Novelty Journals

ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 4, Issue 3, pp: (12-18), Month: September - December 2017, Available at: www.noveltyjournals.com

Code Mobility Model in Distributed Multifarious Computing Environment

Omoghenemuko, Greg I.[1], Asagba, Prince O.[2]

[1]Department of Computer Science, College of Education, Warri, Delta State, Nigeria
[2]Department of Computer Science, University of Port Harcourt, Rivers State Nigeria

Abstract: **There are complex issues as regards bulky-scale distributed multifarious computing environments, but a sure way to successfully interestingly overcome this historic disability, is mobile computing. Mobile code paradigmatics in computer programming has brought about the advent of intelligent, transportable and autonomous pieces of code known as mobile code - a code, that freight itself and it is self-controlled to obtain objectives of manipulating data and annexing output dynamically through migration on a multifarious network environment; to get task of data computation and information retrieving accomplished. The researchers x-ray code mobility and define mobile agent as a specific mobile code. This research gives cursory attention to typology of code mobility, process migration, taxonomy of code mobility and typology of code jumps exhibited by mobile agents, and then proposed gratuitously an intelligent representation of code mobility model and presented some merits of code mobility in distributed multifarious network environment.**

Keywords: **Code mobility; mobile agent; model; code jump; paradigmatics; applets; process migration; taxonomy and distributed multifarious environment.**

1. INTRODUCTION

The mobility of code and autonomy is historic reality and actuality hinged on Internet. Code mobility and autonomy are ultimate expressions of distributive computing in a multifarious environment. This new computer networking paradigm is a veritable tool for global competitiveness and it has made Internet not only popular, interesting but very welcome development. The merit, therefore, of using the mobile code technology is that, interaction cost for the mobile code owner is decreased remarkably since the mobile code after leaving its owner it transports from a host platform to another autonomously. The Remote Programming approach views two computers' interaction as one computer ability to require procedures in another and to launch or give the procedures to execute. The *procedure* and its *state* are together termed *mobile code* or *mobile agent* as it represents the sending computer though exist on the receiving computer. This is mobile agents-based computing paradigm. An innovation on the client-server technology gave birth to mobile agents. The conception of mobile agent is described as unique due to its capability to transport code and data to a location on a network environment. Mobile agent concept brings the computation to the data and the mobility and autonomy feature makes lasting connections unnecessary (Elmarie, 2004). Mobile agent paradigmatics is an advanced technology, which introduced code mobility. Code mobility facilitates various tasks performance in effective manner. Mobile agents prove to be more efficient for usage in a distributed multifarious environment where data from various systems may be needed to perform a particular task.

2. CODE MOBILITY

Code mobility is the translocation of program between nodes in heterogeneous computing network environments'. Mobile codes that have definite names like Telescript, Ara, plagiarism CheckerX, Tacoma, Grasshopper etc., are called mobile agents. Mobile codes are mobile programs, that work as agents unbehalf of their owners' or users', hence, mobile agents.

Code mobility is achieved using programming languages like Java and visual C#, and programming statement like *go to*, *If Then* construct and *While Do* statements. Mobile codes or mobile agent are different from Java applet- a program that is downloaded by user's on a browser and executed from starting to end by host as part of web page and used as advertorial.

One of the first incarnations of mobile software entities, is *worms*, which could spread across nodes and arbitrarily cloned (Shoch and Hupp, 1982) cited in (Dejan *et al.*, 1998). Unrestricted implementations of worms and viruses have negative connotations, due to security breaches like denial-of-service (DoS) attacks (Ford and Baum, 1997) cited in (Dejan *et al.*, 1998).

Process migration was the next technology innovation of mobile-entities' executed at the level of operating-system. Implementations of migration of process and object were possible but unaccepted for their complexities, because of impossibility to introduce process migration without hindering stability and boisterousness of fundamental operating system.

Mobile objects and agents have drawn salient attention. 'Mobile code like applets, has non-transient state that travel between nodes in distributed system like intranet or Internet (Dejan *et al.*, 1998). Compared to mobile objects, mobile agents also represent someone; they can perform autonomous actions for a user or another (Peter and Arkady, 1998).

