Sequence Diagrams: An Aid for Digital Ecosystem Developers

David A. McMeekin, Maja Hadzic, Elizabeth Chang
Digital Ecosystems and Business Intelligence Institute
Curtin University of Technology
GPO Box U1987
Perth WA 6845, Australia
{D.McMeekin, M.Hadzic, E.Chang}@curtin.edu.au

Abstract— **Creating, modifying and/or maintaining a software system is a time consuming and complex process. The digital ecosystem is a new concept introduced into the computer science and information systems world. New and already existing systems need to be integrated into these wider digital ecosystems. Here, it is crucial for the developer to understand the systems that need to be integrated to ensure that the modifications are effective and do not introduce new defects into the system. Developers need to be able to effectively model such systems to assist them in their system understanding, enabling them to create, modify and evolve the system into the associated digital ecosystems. This paper proposes sequence diagrams be used to represent and model the collaborative nature of digital species within digital ecosystems to assist developer understanding and comprehension of these systems. An illustration from the health domain, specifically, an electronic health records application is given. Also, a conjecture is also made within the paper that some software systems, in and of themselves, reflect and resemble the attributes and characteristics of a digital ecosystem.**

I. Introduction

Large and complex software systems have become ubiquitous and integral to the functioning of modern society. Many products, services and infrastructure are designed, produced and delivered through the use of these complex software systems [15], [10]. As a society we have become dependent upon these systems, many of which contain interactions not well understood, and can also behave in unexpected ways [4]. A recent example: Qantas flight QF72, 7 October 2008, abruptly pitched nose-down twice in the space of 3 minutes resulting in a MAYDAY being declared and an emergency occurring. Incorrect filtering by the flight control computers has been identified as of the causes of the incident [1].

Software is found in home appliances such as toasters, microwaves and refrigerators and is also relied upon for air traffic control, automobile traffic management and homeland defense, to name just a few. These software systems are increasing in size and complexity as government, business, education and community organizations attempt to digitize their "essential services" in order to deliver goods and services in a more simplified and efficient manner.

These systems are required, more often, to communicate and interact with each other in a mutually beneficial environment. Moreover, there is an increasing need for these systems to act autonomously, to adapt quickly to changing environments, to self-organize, and display many other characteristics of living beings. For this reason, these interacting and communicating systems are now being named *Digital Ecosystems* (DES) and the interacting entities named *Digital Species* (DS) [7].

Digital ecosystem is a relatively new term that has been embraced by the computing and information sciences world. The concept is borrowed from the biological sciences description of the natural ecosystem. It is into this digital ecosystem concept that software developers must create new or modify existing systems to function according to the needs specified by the digital ecosystem.

Software maintenance has been described as making alterations to an already delivered system [17]. The IEEE Std 1219-1993 defines it as "the modification of a software product after delivery... to adapt the product to a modified environment" [9]. It is well understood that software systems change and become more complex over time [11]. Modifying an existing system to enable it to function within a digital ecosystem represents maintenance on an existing system. Here, it is crucial for the developer to understand the system that needs to be integrated, into the digital ecosystem ,to ensure that the modifications are effective and do not introduce new defects into the system.

The digital ecosystem, with its inter-connectedness and interdependence introduces additional complexity into the development environment in which software engineers must operate. It has been reported that in the maintenance life-cycle, between 50% and 90% of the time is spent by developers trying to understand the program and system they are working on [5].

The digital ecosystem concept is quite recent and hence there is very little discussed in the literature regarding ways to represent the interactions that occur within an ecosystem to aid the actual software developer.

This paper examines representing digital ecosystems' collaborative aspects through sequence diagrams. Traditionally, sequence diagrams have been used to represent interactions between objects [3], [13], [14]. This concept is then used to show how the interactions between different digital species and interactions between digital ecosystems can be represented using sequence diagrams. The goal is to assist developers in understanding and creating digital ecosystems in a more comprehendible and controlled manner.

