REFERENCES

Alexander, R.T. & Bieman, J.M. (2002) 'Challenges with Aspect-oriented Technology', ICSE Workshop on Software Quality, Orlando, Flordia, 25 May 2002.

Anderson, R.J. (2001) *Security engineering: a guide to building dependable distributed systems*, Wiley, Computer Publishing, New York, USA.

Andrews, G.R. & Reitman, R.P. (1980) 'An axiomatic approach to information flow in programs', *ACM Transactions on Programming Languages and Systems*, vol. 2, no. 1, pp. 56-76.

Baniassad, E. & Clarke, S. (2004) ' Finding Aspects in Requirements with Theme/Doc', in *Proceedings of Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design*, Lancaster, UK, 22 March 2004.

Bell, D.E. & La Padula, L.J. (1976) *Secure computer systems: Unified Exposition and Multics Interpretation*, Technical Report ESD-TR-75-306, Electronics Systems Division, Bedford USAF Base.

Bodkin, R. (2004) 'Enterprise Security Aspects', AOSD'04 International Conference on Aspect-Oriented Software Development, Lancaster, UK, March 2004, <[Online] Available: http://www.cs.kuleuven.ac.be/~distrinet/events/aosdsec/papers.html>.

Boehm, B. (2002) 'Get Ready for Agile Methods with Care', *Computer*, vol. 35, no. 1, pp. 64-9.

Boström, G. (2004) 'A case study on estimating the software engineering properties of implementing Database Encryption as an aspect', Proceedings of the 3rd international conference on Aspect-oriented software development, Lancaster, UK, 22-24 March 2004.

Briscoe, B., Rizzo, M., Tassel, J. & Damianakis, K. (2000) 'Lightweight policing and charging for packet networks', in *3rd IEEE Conference on Open Architectures and Network Programming*, Tel Aviv, Israel, 26-27 March 2000, pp. 77-87.

Cederquist, J.G., Corin, R., Dekker, M.A.C., Etalle, S., den Hartog, J.I. & Lenzini, G. (2006) *The Audit Logic: Policy Compliance in Distributed Systems*, Centre for Telematics and Information Technology, University of Twente, Enschede.

Chan, M., Woon, I. & Kankanhalli, A. (2006) 'Perceptions of information security in the workplace: linking information security climate to compliant behavior', *Journal of Information Privacy and Security*, vol. 1, no. 3, pp. 18-41.

CHAPTER 9: A DRAFT BILL ON THE PROTECTION OF PERSONAL INFORMATION (2005), viewed 30 November 2009, <Available [Online] http://old.ispa.org.za/regcom/privacyfiles/chapter-9-draft-bill-protection-personal-info.pdf>.

Chen, L. (2004) *Aspect-Oriented Programming in Software Engineering*, Technical Report, Wake Forest University, Department of Computer Science.

Chon, R., Enokido, T. & Wietrzsk, V. (2004) 'Role Locks to Prevent Illegal Information Flow among Objects', in *18th International Conference on Advanced Information Networking and Applications (AINA'04) Volume 1*, Fukuoka, Japan, 29-31 March 2004, pp. 196-201.

Chou, S.-C. (2003) 'Information Flow Control among Objects: Taking Foreign Objects into Control', in *36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Big Island, Hawaii, 6-9 January 2003, pp. 335-44.

Clarke, R.E. (2002) 'e-Consent: A Critical Element of Trust in e-Business', 15th Bled Electronic Commerce Conference: eReality:Constructing the eEconomy, Bled, Slovenia, 17-19 June 2002.

Constantinides, C. & Hasson, Y. (2002) 'Beyond objects: Improving the modularity of complex software', Workshop on Grand Challenges for Computing Research, Edinburgh, Scotland, 24-26 November 2002.

D' Arcy, D. & Hovav, A. (2007) 'Deterring internal information systems misuse', *Communications of the ACM*, vol. 50, no. 20, pp. 113-7.

De Win, B., Joosen, W. & Piessens, F. (2002) 'Developing Secure Applications through Aspect-Oriented Programming', in Aksit, M., Clarke, S., Elrad, T. & Filman, R.E. (eds), *Aspect-Oriented Software Development*, Addison-Wesley, Boston, p. 633–50.

De Win, B., Joosen, W. & Piessens, F. (2003) 'AOSD & Security: A Practical Assessment', Workshop on Software engineering Properties of Languages for Aspect Technologies (SPLAT03), Boston, Massachusetts, 17-21 March 2003.

De Win, B., Piessens, F. & Joosen, W. (2002) 'On the importance of the separation-of-concerns principle in secure software engineering', Workshop on the Application of Engineering Principles to System Security Design, Boston, Massachusetts, 6-8 November 2002.

De Win, B., Vanhaute, B. & De Decker, B. (2002) 'How aspect-oriented programming can help to build secure software', *Informatica*, vol. 26, no. 2, pp. 141-9.

De Win, B., Vanhaute, B. & Decker, B. (2001) 'Security Through Aspect-Oriented Programming', in Decker, B.D., Piessens, F., Smits, J. & Herreweghen, E.V. (eds), *Advances in*

Network and Distributed Systems Security, IFIP TC11 WG11.4 First Working Conference on Network Security, Leuven, Belgium, 26-27 November 2001, pp. 125-38.

Denning, D.E. & Denning, P.J. (1977) 'Certification of Programs for Secure Information Flow', *Communications of the ACM*, vol. 20, no. 7, pp. 504 -13.

Devanbu, P.T. & Stubblebine, S. (2000) 'Software engineering for security: a roadmap', in *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland, 4-11 June 2000, pp. 227-39.

Dewan, P., Grundin, J. & Horvitz, E. (2007) 'Towards a mixed-initiative access control', in *COLCOM '07: Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, New York, USA, November 12-15, 2007, pp. 64-71.

Downs, D., Rub, J.R., Kung, K.C. & Jordan, C.S. (1985) 'Issues in Discretionary Access Control', in *1985 IEEE Symposium on Security and Privacy, 1985*, Oakland, CA, 22-24 April 1985, pp. 208-15.

Elrad, T.M., Askit, G., Kiczales, K., Lieberherr, H. & Ossher. (2001) 'Discussing Aspects of AAOP', *Communications of the ACM*, vol. 44, no. 10, pp. 33-8.

Engel, M. & Freisleben, B. (2005) 'Supporting autonomic computing functionality via dynamic operating system kernel aspects', in *Proceedings of the 4th international conference on Aspect-oriented software development*, Chicago, Illinois, 22-26 March 2005, p. 51 – 62.

English, C., Nixon, P., Terzis, S., McGettrick, A. & Lowe, H. (2002) 'Security Models for Trusting Network Appliances', 5th Annual Workshop on Networked Appliances, Liverpool, England, October 2002.

Esquivel, A., Haya, P.A. & Garc´ıa-Herranz, M. (2007) 'Managing Pervasive Environment Privacy Using the “fair trade” Metaphor', in Meersman, R., Tari, Z. & Herrero, P. (eds), *On the move to meaningful Internet systems: OTM 2007 Workshops*, Springer-Verlag, Berlin, Germany, vol. 4806, Lecture Notes in Computer Science, pp. 804-13.

Etalle, S. & Winsborough, W.H. (2007) 'A Posteriori Compliance Control', in *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, Sophia Antipolis, France, 20-22 June 2007, pp. 11-20.

Falcarin, P., Baldi, M. & Mazzocchi, D. (2004) 'Software Tampering Detection using AOP and mobile code', 3rd International Conference on Aspect-Oriented Software Development (AOSD'04), Lancaster, UK, 22-24 March 2004.

Ferreira, A., Cruz-Correia, R., Antunes, L., Farinha, P., Oliveira-Palhares, E., Chadwick, D.W. & Costa-Pereira, A. (2006) 'How to break access control in a controlled manner', in *Proceedings of the 19th IEEE International Symposium on Computer-Based Medical Systems*, Salt Lake City, Utah, 22-23 June 2006, pp. 847-51.

Georgiev, I.K. & Georgiev, I.I. (2001) 'A security model for distributed computing', *Journal of computing sciences in colleges*, vol. 17, no. 1, pp. 178-86.

Grandison, T.W.A. (2003) 'Trust Management for Internet Applications', PHD thesis, Imperial College London.

Groher, I. & Schulze, S. (2003) 'Generating Aspect Code from UML Models', Workshop on Aspect-Oriented Modeling with UML, AOSD, Boston, Mass. USA, 17-21 March 2003.

Grundy, J. & Ding, G. (2002) 'Automatic Validation of Deployed J2EE Components Using Aspects', in *17th IEEE International Conference on Automated Software Engineering (ASE'02)*, Edinburgh, UK, 23-27 September 2002, pp. 47-57.

Haldar, V., Chandra, D. & Franz, M. (2005) 'Practical, Dynamic Information Flow for Virtual Machines', in *PLID'05 2nd International Workshop on Programming Language Interference and Dependence*, London, UK, 6 September 2005.

Harrison, W. & Ossher, H. (1993) 'Subject-oriented programming: a critique of pure objects', in *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications*, Washington D.C., 26 September – 1 October, pp. 411-28.

Herath, T. & Rao, H.R. (2009) 'Protection motivation and deterrence: a framework for security policy compliance in organisations', *European Journal of Information Systems*, vol. 18, no. 2, pp. 106-25.

Higgins, G.E., Wilson, A.L. & Fell, B.D. (2005) 'An Application of Deterrence Theory to Software Piracy', *Journal of Criminal Justice and Popular Culture*, vol. 12, no. 3, pp. 166-84.

Hong, J.I. & Landay, J.A. (2004) 'An Architecture for Privacy Sensitive Ubiquitous Computing', in *Proccedings of the International Conference on Mobile Systems, Applications and Services*, Boston, Massachusetts, USA, pp. 177-89.

Houmb, S.H., Georg, G., France, R. & Matheson, D. (2004) 'Using aspects to manage security risks in risk-driven development', in *3rd International Workshop on Critical Systems Development with UML*, Lisbon, Portugal, 11-15 October, pp. 71-84.

Imine, A., Cherif, A. & Rusinowitch, M. (2009) *An Optimistic Mandatory Access Control Model for Distributed Collaborative Editors*, Technical Report, INRIA.

