2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems

Clonal Plasticity: A Method for Decentralized Adaptation in Multi-Agent Systems

Vivek Nallur, Nicolás Cardozo, Siobhán Clarke
Future Cities, DSG, Trinity College Dublin
College Green, Dublin, Ireland
{vivek.nallur | cardozon | siobhan.clarke}@scss.tcd.ie

ABSTRACT

This paper introduces a new plant-inspired mechanism for decentralized adaptation, called *clonal plasticity*, which does not make use of the MAPE loop. The mechanism acts on every individual in a population and does not require centralized control. The paper presents a case-study demonstrating its utility and feasibility in adaptation. The paper concludes with some thoughts about the costs and limitations of this mechanism, and possible future strands of research in this direction.

CCS Concepts

•**Computing methodologies** → *Self-organization;*

Keywords

decentralized self-adaptation; plasticity;

1. INTRODUCTION

Technologically, society is moving towards an ever-increasing number of connected devices, which are highly sophisticated in terms of computing power. Concepts like *Smart Cities* envisage a world, where systems from multiple domains, currently silo-ed in their particular function, interact in an integrated manner to provide citizens with efficient and sustainable infrastructural support in transportation, energy, water, etc [23, 7]. Other concepts like *Internet of Things* envisage a hyper-connected world, where devices, hitherto considered 'dumb', would be able to connect to other devices and make intelligent decisions, even in the face of changing human needs [26, 29]. All of these concepts assume that the underlying technological systems and infrastructure are able to adapt to environmental change and preserve their function [3]. Thus, self-adaptive systems are a foundational necessity to the realization of such techno-fused societies. Self-Adaptive systems are those that are able to preserve at least one of the so-called self-* properties (self-protection, self-healing, self-configuring, self-optimizing, etc.). This has traditionally been done using the MAPE-K loop [17]. However, this is difficult to scale for highly distributed systems, since by the time the *monitoring, analysis, planning* and *execution* steps are performed, the original problematic environment could have already changed. Also, a centralized MAPE-K loop based self-adaptation makes little sense in a Multi-Agent System (MAS) where each agent is not only potentially autonomous, but also has potentially different goals. MASs are typically used in situations where distributed, autonomous decision-making is a natural fit. These situations include modeling city traffic [8], socio-technical systems [22], smart-grids [14, 20], smart-parking systems [24], cooperative control [25], etc. Most multi-agent system based adaptation relies on the distributed and autonomous nature of the constituent agents [11], but there is little literature on how to configure the agents themselves. Applications such as collision avoidance [6], grid management [16] use collaborative learning, and hierarchically deferring to a controller agent, to achieve the application's objectives. The most popular technique of self-configuration in multi-agent systems is reinforcement-learning [4]. However, in competitive situations where collaboration is not possible, these techniques are not readily applicable. In this context, Prof. David Garlan presented a talk at SEAMS'15 entitled "*The MAPE-loop considered harmful*", where it was speculated that a MAPE-less adaptation technique could potentially be useful in multiple scenarios. *Coker et al.* [9] also posited that large-scale software adaptation could be enriched by techniques from other domains, such as stochastic search. We wish to position our paper in this line of thought, and present a plant-inspired adaptation technique called *clonal plasticity*. Many plants exhibit adaptive behavior through the use of **phenotypic plasticity**, where the immediate environment influences the height of the stem, width of the leaves, size of the roots, etc. Another phenomenon that occurs in plants is **clonal reproduction**, where all the individual plants in a given location are genetically identical copies of a single ancestor, without any sexual reproduction. We combine these two features to create a strategy called *clonal plasticity*. In the following sections, we first define what we mean by clonal plasticity for a software (Section 2), show through a case-study (Section 3), its utility in decentralized adaptation. Finally, we reflect from a software engineering perspective (Section 4) on the costs and limitations of using such an adaptation strategy, before concluding the paper.

