iNet: An Extensible Framework for Simulating Immune Network

J. Suzuki and Y. Yamamoto
Graduate School of Science and Technology,
Keio University
Yokohama City, 223-8522, Japan

Abstract

The natural immune system is a subject of great research interests because it provides powerful and flexible information processing capabilities as a complex adaptive system. This paper describes our extensible framework named iNet for building artificial immune systems, particularly artificial immune networks, which has been used in our several projects developing biologically-inspired intelligent applications. We describe the iNet architecture, design principles and features, highlighting its versatility and component reusability. We also describe an autonomous decentralized network application built with iNet as a sample use case for showing how to use it.

1 Introduction

The natural immune system is a subject of great research interests because it provides powerful and flexible information processing capabilities as a decentralized intelligent system. It has some important computational aspects such as self/non-self discrimination, learning, memory, retrieval and pattern matching. The immune system also provides an excellent model of adaptive operation at the local level and of emergent behavior at the global level. There exists several theories to explain immunological phenomena and their software models. They have been used for machine learning, computer security, fault detection, change management, image processing, searching and robot navigation [1].

We have conducted research projects applying the principles and mechanisms in the natural immune system, particularly immune network, into practical software applications. For example, we have developed an autonomous policy negotiation and system reconfiguration facility for communication endsystems including HTTP server [2, 3]. Also, we are building a personalization engine that customizes web contents so that they become best suited to an individual user, and multi-agent systems that optimize strategies and behaviors of agents in a pursuit game and Robocup soccer game simulation. Our goal is to demonstrate the artificial immune system augments the autonomous decentralized adaptability of software systems.

This paper describes our extensible framework for building artificial immune networks, named iNet. iNet has been developed and maintained to support our research projects using artificial immune systems described above. It has been open for public use at Keio University since 1999, and will be released for researchers simulating the immune network mechanisms and exploring the design space of artificial immune networks. iNet serves as an infrastructure for our research efforts, and can be used by others exploring related mechanisms. It is an object-oriented reusable framework implemented with Java, which is customizable for various applications. In fact, we have developed the above applications by customizing iNet components without breaking a single architecture. This paper describes the iNet features, design principles and architecture, highlighting its versatility and component reusability. We also describe an intelligent network application built with iNet as a sample use case for showing how to use it.

The remainder of this paper is organized as follows: Section 2 overviews the natural immune system. Section 3 describes the iNet architecture and its design. Section 4 presents an intelligent network application using iNet.

2 Overview of the Natural Immune System

This section overviews two aspects in the natural immune system: self-protection and self-regulation mechanisms.