978-1-4244-2346-0/09/$25.00 ©2009 IEEE

In this paper the digital ecosystem concept is also expanded. Our conjecture is that a piece of software is often more than just a digital species [7], in many cases it can be defined as a type of digital ecosystem in and of itself. Obviously, not all software can be defined this way but many large and complex systems demonstrate characteristics synonymous with digital ecosystems.

II. Digital Ecosystems and Sequence Diagrams

The challenge to significantly improve software productivity and quality improvements is now located in the human complexity side of software development, that is, developers' time and effort needed to create the required software systems [10]. Also, a key endeavour of a software developer is to create the impression of simplicity, removing the user from any contact with the underlying complexity of a software system [3].

According to Kothari [10] a reduction in this human complexity should make a significant impact on the increased development of complex software systems. These complex software systems, demonstrate a large and rich set of functional behaviors while another distinguishing attribute of these systems is that it is almost impossible for a single developer to fully comprehend the system and all its subtleties [3].

A. Digital Ecosystems

Digital ecosystems have been classified as Complex Adaptive Systems (CAS). A CAS has many interacting parts or agents, from which behavior patterns evolve [18]. The behaviors arise due to the interacting parts or agents. These interactions cause behaviors to evolve that would not have evolved had the interactions not occurred [8].

Digital ecosystems are based on the concept of a biological ecosystem. The term ecosystem was first used in 1930 by Arthur Roy Clapham to describe the relationships between physical and biological elements in an environment. Following this concept, digital ecosystems have been defined as: "an infrastructure that is a pervasive digital environment which is populated by digital components which evolve and adapt to local conditions thanks to the recombination and evolution of its digital components [6]" or "is the dynamic and synergistic complex of digital communities consisting of interconnected, interrelated and interdependent Digital Species [7]." A digital ecosystem is a collaborative, domain–clustered and demand–driven environment [2]. These definitions clearly show that a digital ecosystem is a complex infrastructure containing interconnected and interrelated digital components or digital species.

A framework and methodology was proposed for the design of digital ecosystems [7]. The methodology contained five steps:

1) defining the digital species goals,
2) creating intelligent digital species,
3) defining collaborations of the digital species,
4) enabling, improving and constructing individual digital species, and
5) implementing security requirements to protect the digital ecosystem.

Although not explicitly stated or addressed within this proposed methodology, each step requires a software engineer/developer to write code for use within a digital ecosystem. For example, step 1, when the capabilities for "intuitive actions, information and transaction flow" are decided upon, the developer will be presented with this information and be required implement it in the chosen language. The steps provide an excellent framework and methodology for designing digital ecosystems, however no provision is made for low level assistance to the implementing developer.

In this paper the methodology's point 3 is elaborated upon, and how sequence diagrams can be used to model digital ecosystems' collaborative nature.

B. Sequence Diagrams

A sequence diagram is part of the Object Interaction Diagram family, found in the Unified Modeling Language diagrams. Sequence diagrams, also known as collaboration diagrams, diagrammatically represent a specific operation or situation, and the interactions through message passing that occur for the specific operation to take place [13], [14].

Collaborating objects are represented on a sequence diagram. These objects pass messages, and exchange information, as well as perform actions upon data. This results in the completion of the task requested by the actor [1].

Figure 1 is a simple sequence diagram that shows an actor interacting with an object of type `ClassOne`. The "require details" call from the actor to an object of `ClassOne` shows this initial interaction between the actor and the system. The actor requests information from the system. The `ClassOne` object interacts with the `ClassTwo` object. The object interaction is represented by the method call `getDetails()`. The `ClassTwo` object then processes the `getDetails()` method call and returns the process' results in the variable labelled `details` to the `ClassOne` object. The actor now has access to the returned `details` information and can use them for their own purposes.