Izaki, K., anaka, K. & Takizawa, M. (2001) 'Information Flow Control in Role-Based Model for Distributed Objects', in *Eighth International Conference on Parallel and Distributed Systems*, Kyongju City, Korea, 26-29 June 2001, pp. 363-70.

Jones, R.L. & Rastogi, A. (2004) 'Secure Code: Building Security into the Software Development Life Cycle', *Information Security Journal: A Global Perspective*, vol. 15, no. 5, pp. 29-39.

Jøsang, A. & Patton, M. (2001) *User Interface Requirements for Authentication of Communication*, Technical Report, Distributed Systems Technology Centre, Brisbane, Australia.

Kersten, M. (2005) *AOP Tools Comparison, AOP@Work, DeveloperWorks, IBM*, viewed 1 December 2005, <[Online] Available: http://www.ibm.com/developerworks/library/j-aopwork1>.

Kiczales, G. (1996) 'Aspect-Oriented Programming', *Computing Surveys(CSUR)*, vol. 28, no. 4, p. 154.

Kiczales, G., Hillsdale, E., Hugunin, J., Kersten, M., Palm, J. & Griswold, W.G. (2001) 'Getting Started with AspectJ', *Communications of the ACM*, vol. 44, no. 10, pp. 59-65.

Kiczales, G., Irwin, J., Lamping, J., Loingtier, J., Lopes, C.V., Maeda, C. & Mendhekar, A. (1997) 'Aspect-Oriented Programming ', in Aksit, M. & Matsuoka, S. (eds), *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP)*, Springer-Verlag, Jyväskylä, Finland, vol. 1241, Lecture Notes in Computer Science, pp. 220-40.

Kim, S. & Leem, C.S. (2004) 'An Information Engineering Methodology for the Security Strategy Planning', in Lagana, A. (ed.), *Computational Science and Its Applications - ICCSA 2004*, Springer, Berlin/Heidelberg, vol. 3043, Lecture Notes in Computer Science, pp. 597-607.

Kumar, A., Singh, A.K. & Babu, R.S. (2001) 'A security assurance framework for component based software development', *Informatica*, vol. 25, no. 4, pp. 509 - 15.

Läufer, K., Thiruvathukal, G.K., Elrad, T. & Bader, A. (2003) 'Enhancing the CS Curriculum with Aspect Oriented Software Development (AOSD), Working Paper', International Conference on aspect-oriented software development, Boston, 17-21 March 2003.

Lee, G., Kim, W. & Kim, D.-K. (2004) 'Novel Method to Support User's Consent in Usage Control for Stable Trust in E-business', in Lagan, A., Gavrilova, M.L., Kumar, V., Mun, Y., Tan, C.J.K. & Gervasi, O. (eds), *Computational science and its applications - ICCSA 2004*, vol. 3045, Lecture Notes in Computer Science, pp. 906-14.

Li, N., Moa, Z. & Chen, H. (2009) 'Usable Mandatory Access Control for Operating Systems', in Roa, H.R. & Upadhyaya (eds), *Information Assurance, Security and Privacy (Handbooks in Information Systems)*, Emerald Group Publishing Limited, Bingley, UK, vol. 14, pp. 335-63.

Li, X., Naeem, N.A. & Kemme, B. (2005) 'Fine-Granularity Access Control in 3-tier Laboratory Information Systems', in *Proceeding of the 9th Database Engineering and Application Symposium, (IDEAS '05)*, Montreal, Canada, 25-27 July 2005, pp. 391- 7.

Mao, Z., Li, N., Chen, H. & Jiang, X. (2009) 'Trojan horse resistant discretionary access control', in *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, Stresa, Italy, pp. 237-46.

March, M.T. & Smith, G.F. (1995) 'Design and natural science research on information technology ', *Decision Support Systems*, vol. 15, no. 4, pp. 251-66.

Masuhara, H. & Kawauchi, K. (2003) 'Dataflow Pointcut in Aspect-Oriented Programming', in Ohori, A. (ed.), *Proceedings of The First Asian Symposium on Programming Languages and Systems (APLAS'03)*, Springer, Beijing, China, vol. 2895, Lecture Notes in Computer Science, pp. 105-21.

McCollum, C.J. & Messing, J.R.N.L. (1990) 'Beyond the Pale of MAC and DAC Defining new forms of access control', in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, USA, 17-19 May 1990, pp. 190-200.

Miller, S.K. (2001) 'Aspect-Oriented Programming Takes Aim at Software Complexity', *Computer*, vol. 34, no. 4, pp. 18-21.

Murphy, G.C., Walker, R.J. & Baniassad, E.L.A. (1999) 'Evaluating Emerging Software development Technologies: Lessons learned from Assessing Aspect-Oriented Programming', *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 483-55.

Murphy, G.C., Walker, R.J., Baniassad, E.L.A., Rollibard, M.P., Lai, A. & A., K.M. (2001) 'Does aspect-oriented programming work?' *Communications of the ACM*, vol. 44, no. 10, pp. 75-7.

Offerman, P., Levina, O., Schonherr, M. & Bub (2009) 'Outline of a design science research process', in *4th International Conference on Design Science Research into Systems and Technology*, Malvern, Pennsylvania, 6-9 May 2009.

Osborn, S., Sandhu, R. & Munawer, Q. (2000) 'Configuring Role-Based Access Control to Enforce Mandatory and Discretionary', *ACM Transactions on Information and System Security*, vol. 3, no. 2, pp. 85-106.

Padayachee, K. (2007) 'Instrumentation of AspectJ Programs: An Exploratory Study', in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2007*, Hong Kong, vol. 1, 21- 23 March 2007, pp. 1077-81.

Padayachee, K. & Eloff, J.H.P. (2006) 'The Next Challenge: Aspect-Oriented Programming', in Nyongesa, H. (ed.), *Proceedings of the Sixth IASTED International Conference on Modelling, Simulation and Optimization*, Gaborone, Botswana, 11-13 September 2006, pp. 304-7.

Padayachee, K. & Eloff, J.H.P. (2007) 'Enhancing Optimistic Access Controls with Usage Control', in Lambrinoudakis, C., Pernul, G. & Tjoa, A.M. (eds), *Trust, Privacy and Security in Digital Business*, Springer, Regensburg, Germany, vol. 4657, Lecture Notes in Computer Science, pp. 75 - 82.

Padayachee, K. & Eloff, J.H.P. (2009) 'Adapting usage control as a deterrent to address the inadequacies of access controls', *Computers and Security*, vol. 28, no. 7, pp. 536-44.

Padayachee, K. & Wakaba, N. (2007) 'A Taxonomy of Aspect-Oriented Security', The 2007 European Applied Business Research Conference, Venice, Italy, 4-7 June 2006.

Park, J., Zhang, X. & Sandhu, R. (2004) 'Attribute Mutability in Usage Control', in *Proceedings of the annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Sitges, Catalonia, Spain, 26 July 2004, pp. 15-29.

Pavlich-Mariscal, J., Michel, L. & Demurjian, S. (2005) 'A Formal Enforcement Framework for Role-Based Access Control using Aspect-Oriented Programming', in *Proceedings of ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2005)*, Montego Bay, Jamaica, 2-7 October 2005, pp. 537-52.

Pfleeger, C.P. (1997) *Security in Computing*, 2nd edn, Engelwood Cliffs, NJ.:Prentice Hall, United States of America.

Pfleeger, C.P. & Pfleeger, S.L. (2003) *Security in Computing*, 3rd edn, Prentice Hall, Upper Saddle River, New Jersey.

Pieprzyk, J., Hardjono, T. & Seberry, J. (2003) *Fundamentals of computer security*, Springer, Berlin.

Pohl, C., Charfi, A., Gilani, W., Göbel, S. & B.G., H. (2008) 'Adopting Aspect-Oriented Software Development in Business Application Engineering', 7th International Conference on Aspect-Oriented Development, Brussels, Belgium, 31 March - 4 April 2008.

Povey, D. (1999) 'Optimistic Security: A New Access Control Paradigm', Proceedings of the 1999 workshop on New security paradigms, Caledon Hills, Ontario, Canada, 22 - 24 September 1999.

Pretschner, A., Hilty, M., Schutz, F., Schaefer, C. & Walter, T. (2008) 'Usage Control Enforcement: Present and Future', *IEEE S ecurity & Privacy*, vol. 6, no. 4, pp. 44-53.

Pretschner, A. & Walter, T. (2008) 'Negotiation of Usage Control Policies - Simply the Best?' in *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, Washington, DC, USA, 4-7 March 2008, pp. 1135-6.

Pudney, P. (2003) *e-Consent in consumer health & telemedicine*, University of South Australia, viewed 30 November 2009, <[Online] Available: http://www.pudney.net.au/~phillip/papers/econsent.pdf>.

Raje, R.R., Zhong, M. & Wang, T. (2001) 'Case Study: A Distributed Concurrent System with AspectJ', *ACM SIGAPP Applied Computing Review*, vol. 9, no. 2, pp. 17-23.

Ramachandran, R., Pearce, D.J. & Welch, I. (2006) 'AspectJ for Multilevel Security', The 5th AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS), Bonn, Germany, 2006, 21 March 2006.

Rjaibi, W. & Bird, P. (2004) 'A Multi-Purpose Implementation of Mandatory Access Control in Relational Database Management Systems', in *Proceedings of 30th VLDBases Conference*, Toronto, Canada, pp. 1010-20.

Robinson, P., Rits, M. & Kilian-Kehr, R. (2004) 'An Aspect of Application Security Management', Proceedings of the 2nd International Workshop AOSDSEC'04, Lancaster, UK, March 2004.

Russell, D.F. & Gangemi, G.T. (1991) *Computer Security Basics*, O'Reilly Media and Associate, Sebastopol, California.

Samarati, P., Bertino, E., Ciampichetti, A. & Jajodia, S. (1997) 'Information Flow Control in Object-Oriented Systems', *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 4, pp. 524-38.

Samarati, P. & de Capitani di Vimercati, S. (2001) 'Access control: Policies, models, and mechanisms', in Focardi, R. & Gorrieri, R. (eds), *Foundations of Security Analysis and Design*, Springer-Verlag, Berlin, vol. 2172, Lecture Notes in Computer Science, pp. 137-96.

