Emergent Robustness in Software Systems through Decentralized Adaptation: an Ecologically-Inspired ALife Approach

Franck Fleurey, Benoit Baudry, Benoit Gauzens, André Elie, Kwaku Yeboah-Antwi

Emergent Robustness in Software Systems through Decentralized Adaptation: an Ecologically-Inspired ALife Approach

Franck Fleurey[1], Benoit Baudry[2], Benoit Gauzens[3], André Elie[2] and Kwaku Yeboah-Antwi[2]

[1]SINTEF, Norway
[2]INRIA, France
[3]Université de Rennes 1, France
benoit.baudry@inria.fr

Abstract

The ecosystem of web applications faces a critical paradox: on one hand, the Internet is a constantly evolving and unpredictable computing platform, on the other hand, the software services that run on top of it hardly have the ability to adapt to the evolution of this platform. Among the software services, we distinguish between *service providers* that provide micro services and *service consumers* that aggregate several micro services to deliver macro services to customers. Providers and consumers must handle uncertainty: providers cannot know in advance what consumers need; consumers rely on third-parties that can disappear at any time. Our proposal analogizes the software consumer / provider network to a bipartite ecological graph. This analogy provides the foundations for the design of EVOSERV, an individual-based ALife simulator used to experiment with decentralized adaptation strategies for providers and consumers. The initial model of a software network is tuned according to observations gathered from real-world software networks. The key insights about our experiments are that, 1) we can successfully model software systems as an ALife system, and 2) we succeed in emerging a global property from local decisions: when consumers and providers adapt with local decision strategies, the global robustness of the network increases. We show that these results hold with different initial situations, different scales and different topological constraints on the network.

Introduction

The infrastructure of the Internet (computers, routers, servers and connections) can be considered an evolving complex adaptive system (Park and Willinger, 2005; Albert et al., 1999). Mapping the internet as a graph, it is possible to observe that the internet is an extremely adaptive network which is highly unpredictable due to its attrition/churn rate (i.e. nodes or connections frequently appear and disappear with no warning). Software companies tackle this unpredictability using loosely coupled architectures (Huhns and Singh, 2005): on one hand, *service providers* develop and maintain micro services that handle basic functionalities (e.g., database management, access control, etc.) which they provide over the web; on the other hand, *service consumers* access several micro services over the Internet to build macro services that they provide to customers (e.g., salary management or travel planning). Despite these efforts, both providers and consumers still face uncertainty. Providers must develop services with no certainty about what consumers exactly need, how often they need it and what level of granularity is needed. This poses an essential challenge when it comes to deciding what services to provide and in which quantities. Consumers aggregate third-party services, but they cannot predict if the provider will fail, or if the connection to the provider will fail or which provider provides the most of the services they need. In complex adaptive systems, entities that cannot evolve in response to environmental changes contribute to the imperilment of the robustness of the whole system (e.g., as demonstrated in food webs (Staniczenko et al., 2010)). In the context of software networks, this means that the lack of evolutionary capabilities in software service providers and consumers greatly reduces the robustness of the global software network (providers, consumers and connections). Today, the robustness of these software networks relies on either over-approximated redundancy (providers provide much more than needed in case the consumer's demand increases) or on centralized techniques that assume a global view and knowledge of the network and its topology. The former solution which is the most commonly used today implies and results in a waste of resources, while the latter is rarely applied because it is often impossible for software companies to build and have an accurate global view of the system (especially concerning the interaction between them and their consumers). There exists a need for novel software engineering approaches to handle the uncertainty and dynamicity of Internet applications (Bertolino et al., 2015).

In this work, we investigate a novel approach to engineering software services deployed on evolving computing platforms. Following the intuition that "*computer systems can be better understood, controlled, and developed when viewed from the perspective of living systems*" (Forrest et al., 2002), we developed EVOSERV, an ecologically-inspired ALife individual-based model and simulator which models software systems as an artificial life system. EVOSERV analogizes provider/consumer software interactions to mu-

tualistic ecological interactions (e.g., bee/pollinator networks). Treating consumers and providers as individuals in an ecological system, EVOSERVenables the modeling of evolution in a software system during the course of its lifecycle thereby allowing the investigation of the changes in robustness of the software system during its lifecycle. We tune EVOSERVwith two large real-world software networks.