SEAMS'16, May 16-17 2016, Austin, TX, USA

ISBN 978-1-4503-4187-5/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2897053.2897067

2. *CLONAL PLASTICITY*

In clonal reproduction, a 'daughter' plant is formed "through

sequential reiteration of a basic structural unit or module consisting of the leaf with its associated auxiliary bud and a segment of stem that connects it to other units" [13]. Basically, the individuals that are formed after reproduction are genetically identical to the original parent. This form of reproduction (*clonal reproduction*) has long been known to botanists as a very successful survival strategy. *For instance,* one aspen tree has been known to cover 43 hectares of land in Fish Lake County, Utah, with around 47000 clones, with an average age of 100 years [10]. It is important to note that though these clones are genetically (and sometimes physically) connected, they function as independent organisms. Also, in spite of an identical genome, they do not necessarily look or behave identically. This variation in structure and/or behavior is due to a phenomenon called *phenotypic plasticity*. Plasticity, in this context, is the ability of an organism to vary its structure (height of stem, width of leaves, etc.) or behaviour (connect to its clonal sibling or disconnect) according to environmental influences.

Phenotypic Plasticity is shown by the genotype of an organism when its expression is able to be altered by environmental influences [2]. Plasticity, by this definition, does not include variation in an organism due to genetic factors.

We combine these two phenomenon, exhibited in plants, to propose a self-adaptation mechanism called **clonal plasticity**. The fundamental difference between *clonal plasticity* and other evolutionary mechanisms, such as *genetic algorithms*(GA), is the absence of a selection function, that identifies fit and unfit individuals. Clearly the absence of a selection function, from an adaptation perspective, implies that a system consisting of many individually adapting clones would adapt slower than a system implementing a GA. However, in some uncertain environments, it is better to approach a good solution in graduated steps, than in a fast convergence. Another feature of clonal plasticity is that all of its adaptations are reversible. This may not be the case with a genetic algorithm. Depending on the sequence of steps taken, a GA might not be able to revisit a certain solution, since unfit individuals containing a portion of the solution may have been previously completely removed from a population. In the general case, it is not possible for any adaptive mechanism to guarantee that the adaptation step will definitely lead to a better situation than the current one. However, a good adaptive mechanism must recognize that sometimes environments change fast, and therefore changes made to a system must be reversible.

2.1 Pre-requisites

A pre-requisite for clonal plasticity is plasticity in the genome. That is, depending on environmental feedback, the genome must be able to express behavioral changes. Note, this does not mean that the genome itself is changed, but merely that the organism responds behaviorally to changes in the environment. This is in contrast with other mechanisms that have been studied under the umbrella of *evolutionary computing*. Whether by schemes of *mutation* or *crossover*, evolutionary computation makes changes in the underlying genome of the organism, whereas with clonal plasticity, the genome remains unchanged.

From a software system's perspective, this feature of clonal plasticity means that functional goals of a plastic system will continue to be met in the same manner, as the original system. For certain domains (*e.g.,* critical systems [5]), the guarantee of functional correctness is an important consideration.

2.2 The Process of Plastic Reproduction

In clonal plasticity, the principal method of reproduction is *clonal*, which means that child organisms are substantially similar to their parent organism. The differentiation between a parent and a child occurs due to environmental influence, and the degree of plasticity exhibited by the organism. The process of plastic reproduction, depicted in Figure 1, is given by the following steps:

1. **Identify Plasticity Points**: At a given moment in time, `t`, of the system's lifetime, each individual in the system (depicted a circle in Figure 1) identifies all its parts that can take multiple representations, values, or behaviour.

2. **Evaluate Environmental Input**: Simultaneously, the fitness of each individual is self-evaluated in the following way: for each plastic point, environmental feedback is evaluated to check whether it impedes or favors that point. Feedback from the environment may come from different sensors or neighboring individuals or chemotaxis, or the individual's own reproduction memory (`m` in Figure 1).

3. **Plasticity Memory**: All individuals that have reproduced before, keep a record of the previous Plasticity Range chosen during the last reproduction cycle (variable `m` in Figure 1). This allows an individual to repeat a good adaptation strategy, if it previously had a positive reward, or alternatively abandon a poor adaptation strategy.

4. **Choose Plasticity Range**: During the following time-step (`t+1`), each individual chooses a plastic response, based on the evaluated environmental input. The available responses may be one of: (1) Exact Clone, (2) Low Plasticity, and (3) High Plasticity.

5. **Clone and Modify Plastic Points**: During the same time-step, each individual makes a clone of itself. The individual's plasticity points may be adapted according to the chosen strategy. The entire process starts again with a new time-step, $\mathtt{t}'$, on all clones.

