format, each child variable is further refined and related to three variables until both the specified number of decisions and minimum number of model variables have been attained. For synthetic models without decision dependencies, each expression corresponding to the individual option of an *OrRefinement* is always a parameter estimation. However, for models with decision dependencies, each expression corresponding to the individual option of an *OrRefinement* could be an *ANDRefinement*, *ParameterEstimation* or *OrRefinement*. The number of decision dependencies specified determines the number of times an *OrRefinement* variable is linked to the option of another *OrRefinement* variable. The model generator and all models generated for the experiments below are available from the tool's website (http://www0.cs.ucl.ac.uk/staff/S.Busari/RADAR/). All our experiments are run on a machine running Linux with a four-core 2.6 GHz processor and 10GB RAM.

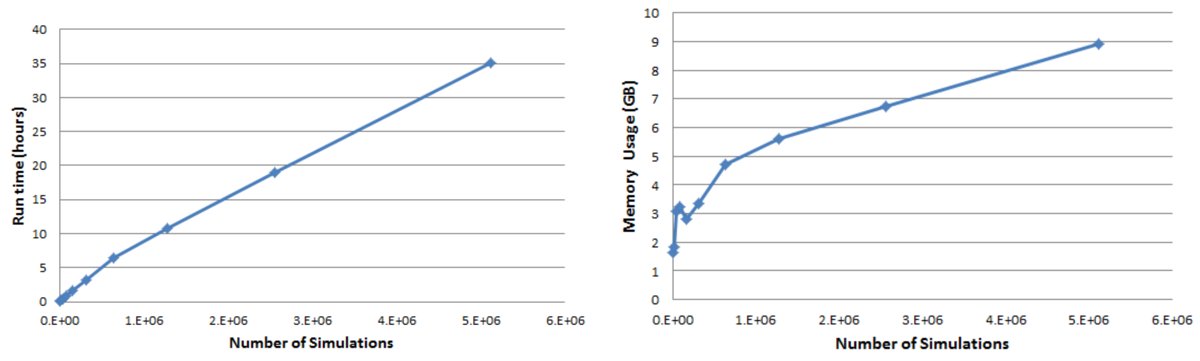

Fig. 1: Total run-time (left) and memory usage (right) measured for doubling the number of simulations, N, from 10^4 to 512×10^4.

RQ1: What is RADAR's Scalability with respect to the number of simulations

To evaluate how RADAR run-time and memory usage increases as the number of simulation, N, increases, we have generated a synthetic model whose characteristics are similar to that of the emergency response system, i.e. it contains 2 objectives, 10 decisions, 3 options per decisions, and no decision dependencies. We have then measured the running times and memory consumption of analysing this model when doubling N 10 times from 10^4 to 512×10^4. The results are shown in Figure 1 and indicate that the running time and memory usage increase linearly with N.

RQ2: What is RADAR's Scalability with respect to design space size

To evaluate how RADAR run-time and memory usage increases when the design space size increases, we have generated synthetic models with decision dependencies by incrementally and separately increasing the number of decisions and options per decisions until the resulting models could no longer be analysed in less than an hour. The synthetic models generated have 2 objectives and at least 100 model variables. Figure 2 shows the result of this experiment. RADAR was able to evaluate in less than one hour models with a design space of up to 153,751 solutions. The model with the largest design space included 11 decisions with 7 options. The figure also show that on our synthetic models the run-time and memory usage increase roughly linearly with the size of the design space.

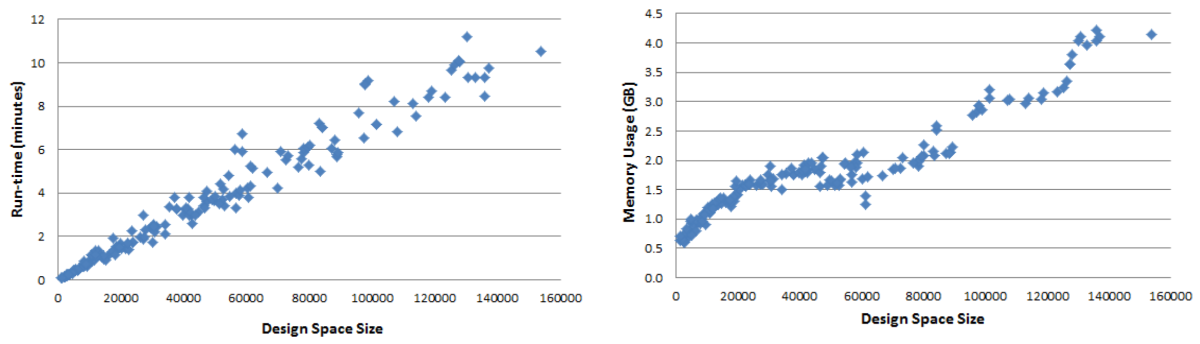

Fig. 2: Total run-time (left) and memory usage (right) measured for 180 RADAR models with different design space size.