We propose four (4) major classes of localized strategies that describe and govern how consumers and providers evolve and adapt. These four (4) classes of strategies are generalized from the set of localized evolutionary/adaptation actions that are possible in a distributed software system. Using EVOSERV, we empirically compare and contrast the adaptability of various software systems subjected to these strategies focusing on the changes in robustness of the global software systems. We define *robustness* as the ability of consumers to survive the extinction of providers and adapt the notion of extinction sequence from ecology (Burgos et al., 2007) to measure it.

Our results empirically show that, localized adaptation of consumers and providers in software systems enables the emergence of global functional robustness. They empirically prove that systems utilizing our localized adaptation strategies perform better than a random adaptation strategy (33% robustness with random adaptation vs. 46% with our adaptation strategies). We also run experiments that confirm the stability of these results at different scales and on different software systems.

In this paper, we develop three main contributions

- EVOSERV[1], an ALife, individual-based simulator that accurately models the evolution and adaptation of a complex software system during its lifecycle and allows the investigation of the resultant robustness.
- Empirical evidence showing that global robustness can emerge from localized evolution rules in provider/consumer software networks.
- Empirical data about two real-world provider/consumer software graphs in which we consistently observe a power-law distribution for (i) the number of services in consumers and (ii) the rate of service usage among consumers. We use this data to tune the initial bipartite network used in EVOSERV.

Bipartite interactions in Internet Computing

In this section, we introduce the bipartite graph model upon which our simulation is built, as well as the different evolutionary strategies and mechanisms that we experiment with. We chose a bipartite graph model as the foundation because it allows us to capture and represent the interactions and relationships present in a software network.

[1] https://github.com/DIVERSIFY-project/EVOSERV

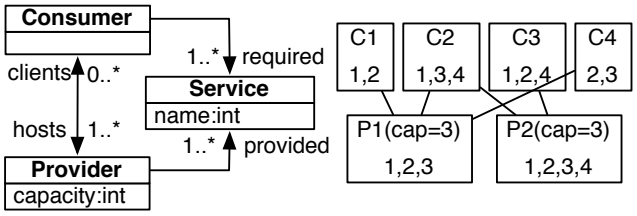

Figure 1: Model and one instance of software bipartite graph

Bipartite software interactions

The left part of Fig. 1 describes the data model we use to abstract bipartite software relationships between the different components of a software system.

- `Provider` individuals are an abstract representation of nodes on the Internet that provide micro-services to other nodes (e.g. a web server). A `Provider` P, is characterized by three attributes: `provided`, a list of services provided by P; `clients`, the list of consumers that consume at least one service of P; a `capacity`, the maximum number of Consumers that can access one or more of P's services (every individual accessing P is counted once irrespective of the number of services accessed).
- `Consumer` individuals aggregate remote services in order to offer macro-services to their customers (e.g., a travel web site such as Expedia). A `Consumer` C is characterized by two attributes: `required`, a list that contains all the services that C needs; `hosts`, the set of providers to whom C connects to over the Internet in order to get all the services it needs (we do not impose a limit on the number of providers to whom a consumer can connect to).

A `Service` in our model refers to a resource that is being required or provided. A service is referenced by its name(an integer identifier in our model).

The right part of Fig. 1 displays an example of a software system modeled in EVOSERV. It contains 4 consumers and 2 providers with both providers having a capacity of 3. Consumer $C1$ requires services 1 and 2 to function correctly and accesses those services through a connection to $P1$; $C2$ requires services 1, 3 and 4 and accesses them through connections to $P1$ and $P2$. Provider $P1$ is currently at maximum capacity while $P2$ can still serve one more client.

The following sub-sections describe the dynamic aspects of software bi-partite graphs. We distinguish between two types of changes: adaptation and evolution. *Adaptation* refers to changes to individuals nodes and it does not vary the composition of the set of nodes in the system, while *evolution* refers to changes which vary the composition of the system(ie. providers and consumers can appear and disappear over time).

Adaptation in EVOSERV

Provider Adaptation Providers adapt by changing the set of services they offer. We experiment with two different types of adaptation operators

Random: A provider P chooses at random to either add a service or drop one of the services it provides.

Popular: A provider P chooses either to add a popular service to its list of services provided or to drop a unpopular service from its list of services provided. When P adds a service, it favors services that are popular among its consumers with more popular services having higher probabilities of being selected. Conversely, when P drops a service, services that are not popular among P's consumers have higher probabilities of being dropped. The probability of dropping a service increases with the number of consumers connected to P. This is in order to reduce the pressure on this provider when it's at capacity. We assume that P can access the set of services required by its consumers.