Sandhu, R. & Park, J. (2003) 'Usage Control: A Vision for Next Generation Access Control', in Gorodetsky, V., Popyack, L.J. & Skormin, V.A. (eds), *Computer Network Security*, Springer, Berlin/Heidelberg, vol. 2776, Lecture notes in Computer Science, pp. 17-31.

Sandhu, R.S. (2001) 'Future Directions in Role-Based Access Control Models', in Gorodetski, V.I., Skormin, V.A. & Popyack, L.J. (eds), *Information Assurance in Computer Networks*, Springer, Berlin, Heidelberg, vol. 2052, Lecture Notes in Computer Science, pp. 22-6.

Sandhu, R.S., Coyne, E.J., Feinstein, H.L. & Youman, C.E. (1996) 'Role-Based Access Control Models', *IEEE computer*, vol. 29, no. 2, pp. 38-47.

Shah, V. & Hill, F. (2003) *An Aspect-Oriented Security Assurance Solution, Defence Advanced Research Projects Agency*, viewed 1 December 2005, <[Online] Available: http://www.stormingmedia.us/50/5039/A503914.html>.

Shin, W. & Yoo, S.B. (2007) 'Secured Web Services Based on Extended Usage Control', in Washio, T.Z., Z-H., Huang, J.Z., Hu, X., Li, J., Xie, C., He, J., Zou, D., Li, K.-C. & Freire, M.M. (eds), *Emerging Technologies in Knowledge Discovery and Data Mining, PAKDD 2007, International Workshops, Nanjing, China, May 22-25, 2007*, Springer, Berlin, vol. 4819, Lecture Notes in Computer Science, pp. 656-63.

Singh, A. (2005) 'The Scalablity of AspectJ', MSC thesis, University of California, Davis.

Slowikowski, P. & Zielinski, K. (2003) 'Comparison Study of Aspect-oriented and Container Managed Security', Proceedings of the ECOOP workshop on analysis of Aspect-Oriented Software, Darmstadt, Germany, 21-25 July 2003.

Stevens, G. & Wulf, V. (2002) 'A New Dimension in Access Control: Studying Maintenance Engineering across Organizational Boundaries', Proceedings of the ACM conference on Computer Supported Cooperative Work (CSCW), New Orleans, Louisiana, USA, 16 -20 November 2002.

Syalim, A., Tabata, T. & Sakurai, K. (2005) 'Usage Control Model and Architecture for Data Confidentiality in a Database Service Provider', in *Indonesia Cryptology and Information Security Conference*, Jakarta, Indonesia, 30-31 March 2005, pp. 155-60.

Tolone, W., Ahn, G.-J., Pai, T. & Hong, S.-P. (2005) 'Access Control in Collaborative Systems', *Acm Computing Surveys*, vol. 37, no. 1, pp. 29-41.

Tymann, P.T. & Schneider, G.M. (2008) *Modern Software Development using Java*, 2nd edn, Thomson Course Technology, Boston, Massachusetts.

Ubayashi, N., Masuhara, H. & Tamai, T. (2004) 'An AOP Implementation Framework for Extending Joint Point Models', Proceedings of the ECOOP' 2004 Workshop on Reflection, AOP and Meta-Data for Software Evolution, Oslo, Norway, 15 June 2004.

Vanhaute, B. & De Win, B. (2001) 'AOP, Security and Genericity', 1st Belgian AOSD Workshop, Vrije Universiteit Brussel, Brussels, Belgium, 8 November 2001.

Verhanneman, T., Piessens, F., De Win, B. & Joosen, W. (2005) 'Uniform Application-level Access Control Enforcement of Organizationwide Policies', in *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, Tucson, Arizona, 5-9 December 2005, pp. 431-40.

Viega, J., Bloch, J.T. & Chandra, P. (2001) 'Applying Aspect-Oriented Programming to Security', *Cutter IT Journal*, vol. 14, no. 2, pp. 31-9.

Viega, J. & Evans, D. (2000) 'Separation of concerns for security', in Tarr, P., Harrison, W., Ossher, H., Finkelstein, A., Nuseibeh, B. & Perry, D. (eds), *ICSE 2000 Workshop on Multi-Dimensional Separation of Concerns in Software Engineering*, Limerick, Ireland, 10 June 2000, pp. 125-38.

Viega, J. & Voas, J. (2000) 'Can Aspect-Oriented Programming Lead to More Reliable Software', *IEEE Software*, vol. 17, no. 6, pp. 19-21.

Wakaba, N. 2004, 'A Taxonomy of Aspect-Oriented Security (Honours Project), Unpublished Dissertation', University of South Africa.

Walker, R.J., Baniassad, E.L.A. & Murphy, G.C. (1999) 'An initial assessment of aspect-oriented programming', in *Proceedings of the 21st international conference on Software engineering*, Los Angeles, California, 16-22 May 1999, pp. 120-30.

Wang, H., Zhang, Y. & Cao, J. (2006) 'Ubiquitous Computing Environments and Its Usage Access Control', in *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, Hong Kong, 29 May - 1 June 2006, pp. 6-16.

Wegner, P. (1990) 'Concepts and Paradigms of Object-Oriented Programming', *ACM SIGPLAN OOPS Messenger*, vol. 1, no. 1, pp. 7-87.

Weippl, E. & Essmayr, W. (2003) 'Personal Trusted Devices for Web Services: Revisiting Multilevel Security', *Mobile Networks and Applications*, vol. 8, no. 2, pp. 151-7.

Whitten, A. & Tygar, J.D. (1999) 'Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0', Proceedings of the 8th USENIX Security Symposium, Washington DC, America, 23-26 August 1999.

Xu, Z., Feng, D., Li, L. & Chen, H. (2003) 'UC-RBAC: A Usage Constrained Role-Based Access Control Model ', in Qing, S., Gollmann, D. & Zhou, J. (eds), *Information and Communications Security*, Springer, Berlin, vol. 2836, Lecture Notes in Computer Science, pp. 337-47.

Zakrzewski & Haddad, I. (2002) *Linux Distributed Security Module*, viewed 26 October 2007, <[Online]. Available: http://www.linuxjournal.com/article/6215>.

Zhang, X., Nakae, M., Covington, M.J. & Sandhu, R. (2006) 'A Usage-based Authorization Framework for Collaborative Computing Systems', in *Proceeding of Symposium on Access Control Models and Technologies (SACMAT'06)*, Lake Tahoe, California, USA, 7-9 June 2006, pp. 180-9.

Zhao, B., Sandhu, R., Zhang, X. & Qin, X. (2007) 'Towards a Times-Based Usage Control Model', in Barker, S. & Ahn, G.-J. (eds), *Data and Applications Security XXI*, Springer, Berlin/Heidelberg, vol. 4602, Lecture Notes in Computer Science, pp. 227-42.

Zhao, X. & Johnson, M.E. (2008) 'Access Flexibility with Escalation and Audit', in *20th Workshop on Information Systems and Economics (WISE 2008)*, Paris, France, 13-14 December 2008.

Zheng, L.M. & Myers, A.C. (2004) *Dynamic Security Labels and Noninterference,Technical Report 2004.*, viewed 26 October 2005, <[Online]. Available: www.cs.cornell.edu/andru/papers/dynlabel.pdf>.

Zurko, M.E. (2005) 'User-Centered Security: Stepping Up to the Grand Challenge', 21st Annual Computer Security Applications Conference 2005, Tucson, Arizona, USA, 5-9 December 2005.

INDEX

U

W

APPENDIX A: PUBLICATIONS

The next challenge: Aspect-Oriented Programming

Abstract:Computer Science educationists face many challenges due to the rapid evolution in technology. One of the more recent challenges was the introduction of object-oriented programming to the computing curriculum. There have been many articles based on the difficulties encountered in teaching object-oriented programming and many solutions proposed in response. While some problems remain unresolved, the pressure to keep abreast of technology remains. The next hurdle that academics may face will be incorporating aspect-oriented programming into the curriculum. Although aspect-oriented programming is not yet ubiquitous in industry it is receiving considerable attention from research and practitioner communities alike. Increasingly academics will encounter the tension between teaching the fundamentals and introducing real-world technologies such as aspect-oriented programming that address real-world concerns. This paper addresses this particular notion, together with the challenges that will be faced if aspect-oriented programming is introduced into the computer science curriculum.

Reference:

Padayachee K. & Eloff J.H.P. 2006. The Next Challenge: Aspect-Oriented Programming, In: The Sixth IASTED International Conference on MODELLING, SIMULATION, AND OPTIMIZATION (MSO 2006) ACTA Press, Gaborone, Botswana, 11-13 September 2006, pages 123-127

An Aspect-Oriented Implementation of e-Consent to Foster Trust

Abstract: As society becomes increasingly dependent on software, there is an increasing expectation of information systems to protect the individual's right to privacy. The process of attaining electronic consent (e-Consent) may perhaps improve the trust that society has in information systems to protect these rights. However, an issue such as e-Consent is usually not given due consideration, as it is a non-functional issue and the implementation of the e-consent mechanism in disparate and legacy systems is difficult. Hence many systems are implemented without such types of controls. Evidently, aspect-oriented software design is highly extensible, as security concerns may be easily integrated into a completed software product. In this paper it is proposed that aspect-oriented programming be used to augment an existing system with electronic consent.

Reference: Padayachee, K. & Eloff J.H.P. 2006. Aspect-Oriented Implementation of e-Consent to foster Trust, In: SAICSIT 2006: Service-oriented software and Systems, Cape Town, South Africa, 9 - 11 October 2006, pages 164-169

An Aspect-Oriented Model to Monitor Misuse

Abstract: The efficacy of the aspect-oriented paradigm has been well established within several areas of software security as aspect-orientation facilitates the abstraction of these security-related tasks to reduce code complexity. The aim of this paper is to demonstrate that aspect-orientation may be used to monitor the information flows between objects in a system for the purposes of misuse detection. Misuse detection involves identifying behavior that is close to some previously defined pattern signature of a known intrusion.