Mobile code is a code that perambulates on network and computes on a destination computer. The capability to move code to new location, is a powerful concept that originated very interesting range of developments. The risks of having such an arbitrary code executing on a local machine is evident. Virus may be added to code that could affect the machine when the code is run. The program may also be a Trojan horse, appearing to do something useful while taking over the machine (Satish *et al.*, 2002). Mobile code may be a virus, a worm or a Trojan. Some techniques like sand-boxing and certificates are suggested to alleviate these problems. From perspective of sender of agent, execution of mobile code on the recipient's computer can be manipulated or spied because recipient has full advantage over the agent's code. As we explore Internet- the global information architectural infrastructure, the more mobile agents (or mobile applications) usability and the more mobile agents' susceptibility to malignant host platforms' attacks. Therefore, mobile agent protection is great challenging emerging issue confronting computer science advancement in 21st century.

3. TYPOLOGY OF CODE MOBILITY

Two types of code mobility (ie. mobile agency) exist in Mobile agent Paradigmatics. They are: *physical mobility* and *logical mobility*.

Physical mobility*:* This is ability for host platform to change geographic location while sustaining capability to communicate and perform computation (ie. execute). Physical mobility is a requirement by user

Logical mobility: Software (ie. application) executable components can be relocated to another environment at execution time (or run time). Logical mobility is mobile agent developer's choice.

There are variations in logical code mobility. An entire code can translocate to another destination to fulfill computation tasks'. For sure, mobility of live-code is *computation mobility* (Cardelli, 1997). If the entire computation tasks are taken to another destination (host) for execution and results retuned to local host platform, it is called *total computation mobility*. When only portion of computations', is translocated to another destination (host) for execution, it is called *partial computation mobility*.

Mobility of live Data and Information is called *Data-Information mobility.* For instance, Java's ability to execute applets directly in clients browsers, is Java's most dazzling and captivating incarnation

Data mobility is present since the beginning of networked computing, in applications' ranging from remote consultation of databases to Web browsing (Paul and Veronica, 2003).

4. TAXONOMY OF CODE MOBILITY

Physical mobility and logical mobility in mobile agency are classified into three distinct groups. They are:

Unstable Mobility. This is mobile code (or agent) sent on demand from host-server to client machine to execute, and after execution, arrives home with results to owner that sent it.

Stable or Strong Mobility is mobile agent's ability to convey state within an execution unit to a different computation environment. Strong migration captures agent object's state, code and state, allowing it to continue execution from exact point at which it left-off. The strong migration is better for end programmer, but work for the system developer since routines to capture control state must be added to the existing interpreters.

Fairly Stable or Weak Mobility is Mobile agents competence to move over many environments of computing. The code can contain some initialisation data, but no execution state is transferred (Alfred *et al.,* 1986). Migration either results from programmer's hard coded itinerary or reactiveness property of mobile agent to its environment (Elmarie, 2004). Weak migration captures only the agent's object state and code, and then calls a known entry point inside its code to restart the agent on the new machine. All the java-driven mobile codes do not capture an agent's thread (or control) state during migration, which is weak migration. For thread capture requires modifications to standard Java's virtual machine. That is, thread capture means that the systems are usable with one specific virtual machine. Programmers generally address mobility in two directions; *weak mobility*, whereby the agent's state is denoted in well-defined data structures in code, permitting migration only at specific points in agent code; and *strong mobility* captures agent's states at the underlying thread or process and allows migration when executing agent (Guido *et al.,* 2009). Weak mobility means agent capability to restarts on the new host from where it was transferred (Horvat *et al.,* 2000). To retain thread in agent system supporting *weak mobility*, additional programming is needed to save execution manually. In a system with *strong mobility*, migration is open, reducing programmer's efforts and size of code transmitted (Picco, 2001).

Mobile agent chooses strategy for migration, such as pull-code where the code is downloaded by host executing from a specified sources and push-code where they code is sent in advance to the executing host. In the pull-code strategy, agent's code is transmitted with its data and dynamically loaded by new host during migration (Elmarie, 2004). The transmission strategy defines the way mobile agent is physically transmitted to host. It states how the data and code are transmitted on a protocol level, such s TCP/IP (Jürgen *et al.*, 2007).

Class files are loaded as package once a specific file in class is needed. Push- code strategy requires agent's transmission to hosts' specified on itinerary. Considering migration of mobile codes as a jump from one host to another to accomplish computation tasks in a heterogeneous environment, we refer to mobile codes (that is, mobile agents) by the type of jump or migration that they exhibit:

(a) **One-jump Agents** (**Unstable Mobility**) e.g. Applets, Aglets, Concordia.