As noted in [3], the objects described in Figure 1 contributed to the system's behavior through their collaborations. This interaction between the different objects is similar to interactions that occur within a digital ecosystem. Behavior coordination within a digital ecosystem requires digital species to exchange and process messages between one another.

Sequence diagrams are used to represent interactions and communications between different object instances within a system. Analogously sequence diagrams can be used within digital ecosystem development to represent interactions between different digital species and different digital ecosystems. For the developer this gives them a way to represent the interactions between digital species, digital organs, and digital ecosystems assisting them in the task of

[1] an actor is anyone or anything that acts upon or is acted upon by the system.

preparing the application to function within the wider digital ecosystem.Using Figure 1, this could be represented through having `ClassOne` correspond to `DigitalSpecieOne` or `DigitalEcosystemOne` and `ClassTwo` to `DigitalSpecieTwo` or `DigitalEcosystemTwo`.

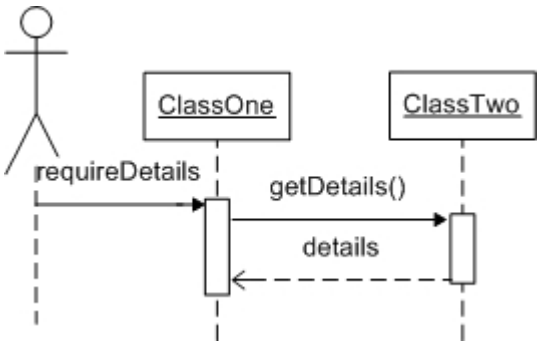

Fig. 1. An example sequence diagram.

III. Software as a Digital Ecosystem

It has been seen that some software, in and of itself, can be looked at as a small digital ecosystem. For example, it was reported that the launching of Microsoft Office 2008 (while in testing) with the Remote Home Folder in use, required the application to scan more than 20000 different files. Each file contained various information and data required for the product's successful launch [12]. These interactions lend to the concept that Microsoft Office can be seen as a digital ecosystem.

For the software to function correctly when the code is executed it must deliver that which is expected. In the case of a digital ecosystem, each digital species must provide the expected service and result. Without the digital species interaction, the digital ecosystem will not function.

It can be said that the Microsoft Office package is a digital ecosystem and that the associated tools, Microsoft Word, Microsoft Excel, Microsoft Access, Microsoft PowerPoint, Microsoft Publisher and Microsoft Outlook are digital species. These digital species populate the Microsoft Office digital ecosystem.

Each digital species consists of digital organs and these digital organs have different functions within the body of the digital species. For example, Microsoft Excel is a digital species and one of it's digital organs function is enabling users to create graphs. It can also be said that the Microsoft Office package is a digital species and that the associated tools, Microsoft Word, Microsoft Excel, Microsoft Access, Microsoft PowerPoint, Microsoft Publisher and Microsoft Outlook are digital organs. So how do we distinguish between the two? Can we formalize a definition to distinguish between the two?

For this, we go back to our original source of inspiration, the biological ecosystem. To understand the difference between two different species, we look at the differences between, for example, two different trees. The different appearances cannot be defined to be the most obvious difference between the two, as the leaves and roots of trees are also different in appearance. An obvious difference between the different trees is their fruit. Different trees produce different fruits. In the digital world, different fruit can be considered to correspond to different outputs. Microsoft Word output is different from Microsoft Excel output so we can say that Microsoft Word and Microsoft Excel are two different digital species of the Microsoft Office digital ecosystem.

To understand the difference between two different organs, we look at differences between, for example, the leaves and roots of a tree. Neither the leaves nor the roots of the tree can produce fruit on their own. They are dependent upon each other for fruit production. In the same way, the Microsoft Excel graph function cannot create a graph without the data available through other Microsoft Excel digital organs. Different functions of the Microsoft Excel digital species represent different digital organs of this species.