Choosing the Plasticity Range:.

The process of modifying the plastic points considers both positive and negative feedback. For a given time instant `t`, the adaptation decision depends on two factors:

1. Feedback from environment at time `t`, whether it is positive or negative.

2. Memory of the modification action, taken at time `t-1`.

If the feedback from the environment is positive, and memory of the previous action is also positive, then the Plasticity Range is *Exact Clone* If the feedback from the environment is positive, but the memory of the previous action is negative, then the Plasticity Range is *Low Plasticity*. If the feedback from the environment is negative, then the Plasticity Range is always *High Plasticity*

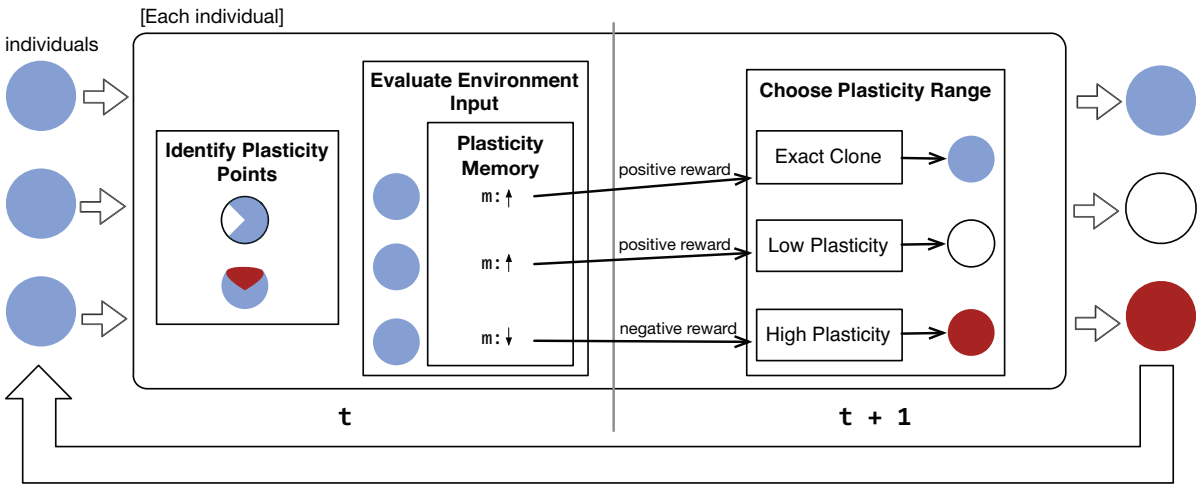


Figure 1: Clonal plasticity process

Exact Clone.
In this case, the individual simply makes a copy of itself, with no change at all.

Low Plasticity.
Modification with Low Plasticity implies that the individual chooses the same plasticity point, as chosen during the previous cloning process, and moves in the same direction as before.

High Plasticity.
Modification with High Plasticity implies that the individual chooses a different plasticity point, than was chosen during the previous cloning process, and moves in a random direction.

Presence of Competition from Other Agents.
Depending on the particular system, plasticity can also accommodate competition from other agents. If the feedback from the environment is negative, and the memory of the previous action is also negative, then that agent is deemed to be too weak to clone itself. Rather, it is 'taken over' by any random neighbouring agent that is stronger due to a positive feedback from its environment.

2.3 Differences from a GA

Clonal Plasticity is still an instance of an evolutionary process(since it is inspired from plants). However, there are some important differences between Clonal Plasticity and well-known evolutionary techniques, like a Genetic Algorithm. These can be summarized as follows:

1. *No Selection*: Plasticity acts on every individual and the degree of plastic change exhibited is solely dependent on the individual's fitness in its local environment. This raises the possibility of an individual being *fit* in one part of the environment, and being *unfit* in another part of the environment. The hyper-local environment for fitness and the absence of a selection function completely decentralizes the change process.

2. *No change to genome*: Phenotypic plasticity occurs due to environmental conditions (this may be neighbouring individuals or physical/temporal environment), but there is no change in the genotype —that is, only parametrizable changes are allowed. This implies that the boundary of change is well-defined.

3. *Slow change*: Unlike conventional genetic or evolutionary algorithms, the new 'child' is a faithful copy of the parent, and hence, the amount of change that can be exhibited from generation to generation is very little.