Reference: Padayachee, K. & Eloff, J.H.P. 2006. An Aspect-Oriented Model to Monitor Misuse, International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering, In: Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, Springer (Netherlands), pages: 273 -278, December 2006

An Aspect-Oriented Approach to Enhancing Multilevel Security with Usage Control: An Experience Report

Abstract: The aim of this paper is to document experiences with augmenting multilevel security with usage control at the application level within the aspect-oriented paradigm. Multilevel access control is an access control policy that supports systems that process especially sensitive data. However, attribute-based access control is sometimes insufficient and needs to be combined with additional features in order to meet the demands of modern applications and systems. Usage control enables finer-grained control over the usage of digital objects than do traditional access control policies and models.

Reference:

Padayachee, K. & Eloff, J.H.P. 2007. An Aspect-Oriented Approach to Enhancing Multilevel Security with Usage Control: An Experience Report, In: IAGENG: Lecture Notes in Engineering and Computer Science Volume 1 - International, Conference on Software Engineering (ICSE'07), Hong Kong, 21 - 23 March 2007, Hong Kong: Newswood Ltd. International Association of Engineers (Hong Kong), pages: 1060 - 1065

Enhancing Optimistic Access Controls with Usage Control

Abstract: With the advent of agile programming, lightweight software processes are being favoured over the highly formalised approaches of the past. Likewise, access control may benefit from a less prescriptive approach with an increasing reliance on users to behave ethically. These ideals correlate with optimistic access controls. However, ensuring that users behave in a trustworthy manner may require more than optimistic access controls. This paper investigates the possibility of enhancing optimistic access controls with usage control to ensure that users conduct themselves in a trustworthy manner. Usage control enables finer-grained control over the usage of digital objects than do traditional access control policies and models. Further to ease the development and maintenance of usage control measures, it is posited that it is completely separated from the application logic by using aspect-oriented programming.

Reference: Padayachee, K. and Eloff J.H.P. 2007. Enhancing Optimistic Access Controls with Usage Control, In: Lecture Notes in Computer Science: Trust, Privacy and Security in Digital Business, Springer (Germany), Volume 4657 Regensburg, Germany, September 3-7, 2007, pages: 75 – 82.

Adapting Usage Control as a Deterrent to address the Inadequacies of Access Controls

Abstract: Access controls are difficult to implement and evidently deficient under certain conditions. Traditional controls offer no protection for unclassified information, such as a telephone list of employees that is unrestricted, yet available only to members of the company. On the opposing side of the continuum, organizations such as hospitals that manage highly sensitive information require stricter access control measures. Yet, traditional access control may well have inadvertent consequences in such a context. Often, in unpredictable circumstances, users that are denied access could have prevented a calamity had they been allowed access. It has been proposed that controls such as auditing and accountability policies be enforced to deter rather than prevent unauthorized usage. In dynamic environments preconfigured access control policies may change dramatically depending on the context. Moreover, the cost of implementing and maintaining complex preconfigured access control policies sometimes far outweighs the benefits. This paper considers an adaptation of usage control as a proactive means of deterrence control to protect information that cannot be adequately or reasonably protected by access control.

Reference: Padayachee K, Eloff JHP (2009), Adapting usage control as a deterrent to address the inadequacies of access controls, Computers and Security (2009), Vol 28, No. 7, pages 536-544

APPENDIX B:
OOP DOCUMENTATION

For full documentation refer to the accompanying CD.

Hierarchy For All Packages

Package Hierarchies:

accessobject, authenticationSim, authorizationSim, components, testutilities, usagecontrol

---

Class Hierarchy

- class java.lang.Object
  - class accessobject.**AccessInformation** (implements java.lang.Runnable)
    - class accessobject.**Access**
    - class usagecontrol.**BreakTheGlass**
    - class usagecontrol.**OngoingConditions**
    - class usagecontrol.**OngoingObligations**
  - class javax.swing.plaf.basic.BasicComboBoxEditor (implements javax.swing.ComboBoxEditor, java.awt.event.FocusListener)
    - class components.**JSearchableComboBox.SearchEditor**
  - class components.**CharUtility**
  - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.Container
      - class javax.swing.JComponent (implements java.io.Serializable)
        - class javax.swing.JComboBox (implements javax.accessibility.Accessible, java.awt.event.ActionListener, java.awt.ItemSelectable, javax.swing.event.ListDataListener)
          - class components.**JSearchableComboBox**
        - class javax.swing.JPanel (implements javax.accessibility.Accessible)
          - class components.**CheckBox** (implements java.awt.event.ItemListener)
          - class components.**Demo**
          - class components.**image** (implements java.awt.event.ActionListener)
      - class java.awt.Window (implements javax.accessibility.Accessible)
        - class java.awt.Frame (implements java.awt.MenuContainer)
          - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class components.**ImageFrame**
  - class components.**DoublyLinkedList**
  - class components.**DoublyLinkedList.DLLIterator**
  - class components.**DoublyLinkedList.DLLNode**
  - class testutilities.**MemoryUsage**
  - class authorizationSim.**MyCallbackHandler** (implements javax.security.auth.callback.CallbackHandler)
  - class authorizationSim.**SampleAuthorization** (implements java.security.PrivilegedAction)
  - class authorizationSim.**SampleAzn**
  - class authenticationSim.**SampleLoginModule** (implements javax.security.auth.spi.LoginModule)
  - class authorizationSim.**SamplePrincipal** (implements java.security.Principal, java.io.Serializable)
  - class components.**TernarySearchTree**
  - class components.**TernarySearchTree.TSTNode**

- class usagecontrol.**UsageControl**

Java Documentation for Class BreakTheGlass

usagecontrol

Class BreakTheGlass

```
java.lang.Object
  └accessobject.AccessInformation
      └usagecontrol.BreakTheGlass
```

All Implemented Interfaces:

java.lang.Runnable

---

public class **BreakTheGlass**
extends AccessInformation

Field Summary

Fields inherited from class accessobject.AccessInformation
AccessType, aThread, ObjectName, SubjectName

Constructor Summary

(package private)	**BreakTheGlass**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary

(package private) boolean	**display**()

Methods inherited from class accessobject.AccessInformation
endrequest, getAccessType, getObject, getSubjectName, run

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BreakTheGlass

```
BreakTheGlass(java.lang.String SubjectName,
              java.lang.String ObjectName,
              java.lang.String AccessType)
```

Method Detail

display

```
boolean display()
```

Java Documentation for Class OngoingConditions

usagecontrol

Class OngoingConditions

```
java.lang.Object
  └accessobject.AccessInformation
      └usagecontrol.OngoingConditions
```

All Implemented Interfaces:

java.lang.Runnable

---

public class **OngoingConditions**
extends AccessInformation
Author:

Keshnee Padayachee

Field Summary

private static long	**condition** Controls actions relating to the conditions of access
private boolean	**stop**

Fields inherited from class accessobject.AccessInformation
AccessType, aThread, ObjectName, SubjectName

Constructor Summary

OngoingConditions(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary

boolean	**conditionisValid**()
void	**conditionsWarning**()
void	**endOngoingConditions**()
long	**getCondition**()
void	**run**()

Methods inherited from class accessobject.AccessInformation
endrequest, getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

condition

```
private static long condition
```

Controls actions relating to the conditions of access

stop

```
private volatile boolean stop
```

Constructor Detail

OngoingConditions

```
public OngoingConditions(java.lang.String SubjectName,
                         java.lang.String ObjectName,
                         java.lang.String AccessType)
```

Method Detail

conditionsWarning

`public void` **`conditionsWarning`**`()`

getCondition

`public long` **`getCondition`**`()`

conditionisValid

`public boolean` **`conditionisValid`**`()`

run

`public void` **`run`**`()`

Specified by:
`run` in interface `java.lang.Runnable`
Overrides:
`run` in class `AccessInformation`

endOngoingConditions

`public void` **`endOngoingConditions`**`()`

Java Documentation for Class OngoingObligations

usagecontrol

Class OngoingObligations

```
java.lang.Object
  └accessobject.AccessInformation
      └usagecontrol.OngoingObligations
```

All Implemented Interfaces:
java.lang.Runnable

public class **OngoingObligations**
extends AccessInformation

Field Summary

private ImageFrame	**OngoingObligationsRequest** Controls actions relating the OngoingObligations of the Access

Fields inherited from class accessobject.AccessInformation
AccessType, aThread, ObjectName, SubjectName

Constructor Summary

OngoingObligations(java.lang.String SubjectName, java.lang.String ObjectName,java.lang.String AccessType)

Method Summary

void	**endOngoingObligations**()
void	**run**()

Methods inherited from class accessobject.AccessInformation
endrequest, getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

OngoingObligationsRequest

```
private ImageFrame OngoingObligationsRequest
```

Controls actions relating the OngoingObligations of the Access

Constructor Detail

OngoingObligations

```
public OngoingObligations(java.lang.String SubjectName,
                          java.lang.String ObjectName,
                          java.lang.String AccessType)
```

Method Detail

run

```
public void run()
```

Specified by:
run in interface java.lang.Runnable
Overrides:
run in class AccessInformation

---

endOngoingObligations

```
public void endOngoingObligations()
```

Java Documentation Class UsageControl

usagecontrol

Class UsageControl

```
java.lang.Object
  └ usagecontrol.UsageControl
```

public class **UsageControl**
extends java.lang.Object

Field Summary

private Access	**accessObject**
private static boolean	**accessOpen** Controls the Usage control of an Access to an Object
private java.lang.Thread	**accessThread**
(package private) java.lang.String	**AccessType**
private BreakTheGlass	**breakTheGlass**
static boolean	**conditionsInvalid**
private java.lang.Thread	**conditionsThread**
static boolean	**endAccess**
static boolean	**endObligations**
(package private) java.lang.String	**ObjectName**
private java.lang.Thread	**obligationsThread**
private OngoingConditions	**OnConditions**
private OngoingObligations	**Onobligations**
private static boolean	**preCondition**
(package private) java.lang.String	**SubjectName**

Constructor Summary

UsageControl(Access AccessObject)

Method Summary

(package private) boolean	**breakTheGlass**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)
boolean	**checkAccessType**()
(package private) void	**initiateBreakTheGlassFacility**()
boolean	**initiateUsageControl**()
(package private) void	**logAccess**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType, java.lang.String Notice, java.lang.String RedFlag)
(package private) void	**postAccess**()
void	**postObligations**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AcccessType)
(package private) boolean	**preConditions**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)
(package private) boolean	**preObligations**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)
(package private) void	**stopAccess**()
(package private) void	**stopOngoingObligations**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

accessOpen

`private static boolean` **accessOpen**
Controls the Usage control of an Access to an Object