Constraints on Provider's adaptive behavior: Both adaptive behaviors described above are constrained such that, all consumers of a given provider, P, can always access what they require. Thus, if P decides to drop a Service, it first checks that none of its consumers would go extinct (each consumer provides a routine that assesses whether the Service is strictly required).

Consumer Adaptation Consumers adapt by changing the links to providers through which they access their required services. We experiment with two different kinds of adaptive operators

Random: A consumer, C, chooses at random a new provider to link to.

Equitable: A consumer, C, attempts to optimize the diversity of its service provisioning by accessing a set of providers that maximizes its equitability (`i.e.` the number of times each required service is provided). Given $\mathcal{S} = \{s_1, \ldots, s_n\}$, the set of services that a consumer requires and $\#occ_{s_i}$, the number of times a service s_i is provided through a link to a consumer, the Shannon index for a consumer is:

$$H'_{i,\mathcal{P}} = -\sum_{k=1}^{n} \frac{\#occ_{s_k}}{n} \ln \frac{\#occ_{n_k}}{n}$$

and its equitability is $\exp(H'_{i,\mathcal{P}})$. Maximizing equitability ensures that a consumer has a set of connections which offers each required service an even number of times.The concept of equitability is taken from ecology where it is used to evaluate biodiversity.

In practice, many different advertisement or discovery mechanisms can be used to find the set of potential providers. Our experiment does not intend to replicate any particular mechanism but only assumes that at any point in time, consumers have access to a neighbourhood of providers, which we pick as a random subset of the set of all providers.

Constraints on consumers adaptation: Both adaptation operators described above are constrained such that, each required service that was already provided remains provided. Any existing link that only provides consumer C with services that are also provided through other links can be discarded, thereby allowing new links to be made to any provider in the neighbourhood who provides at least one of C's required services thats currently unprovided.

Global constraints over the model to keep costs comparable between different experiments :

The total number of links in the network or the total number of services provided in the network can change over the adaptation process and can introduce a confounding bias. Allowing more links or introducing more provided services can have a positive impact on robustness by providing consumers with more opportunities to link to their required services. However, we wanted to prevent this phenomenon in order to isolate the effect of adaptation rules when evaluating the impact on global robustness.

The number of services and links are therefore kept constant. This constraint is enforced in the following manner: the simulator allows a random set of nodes to run adaptations which decrease the number of links and services and collects tokens corresponding to these deletions. In a second step, those tokens are distributed randomly in order to allow some nodes to add links and services. This token system ensures the decentralized nature of our simulations while conserving some global constraints. In the real world, this corresponds to fixing the total amount of available resources in the software system (e.g., the number of services and the amount of bandwidth).

Ecological concepts for robust software networks

In this work, we analogize a network of software services that run over the Internet to bipartite ecological graphs. This analogy comes with two essential principles that we build upon in this work: 1) species in ecological communities adapt in a completely decentralized manner to cope with a continuously evolving environment; 2) ecologists have developed sound measures to quantify the robustness of a bipartite graph as represented by interactions between species. In the following, we summarize these two ecological principles.

Species adaptation in ecosystems is completely decentralized and is not driven by a global goal at the ecosystem level. Adaptation is necessary in the face of an evolving environment; as environmental conditions change over time, species have to adapt (i.e. keep a good reproductive success) to their local conditions. The *Red Queen hypothesis* (Van Valen,

1973), named in reference to a statement[2] made to Alice by the Red Queen in Lewis Carroll's *Through the looking glass*, crystallizes the idea that, species need to constantly evolve in order to survive, in the same way as Alice and the Red Queen need to constantly run to stay in the same place. The *Red Queen Hypothesis* provides a good theoretical raison d'être for adaptation, stating that, in order to increase the persistence of our Consumers and Providers individually, a constant adaptation is key to keeping a good fit to specific conditions. This hypothesis however lacks a comprehensive view at the macroscopic scale: is the robustness of our system an emergent property of such behavior?

Some experiments have shown that large levels of diversity and redundancy promote the stability of functional processes in ecosystems like biomass production (Tilman and Downing, 1994). However, the mechanisms underlying this experimental evidence remains elusive in the case of complex networks of interacting species. Mathematical models show that more complex systems (in terms of species richness and number of interactions) are more likely to collapse. This incompatibility between theoretical and empirical observations is known as May's paradox (May, 1972). This paradox and more broadly, the relationship between complexity, evolution and stability is a still open fundamental question in ecology. Most of this previous work reasons on ecological graphs where nodes represent species, and edges ecological interactions (e.g. predation or parasitism). The topology of ecological graphs (emerging from ecological processes such as extinction, colonization, but also evolution) is non random, and its structure seems to favour stability (Yodzis, 1981).