4. *No objective function*: The absence of an objective function means that for a long-lived system, as the environment changes, plasticity will cope with the changes as well. Also, the system designer does not have to inject an artificial stopping condition, since the system, in the aggregate, remains largely stable in the absence of change.

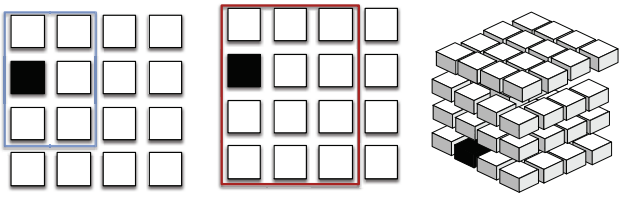
By its very nature, clonal plasticity is well-suited to systems where there are a large number of distributed, functional units, each requiring adaptation that is local to its particular environment. It is also suited to multi-agent systems, where each agent could potentially adapt on its own terms. To evaluate the effectiveness of plasticity, we use a case study that showcases adaptation in the face of environmental change (Section 3).

3. PLASTIC ADAPTATION CASE STUDY

This section showcases the utility and feasibility of a plasticity model to realize decentralized adaptation, by implementing an adaptive version of Conway's Game of Life (GoL). We use GoL, as its dynamics are readily known, and the application can be easily decomposed into multiple individuals, each reacting to adaptation scenarios from the environment. The purpose of this case study is to show how individuals in a system can adapt their behavior with respect to the their own perception of the surrounding execution environment, *i.e.,* adaptation is not uniform for all individuals, but rather is heterogeneous. The objective for each individual to adapt, is to offer the most advance game experience with respect to the surrounding environment. In order to adapt each individual's behavior, we follow a clonal plasticity approach,

where individuals adapt immediately after they have sensed a change in their surrounding execution environment.

The initial setting for our case study is that of the regular GoL, *i.e.,* a two dimensional grid of cells, each cell being in one of two possible states, alive or dead. The game progresses by changing the state of each cell with respect to the state of their immediate neighbors. Our version of GoL is decentralized, detaching cells from a central grid arrangement. Rather, as shown in Figure 2a, each cell is a free floating individual that finds its neighbors according to their proximity (given by cells' position). Cells use a 1-radius neighborhood to evaluate game progress (*e.g.,* demarcated by the blue square centered at the live cell in the figures). Each individual cell in the game can react to two changes in their surrounding execution environment, leading to different adaptations in its behavior. (1) The first environment change, causes cells to use larger neighborhoods to update their state as the game progresses. This is shown in Figure 2b for a neighborhood of radius 2 (demarcated by the red square center around the alive cell). (2) The second environment change, causes cells to transcend into a three--dimensional space. This is shown in Figure 2c, where all cells are adapted to live in a three-dimensional space.



(a) Decentralized GoL (b) Large neighborhoods adaptation (c) 3D adaptation

Figure 2: Game of Life set-up and adaptations

3.1 Implementing Decentralized Adaptations

The two aforementioned adaptation scenarios for GoL are implemented using the adaptation mechanisms proposed in Context-oriented Programming (COP) [15], to adhere to a plastic reproduction mechanism. In particular, we use the Context Traits [12] language for our implementation. Context Traits enables the behavioral adaptation of a software system in response to input from its surrounding execution environment. COP is chosen as an implementation mechanism since it satisfies the five steps of the plastic reproduction process introduced in Section 2.2.

1. Our adaptation scenarios affect the interaction between cells. Therefore, the Plasticity Points for all cell individuals in GoL are identified precisely as those points where cells interact with each other (*i.e.,* the `countAliveNeighbors()`, `livenessConditions()`, and `step()` functions). In Context Traits such adaptations are defined as stand-alone behavior units, defined as shown below for the large neighborhood adaptation as an example. The two outermost loops in this adaptation search for cells in a 2-radius neighborhood, while the inner loop, goes over all cells discovered in the environment. If there are alive cells in the neighborhood, then these are counted.

```
LargeNeighborhoodCell = Trait({
  countAliveNeighbors: function() {
    var count = 0;
      for(dx=-2; dx<= 2; dx++){
        for(dy = -2; dy <= 2; dy++) {
          if(dx == 0 && dy == 0) {}
           else {
             for(i=0; i<cells.length; i++) {
               if(this.x+dx == cells[i].x &&
                   this.y+dy == cells[i].y &&
                   cells[i].state) {
                 count ++;
               }
             }
           }
        }
      }
    return count;
  }
});
```