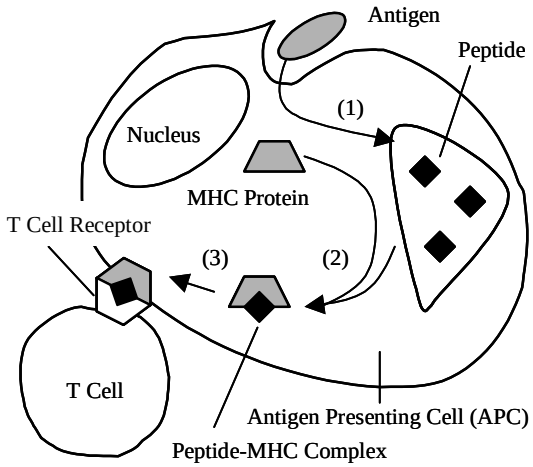

Figure 1: Processing and presentation of an antigen

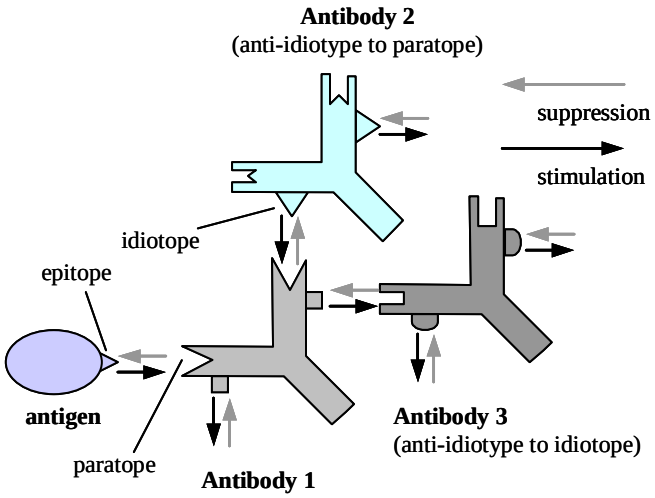

Figure 2: Interactions in the immune network

2.1 Self-protection mechanism in Immune System

The immune system can discriminate between foreign molecules (i.e. non-self) and the body's own cells and proteins (i.e. self). Once non-self is recognized, the immune system enlists the participation of a variety of cells and molecules to mount an appropriate response in order to eliminate or neutralize it.

The immune response involves *antigen-presenting cells*, *lymphocytes* and *antibodies*. Lymphocytes are one of many types of white blood cells, and two major population of lymphocytes are *B lymphocytes* (a.k.a. B cells) and *T lymphocytes* (a.k.a. T cells). B cells have receptors on their surface, which can recognize antigens invading a human body, e.g. viruses, and then produce antibodies specific to the recognized antigen. T cells have receptors called T cell receptor, which can recognize antigen associated with cell-membrane proteins known as *major histocompatibility complex molecules* (MHC molecules). Once the T cell recognizes and interacts with an antigen-MHC molecule complex, the cell secretes various growth factor known collectively as *cytokines*. They activate B cells and helps T cells to kill the recognized antigen. The recognition of antigen by T cells should be carefully regulated because an inappropriate T cell response to self-components can cause fatal autoimmune consequences (e.g. allergy). To ensure carefully regulated recognition, T cells only can interact with antigen that is displayed together with MHC molecules on the surface of antigen-presenting cells (APCs). These specialized cells, such as *macrophages*, roam the body ingesting the antigens.

Figure 1 depicts the process for an antigen-presenting cell to process and present an antigen. Once APCs find an antigen, it internalizes the antigen and fragments it into antigenic peptides (1) . Pieces of these peptides are joined to MHC molecules (2) and are displayed on the surface of the cell (3). The T cell recognizes and interacts with this peptide-MHC molecule complex using its receptor (T cell receptor). Like this, T cells do not interact with antigen directly. Instead, they can attach to the only cell displaying foreign antigen complexed to an MHC molecule. Such cells are called *altered self-cells*. MHC is a tightly linked cluster of genes, and MHC molecules are encoded with them. MHC genes are the same in all of an individual's cells but differ from person to person. T cells can recognize the genetic difference in MHC molecules and does not attack cells encoded the same type of MHC.

2.2 Dynamic Self-regulation mechanism through Immune Network

B cells are the cells maturing in the bone marrow. Roughly 10^7 distinct types of B cells exist in a human body, each of which has a distinct molecular structure and produces antibodies from its surface. An antibody recognizes and eliminates a specific type of antigens. The key portion of antigen that is recognized by the antibody is called *epitope*, which is the antigen determinant (see Figure 2). *Paratope* is the portion of antibody that corresponds to a specific type of antigens. Once an antibody combines an antigen via their epi-

tope and paratope, the antibody start to eliminate the antigen. Recent studies in immunology have clarified that each type of antibody also has its own antigenic determinant, called an *idiotope*. This means an antibody is recognized as an antigen by another antibody.

Based on this fact, Jerne proposed the concept of the *immune network*, or idiotypic network [4], which states that antibodies and lymphocytes are not isolated, but they are communicating with each other (Figure 2). The idiotope of an antibody is recognized by another antibody as an antigen. This network is formed on the basis of idiotope recognition with the stimulation and suppression chains among antibodies. Thus, the immune response eliminating foreign antigens is offered by the entire immune system (or, at least, more than one antibody) in a collective manner, although the dominant role may be played by a single antibody whose paratope fits best with the epitope of the specific invading antigen. The immune network also helps to keep the quantitative balance of antibodies. Through stimulation/suppression interactions, the populations of specific antibodies increase very rapidly following the recognition of any foreign antigen and, after eliminating the antigen, decrease again. Performed based on this self-regulating mechanism, the immune response has an emergent property through many local interactions.