---

preCondition
`private static boolean` **preCondition**

---

obligationsThread
`private java.lang.Thread` **obligationsThread**

---

conditionsThread
`private java.lang.Thread` **conditionsThread**

---

accessThread
`private java.lang.Thread` **accessThread**

OnConditions
`private` OngoingConditions **OnConditions**

---

accessObject
`private` Access **accessObject**

Onobligations
`private` OngoingObligations **Onobligations**

---

breakTheGlass
`private` BreakTheGlass **breakTheGlass**

SubjectName
`java.lang.String` **SubjectName**

ObjectName
`java.lang.String` **ObjectName**

AccessType
`java.lang.String` **AccessType**

---

conditionsInvalid
`public static boolean` **conditionsInvalid**

---

endAccess
`public static boolean` **endAccess**

---

endObligations
`public static boolean` **endObligations**

Constructor Detail

UsageControl
`public` **UsageControl**`(`Access `AccessObject)`

Method Detail

checkAccessType
`public boolean` **checkAccessType**`()`

---

initiateUsageControl

```
public boolean initiateUsageControl()
```

---

initiateBreakTheGlassFacility

```
void initiateBreakTheGlassFacility()
```

---

stopAccess

```
void stopAccess()
```

---

stopOngoingObligations

```
void stopOngoingObligations()
```

---

preObligations

```
boolean preObligations(java.lang.String SubjectName,
                       java.lang.String ObjectName,
                       java.lang.String AccessType)
```

---

preConditions

```
boolean preConditions(java.lang.String SubjectName,
                      java.lang.String ObjectName,
                      java.lang.String AccessType)
```

---

postAccess

```
void postAccess()
```

---

postObligations

```
public void postObligations(java.lang.String SubjectName,
                            java.lang.String ObjectName,
                            java.lang.String AcccessType)
```

---

breakTheGlass

```
boolean breakTheGlass(java.lang.String SubjectName,
                      java.lang.String ObjectName,
                      java.lang.String AccessType)
```

---

logAccess

```
void logAccess(java.lang.String SubjectName,
               java.lang.String ObjectName,
               java.lang.String AccessType,
               java.lang.String Notice,
               java.lang.String RedFlag)
```

Source Code for class Access

```
package accessobject;
import javax.swing.UIManager;
import usagecontrol.UsageControl;
import components.image;


public class Access extends AccessInformation {
   /**
    * This class controls the object being accessed
    */
   public void request(){
      UIManager.put("swing.boldMetal", Boolean.FALSE);
      image.createAndShowGUI(ObjectName);
   }
```

```
  public Access(String SubjectName, String ObjectName, String AccessType) {
   super(SubjectName,ObjectName,AccessType) ;
  }

public void endrequest() {
   super.endrequest();
   image.close();
}

public void run() {

   super.run();
   UIManager.put("swing.boldMetal", Boolean.FALSE);
   image.createAndShowGUI(ObjectName);

   aThread = Thread.currentThread();
   // Keep going as long as myThread is the same as the current thread.

   while (image.WindowOpen) {
      try {
           Thread.sleep(500); // Tell the thread to sleep for half a second.
      }
      catch (InterruptedException e) {}
      }

      if (!image.WindowOpen){
         endrequest();
         //object-oriented version
         UsageControl.endAccess = true;
         //end object-oriented version
      }
}

}
```

Source code for class AccessInformation

```
package accessobject;

public class AccessInformation implements Runnable{
   /**
    * This class maintains all the details relating to the access
    */
   protected String SubjectName;
   protected String ObjectName;
   protected String AccessType;
   protected Thread aThread;
   public AccessInformation(String subName, String OName, String type) {
      SubjectName = subName;
      ObjectName =  OName;
      AccessType =  type;
   }
   public String getSubjectName()
   {
      return SubjectName;
   }

   public String getObject()
   {
      return ObjectName;
   }
   public String getAccessType()
   {
      return AccessType;
   }
   public void run(){aThread = Thread.currentThread(); }

   public void endrequest() {
      aThread = null;
   }

}
```

Source code for class BreakTheGlass

```
package usagecontrol;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import accessobject.AccessInformation;

public class BreakTheGlass extends AccessInformation{
   /**
    * Provides the BreakTheGlass Interface
    */
   BreakTheGlass(String SubjectName, String ObjectName, String AccessType ) {
            super(SubjectName,ObjectName,AccessType);
   }

   boolean display(){
      String message  = "<html>" + SubjectName +" are you <font color = green> SURE <font
   color=black>" +
      "you want to continue with this access?"
      +"<br>(a) This access will be <font color=red>RED-FLAGGED<font color=black>!!!"
      +"<br>(b) You will have justify this usage to the system adminstrator" ;

      ImageIcon icon = new ImageIcon("c:\\icons\\policestop.gif");

      int answer = JOptionPane.showConfirmDialog(null, message,"BREAK THE GLASS IN CASE OF
      EMERGENCY",
      JOptionPane.YES_NO_OPTION,JOptionPane.INFORMATION_MESSAGE, icon);

      if (answer == JOptionPane.YES_OPTION) {
         return true;
      }
      else if (answer == JOptionPane.NO_OPTION) {
         return false;
      }
      return false;
   }
}
```

Source code for class OngoingConditions

```
package usagecontrol;
import javax.swing.*;
import accessobject.AccessInformation;

public class OngoingConditions extends AccessInformation{
   /**
    * Controls actions relating to the conditions of access
    */
   private static long condition = 0;
   private volatile boolean stop = false;
   public OngoingConditions(String SubjectName, String ObjectName, String AccessType ) {
         super(SubjectName,ObjectName,AccessType);

   }

   // This will terminate the run() method.
    public void conditionsWarning(){
      ImageIcon icon = new ImageIcon("c:\\icons\\warn1.gif");
      String message = "<html> <font color=blue> "+ SubjectName
                  +", is <font color = red> PROHIBITED<font color=blue> "
                  +"from accessing client file: " + ObjectName + " after    working hours";
      JOptionPane.showMessageDialog(null, message ,"CONDITIONS WARNING",
      JOptionPane.INFORMATION_MESSAGE,icon);

      //object-oriented version
      UsageControl.conditionsInvalid = true;
      //end of object-oriented version
    }

    public long getCondition(){
      condition++;
      return condition;
   }
```

```
    public boolean conditionisValid()
    {   condition++;
      if (condition%10 == 0)
         return false;
      else
         return true;
    }
     public void run() {
      super.run();
      while(conditionisValid()){
      try {
           Thread.sleep(1000); // Tell the thread to sleep for a second.
      }
      catch (InterruptedException e) {}
      }
      if (!stop){
         conditionsWarning();
      }
      }
    public void endOngoingConditions(){
      stop = true;
    }
}
```

Source code for class OngoingObligations

```
package usagecontrol;


import java.awt.Color;
import javax.swing.ImageIcon;
import components.ImageFrame;
import accessobject.AccessInformation;

public class OngoingObligations extends AccessInformation{
  /**
   * Controls actions relating the OngoingObligations of the Access
   */
  private ImageFrame OngoingObligationsRequest;
  public OngoingObligations(String SubjectName, String ObjectName, String AccessType) {
       super(SubjectName,ObjectName,AccessType);
  }

  public void run() {
    super.run();
    String Message = "<html><font color = green>" + SubjectName+ "   ACCESSING...client
  file: "
                    + ObjectName + " WITH RIGHTS "+         AccessType+". <br> ";
    Message.toUpperCase();
    ImageIcon icon = new ImageIcon("c:\\files\\OBS.jpg");
    OngoingObligationsRequest = new ImageFrame(600,300,400,400,"Ongoing
    Obligations",Message,icon);
    OngoingObligationsRequest.setForeground(Color.BLUE);
    OngoingObligationsRequest.setResizable(false);
    // Keep going as long as myThread is the same as the current thread.

    while (OngoingObligationsRequest.windowOpen()) {
       try {
          Thread.sleep(500); // Tell the thread to sleep for half a second.
       }
       catch (InterruptedException e) {}
    }
    if (!OngoingObligationsRequest.windowOpen()){
          OngoingObligationsRequest.close();
          //object-oriented version
          UsageControl.endObligations = true;
          //end of object-oriented version
    }
    }
  public void endOngoingObligations()
  { UsageControl.endObligations = true;
    if (OngoingObligationsRequest.windowOpen()){
```

```
      OngoingObligationsRequest.close();
  }
  }
}
```