2. Program entities (*i.e.,* object instances) sense their environment and evaluate whether their Plasticity Points (*i.e.,* behavior) requires adaptation. If this is the case, then adaptations are enacted immediately. In our example, the need for adaptation is signaled to an object instance via a *context* object. Contexts associate object instances with their behavior adaptations, for example, the large neighbourhood adaptation requires a new $\texttt{LargeNeighborhood}_{i,j}$ context is created, associating the context to a $\texttt{cell}_{i,j}$ instance, and the behavioral adaptation defined previously, as follows

```
var LargeNeighborhood_{i,j} = new cop.Context({
   name: 'LargeNeighborhood'_{i,j}
  });

LargeNeighborhood_{i,j}.adapt(cell_{i,j},
    LargeNeighborhoodCell);
```

3. In our implementation, individuals do not keep an active record of their last adaptation taken. Rather, each individual is aware of all its adaptations at all time. However, these previous actions do not influence individuals' behavior, and Environmental Input is gathered from external sensors and self.

4. When adapting an object instance, in our COP model, we always choose a low plasticity strategy. That is, adaptations are always assumed to provide the best possible behavior, and hence provide a positive effect, for the object instance with respect to its surrounding environment.

5. Adaptations have an instantaneous effect on the object instances they modify, *i.e.,* immediately after the context signals the adaptation, by calling the **activate()** construct on the context (*e.g.,* `LargeNeighborhood.activate()`), the object instance associated with such context will immediately use the behavior adaptation associated with the context.

   Also, similar to the introduction of adaptations by means of context activation, in our approach, adaptations can be reverted by withdrawing adaptive behavior from instance objects using the **deactivate()** construct on the context (*e.g.,* `LargeNeighborhood.deactivate()`). Reverting adaptations also has an immediate effect on object instances, withdrawing the corresponding behavior from the Plasticity Point.

3.2 Environment Dynamics and Experimental Set-up

To show the feasibility of the plasticity adaptation mechanism for decentralized environments, we run a GoL consisting of n^2 cells (*i.e.,* individuals).[1] At any moment in time, any of the individuals can engage in any of the adaptations, if it is so required in the surrounding environment.

Our GoL implementation has three Plasticity Points, the `countAliveNeighbors()`, `livenessConditions()`, and `step()` functions. The first two functions are behavioural adaptations in the `LargeNeighborhood` context, while the first and last functions are behavioural adaptations in the `3D` context.

Figure 3 shows the possible adaptation scenarios for our GoL implementation. Starting from the top, the game progresses as stipulated by the original rules, where each cell uses a neighborhood of radius 1 (*e.g.,* blue square centered on the alive cells in Figure 3) to verify the conditions to progress to the next step. From this stage the environment can influence change in two ways, the rules change to include larger neighborhoods (moving to the game on the lefthand side of Figure 3), or the rules change to work on a three-dimensional space (moving to the right-hand side of Figure 3). Note here that, no matter which of the adaptations is taken, this is not a uniform change for all individuals, but only those cells that sensed the change in the surrounding environment adapt. For example, in Figure 3, in the first case only the leftmost live cell adapted to take into account large neighborhoods in its game rules, while, in the second case, five cells adapted to a three-dimensional space. Cells that have adapted, have two choices for a new adaptation, they can either revert the adaptation taken, or they can incorporate the other adaptation, in response to their environment. If the environment is sensed to revert the adaptation, then the game will revert to its original state. In the case of cells that adapted to large neighborhoods, if they now adapt two three-dimensional spaces, then the game will behave as in the bottom of Figure 3. In the case of cells that adapted to a three-dimensional space, if they now adapt to large neighborhoods, then they will also behave as in the bottom of Figure 3 *i.e.,* a three-dimensional space with neighborhoods of radius 2. From such state, either of the adaptations can be reverted, going back to the adaptation signaled by the environment. For example, if the large neighbourhoods is reverted, the behavior will be that of the three-dimensional space.