Therefore, a digital ecosystem can be a collection of interrelated and interconnected digital species where each digital species produces unique output. Digital species differ from each other based on their output. Digital organs on their own cannot produce this output. Digital organs work together with other digital organs of the same digital species toward the production output specific to this digital species.

It should be noted here that software needs hardware infra–structure to manifest and display its nature. The hardware is analogous to the body of a biological species and the software is analogous to the biological species' spirit [7].

Until this point in time, the digital ecosystems literature has not discussed software as a digital ecosystem. Some software, especially object oriented software, operates and functions in similar ways to which digital ecosystems operate and function.

There is a strong enough case identifying the way in which software operates that allows for some software in and of itself to be listed as a type of digital ecosystem. The example cited within this paper is not unique to that product. Many products and individual pieces of software interact and depend upon their surrounding environments for successful operation. For these pieces of software to function successfully they need to work, communicate and deliver results. In accordance with our earlier digital ecosystem definition, a complex infrastructure containing interconnected and interrelated digital components, this and other large pieces of software, are defined by many of the same characteristics that also characterize digital ecosystems.

IV. Sequence Diagrams for Digital Ecosystems

A working relationship between two or more digital species means that that relationship encloses assumptions that must be made about each other: operations that can be performed and resulting behavior. These relationships occur via links between the digital species. The links may be conceptual or physical, and it is via these links that digital species can obtain the services of another digital species, or via which digital species may traverse to each other [3], [16].

978-1-4244-2346-0/09/$25.00 ©2009 IEEE

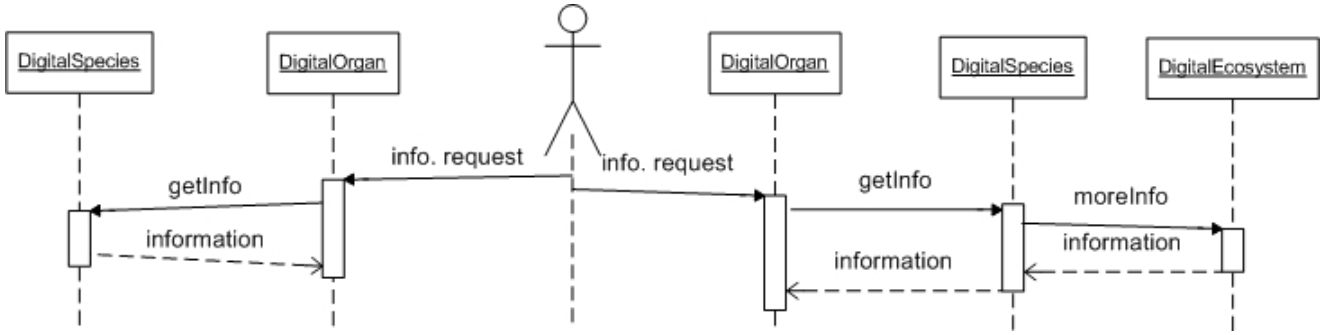

Fig. 2. A sequence diagram with the actor surrounded by digital organs, species and ecosystems.

The relationships and interactions between different digital species or different digital ecosystems can be represented using sequence diagrams. The sequence diagrams, are similar to the hierarchies, architectures and models described at the higher level of digital ecosystems. The difference between the models, architectures and hierarchies and the sequence diagrams is that the sequence diagrams give the visual understanding of the digital ecosystem from a developer's perspective, rather than from a designer's point of view.

Normal sequence diagram usage has the actor to one side of the sequence of events. Here in Figure 2 the actor is shown in the middle of the diagram. This represents the fact that the actor is surrounded by digital organs, digital species or digital ecosystems. At any given point in time the actor may request information requiring data to be returned from multiple locations.

The attributes described and the description given in Figure 1 are simplistic. The goal here was to unambiguously highlight the similarities that exist between modeling object interactions within software (Figure 1) and modeling collaborating components within a wider digital ecosystem (Figure 2).