Source Code for class UsageControl

```
package usagecontrol;

import javax.swing.JOptionPane;
import javax.swing.ImageIcon;
import accessobject.Access;
import components.CheckBox;

public class UsageControl {
  /**
   * Controls the Usage control of an Access to an Object
   */
  private static boolean accessOpen;
  private static boolean preCondition = true;
  private Thread obligationsThread;
  private Thread conditionsThread;
  private Thread accessThread;
  private OngoingConditions OnConditions;
  private Access accessObject;
  private OngoingObligations Onobligations;
  private BreakTheGlass breakTheGlass;
  String SubjectName;
  String ObjectName;
  String AccessType;

  public static boolean conditionsInvalid = false;
  public static boolean endAccess = false;
  public static boolean endObligations = false;

  public UsageControl(Access AccessObject)
  {
    accessObject = AccessObject;
    SubjectName = AccessObject.getSubjectName();
    ObjectName = AccessObject.getObject();
    AccessType = AccessObject.getAccessType();
  }

  public boolean checkAccessType(){
    return true;
  }

     public boolean initiateUsageControl(){
    accessOpen = true;
    conditionsInvalid = false;
    endObligations = false;
    endAccess = false;
    if (preObligations(SubjectName, ObjectName, AccessType))
       {
       if (preConditions(SubjectName, ObjectName, AccessType)
           || breakTheGlass(SubjectName, ObjectName, AccessType)){
           accessThread = new Thread(accessObject);
           accessThread.start();

           Onobligations = new OngoingObligations( SubjectName, ObjectName, AccessType);
           obligationsThread = new Thread(Onobligations);
           obligationsThread.start();

           OnConditions = new OngoingConditions(SubjectName,ObjectName,AccessType);
           conditionsThread = new Thread(OnConditions);
           conditionsThread.start();

           while(accessOpen){
           try {
             Thread.sleep(500);
             if (conditionsInvalid){
                initiateBreakTheGlassFacility();
             }
             if (endAccess){
```

```
                stopAccess();
             }
             if (endObligations){
                stopOngoingObligations();
             }

          } catch (InterruptedException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
          }
       }
       postObligations(SubjectName,ObjectName,AccessType);
       }
       else
       System.out.println("no conditions");
       }
    else
       System.out.println("no obligations");
       return true;
  }

  /**
   *
   */

     void initiateBreakTheGlassFacility(){

      if (!breakTheGlass(OnConditions.getSubjectName(), OnConditions.getObject(),
OnConditions.getAccessType()))
       {   Onobligations.endOngoingObligations();
           postAccess();
       }
       conditionsInvalid = false;
       }

     void stopAccess(){
       Onobligations.endOngoingObligations();
       if (accessOpen)
       postAccess();
  }

     void stopOngoingObligations(){
       if (accessOpen)
       postAccess();
  }

  boolean preObligations(String SubjectName, String ObjectName, String AccessType){

    ImageIcon icon = new ImageIcon("c:\\icons\\hand.gif");
    String message = "<html>" + SubjectName + ", if you click <font color=green> YES " +
                  "<font color=black> you agree to <font           color=red>NOT<font
color=black>" +
                  " distribute client file: "
                  + ObjectName;
    int answer = JOptionPane.showConfirmDialog(null, message,"PRE-OBLIGATIONS",
              JOptionPane.YES_NO_OPTION,JOptionPane.WARNING_MESSAGE,icon);
    if (answer == JOptionPane.YES_OPTION) {
       return true;
    }
    else if (answer == JOptionPane.NO_OPTION) {
        return false;
    }
    return false;
  }

  boolean preConditions(String SubjectName, String ObjectName, String AccessType){

  if (preCondition) {
    ImageIcon icon = new ImageIcon("c:\\icons\\warn1.gif");
    String message = "<html> <font color=blue>" + SubjectName +", "
    +" <font color=red>PROHIBITED<font color = blue> from accessing client file: " +
    ObjectName + " at this time !" ;
    JOptionPane.showMessageDialog(null, message ,"PRE-CONDITIONS WARNING",
    JOptionPane.INFORMATION_MESSAGE,icon);
    preCondition = false;
```

```
    return false;
  }
  return true;
  }

  void postAccess(){
    accessOpen = false;
    accessObject.endrequest();
    Onobligations.endOngoingObligations();
    OnConditions.endrequest();
    //update logs
  }

  public void postObligations(String SubjectName, String ObjectName, String AcccessType){
    CheckBox.createAndShowGUI();
  }

  boolean breakTheGlass(String SubjectName, String ObjectName, String AccessType){
    breakTheGlass = new BreakTheGlass(SubjectName,ObjectName,AccessType);
    if (breakTheGlass.display()){
       logAccess(SubjectName,ObjectName,AccessType,"Illegal Access", "YES");
       return true;
    }
    return false;
  }

  void logAccess(String SubjectName, String ObjectName, String AccessType, String Notice,
String RedFlag){
       //WRITE TO LOG FILE
  }
}
```

APPENDIX C:
AOP DOCUMENTATION

For full documentation refer to accompanying CD.

Hierarchy For All Packages

Package Hierarchies:

accessobject, authenticationSim, authorizationSim, components, testutilities, usagecontrol

---

Class Hierarchy

- class java.lang.Object
  - class accessobject.**AccessInformation**
    - class accessobject.**Access**
      - class usagecontrol.**OngoingConditions** (implements java.lang.Runnable)
    - class usagecontrol.**BreakTheGlass**
    - class usagecontrol.**OngoingObligations** (implements java.lang.Runnable)
  - class javax.swing.plaf.basic.BasicComboBoxEditor (implements javax.swing.ComboBoxEditor, java.awt.event.FocusListener)
    - class components.**JSearchableComboBox.SearchEditor**
  - class components.**CharUtility**
  - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - class java.awt.Container
      - class javax.swing.JComponent (implements java.io.Serializable)
        - class javax.swing.JComboBox (implements javax.accessibility.Accessible, java.awt.event.ActionListener, java.awt.ItemSelectable, javax.swing.event.ListDataListener)
          - class components.**JSearchableComboBox**
        - class javax.swing.JPanel (implements javax.accessibility.Accessible)
          - class components.**CheckBox** (implements java.awt.event.ItemListener)
          - class components.**Demo**
          - class components.**image** (implements java.awt.event.ActionListener)
      - class java.awt.Window (implements javax.accessibility.Accessible)
        - class java.awt.Frame (implements java.awt.MenuContainer)
          - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class components.**ImageFrame**
  - class components.**DoublyLinkedList**
  - class components.**DoublyLinkedList.DLLIterator**
  - class components.**DoublyLinkedList.DLLNode**
  - class usagecontrol.**IntertypeDeclarationOnAccess**
  - class testutilities.**MemoryUsage**
  - class authorizationSim.**MyCallbackHandler** (implements javax.security.auth.callback.CallbackHandler)
  - class authorizationSim.**SampleAuthorization** (implements java.security.PrivilegedAction)
  - class authorizationSim.**SampleAzn**
  - class authenticationSim.**SampleLoginModule** (implements javax.security.auth.spi.LoginModule)

- class authorizationSim.**SamplePrincipal** (implements java.security.Principal, java.io.Serializable)
- class components.**TernarySearchTree**
- class components.**TernarySearchTree.TSTNode**
- class usagecontrol.**UsageControlInjector**

Java Documentation for Class Access:

Class Access

```
java.lang.Object
  └ accessobject.AccessInformation
      └ accessobject.Access
```

Direct Known Subclasses:

OngoingConditions

---

public class **Access**
extends AccessInformation

Advised by: usagecontrol.UsageControlInjector.after(): OngoingAccess..

Aspect declarations: usagecontrol.IntertypeDeclarationOnAccess.declare parents: implements Runnable

Author:

KESHNEE TODO To change the template for this generated type comment go to Window - Preferences - Java - Code Style - Code Templates

---

Inter-Type Method Summary	
void	Access.endrequest() **Declared by:** usagecontrol.IntertypeDeclarationOnAccess
void	Access.run() **Declared by:** usagecontrol.IntertypeDeclarationOnAccess

Inter-Type Field Summary	
package Thread	Access.aThread **Declared by:** usagecontrol.IntertypeDeclarationOnAccess

Field Summary

Fields inherited from class accessobject.AccessInformation
accessType, objectName, subjectName

Constructor Summary
Access(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary	
void	**close**()
void	**request**() **Advised by:** usagecontrol.UsageControlInjector.around(accessobject.Access) : Intercept Request..

Methods inherited from class accessobject.AccessInformation
getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Access

```
public Access(java.lang.String SubjectName,
              java.lang.String ObjectName,
              java.lang.String AccessType)
```

Method Detail

request

```
public void request()
```

Advised by: usagecontrol.UsageControlInjector.around(accessobject.Access) : Intercept Request..

---

close

```
public void close()
```

Java Documentation for class AccessInformation

Class AccessInformation

```
java.lang.Object
  └ accessobject.AccessInformation
```

Direct Known Subclasses:

Access, BreakTheGlass, OngoingObligations

---

public class **AccessInformation**
extends java.lang.Object

Author:

KESHNEE TODO To change the template for this generated type comment go to Window - Preferences - Java - Code Style - Code Templates

---

Field Summary	
protected java.lang.String	**accessType**
protected	**objectName**

java.lang.String	
protected java.lang.String	**subjectName**

Constructor Summary
AccessInformation(java.lang.String subName, java.lang.String OName, java.lang.String type)

Method Summary	
java.lang.String	**getAccessType**()
java.lang.String	**getObject**()
java.lang.String	**getSubjectName**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

subjectName
```
protected java.lang.String subjectName
```

objectName
```
protected java.lang.String objectName
```

accessType
```
protected java.lang.String accessType
```

Constructor Detail

AccessInformation
```
public AccessInformation(java.lang.String subName,
                         java.lang.String OName,
                         java.lang.String type)
```

Method Detail

getSubjectName
```
public java.lang.String getSubjectName()
```

getObject
```
public java.lang.String getObject()
```

getAccessType
```
public java.lang.String getAccessType()
```

Java Documentation for Class BreakTheGlass

Class BreakTheGlass

```
java.lang.Object
  └ accessobject.AccessInformation
      └ usagecontrol.BreakTheGlass
```

---

public class **BreakTheGlass**
extends AccessInformation

Field Summary

Fields inherited from class accessobject.AccessInformation
accessType, objectName, subjectName

Constructor Summary	
(package private)	**BreakTheGlass**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary	
(package private) boolean	**display**()

Methods inherited from class accessobject.AccessInformation
getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BreakTheGlass

```
BreakTheGlass(java.lang.String SubjectName,
              java.lang.String ObjectName,
              java.lang.String AccessType)
```

Method Detail

display

boolean **display**()

AJDocumentation on Aspect IntertypeDeclarationOnAccess

java.lang.Object
└ **usagecontrol.IntertypeDeclarationOnAccess**

public aspect **IntertypeDeclarationOnAccess**
extends java.lang.Object
Author:
KESHNEE

Declare Summary	
	declare parents: implements Runnable **Declared on:** accessobject.Access
package Thread	Access.aThread **Declared on:** accessobject.Access
void	Access.endrequest() **Declared on:** accessobject.Access
void	Access.run() **Declared on:** accessobject.Access

Constructor Summary
IntertypeDeclarationOnAccess()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Declare Detail

declare parents: implements Runnable

Declared on: accessobject.Access

Access.aThread

package Thread **Access.aThread**

Declared on: accessobject.Access

Access.endrequest()

public void **Access.endrequest()**

Declared on: accessobject.Access

Access.run()

public void **Access.run()**

Declared on: accessobject.Access

Constructor Detail

IntertypeDeclarationOnAccess
public **IntertypeDeclarationOnAccess**()

Java Documentation on Class OngoingConditions

java.lang.Object
  └ accessobject.AccessInformation
    └ accessobject.Access
      └ **usagecontrol.OngoingConditions**

All Implemented Interfaces:
java.lang.Runnable

---

public class **OngoingConditions**
extends Access
implements java.lang.Runnable
Author:
PADAYK TODO To change the template for this generated type comment go to Window - Preferences - Java - Code Style - Code Templates

---

Field Summary	
private static long	**condition** Controls actions relating to the conditions of access
private boolean	**stop**

Fields inherited from class accessobject.AccessInformation
accessType, objectName, subjectName

Constructor Summary
OngoingConditions(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary	
boolean	**conditionisValid**()
void	**conditionsWarning**()
void	**endOngoingConditions**()

long	**getCondition**()
void	**run**() **Advised by:** usagecontrol.UsageControlInjector.after(): OngoingCondition..