We executed GoL in two different settings to ensure its usability, and the feasibility of our approach. The first setting consisted of controlled adaptations of the game. That is, we implemented two fixed scenarios in which contexts will be activated and deactivated deliberately to verify that the game behaved correctly for each adaptation scenario as well as a the combination of both. In this setting, at every step the game progressed according to its rules (*i.e.,* regular GoL rules and the adapted ones), and when reverting all cells to the original setting, the game continued working normally. The second setting consisted on an "infinite" run of the game, in which every two seconds each cell would sense the environment and adapt according to the sensed scenario. The cells that sense the adaptation scenario are

[1]Our implementation of GoL and the simulation experiments are available for download at https://bitbucket.org/viveknallur/seams2016.git.

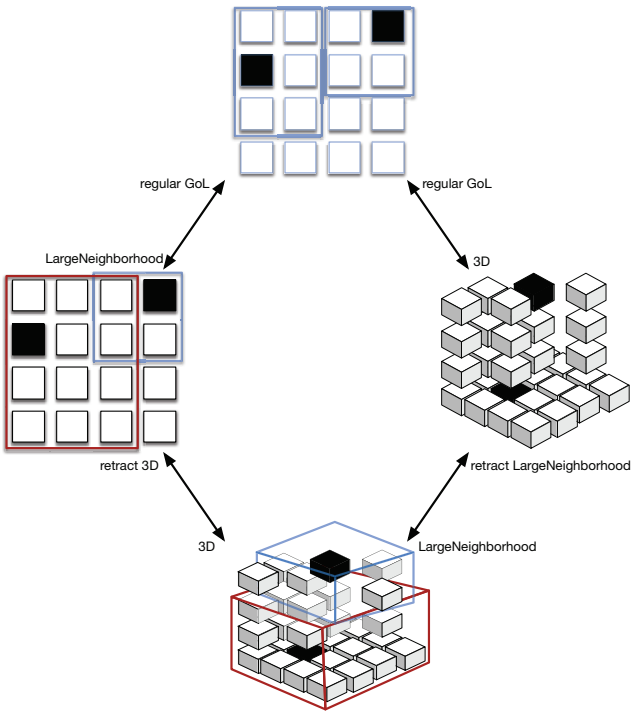


Figure 3: Adaptation scenarios for GoL

chosen randomly, but they all adapt to the same scenario, *i.e.,* either large neighborhoods, or three-dimensional space. After this adaptation step, all cells take a step according to the rules currently applying to them, and then the environment is sensed again.

We see that using the process of clonal reproduction along with plasticity in behaviour allows independent cells to adapt to changes in the rules of GoL. Implementing these kinds of adaptations for an individual cell simplifies the adaptation reasoning, since we do not have to account for the entire system as a whole. Performing a MAPE-K based adaptation for such a system, where the environment can signal change of neighborhood size on one part of the system, or three-dimensional space at another part of the system would have been very difficult. The analysis and planning would have had to deal with multiple conflicting signals from the monitoring aspect of the adaptation framework.

3.3 Clonal Plasticity Beyond GoL

While GoL serves as an example to show the feasibility of using clonal plasticity for the adaptation of software systems in a decentralized fashion, this application also showcases the applicability of this approach to larger and more complex scenarios.

Note that the transition from small scale scenarios as GoL to larger decentralized application domains in smart cities or IoT environments is immediate from the programmers' perspective. As shown with GoL, each individual agent defines the situations from the surrounding environment it responds to (*i.e.,* contexts) and the fine-grained adaptations to interact with the new environment. Such development approach would still stand for larger case studies. Each agent in the system is implemented associating their corresponding adaptations without any additional infrastructure

or required knowledge about other agents. Furthermore, as there is no coordination or synchronization required between agents, the scale of the system will not affect the adaptation time for the individual agents.

4. SOFTWARE ENGINEERING PERSPECTIVE

The benefit of plasticity is the ability to produce better phenotype-environment matches across more environments, than would be possible by producing a single phenotype in all environments [18]. However, there are some costs to be incurred, as with all adaptive mechanisms, and some limitations that accrue to a plastic mechanism.

4.1 Costs

1. *Maintenance Costs*: Although fitness for purpose is usually considered at design-time, creating a continual sensory mechanism for receiving feedback from the environment, induces maintenance costs.