The structure of immune network is not fixed, but varies continuously according to dynamic changes of environment. This flexible self-organizing function is realized mainly by incorporating newly generated B cells and removing useless ones. The new cells are generated by both gene recombination in bone marrow and mutation in the proliferation process of activated cells. Although many new cells are generated every day, most of them have no effect on the existing network and soon die away without any stimulation. Due to such enormous loss, the immune system maintains an appropriate set of cells so that the system can adapt to environmental changes in the piecemeal way.

3 iNet Architecture

This section presents the architectural design of iNet, which is a framework for simulating the natural immune network, and describes how it is customized to develop applications using artificial immune networks.

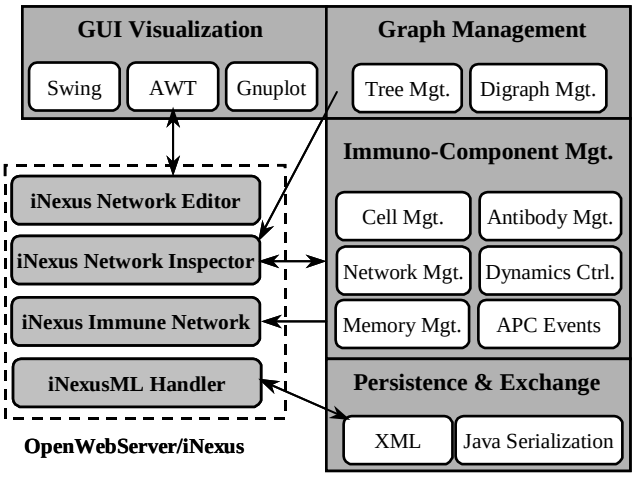

Figure 3: Architectural overview of the iNet framework

3.1 iNet as an Application Framework

An application framework provides reusable software components for applications by integrating sets of abstract classes and defining standard ways that instances of these classes collaborate [5]. The collection of these components forms an application skeleton. This skeleton can be customized by inheriting and instantiating from reusable components in the framework.

A software pattern represents a recurring solution to a software development problem within a particular context [6]. Patterns identify the static and dynamic collaborations and interactions between software components. In general, application frameworks and patterns help to enhance reuse techniques, reduce development cost and improve the quality of applications.

iNet is an application framework to develop a family of applications using artificial immune networks. Reusable components within iNet are designed with several software patterns; thereby iNet can explicitly show developers its design intents and extension points with which they can tailor their own applications.

Figure 3 illustrates the major structural components that comprise the iNet framework. iNet is designed to allow the customization of various strategies to form immune network, e.g. network topology and network dynamics control. It contains four packages[1]: *GUI Visualization*, *Graph Management*, *Immuno-Component Management*, and *Persistence and Exchange*.

[1]What the term package means here is same as the concept of package in Unified Modeling Language (UML) and Java.

The GUI Visualization package allows applications to incorporate the graphical presentation and direct manipulation capabilities using different visualization toolkits. It supports Swing and AWT as general GUI components, and Gnuplot as an off-the-shelf tool.

The Graph Management package allows applications to define and manipulate the graph structures of immune networks based on their preferences. This package provides components that forms directed graphs and tree. The artificial immune network has been modeled with liner network, cyclic network and tree, especially Cayley-tree that is a loop-less tree [7]. iNet supports most of the typical topology for building immune networks. Developers also define their own specific topologies by reusing existing components.

The Immuno-Component Management package is responsible for defining and coordinating components related to immune responses. All the components in this package are provided as interface classes, while other packages contain classes, abstract classes and interface classes, because the role and behavior of every immuno-component varies application to application. This means each application implements the interface classes to tailor an artificial immune network based on its characteristics and needs. The next section presents the structure and interaction of these components and describes how to implement them.