Sequence diagrams are one way to represent digital ecosystems at a low level for developers. For example, consider Figure 3 to be a Digital Health Ecosystem. It has three parts and is a simple adaptation of a sequence diagram demonstrating an actor interacting within this environment. Each part of the figure can be interpreted in a different way:

1) Figure 3a. The actor requests information, and these requests go out to different digital species within the digital ecosystem. Each doctor may have a digital species assisting them in making appointments. A receptionist may desire to obtain information from these digital species to schedule the appointments.
2) Figure 3b. The actor requests information, and these requests go out to different digital ecosystems. As an example, there are two different digital ecosystems to be Digital Health Ecosystem of two different medical centers. A patient, who previously visited both medical centers, has their records in each medical centre. It is assumed that the treatment was different, hence the records will also be different. When the same patient visits a third medical centre, the doctor may need the patient's previous medical records to correctly diagnose and prescribe treatments for this patient. For this reason, the doctor needs to request the patient's records from both medical centers, namely, from the two different medical centres Digital Health Ecosystem.
3) Figure 3c. The actor requests information, and these requests go out to different digital organs of a digital specie. For example, monitoring an apparatus as a digital species that requires information from various digital organs to display reading results.

Using sequence diagrams for this purpose has several advantages. A significant advantage the sequence diagram holds is that of familiarity. A large contingency of developers are already familiar with UML. This means that when the developer is presented with a sequence diagram describing the interactions needing implementation into code, familiarity already exists with this type of representation. This would allow the developer to spend more time understanding and comprehending what is in the sequence diagram rather than attempting to understand the interactions' representation.

V. Conclusion and Future Work

This paper's first goal was to propose the use of UML sequence diagrams to represent interactions between digital ecosystems/digital species/digital organs for two reasons:

1) a sequence diagram is a way to represent sequential interactions between collaborating objects within a system and can be used to represent interactions between digital ecosystems/digital species/digital organs,
2) sequence diagrams are already used by developers, hence the developer is provided with more time to develop the system rather than attempting to understand the system's representation.

The high level design of semantics and architecture for digital ecosystems is extremely important. These designs must then be moved to a lower level where developers implement them into the chosen languages.

An advantage of using UML sequence diagrams is in that UML has become the de-facto language for diagrammatically representing software systems. In the context of the complexity the developer faces to successfully implement digital ecosystems, usage of an already established diagram

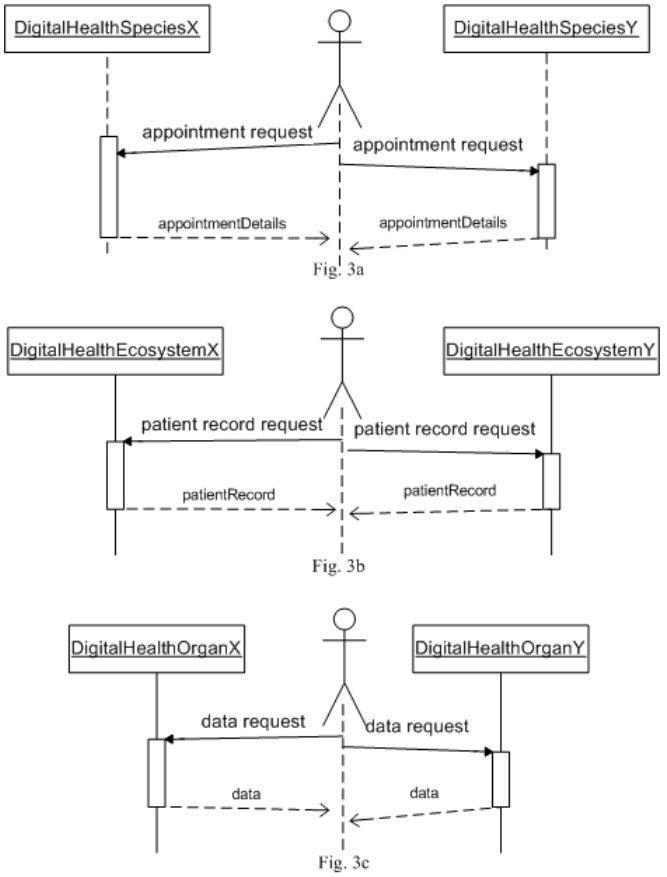

Fig. 3. A sequence diagram representation of a Digital Health Ecosystem.