We investigate evolution and adaptation in software systems in order to experiment with the emergence of global robustness through localized decentralized actions. Given a bipartite graph that models a network in which nodes are species and edges are species interactions, the *robustness measure* (Burgos et al., 2007) quantifies the ability of one level of the graph to survive the extinction of species in the other level. The primary extinction sequence can be performed according to different strategies: randomly removing nodes, removing the most or least connected ones, etc. In the case of mutualistic graphs, as soon as a species looses all of its connections, it is unable to reproduce (for plants) or feed (for pollinators) and thus goes extinct. This is called a *secondary extinction*. Referring back to the network in Fig. 1: if P1 is removed, C4 goes extinct. If C1 has the ability to adapt, it can connect to P2 otherwise it also goes extinct, a *secondary extinction*; subsequently, when P2 is removed, the whole network goes extinct.

Fig. 2 plots the relationship between secondary and primary extinctions. The robustness index is the area under the curve. In our experiments, we normalize the robustness index by considering it a ratio of the maximum robustness value (in Fig. 2 the robustness is thus $\frac{area\_under\_curve}{350\times100}$). Since primary extinctions are random, we compute several robustness indices on a given graph to obtain a mean. For example, Fig. 2 shows 20 runs and the mean value as a thick line: the lowest robustness index is 0.33 and the max is 0.4.

[2]"Now, here, you see, it takes all the running you can do, to keep in the same place."

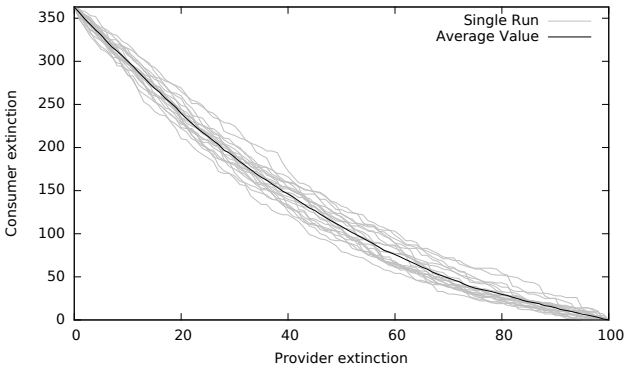

Figure 2: Relationship between secondary and primary extinctions in a bipartite graph (with a random primary extinction sequence)

Experimental design

All of our experiments start with a bipartite graph that models a software system and then simulates the evolution of that graph to evaluate the effect of the decentralized adaptation rules. In this section, we discuss how we set up the initial graph, and how we tune the different experimental parameters.

Mining software interactions to tune EVOSERV

We analyzed two real-world software systems in order to generate real world data to aid us in tuning of our initial graph. We collected data about installations of WordPress [3], an open-source content management system used to easily deploy web sites, and also data about web browsers. We selected these two case studies because both are very popular technologies (WordPress is deployed on 23% of the top 10 million web sites [4], and billions of web browsers are deployed and used daily worldwide) and also because both technologies are open and can be easily extended by their users through the addition of plugins that provide specific functionalities (e.g., display photo, read pdf, etc.). For both case studies, we collected the number of plugins installed on each client in the dataset and the subsequent distribution of plugin usage. This data is used to tune the size of software entities in our graph, as well as the distribution of services on these entities.

[3]https://wordpress.org/

[4]http://w3techs.com/technologies/overview/content_management/all/

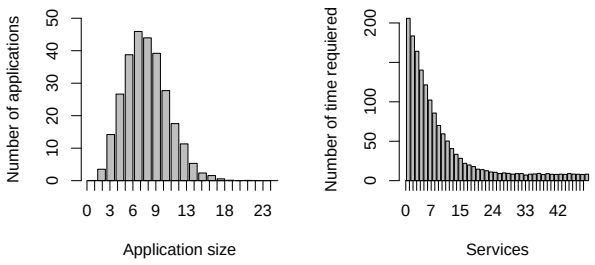

Figure 3: Number of Services per entity (left) and distribution of Service usage (right) in the initial graph

WordPress: Analyzing the 500 000 top web sites [5], we selected the ones that use WordPress. This gave a list of 110 000 WordPress sites. We then crawled these sites to find out which plugins they use. Analyzing this data [6], we found out that the number of plugins per site follows a poisson distribution with an average value of 5. The distribution of plugin usage follows a negative exponential slope.