The Persistence & Exchange package implements the mechanisms to save the state of immuno-components and exchange it between different tools. The current version of iNet can export and import the state of an artificial immune network with eXtensible Markup Language (XML) and Java serialization. It allows applications to access and manipulate XML representations of immune networks through Document Object Model (DOM) interfaces and Simple API for XML (SAX). The state of immune networks can be interoperable between different applications or tools implemented with different programming languages.

The rectangle in a dot line in Figure 3 represents an application developed with iNet, called OpenWebServer/iNexus. We present this application in Section 4.

3.2 Components and Patterns in iNet

Figure 4 is a UML class diagram showing fundamental components in the Immuno-Component Management package in the iNet framework. These components are artificial counterparts of the components within the natural immune system described in Section 2. All

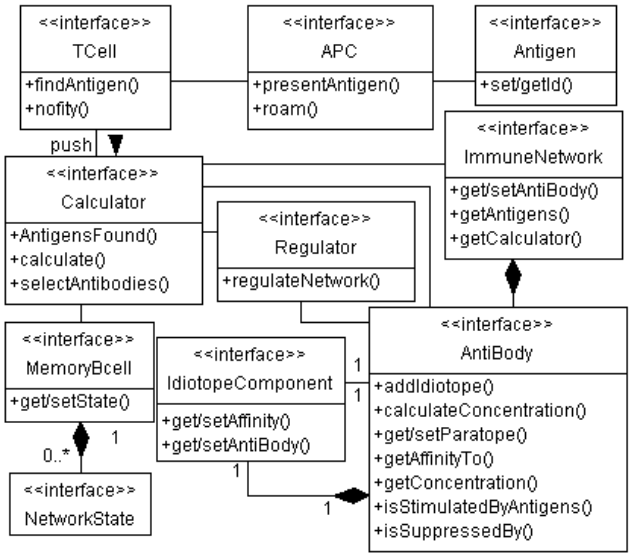

Figure 4: The fundamental classes in the Immuno-Component Management package in the iNet framework. They are described with UML class diagram.

the components are defined with interface classes as described above.

Figure 5 shows how to implement them in an application. `Node`, `Arc` and `WeightenedArc` are the classes definded in the Graph Management package. The classes with the prefix `iNexus`, i.e. `Policy`, `Calculator` and `Link`, are examples of application classes implementing the interface classes in the Immuno-Component Management package. The design principle defining application classes aligns with the *Adapter* design pattern [6]. This design pattern converts the interface of a class into another interface, and let classes work together that could not otherwise because of incompatible interfaces. In Figure 5, `Antibody` adapts to `Node` by communicating to a concrete `Policy` class. As a result, `Policy`, an application class, can override `Antibody`'s behavior as a `Node`. This means that application developers can tailor their artificial immune networks based on any topologies they want. As such, the Graph Management and Immuno-Component Management packages are kept loosely coupled, because it is not possible to predict how an artificial immune network is structured statically.

iNet incorporates other five design patterns: *Observer*, *Strategy*, *Memento*, *Iterator* and *Visitor*. The event notification mechanism between `TCell` and `Calculator` is designed with *Observer*. *Strategy* is used for designing `Calculator` and `Regulator`, which

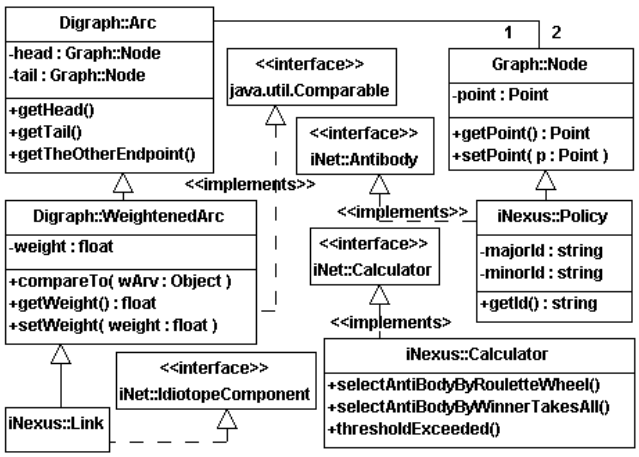

Figure 5: Interface classes in the Immuno-Component Management package and their implementations