(i) Agents sent on demand from host to client machine to execute.

(ii) After execution, agents' arrive home with results to owner that sent it.

Aglets implement a model that is experiment-driven. Like most other commercial agents developed in java, they translocate the agent's data, objects and code, but not thread state while migrating from one machine to other. One-jump agents' is illustrated in (figure-1.)

(b) **Two-jump Agents** (**Fairly Stable Mobility** or **weak mobility**) eg. Messenger

(i) Agent transports or travels from local host to remote, and then returns to local host. Two-jump agents' is illustrated in (figure-2.)

(c) **Multi-jump Agents** *(***Stable Mobility**) eg. Ara, Telescript etc.

(i) Agents sent out on Internet to fulfill tasks.

(ii)These agents may visit multiple agent platforms and relate with other agents' in a multifarious environment. Multi-jump agents' is illustrated in (figure-3.)

In line with our taxonomy of code mobility, Ordille (1996) cited in Elmarie (2004), categorises the travelling of mobile agents into three forms, namely *one-hop agents*, *two-hop boomerang agents* and *multi-hop agents*. A hop is trip to a host after another.

A hop is trip to a host after another. A one-hop agent only travels from local host to a remote host; a two-hop boomerang agent travels from local host to remote and back to local host, but multi-hop agents' translocate or journey to multiple

Novelty Journals

ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 4, Issue 3, pp: (12-18), Month: September - December 2017, Available at: www.noveltyjournals.com

hosts' platforms. The researchers represent a *one-hop agents*, *two-hop boomerang agents* and *multi-hop agents* in Figures 1, 2 and 3 for clarification.

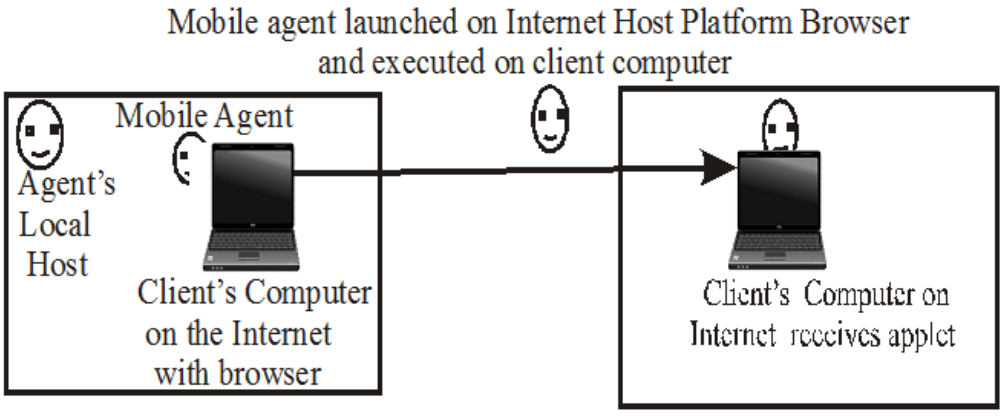

Figure 1:One-jump mobile agent (unstable mobility)

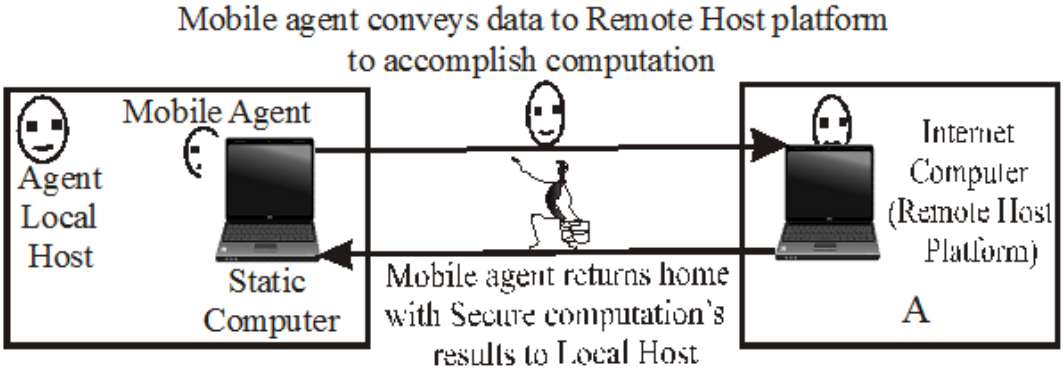

Figure 2: Two-jump mobile agent (fairly stable or weak mobility)