Methods inherited from class accessobject.Access
close, request

Methods inherited from class accessobject.AccessInformation
getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

condition

```
private static long condition
```

Controls actions relating to the conditions of access

---

stop

```
private volatile boolean stop
```

Constructor Detail

OngoingConditions

```
public OngoingConditions(java.lang.String SubjectName,
                         java.lang.String ObjectName,
                         java.lang.String AccessType)
```

Method Detail

conditionsWarning

```
public void conditionsWarning()
```

---

getCondition

```
public long getCondition()
```

---

conditionisValid

```
public boolean conditionisValid()
```

---

run

```
public void run()
```

Advised by: usagecontrol.UsageControlInjector.after(): OngoingCondition..

Specified by:

run in interface java.lang.Runnable

---

endOngoingConditions

```
public void endOngoingConditions()
```

Java Documentation on Class OngoingObligations

```
java.lang.Object
  └ accessobject.AccessInformation
      └ usagecontrol.OngoingObligations
```

All Implemented Interfaces:
java.lang.Runnable

---

public class **OngoingObligations**
extends AccessInformation
implements java.lang.Runnable

Field Summary	
private ImageFrame	**OngoingObligationsRequest** Controls actions relating the OngoingObligations of the Access

Fields inherited from class accessobject.AccessInformation
accessType, objectName, subjectName

Constructor Summary
OngoingObligations(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Method Summary	
void	**endOngoingObligations**()
void	**run**() **Advised by:** usagecontrol.UsageControlInjector.after(): OngoingObligation..

Methods inherited from class accessobject.AccessInformation
getAccessType, getObject, getSubjectName

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

OngoingObligationsRequest

`private ImageFrame OngoingObligationsRequest`

Controls actions relating the OngoingObligations of the Access

Constructor Detail

OngoingObligations

```
public OngoingObligations(java.lang.String SubjectName,
                          java.lang.String ObjectName,
                          java.lang.String AccessType)
```

Method Detail

run

`public void run()`

Advised by: usagecontrol.UsageControlInjector.after(): OngoingObligation..

Specified by:

`run` in interface `java.lang.Runnable`

---

endOngoingObligations

`public void endOngoingObligations()`

AJDocumentation on Aspect UsageControlInjector

```
java.lang.Object
  └ usagecontrol.UsageControlInjector
```

---

public aspect **UsageControlInjector**
extends java.lang.Object

Pointcut Summary	
(package private)	Intercept_Request(accessobject.Access)
(package private)	OngoingCondition()
(package private)	OngoingAccess()
(package private)	OngoingObligation()

Advice Summary
around(accessobject.Access): Intercept_Request..
Advises: accessobject.Access.request
after(): OngoingCondition..
Advises: usagecontrol.OngoingConditions.run
after(): OngoingAccess..
Advises: accessobject.Access
after(): OngoingObligation..
Advises: usagecontrol.OngoingObligations.run

Field Summary	
private Access	**accessObject**
private static boolean	**accessOpen**
private java.lang.Thread	**accessThread**
private java.lang.String	**AccessType**
private java.lang.Thread	**conditionsThread**
private java.lang.String	**ObjectName**
private java.lang.Thread	**obligationsThread**
private OngoingConditions	**OnConditions**
private OngoingObligations	**Onobligations**
private static boolean	**preCondition**
private java.lang.String	**SubjectName**

Constructor Summary
UsageControlInjector()

Method Summary	
(package private) boolean	**breakTheGlass**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)
(package private) void	**initiateBreakTheGlassFacility**()
(package private) void	**logAccess**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType, java.lang.String Notice, java.lang.String RedFlag)
boolean	**OptimisticRights**()
(package private) void	**postAccess**()
void	**postObligations**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AcccessType)

(package private) boolean	**preConditions**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)
(package private) boolean	**preObligations**(java.lang.String SubjectName, java.lang.String ObjectName, java.lang.String AccessType)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Pointcut Detail

Intercept_Request(accessobject.Access)

OngoingCondition()

OngoingAccess()

OngoingObligation()

Advice Detail

around

around(accessobject.Access): Intercept_Request..

Advises: accessobject.Access.request

---

after

after(): OngoingCondition..

Advises: usagecontrol.OngoingConditions.run

---

after

after(): OngoingAccess..

Advises: accessobject.Access

---

after

after(): OngoingObligation..

Advises: usagecontrol.OngoingObligations.run

Field Detail

accessOpen
```
private static boolean accessOpen
```

preCondition
```
private static boolean preCondition
```

obligationsThread
```
private java.lang.Thread obligationsThread
```

conditionsThread
```
private java.lang.Thread conditionsThread
```

accessThread
```
private java.lang.Thread accessThread
```

accessObject
```
private Access accessObject
```

OnConditions
```
private OngoingConditions OnConditions
```

Onobligations
```
private OngoingObligations Onobligations
```

SubjectName
```
private java.lang.String SubjectName
```

ObjectName
```
private java.lang.String ObjectName
```

AccessType
```
private java.lang.String AccessType
```

Constructor Detail

UsageControlInjector
```
public UsageControlInjector()
```

Method Detail

OptimisticRights
```
public boolean OptimisticRights()
```

initiateBreakTheGlassFacility
```
void initiateBreakTheGlassFacility()
```

preObligations
```
boolean preObligations(java.lang.String SubjectName,
                       java.lang.String ObjectName,
                       java.lang.String AccessType)
```

preConditions
```
boolean preConditions(java.lang.String SubjectName,
                      java.lang.String ObjectName,
                      java.lang.String AccessType)
```

postAccess
```
void postAccess()
```

postObligations
```
public void postObligations(java.lang.String SubjectName,
                            java.lang.String ObjectName,
```

```
                        java.lang.String AcccessType)
```

breakTheGlass

```
boolean breakTheGlass(java.lang.String SubjectName,
                      java.lang.String ObjectName,
                      java.lang.String AccessType)
```

logAccess

```
void logAccess(java.lang.String SubjectName,
               java.lang.String ObjectName,
               java.lang.String AccessType,
               java.lang.String Notice,
               java.lang.String RedFlag)
```

Source Code for class Access

```
package accessobject;

import javax.swing.UIManager;

import components.image;

public class Access extends AccessInformation{

   public void request(){
      System.out.println("Inside request");
      UIManager.put("swing.boldMetal", Boolean.FALSE);
      image.createAndShowGUI(objectName);
   }

   public Access(String SubjectName, String ObjectName, String AccessType) {
      super(SubjectName, ObjectName, AccessType) ;
   }

   public void close(){
       image.close();
   }
}
```

Source code class AccessInformation

```
package accessobject;

public class AccessInformation {
   protected String subjectName;
   protected String objectName;
   protected String accessType;
   public AccessInformation(String subName, String OName, String type) {
      subjectName = subName;
      objectName =  OName;
      accessType =  type;
   }
   public String getSubjectName()
   {
      return subjectName;
   }

   public String getObject()
   {
      return objectName;
   }
   public String getAccessType()
   {
      return accessType;
   }

}
```

Source Code for BreakTheGlass

```
package usagecontrol;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

import accessobject.AccessInformation;
public class BreakTheGlass extends AccessInformation{
   /**
    * Provides the BreakTheGlass Interface
    */
BreakTheGlass(String SubjectName, String ObjectName, String AccessType ) {
      super(SubjectName,ObjectName,AccessType);
}

boolean display(){
String message   = "<html>" + subjectName +" are you <font color = green> SURE <font
color=black>" +
      "you want to continue with this access?"
      +"<br>(a) This access will be <font color=red>RED-FLAGGED<font color=black>!!!"
      +"<br>(b) You will have to justify this usage to the system administrator" ;

      ImageIcon icon = new ImageIcon("c:\\test0\\policestop.gif");

      int answer = JOptionPane.showConfirmDialog(null, message,"BREAK THE GLASS IN CASE OF
      EMERGENCY",
      JOptionPane.YES_NO_OPTION,JOptionPane.INFORMATION_MESSAGE, icon);

      if (answer == JOptionPane.YES_OPTION) {
         return true;
      }
      else if (answer == JOptionPane.NO_OPTION) {
         return false;
      }
      return false;
   }
}
```

Source code for InterTypeDeclarationOnAccess

```
package usagecontrol;
import javax.swing.UIManager;
import accessobject.*;
import components.image;
/**
 * @author KESHNEE
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public aspect IntertypeDeclarationOnAccess {
   declare parents: Access implements Runnable;
   Thread Access.aThread;
   public void Access.endrequest() {
      aThread = null;
      close();
   }
   public void Access.run() {
      aThread = Thread.currentThread();
      UIManager.put("swing.boldMetal", Boolean.FALSE);
      image.createAndShowGUI(getObject());
      // Keep going as long as myThread is the same as the current thread.

      while (image.WindowOpen) {
         try {
            Thread.sleep(500); // Tell the thread to sleep for half
         }
         catch (InterruptedException e) {}
         }

         if (!image.WindowOpen){
               endrequest();
         }
   }
}
```

```
      }
      return false;
   }
}
```