are responsible for calculating and regulating populations of antibodies respectively. The calculation and regulation algorithms are defined as implementation classes of `Calculator` and `Regulator` by application developers. `NetworkState` is a Memento object, which captures and externalizes an immune network state. `MemoryBCell` uses it for providing the capability of immunological memory. *Iterator* provides a way to access antibodies in an immune network sequentially without exposing their underlying representation. *Visitor* allows developers to define a new method without changing the antibody class structure. A series of software patterns allows application developers to understand where the extensible and configurable points are in the iNet framework and how they can design their own artificial immune networks, only if they are familiar with patterns. Due to space limitations it is not possible to describe each software pattern in detail. Every pattern used in iNet is described in [6] at large.

Figure 6 depicts a part of standard interactions among components in the Immuno-Component Management package with UML sequence diagram. Each objects shown in Figure 6 is an instance of application class implementing the iNet interfaces presented in Figure 4.

4 OpenWebServer/iNexus: An Application using iNet

OpenWebServer/iNexus is our research vehicle for investigating and demonstrating the system adaptabil-

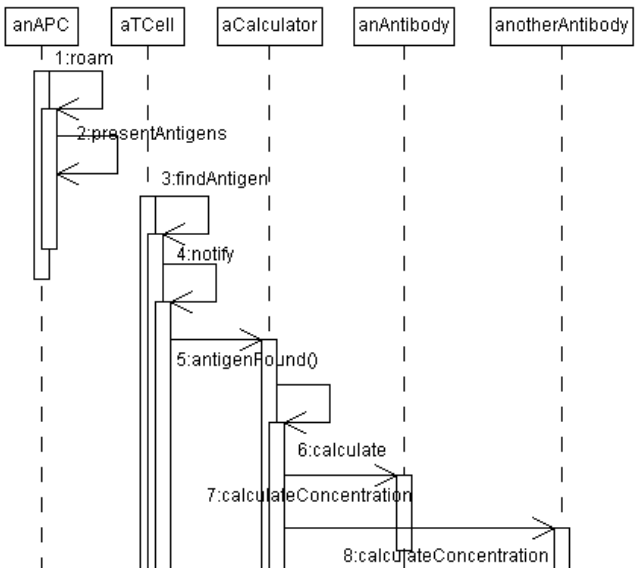

Figure 6: A standard sequence of method invocations among the iNet components

ity of communication system. OpenWebServer is an open-ended and adaptive framework for building optimally configured Internet servers [8] (see Figure 7). iNexus is an autonomous decentralized policy negotiation and system reconfiguration facility allowing OpenWebServer to adapt to an ever-changing environment [2, 3, 9]. The structure and dynamics of iNexus are designed based on the principles and mechanisms in the natural immune network. It manages a wide range of policies, even inter-dependent ones, and determines the most appropriate set of policies for a given system condition by relaxing constraints between them. The policy negotiation process is performed through decentralized interactions among policies without a single point of control, as the natural immune system does.

As shown in Figure 3, iNexus is developed as an application of iNet. It maintains an artificial immune network with a directed graph topology. The iNexus Network Editor is a GUI-based editor to define an initial structure of immune network. The iNexus Network Inspector is used to inspect the realtime status of an artificial immune network, e.g. the network structure and antibody polulations. The iNexusML Handler is an the facility to import and export network representation formatted with iNexusML (iNexus Markup Language), an XML-based format. The algorithms to control the network dynamics are described in [3]. An algorithms to provide the immunological memory in iNexus is described in [9].