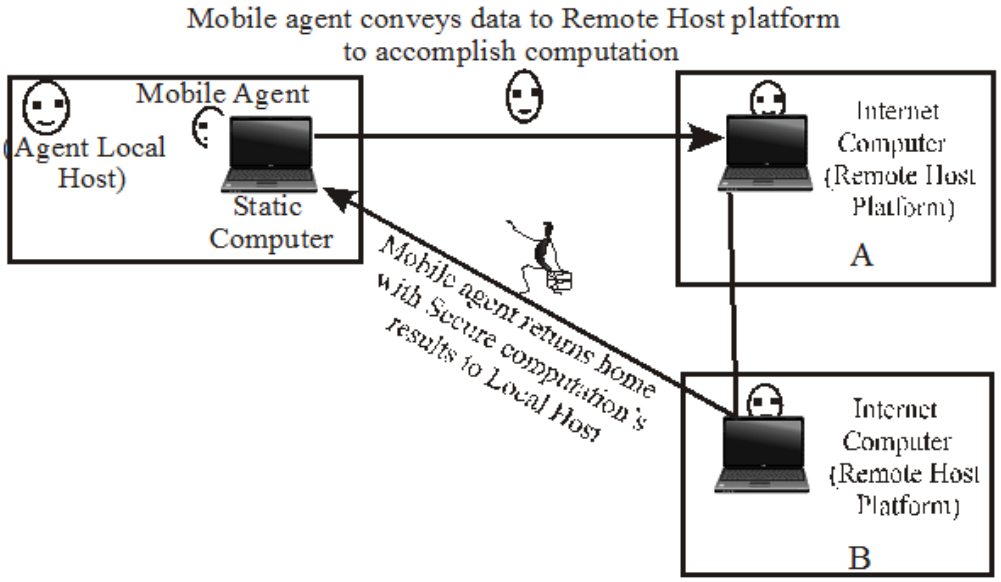

Figure. 3: Multi-jump agent (stable or strong mobility)

Novelty Journals

ISSN 2394-7314

International Journal of Novel Research in Computer Science and Software Engineering
Vol. 4, Issue 3, pp: (12-18), Month: September - December 2017, Available at: www.noveltyjournals.com

According to the categorisation of mobile agents into: one-hop agents, two-hop boomerang agents and multi-hop agents, trust is established by policy. For example, establishing trust for one-hop agents is simpler than multi-hop agents in that one remote host is visited and the agent does not travel any further.

The trust in these mobile agents, is determined by a risk policy; if the mission of agent is only to carry data to destination, then the agent owner only has to trust remote host to receive mobile agent and its data (Elmarie, 2004).

Proposed Code Mobility Model:

Mobility model defines creation and behaviour of mobile agent on its local host, and when mobile agent translocates to other hosts; either to do computation or retrieve information before returning to local host. The mobility model views migration capabilities from three dynamic dimensions, namely:

1. How programmers implement migration;
2. Migration strategy mobile agents' choose; and
3. Effect of transmission strategy on the ubiquitous computational network environment.