Web browsers: Since browsers are installed on user machines, we could not access this data directly. We therefore set up `https://amiunique.org/`, a website where we collect anonymous information about the browser, the list of fonts and plugins it contains from every visitor. By February 2015, the site had been visited by 63,000 unique visitors. All of the observed browsers contained a total of 1,920 different plugins. We observeed distributions very similar to the ones in WordPress: a negative exponential slope for the plugin usage distribution and poisson distribution with an average of 6 for the number of plugins per browser.

Tuning the initial model The initial graph for our simulation had the following size: 300 Consumers, 100 Producers and 50 Services. The 1:3 ratio between the 2 first parameters was loosely based on a Cloud Service Brokerage example, with the values increased by an arbitrary multiplying factor of 100. We fixed the number of evolution cycles through which the bipartite graph is run to 500. We then used the data observed on real systems to tune the size of Consumers, as well as the distribution of the Service usage among these Consumers. As seen in Fig. 3, the size of Consumers follows a poisson distribution of parameter 5.47, and the Service usage, a power law distribution of parameter 2.08.

Experiments

In order to evaluate the effect of the adaptation strategies described in section II, we run the simulation with four different combinations of adaptive behaviors for Consumers and Providers.

random-random: Consumers and Providers adapt randomly. This is the baseline

random-popular: Consumers adapt randomly and Providers adapt according to the *popular* adaptation behavior.

equitable-random: Consumers adapt according to the *equitable* adaptive behavior and Providers adapt randomly.

equitable-popular: Consumers adapt according to the *equitable* adaptive behavior and Providers adapt according to the *popular* adaptation behavior.

To ensure statistical validity, we needed to run the simulation and robustness computations a sufficient number of times. We utilized Monte Carlo estimation to determine the number of times needed.

- We determined that the robustness value of one graph should be the mean value computed over 50 extinction sequences (each sequence randomly picks providers that go extinct). Monte Carlo methods showed that, after 50 sequences, the variance of a new robustness value was below 0.05%
- We run each simulation over 50 different initial graphs and determined that, after 50 graphs, the variance for results from a new simulation was below 0.01% .

Results

RQ1. Can local adaptation lead to the emergence of global robustness in software systems?

This was the key research question for our work. We evaluated the impact of the different adaptation strategies on the evolution of global system robustness. The "random-random" strategy was used as a baseline, mimicking how software systems currently do adapt. All of the adaptation strategies were subject to the same global constraints in the model; `i.e.` all consumers remain satisfied and the cost of the model (total number of services provided and total number of links) is kept constant.

Fig. 4 shows the evolution of the robustness of the global system for the four strategies. This plot was obtained by averaging the results from 50 runs of the simulator on 50 randomly generated initial models. Table 1 presents the values for the average final robustness for the different strategies and their standard deviation.

These results show that, the localized adaptation operators had a positive impact on the global robustness of the system when only one class of nodes(Providers and Consumers) was utilizing them (rows 2 and 3 in Table 1). More interestingly, the hybrid approach of all nodes utilizing their localized adaptation operators simultaneously(row 4 in Table 1) resulted in a much higher benefit than when only one was. This shows that there is a synergy between those decentralized strategies and this is in favor of the global robustness

[5] `http://www.alexa.com/topsites`

[6] Wordpress data available here: `http://diversify-project.eu/wordpress/`

Adaptation Strategy		Robustness	
Consumer	Providers	Average	Std. Dev.
Random	Random	33.4%	0.896
Equitable	Random	37.7%	1.084
Random	Popular	35.9%	1.018
Equitable	Popular	46.2%	1.584

Table 1: Final robustness of the global system with four different adaptation strategies (observations on 25 runs)

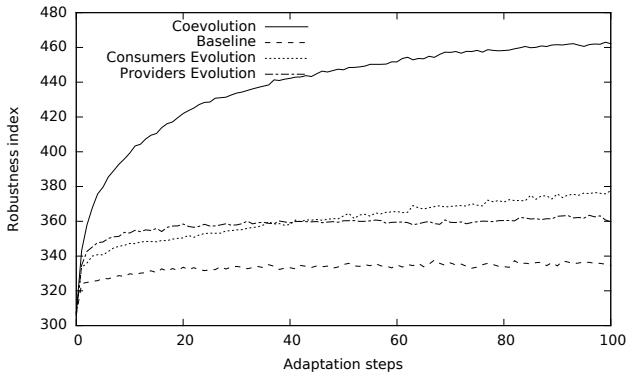

Figure 4: Evolution of globalized robustness using different decentralized adaptation strategies

of the system. This result is interesting because the strategies used were only based on local knowledge and target local optimizations for individual Providers and Consumers. There was no explicit push for these individual strategies to have a positive effect on the global robustness of the system.