2. *Production Costs*: In memory-constrained environments, the binary footprint of a particular piece of software could play an important role. In such deployment scenarios, creating a regulatory mechanism to sense feedback and then adjust parameters incurs a cost in terms the of size of the deployed code.

3. *Information Acquisition Costs*: The process of sensing the environment uses up CPU cycles and memory, which may/may not be significant, depending on the domain.

4. *Development Costs*: From a human perspective, adding a plasticity mechanism, while less cumbersome than other adaptation mechanisms, is still a developmental addition. Any developmental addition also has knock-on effects on testing and validation steps in the software development life-cycle.

4.2 Limitations

1. *Information Reliability Limit*: Depending on the granularity of the sensory/feedback mechanism, the precise environmental cue for plastic adaptation may not be available. This is dependent more on the domain, and less on the ability of the software. However, the adaptation performed by the plastic response might not be correct at all times. This could lead to see-saw-ing of the plastic response, which may be counter-intuitive to the system-maintainer.

2. *Lag-time Limit*: A system can only adapt as quickly as it can detect a relevant change in its environment. Depending on the implementation of the feedback mechanism and frequency of cloning, plasticity's adaptive response may not be in sync with the environment.

It must be noted, that all of these costs and limitations are not exclusive to Clonal Plasticity, but rather are valid for any adaptation mechanism, including those that use the MAPE-K loop.

5. RELATED WORK

There has been a plethora of work on decentralized adaptation [27, 28] and the use of natural metaphors [21, 19] for such goals. Our work falls squarely within this field of using nature-inspired mechanisms. Other work such as *Morphogenetic engineering* also draws its inspiration from the self-organized-yet-architected natural systems, however their emphasis is on the architectural properties enabled through self-organization and self-assembly processes. These can also accommodate adaptation to environmental changes, but the focus is on the complex structure that is autonomously created and sustained. In this paper, we examine a more immediate goal of adaptation, that of responding quickly to changes in the environment. Philosophically, this paper is closest to the notion of *functional blueprints* [1]. *Beal* [1] recognizes that " stress-tolerant system can exploit its tolerance to navigate dynamically through the space of viable designs". Clonal Plasticity exploits this very tolerance to *differentiated-but-good-enough* functionality to adapt to the changing environment.

6. FUTURE WORK AND CONCLUSION

We propose clonal plasticity as an alternative to existing adaptation mechanisms. In our experiment, we demonstrated the feasibility and performance of the approach. Nonetheless, to establish the efficacy of clonal plasticity, it needs to be implemented and evaluated in multiple settings, and in multiple domains. We expect that certain domains would be a more natural fit, than others, especially those that require self-configuration of parameters. In the longer term, we would like to establish a taxonomy of problems which allows to systematically pick, whether to use clonal plasticity or some other self-adaptive mechanism. In doing so, we hope to enlarge the self-adaptation toolkit available to system designers.

In this paper, we have introduced a new mechanism for adaptation inspired by plants, which is completely decentralized and does not require an objective function. This allows a system that is clonally plastic to evolve at runtime, without the need for a human to oversee its performance over the long-term. We have shown its usability and feasibility for adaptation to environmental changes. We further discussed potential costs and limitations of implementing clonal plasticity. From a software engineering perspective, clonal plasticity represents a new decentralized pattern of adaptation, where we have explored both, advantages and potential pitfalls of using this pattern.

7. ACKNOWLEDGMENTS

This work was partially supported by the EU FET-Project Diversify FP7-ICT-2011-9. We thank the external reviewers for their comments to earlier versions of the paper.

8. REFERENCES

[1] J. Beal. Functional blueprints: an approach to modularity in grown systems. *Swarm Intelligence*, 5(3):257–281, 2011.

[2] A. Bradshaw. Evolutionary significance of phenotypic plasticity in plants. *Genetics*, 13:115–155, 1965.

[3] M. Brenna, M. Falvo, F. Foiadelli, L. Martirano, F. Massaro, D. Poli, and A. Vaccaro. Challenges in

energy systems for the smart-cities of the future. In *Energy Conference and Exhibition (ENERGYCON), 2012 IEEE International*, pages 755–762, Sept 2012.

[4] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, March 2008.

[5] R. Capilla, M. Hinchey, and F. J. Díaz. Collaborative context features for critical systems. In *Proceedings of the Ninth International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '15, pages 43:43–43:50, New York, NY, USA, 2015. ACM.