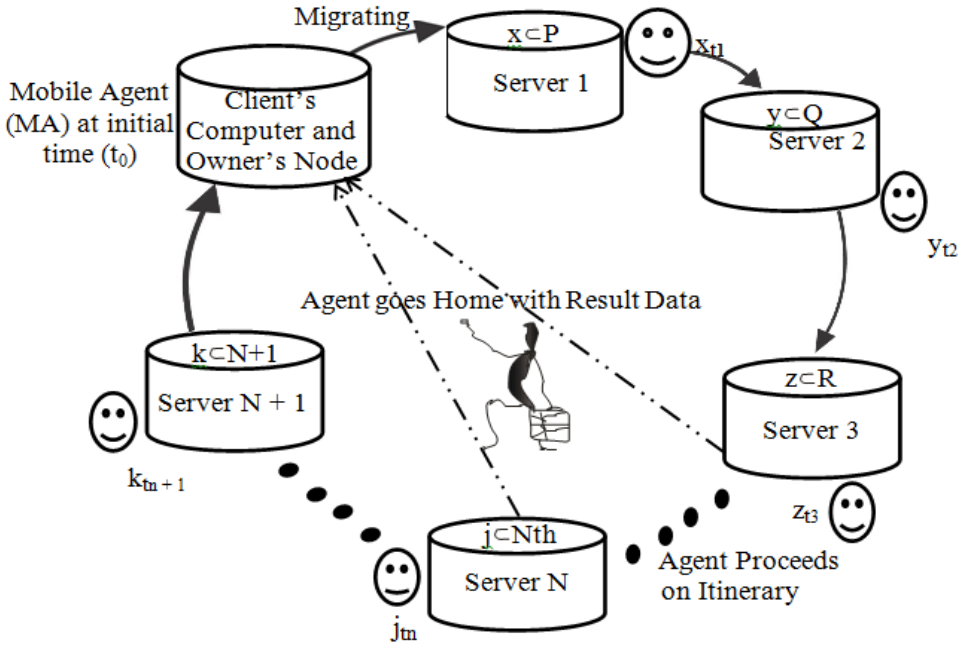

Figure 4: proposed code mobility model

Suppose from figure-4 modeling diagram, a Mobile Agent (MA) has a travel plan (itinerary) to visit three remote hosts (or remote server nodes) P , Q and R on a multifarious network environment (ie. Internet). If time is expressed in second, time (t_0) is initial time before embarking on itinerary and suppose it spends time (t_1) to collect data (x) from Database on remote *host P* before migrating to another remote *host Q to* spend time (t_2) to collect data (y) and then spends time (t_3) at remote *host R* to collect data (z), at remote *host N* to collect data (k) before returning home, then we can formulate a mathematical travel model for this computation and information retrieving agent as follows:

$$MA\ (t_0) \neq MAx_{t1} + MAy_{t2} + MAz_{t3} \quad (1)$$

Where $X \neq Y \neq Z \neq J \neq K$ are resources collected at remote hosts' platforms (Servers) 1, 2, 3, N and N+1 having total resources P, Q, R, N and N+1

$$MA\ (t_{1,2,3,n\ldots n+1}) = MA_{xt1} + MA_{yt2} + MA_{zt3}\ldots+\ldots MA_{Ntn}\ldots + \ldots MA_{Ntn+1} \quad (2)$$

$$= MA(x_{t1} + y_{t2} + z_{t3} + \ldots_{Ntn}\ldots + _{Ntn+1}) \quad (3)$$

N is the Nth remote host where agent migrate to do computation and t_n is the computation's time of agent at remote *host N*. Where n is integer from 1,2,3,4,5, n,…,n+1.

Now, suppose the mobile agent (MA) described in our research, is an information gathering or retrieving agent, and it decides to continue on itinerary to N+1 existing server node to get a task accomplished, the time to accomplish this additional task, is t_{n+1} . Where *n* is integer from 1,2,3,4,5, n….n+1

The total time on itinerary is computed in Seconds as:

Total Time of Mobility (TTM) =Time to collect resources at Server 1 (t_1) + Time to collect resources

at Server 2 (t_2) + Time to collect resource at Server 3(t_3) + Time to collect

resource at Server N (t_n)…+…Time to collect resource at Server N+1(t_{n+1})

$= t_1 + t_2 + t_3 + t_n + \ldots + t_{n+1}$ (Where $t_1 \neq t_2 \neq t_3 \neq t_n \neq t_{n+1}$)

Note that '1' added to *n* as (n+1) does not mean numeric '1', but an extra time by implication

Therefore, the mobile agent's Travel Model (MTM) for an information retrieving agent may be formulated mathematically as:

$$MA(t_{1,\,2,\,3,\,n,\ldots,n+1}) = MA_{xt1} + MA_{yt2} + MA_{zt3} + MA_{Ntn} \ldots + \ldots MA_{Ntn+1}$$

$$= MA(x_{t1} + y_{t2} + z_{t3} + N_{tn} \ldots + \ldots {}_{Ntn+1})$$

Advantages of Code Mobility:

- Mobile code translocates its code to data for computation
- Mobile code translocate code., objects and data from home to abroad for computation and return home with results of computations
- Interminable connectivity between computers is not need in code mobility.
- Mobile codes work autonomously and asynchronously and are not monitored while they migrate
- Mobile codes are needed for gathering, retrieving, and transmitting information for their adaptive, learning and social attributes' and automation.
- Mobile codes' manages network bandwidth and interruption (ie. latency)
- Mobile codes' helps in modularisation (or breaking down) of task into sizeable units that executes independently
- Mobile codes' helps in balancing network load
- Mobile codes' promotes scalability since work is easily moved to a network location that is most convenient.