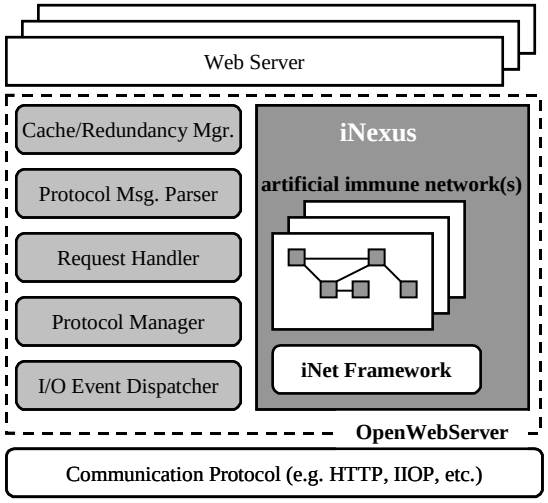

Figure 7: OpenWebServer/iNexus architecture

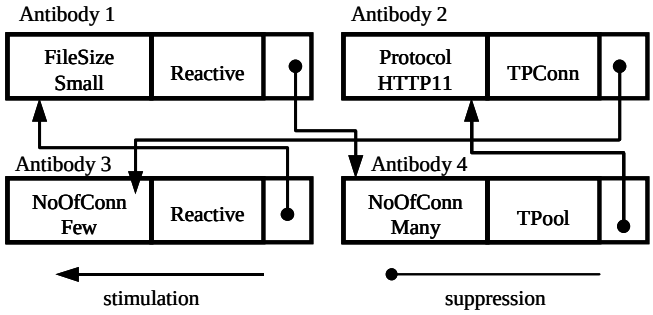

Figure 8: A sample immune network

Figure 8 shows a sample immune network in iNexus. iNexus specifies system's current conditions as antigens, e.g. the number of simultaneous network connections, average size of requested files, types of operating systems, the number of available processors, and supported types of protocols. Policies are regarded as antibodies, e.g. concurrency policies (thread-per-request, active/passive thread pool, thread-per-connection, etc.) and caching policies (FIFO, LRU, etc.). Policies are linked with each other based on the stimulation and suppression relationships. Both stimulation and suppression relationships are assigned to a single link between policies (Figure 8). Every relationship is weighted according to the co-use and conflict constraints between policies. In our existing experiments, OpenWebServer embedding iNexus can autonomously adapt and scale well to ever-changing environments [2, 3].

5 Conclusion

This paper described the iNet framework for simulating and building immune networks. iNet consists of a series of components designed with software patterns, thereby application developers can construct their own artificial immune systems in a flexible and extensible manner. We also present a network application using iNet, OpenWebServer/iNexus, which provides the adaptability through autonomous decentralized system reconfiguration by adopting key biological principles and mechanisms.

References

[1] D. Dasgupta. *Artificial Immune Systems and Their Applications*. Springer, 1999.

[2] J. Suzuki and Y. Yamamoto. A Decentralized Policy Coordination Facility in OpenWebServer. In *Proceedings of SPA 2000*, March 2000.

[3] J. Suzuki and Y. Yamamoto. Building an Artificial Immune Network for Decentralized Policy Negotiation in a Communication Endsystem: OpenWebServer/iNexus Study. In *Proceedings of The 4th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, U.S.A., July 2000.

[4] N. K. Jerne. Idiotypic Networks and Other Preconceived Ideas. *Immunological Review*, 79(5), 1984.

[5] R. Johnson. Framework = Patterns + Components. *Communications of the ACM*, October 1997.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[7] C. Chowdhury. Immune Network: An Example of Complex Adaptive Systems. In D. Dasgupta, editor, *Artificial Immune Systems and Their Applications*, pages pages 89–104. Springer, 1999.

[8] J. Suzuki and Y. Yamamoto. OpenWebServer: an Adaptive Web Server using Software Patterns. *IEEE Communications Magazine*, 37(4), 1999.

[9] J. Suzuki and Y. Yamamoto. Autonomous Decentralized Control of Distributed System Configuration using an Artificial Immune System. In *Proceedings of International Workshop on Autonomous Decentralized System*, Chendu, China, September 2000.