set reduces the need for the developer to learn, understand and implement a wholly new representation methodology. This leads to less cluttering in the developers mind enabling them to focus on developing their ideas rather than focusing on the representation of those to be implemented ideas.

This paper's secondary goal was to suggest that some software in and of itself may be considered a digital ecosystem. The features and requirements needed by software to implement those features are similar to the features and requirements of a digital ecosystem. The complexity involved in implementing software, in many ways, parallels the complexity in implementing a digital ecosystem. It has also been concluded that digital species can be recognised by their own unique output.

Digital organs are part of this digital species and on their own cannot produce this output but they work together with other digital organs of the same digital species toward the production of the output specific to this digital species.

This paper addressed a framework and methodology for low level representation of interactions within digital ecosystems, in order to aid developers, through visual representations of the interacting systems using UML sequence diagrams and suggests that some software systems, in and of themselves, reflect and resemble digital ecosystems.

REFERENCES

[1] Australian Transport Safety Bureau. In-flight upset, 154 km west of learmonth, WA, 7 October 2008, VH-QPA, Airbus A330-303. Technical report, 3 2009.

[2] H. Boley and E. Chang. Digital Ecosystems: Principles and Semantics. In *Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, DEST '07, 2007.*, pages 398–403, 2 2007.

[3] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston. Addison-Wesley Professional, 2007.

[4] P. I. T. A. Committee. Information Technology Research: Investing in our future. Technical report, National Coordination Office for Computing, Information, and Communications, Washington, D.C., 1999.

[5] A. De Lucia, A. R. Fasolino, and M. Munro. Understanding function behaviors through program slicing. In *Proceedings of Fourth Workshop on Program Comprehension, 1996.*, pages 9–18, 3 1996.

[6] European Commission Information Society and Media Directorate General. What is an European Digital Ecosystem? Technical report, 2005.

[7] M. Hadzic, E. Chang, and T. Dillon. Methodology framework for the design of digital ecosystems. In *IEEE International Conference on Systems, Man and Cybernetics, 2007.*, pages 7–12, 10 2007.

[8] J. H. Holland. *Hidden Order: How Adaptation Builds Complexity*. Addison Wesley Publishing Company, 1996.

[9] I. IEEE. IEEE Standard for Software Maintenance (IEEE Std 1219-1993). 1993.

[10] S. C. Kothari. Scalable Program Comprehension for Analyzing Complex Defects. In *The 16th IEEE International Conference on Program Comprehension, 2008. ICPC 2008.*, pages 3–4, 6 2008.

[11] M. M. Lehman and L. A. Belady. *Program evolution: processes of software change*. 1985.

[12] Mac Mojo. Longest remote home folder test 600 miles Web page, last accessed 26 April 2009. http://blogs.msdn.com/macmojo/archive/2007/11/02/longest-remote-home-folder-test-600-miles.aspx November 2007.

[13] B. Oestereich. *Developing software with UML : object-oriented analysis and design in practice*. Addison-Wesley, Harlow, England ; Reading, Mass., 1999.

[14] M. Page Jones. *Fundamentals of Object-Oriented Design in UML*. Addison-Wesley Professional, 2000.

[15] Royal Academy of Engineering and British Computer Society. The Challenges of Complex IT Projects. Technical report, 2004.

[16] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented modeling and design*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1991.

[17] I. Sommerville. *Software Engineering*. Addison-Wesley, eighth edition, 2007.

[18] M. M. Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos*. Simon & Schuster, 1993.