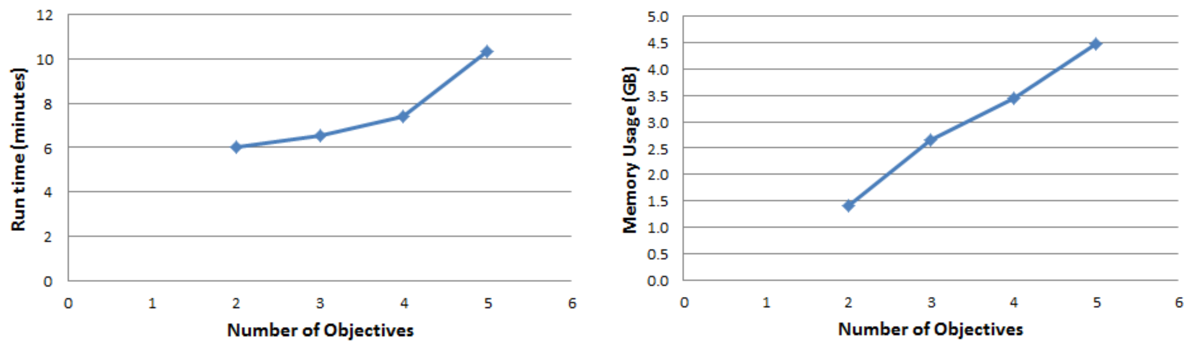

Fig. 3: Total run-time (left) and memory usage (right) measured for 2,3,4 and 5 objectives.

RQ3: What is RADAR's Scalability with respect to the number of objectives

To evaluate how RADAR run-time and memory usage increases when the number of objectives increases, we have generated synthetic models with 10 decisions, 3 options per decisions, and incrementally increased the number of objectives from 2 to 5 until the resulting models could no longer be analysed in less than an hour. The synthetic models generated do not have decision dependencies. Figure 3 shows that on our synthetic models the run-time and memory usage increase roughly linearly with the number of objectives.

Algorithm Step	Average % Total Time	Average % Memory Usage
Generating the design space	0	0
Simulating all solutions in the design space	100	96
Shortlisting the Pareto-optimal solutions	0	1
Computing expected information value over the shortlisted solutions	0	3

TABLE II: Real world RADAR applications and their problem sizes.

RQ4: What is the time spent and memory consumed by each analysis step

For each synthetic model generated in the experiment to answer RQ2, we measured the fraction of time spent and memory used in each of the four analysis step: generating the design space, simulating the design space, shortlisting the Pareto-optimal solutions, and computing expected information value. Table II shows the average fraction of time for each analysis step over all synthetic models. The table shows that the simulation of all solutions takes the largest portion of time (100%) and memory consumption (96%).

III. Conclusion

This report presented the theoretical complexity of RADAR analysis steps: generating the design space, simulating the design space, shortlisting the Pareto-optimal solutions, and computing expected information value. We also evaluated RADAR's scalability by measuring its running time and memory consumption on large synthetic models. Our empirical results show that: the run time and memory consumption of the RADAR analysis steps are linearly proportional to the number of simulations (N), the design space size and the number of objectives; the design space of models without decision dependencies increases exponentially with the number of decisions and options; the simulation of the design space takes the highest average proportion of the run time (100%) and memory usage (96%). Future work involves scaling the RADAR analysis steps, particularly simulation step, using multi-objective optimisation evolutionary algorithms (MOEA), such as NSGAII, SPEAII, MOGA and IBEA.

References

[1] S. Busari and E. Letier, "Radar: A lightweight tool for requirements and architecture decision analysis," in *39th International Conference on Software Engineering (ICSE 2017)*, 2017.

[2] H.-T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM (JACM)*, vol. 22, no. 4, pp. 469–476, 1975.

[3] E. Letier, D. Stefan, and E. T. Barr, “Uncertainty, risk, and information value in software requirements and architecture,” in *36th International Conference on Software Engineering (ICSE 2014)*, 2014, pp. 883–894.

[4] M. Sadatsafavi, N. Bansback, Z. Zafari, M. Najafzadeh, and C. Marra, “Need for speed: an efficient algorithm for calculation of single-parameter expected value of partial perfect information,” *Value in Health*, 2013.