Although the results from the simulations described so far are encouraging, we can only conclude from them that global robustness can emerge in theese specific simulation scenarios. In practice, a number of simplified decisions were made in the simulations. Firstly, the Providers and Consumers were constant. A real world system is more dynamic with Providers and Consumers constantly appearing and disappearing and becoming either temporarily or permanently unavailable. Secondly, the distribution of Providers and Consumers and the distribution of Services were constant but may vary in real life. We decided to investigate the results of the simulation in the absence of such simplifications and also investigated if the results would hold for systems of a different scale. The next 3 research questions discuss these investigations.

RQ2. How sensitive are the adaptation rules when relaxing topological constraints of the software system?

This question focuses on the impact of the hard constraints of previous experiments (fixed size and topology of graph) on our results. We ran a set of experiments where Consumers and Providers evolved randomly (they could go extinct, be mutated, reproduce or be cloned allowing some new entities to appear) in parallel to the adaptation strategy. Reproduction, cloning and extinction probabilities were set at 30%. New Consumers were produced by crossover of two ancestors with the new node containing a sub-set of the services of its ancestors required. New Providers were created using one of the two evolution strategies described below.

Providers' random evolution strategies : New Providers are created by cloning an existing one and mutating the list of provided Services (add a new Service or remove one). Service mutation probability was set at 20%.

Baseline Random adaptation with random evolution of Providers.

Equitable popular Equitable-Popular adaptation with a random evolution of Providers.

Providers' ecological evolution strategies : This improves the random evolution strategy such that, when cloning a Provider, the mutation factor is no longer a static probability. The mutation rate of a provided Service depends on its success among the Consumers connected to it: the more successful a service, the greater the probability it will also be provided by the clone. This strategy introduces a form of environmental pressure in the adaptation, with the aim of creating offsprings that are more fit than their ancestors.

Baseline (ecology). Random adaptation with an ecological evolution of Providers.

Equitable popular (ecology). Equitable-Popular adaptation with an ecological evolution of Providers.

Fig. 5 shows the results for 4 experiments which combine the two adaptation strategies (Random and Equitable-Popular) with the two alternative evolutionary strategies: a fully random one and an ecology inspired one, which is shown to bring improvement slower than the original, but eventually to surpass it.

The impact on robustness was similar to the previous, more constrained, experiments. The main difference is that the rate of convergence is slower: the results of Fig. 5 are for 500 combined adaptation and evolution steps while on Fig. 4 only 100 steps were necessary when no evolution was simulated.

RQ3. How does the distribution of services among consumers influence the results?

The distribution of services among the Consumers is a critical parameter of the simulation. In all previous experiments, the size of Consumers followed a poisson distribution of parameter 5.47 and the Service usage, a power law distribution of parameter 2.08. Since these distributions were tuned by observing real bipartite software interactions, we are confident about their relevance. Yet, the exact parameters of

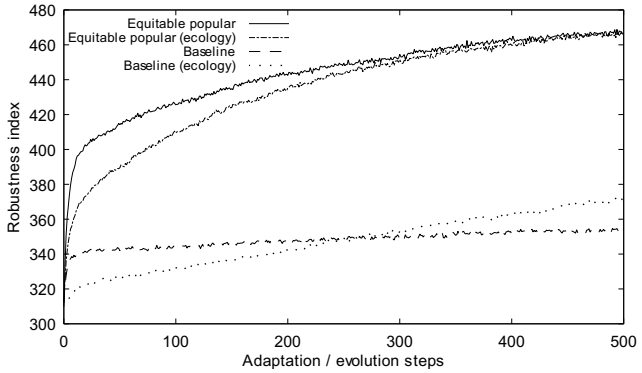

Figure 5: Evolution of the overall robustness using different decentralized adaptation strategies in parallel with different system evolution.

those distributions will vary from one situation to the next (and even possibly over time).

This question addresses the sensitivity of the results with respect to these distributions. In order to do answer it, we replicated the simulation presented on Fig. 4 using graphs generated from different statistical distributions.