5. CONCLUSION

The whole conception of code mobility is quite fascinating and encouraging as it has made computer programming quite challenging but enriching. Although mobile code paradigmatics, is not without some demerits, but the merits exceed its demerits and therefore, we should generally embrace its proficiency and explore its advantages for dynamic computations to facilitate online trade and commerce in distribute multifarious computing environment.

REFERENCES

[1] Alfred, A. V.; Sethi, R.; and Ullman, J. D. (1986): *Compilers: Principles, Techniques and Tools*. Addison-Wesley.[Retrieved From: https://www.researchgate.net/publication/242427147_Aho_R_Sethi_JD_Ullman_ Compilers Principles_Techniques_and_Tools_Addison_Wesley_1986_2]

[2] Cardelli, L.(1998): Mobile Ambients. A Technical Report, Microsoft.

[3] Dejan, S; Williams, L.; and Deepika, C.(1998): *Mobile Objects and Agents (MOA)*: Proceedings' of 4th USENIX Conference on Object-Oriented Technologies and Systems' (COOTS), New Mexico. [Retrieved From: https://www.usenix.org/legacy/publications/library/Proceedings's/coots98/fullull_papers/milojicic/milojicic.pdf]

[4] Elmarie, B.(2004), *Framework for Protecting Mobile Agent Against Malicious Host*. PhD Thesis submitted to Department of Computer Science University of South Africa

[5] Guido, J. V; Benno, J. O; Reinier, J; Frances, M. And Andrew, S.(2009): *Constructing Secure Mobile agent Systems*' *Using Agent Operating System*. Internationall Journal on Intelligent Information and Database Systems', Volume 3, No. 4. [Retrieved..From: https://staff.fnwi.uva.nl/g.j.vantnoordende/publications/ps/ijiids-final.pdf]

[6] Hock, K.T.; and Luc, M. (2002): *Mobile-Code Framework Security Using Certificates.*[Retrieved From: https://www.researchgate.net/publication/220999254_Certificates_for_mobile_code_security]

[7] Horvat, H.; Cretkoviæ, D.; Multinovaiæ, D.; Koêoviæ, P. And Kovaêeviæ, V. (2000): *Mobile agent and Java Mobile agent Toolkits*. Proceedings' of 33rd International Conference on Systems Science, Hawaii.

[8] Ordille, J. (1996): *When-Agents*' *Roam, Who can be Trust*? Proceedings' of 1st Conference on Emerging-Technologies and Applications in Communications'.

[9] Paul, T.; nd Veronica, D. (2003): *High-Level Networking with Mobile Code and First Order AND-Continuations*. [Retrieved From: http://www.cse.unt.edu/~tarau/research/2001/TLP_inet.pdf]

[10] Peter, S.; and Arkady, Z. (1998): *Expressing Dynamics of Mobile Agent Systems Using Ambient Calculus, DEXA*. International Workshop on Database and Expert Systems Applications. Retrieved From: http://www.computer. org/csdl/Proceedingss/dexa/1998/8353/00/83530434-abs.html]

[11] Picco, G. (2001): *Mobile Agents*: An Introduction, Microprocessors and Microsystems, Dipartimento di Elettronica Informazione, Politecnico di Milano, Milan, Italy. Volume 25.

[12] Satish, R.; Govindakrihnan, K.; Venkata, R.; Rajneesh, M.; Balaji, T.; Nilesh, P.; and Sravan, K.(2002): *Security of Mobile Code: New Approach for Mobile Code Security.* Department of Computer Science, University of Kentucky Dan S. W. (1999).. Princeton University. <URL: > Microsoft Application Programmer's Guide.

[13] Wei, J. M. And Mehrdad, J. (2014): *Neural Coding of Uncertainty and Probability.* Annual Review of Neuroscience, Center for Neural Science and Department of Psychology, New York University, New York, 10003; Volume37.[Retrieved From: http://www.annualreviews.org/doi/full/10.1146/annurev-neuro-071013-014017]