[6] Y. Chaaban, J. Hahner, and C. Muller-Schloer. Towards fault-tolerant robust self-organizing multi-agent systems in intersections without traffic lights. In *Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content,Patterns*, pages 467–475, Nov 2009.

[7] H. Chourabi, T. Nam, S. Walker, J. Gil-Garcia, S. Mellouli, K. Nahon, T. Pardo, and H. J. Scholl. Understanding smart cities: An integrative framework. In *System Science (HICSS)*, pages 2289–2297, Jan 2012.

[8] R. Claes, T. Holvoet, and D. Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. *Intelligent Transportation Systems, IEEE Transactions on*, 12(2):364–373, June 2011.

[9] Z. Coker, D. Garlan, and C. Le Goues. SASS: self-adaptation using stochastic search. In *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015*, pages 168–174, 2015.

[10] R. E. Cook. Clonal plant populations: A knowledge of clonal structure can affect the interpolation of data in a broad range of ecological and evolutionary studies. *American Scientist*, 71(3):244–253, May-June 1983.

[11] S. Ghosn, P. Ranganathan, S. Salem, J. Tang, D. Loegering, and K. Nygard. Agent-oriented designs for a self healing smart grid. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 461–466, Oct 2010.

[12] S. González, K. Mens, M. Colacioiu, and W. Cazzola. Context traits: Dynamic behaviour adaptation through run-time trait recomposition. In *12th Intl Conf on Aspect-oriented Software Development*, AOSD'13, pages 209–220. ACM, 2013.

[13] J. L. Harper. Plant demography and ecological theory. *Oikos*, 35(2):244–253, 1980.

[14] C. Harris, R. Doolan, I. Dusparic, A. Marinescu, V. Cahill, and S. Clarke. A distributed agent based mechanism for shaping of aggregate demand on the smart grid. In *Energy Conference (ENERGYCON)*, pages 737–742, 2014.

[15] R. Hirschfeld, P. Costanza, and O. Nierstrasz. Context-oriented programming. *Journal of Object technology*, 7(3):125–151, March 2008.

[16] J. Kantert, S. Edenhofer, S. Tomforde, J. Hahner, and C. Muller-Schloer. Robust self-monitoring in trusted desktop grids for self-configuration at runtime. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2014 IEEE Eighth International Conference on*, pages 178–185, Sept 2014.

[17] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, Jan 2003.

[18] R. Levins. *Evolution in Changing Environments: Some Theoretical Explorations*. Monographs in Population Biology. Princeton University Press, 1968.

[19] K. N. Lodding. The hitchhiker's guide to biomorphic software. *Queue*, 2(4):66–75, June 2004.

[20] A. Marinescu, I. Dusparic, C. Harris, V. Cahill, and S. Clarke. A dynamic forecasting method for small scale residential electrical demand. In *Neural Networks (IJCNN)*, pages 3767–3774, July 2014.

[21] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, AAMAS '02, pages 418–425, New York, NY, USA, 2002. ACM.

[22] V. Nallur, J. Monteil, T. Sammons, M. Bouroche, and S. Clarke. Increasing information in socio-technical mas considered contentious. In *3rd Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*, pages 25–30, Sept 2015.

[23] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris. Smarter cities and their innovation challenges. *Computer*, 44(6):32–39, June 2011.

[24] C. Persson, G. Picard, and F. Ramparany. A multi-agent organization for the governance of machine-to-machine systems. In *Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 421–424, Aug 2011.

[25] P. Shi and Q. Shen. Cooperative control of multi-agent systems with unknown state-dependent controlling effects. *Automation Science and Engineering, IEEE Transactions on*, 12(3):827–834, July 2015.

[26] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. Biswas, and K. Moessner. Enabling smart cities through a cognitive management framework for the internet of things. *Communications Magazine, IEEE*, 51(6):102–111, June 2013.

[27] D. Weyns, S. Malek, and J. Andersson. On decentralized self-adaptation: Lessons from the trenches and challenges for the future. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '10, pages 84–93, New York, NY, USA, 2010. ACM.

[28] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka. *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, chapter On Patterns for Decentralized Control in Self-Adaptive Systems, pages 76–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[29] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *Internet of Things Journal, IEEE*, 1(1):22–32, Feb 2014.