Table 2: Results for different initial distributions of nodes sizes and Services usage. P(λ) for a Poisson distribution, N(μ, σ) for a normal distribution, E(λ, min) for a power law with a minimum value of min and NE(λ, u) for a Negative Exponential distribution with an added constant u.

D_Size	D_Services	Init.	Random	Eq-Pop	Delta
P(3)	NE(0.25,0.005)	35.9	44.8	47	**2.2**
P(6)	NE(0.25,0.005)	39.8	33.3	46.9	**13.6**
P(6)	NE(0.25,0.001)	35.5	39.1	51.9	**12.8**
P(6)	NE(0.25,0.01)	28	31.2	42.9	**11.7**
P(6)	UNIFORM	21.6	27.7	32.4	**4.7**
P(9)	NE(0.25,0.005)	27.7	27.8	44.5	**16.7**
E(8,1)	NE(0.25,0.005)	37.6	38.9	47.8	**8.9**
E(15,3)	NE(0.25,0.005)	32.6	34.4	47.5	**13.1**
N(5,5)	NE(0.25,0.005)	33.4	35.3	47.7	**10.4**

Table 2 presents the results of the simulations. The first and second columns liindicates the parameters for the distribution of Consumer size (D_{size}) and Service usage ($D_{services}$) respectively. The next columns list the robustness value of the initial network ($Init.$) and of the final network evolved with the baseline random ($Rand.$) and the equitable-popular ($Eq - Pop.$) strategies. The last column indicates the difference in final robustness between both adaptation strategies. The results show that the guided strategy always performs significantly better than the baseline. The two lowest results (row 1 and 5) correspond to extreme situations. In row 1, a P(3) distribution makes the average size of Consumers 3, with a good portion of Providers and Consumers of 1 or 2. In that case, the exploration space is very much reduced and the impact of the guided strategy is limited. In row 5 the Services are chosen using a uniform distribution: the nodes are provided with sets of Services that are completely independent from one another. When using a power law (all other rows in the table), the sets of Services are related and more similar. In practice, the uniform distribution is unrealistic: there always are very popular micro-services used by everyone (e.g., Google Analytics).

RQ4. What is the effect of the system's scale?

To assess the generality of the results beyond the fixed network size used in previous experiments (300 Consumers, 100 Producers and 50 Services), we repeated them at different scales and using different ratios between the values. All experiments have shown a clear benefit of the Equitability-Popular strategy over the baseline with various degrees. For example, repeating the experiments with a number of 5000 Consumers and 1000 Providers yielded an initial robustness of 40.3 and a final average robustness of respectively 43.3 and 52.2 for the baseline and Equitability-Popular. With respect to scalability and sensibility to the different parameters, the Equitable-Popular strategy was always significantly better than the random strategy.

Related works

Lim and Bentley (2012) propose an ALife agent-based model to simulate the evolution of an ecosystem of software apps on an app store. They model app developers, app users and apps as agents and experiment with different behaviors for developers. Their goal is to determine what are the best strategies for app developers in order to create successful apps. They manage to identify a combination of strategies and frequency for each strategy with which they can reproduce the development Apple's iOS app ecosystem. Cocco et al. (2014) have recently provided similar modeling to understand what strategies should be adopted by application developers. While these works adopt a model similar to our, their goal is different: understand good strategies for developers in their case, model good adaptation strategies for global robustness in our case.

Goings et al. (2012) present an ecology-based evolution algorithm that produces good behavioral models for complex software systems. They show that adapting ecological principles for evolution produces models of software that are more diverse and evolvable than the models produced by a more classical evolutionary algorithm. Similarly to our simulation, their evolutionary algorithm evolves models in a very constrained environment (fixed population and limited resources), which eventually pushes towards more diversity and good properties for future evolutions.

Another area of related work is the simulation of service-based systems (Wang et al., 2010; Kaur et al., 2013). This is a very active area as shown in the literature reviewed by

Smit and Stroulia (2013). They review 6 major frameworks for the simulation of service-oriented systems. Simulation are either based on the abstract description of the services or on more formal descriptions that rely on discrete-event semantics. These different models support different tasks such as testing, interaction vizualization, code generation or performance prediction. However, none of these frameworks evaluates the impact of perturbations, nor simulates decentralized evolution. Other works have evaluated the ability of service-oriented systems to sustain perturbations, such as denial-of-service attacks (Xu and Lee, 2003), but did not reason about the global dynamics of the system to establish robustness.

Conclusion

In this paper we have proposed EVOSERV, a simulation model to experiment with decentralized adaptation strategies in distributed software systems. This simulation model is calibrated with data from actual software models. We have proposed a set of guided decentralized adaptation strategies and evaluated them against a random baseline in a set of different simulations. The results indicate that the combinations of the proposed Equitable and Popular local strategies significantly increases ability of the system to sustain the extinction of service Providers (i.e., called robustness in this paper). These results mark a significant first step towards the transposition of the ecological *Red Queen hypothesis* into actual software systems. Future work include two main threads: analyze topological properties of the graph over the simulation to characterize the phenomena that favor robustness emergence; deploy these behaviors on a controlled SOA system.

In the future, developers of platforms that provide microservices and of service mash-ups that aggregate services could enhance their products with simple decision making strategies, which benefit the local nodes and participate in a system-wide improvement of robustness.

Acknowledgements

This work is partially supported by the EU FP7-ICT-2011-9 No. 600654 DIVERSIFY project.

References

Albert, R., Jeong, H., and Barabási, A.-L. (1999). Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131.

Bertolino, A., Blake, B., Mehra, P., Mei, H., and Xie, T. (2015). software engineering for internet computing. *IEEE Software*, 32.

Brännström, A., Johansson, J., Loeuille, N., Kristensen, N., Troost, T. A., Lambers, R., and Dieckmann, U. (2012). Modelling the ecology and evolution of communities: a review of past achievements, current efforts, and future promises. *Evolutionary Ecology . . .* , 14:601–625.

Burgos, E., Ceva, H., Perazzo, R. P. J., Devoto, M., Medan, D., Zimmermann, M., and María Delbue, A. (2007). Why nestedness in mutualistic networks? *Journal of theoretical biology*, 249(2):307–13.

Cocco, L., Mannaro, K., Concas, G., and Marchesi, M. (2014). Simulation of the best ranking algorithms for an app store. In *Mobile Web Information Systems*, pages 233–247.

Forrest, S., Balthrop, J., Glickman, M., and Ackley, D. (2002). Computation in the wild. In Park and Willinger (2005).

Goings, S., Goldsby, H., Cheng, B. H. C., and Ofria, C. (2012). An ecology-based evolutionary algorithm to evolve solutions to complex problems. In *Proc. of the Int. Conf. on the Simulation and Synthesis of Living Systems (ALife)*.

Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *Internet Computing, IEEE*, 9(1):75–81.

Kaur, N., McLeod, C., Jain, A., Harrison, R., Ahmad, B., Colombo, A. W., and Delsing, J. (2013). Design and simulation of a soa-based system of systems for automation in the residential sector. In *Proc. of the International Conference onIndustrial Technology (ICIT)*, pages 1976–1981.

Lim, S. L. and Bentley, P. J. (2012). How to be a successful app developer: Lessons from the simulation of an app ecosystem. *SIGEVOlution*, 6(1):2–15.

May, R. M. (1972). Will a Large Complex System be Stable? *Nature*, 238(5364):413–414.

Park, K. and Willinger, W. (2005). *The Internet as a large-scale complex system*, volume 3. Oxford University Press.

Rossberg, a. G., Matsuda, H., Amemiya, T., and Itoh, K. (2006). Food webs: experts consuming families of experts. *Journal of theoretical biology*, 241(3):552–63.

Smit, M. and Stroulia, E. (2013). Simulating service-oriented systems: A survey and the services-aware simulation framework. *Services Computing, IEEE Transactions on*, 6(4):443–456.

Staniczenko, P. P. A., Lewis, O. T., Jones, N. S., and Reed-Tsochas, F. (2010). Structural dynamics and robustness of food webs. *Ecology Letters*, 13(7):891–899.

Tilman, D. and Downing, J. A. (1994). Biodiversity and stability in grasslands. *Nature*, 367(6461):363–365.

Van Valen, L. (1973). A new evolutionary law. *Evolutionary theory*, 30:1–30.

Wang, W., Wang, W., Zhu, Y., and Li, Q. (2010). Service-oriented simulation framework: An overview and unifying methodology. *Simulation*, page 0037549710391838.

Xu, J. and Lee, W. (2003). Sustaining availability of web services under distributed denial of service attacks. *IEEE Transactions on Computers*, 52(2):195–208.

Yachi, S. and Loreau, M. (1999). Biodiversity and ecosystem productivity in a fluctuating environment: the insurance hypothesis. *Proceedings of the National Academy of Sciences of the United States of America*, 96(4):1463–8.

Yodzis, P. (1981). The stability of real ecosystems. *Nature*, 289:674–676.