Source code for OngoingConditions

```
package usagecontrol;
import javax.swing.*;

import accessobject.*;
/**
 * @author PADAYK
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */

public class OngoingConditions extends Access implements Runnable{
   /**
    * Controls actions relating to the conditions of access
    */
   private static long condition = 0;
   private volatile boolean stop = false;
   public OngoingConditions(String SubjectName, String ObjectName, String AccessType ) {
         super(SubjectName,ObjectName,AccessType);

   }

   // This will terminate the run() method.
    public void conditionsWarning(){
      ImageIcon icon = new ImageIcon("c:\\icons\\warn1.gif");
      String message = "<html> <font color=blue> "+ subjectName
                  +", is <font color = red> PROHIBITED<font color=blue> "
                  +"from accessing client file: " + objectName + " after working hours";
      JOptionPane.showMessageDialog(null, message ,"CONDITIONS WARNING",
      JOptionPane.INFORMATION_MESSAGE,icon);
    }

    public long getCondition(){
      condition++;
      return condition;
   }

    public boolean conditionisValid(){
      condition++;
      if (condition%10 == 0)
         return false;
      else
         return true;
    }
   public void run() {
      Thread aThread = Thread.currentThread();
      while(!stop && conditionisValid()){
      try {
           Thread.sleep(1000); // Tell the thread to sleep for a second.
      }

         catch (InterruptedException e) {}
      }
      if (!stop){
         conditionsWarning();
      }
      }
    public void endOngoingConditions(){
      stop = true;
    }
}
```

Source Code for OngoingObligations

```
import java.awt.Color;
import javax.swing.ImageIcon;

import components.ImageFrame;
import accessobject.Access;
import accessobject.AccessInformation;

public class OngoingObligations extends AccessInformation implements Runnable{
   /**
    * Controls actions relating the OngoingObligations of the Access
    */
   private ImageFrame OngoingObligationsRequest;
   public OngoingObligations(String SubjectName, String ObjectName, String AccessType) {
      super(SubjectName,ObjectName,AccessType);
   }

   public void run() {
      Thread aThread = Thread.currentThread();
      String Message = "<html><font color = green>" + subjectName+ " ACCESSING...client file:
      "
      + objectName + " WITH RIGHTS "+ accessType+". <br> ";
      Message.toUpperCase();
      ImageIcon icon = new ImageIcon("c:\\files\\OBS.jpg");
      OngoingObligationsRequest = new ImageFrame(600,300,400,400,"Ongoing
      Obligations",Message,icon);
      OngoingObligationsRequest.setForeground(Color.BLUE);
      OngoingObligationsRequest.setResizable(false);
      // Keep going as long as myThread is the same as the current thread.
      System.out.println("Obligations Window"+objectName);
      while (OngoingObligationsRequest.windowOpen()) {
         try {
            Thread.sleep(500); // Tell the thread to sleep for half a second.
         }
         catch (InterruptedException e) {}
      }
         endOngoingObligations();
      }
   public void endOngoingObligations(){
      if (OngoingObligationsRequest.windowOpen()){
         OngoingObligationsRequest.close();

      }
      }
   }

      stop = true;
    }
}
```

Source code for class UsageControlInjector

```
package usagecontrol;

import javax.swing.JOptionPane;
import javax.swing.ImageIcon;
import testutilities.*;
import accessobject.Access;
import components.CheckBox;

public aspect UsageControlInjector {
   private static boolean accessOpen;
   private static boolean preCondition = true;
   private Thread obligationsThread;
   private Thread conditionsThread;
   private Thread accessThread;
   private Access accessObject;
   private OngoingConditions OnConditions;
   private OngoingObligations Onobligations;
   private String SubjectName;
   private String ObjectName;
   private String AccessType;

   pointcut Intercept_Request(Access AccessObject):
```

```
   execution(* *.request(..)) && !within(UsageControlInjector)
   && target(AccessObject)  ;
   void around (Access AccessObject ): Intercept_Request(AccessObject){
      Runtime runtime = Runtime.getRuntime();
      MemoryUsage.printUsage(runtime);
      accessOpen = true;
      accessObject = AccessObject;
      SubjectName = accessObject.getSubjectName();
      ObjectName = accessObject.getObject();
      AccessType = accessObject.getAccessType();
      if (OptimisticRights()){
         if (preObligations(SubjectName, ObjectName, AccessType))
          {
            if (preConditions(SubjectName, ObjectName, AccessType)
               || breakTheGlass(SubjectName, ObjectName, AccessType)){
                accessThread = new Thread(AccessObject);
                accessThread.start();

                Onobligations = new OngoingObligations( SubjectName, ObjectName, AccessType);
                obligationsThread = new Thread(Onobligations);
                obligationsThread.start();

                OnConditions = new OngoingConditions(SubjectName,ObjectName,AccessType);
                conditionsThread = new Thread(OnConditions);
                conditionsThread.start();

                while(accessOpen){
                try {
                  Thread.sleep(500);
               } catch (InterruptedException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
               }
               }
               postObligations(SubjectName,ObjectName,AccessType);

          }
         }
      }
      MemoryUsage.printUsage(runtime);
   }

   public boolean OptimisticRights(){
      //determine whether this information is subject to optimistic access control
      return true;
   }

   // when there is conditions warning
   pointcut OngoingCondition(): call(* *.conditionsWarning(..) ) && target(OngoingConditions);
   after(): OngoingCondition(){
       initiateBreakTheGlassFacility();
   }

   // when the access ends
    pointcut OngoingAccess()  : call(* *.endrequest(..) ) && target(Access) &&
!within(UsageControlInjector);
   after(): OngoingAccess(){
       postAccess();
   }

   // when the user ends ongoingobligations
   pointcut OngoingObligation()  : call(* *.endOngoingObligations(..) ) &&
target(OngoingObligations) && !within(UsageControlInjector);
   after(): OngoingObligation(){
      postAccess();
   }

   void initiateBreakTheGlassFacility(){
      if (!breakTheGlass(OnConditions.getSubjectName(), OnConditions.getObject(),
OnConditions.getAccessType())){
         Onobligations.endOngoingObligations();
          postAccess();
      }
   }
```

```
   boolean preObligations(String SubjectName, String ObjectName, String AccessType){

      ImageIcon icon = new ImageIcon("c:\\icons\\hand.gif");
      String message = "<html>" + SubjectName + ", if you click <font color=green> YES " +
                  "<font color=black> you agree to <font color=red>NOT<font color=black>" +
                  " distribute client file: "
                  + ObjectName;
      int answer = JOptionPane.showConfirmDialog(null, message,"PRE-OBLIGATIONS",
                JOptionPane.YES_NO_OPTION,JOptionPane.WARNING_MESSAGE,icon);
      if (answer == JOptionPane.YES_OPTION) {
         return true;
      }
      else if (answer == JOptionPane.NO_OPTION) {
          return false;
      }
      return false;
   }

   boolean preConditions(String SubjectName, String ObjectName, String AccessType){

      if (preCondition) {
         ImageIcon icon = new ImageIcon("c:\\icons\\warn1.gif");
         String message = "<html> <font color=blue>" + SubjectName +", "
         +" <font color=red>PROHIBITED<font color = blue> from accessing client file: " +
         ObjectName + " at this time !" ;
         JOptionPane.showMessageDialog(null, message ,"PRE-CONDITIONS WARNING",
                                    JOptionPane.INFORMATION_MESSAGE,icon);
         preCondition = false;
         return false;
      }
      return true;
   }

   void postAccess(){
      accessOpen = false;
      accessObject.endrequest();
      Onobligations.endOngoingObligations();
      OnConditions.endOngoingConditions();
      //update logs
   }

   public void postObligations(String SubjectName, String ObjectName, String AcccessType){
      CheckBox.createAndShowGUI();
   }

   boolean breakTheGlass(String SubjectName, String ObjectName, String AccessType){
      BreakTheGlass breakTheGlass = new BreakTheGlass(SubjectName,ObjectName,AccessType);
      if (breakTheGlass.display()){
         logAccess(SubjectName,ObjectName,AccessType,"Illegal Access", "YES");
         return true;
      }
      return false;
   }

   void logAccess(String SubjectName, String ObjectName, String AccessType, String Notice,
String RedFlag){
         //WRITE TO LOG FILE
   }
}
```

APPENDIX D: PROTOTYPE EVALUATION

D1. Research Methodology

The Design Science Research method was applied. This methodology involves problem identification, design and development, and an evaluation.

D.2 Problem Statement

To validate the concept elucidation of the OAC(UCON) model via evaluative prototyping and to determine if the OAC(UCON) model is perceived as an effective countermeasure against data misuse.

D.3 Design and Development

The product concept was implemented by the researcher using both object-oriented and aspect-oriented techniques. However, in order to remove researcher bias, the product concept was introduced to 14 Honours students at the University of Pretoria. They were not shown the working version as to not to bias their judgement of the concept. Furthermore, as the model was intended to scale up to a real-world scenario, the product specification was placed in a context where optimistic access control enhanced with usage control was viewed within a traditional role-based access control with trust measures. The participants were not expected to use the aspect-oriented approach, as this concept is not currently taught in the syllabus.

Participants were given the following specification as a term assignment:

Using an appropriate open source database application, you will create a simple database to store information about a typical organisation with employees and clients to be serviced. A client record includes inter alia the name, occupation, employer, address, contact details and account details of the client. The employee record includes inter alia the name, salary and period of employment of employees. You will use SQL statements to query the database.

For collaborative purposes the client information is relegated to the public domain, while the employee data is protected by role-based access controls. You need to enforce mixed initiative access controls when the user attempts to access the database.

If the user attempts to access data in the public domain, then he/she is subject to the following usage control mechanisms:

Pre-obligation: The user must click on a button in a dialog box, thereby indicating that he/she agrees not to distribute this information.

Pre-condition: This information may be accessed during business hours only.

Ongoing obligation: A window with the following warning "This dataset must be used EXCLUSIVELY for work-related purposes" is to remain open while the user accesses the information.

Ongoing condition: This information may be accessed during business hours only (same as pre-condition, as it is time dependent). While the pre-condition may have been valid at the time of access, it may become invalid during the access.

Post-obligation: The user must send an e-mail to the administrator if he accessed these databases outside of business hours.

Break-the-glass (BTG): While the user will not be permitted to access the information unless the obligations have been satisfied, he/she will under special circumstances be allowed to access it by utilising the BTG facility even if the pre-conditions or ongoing conditions are invalid.

Post-update: A user's rights to information in the public domain can be modified based on prior usage. Your program should log all access in such a way that there is a secure audit

trail. At the onset each user has a trust level of *high*. However, as they demonstrate their untrustworthiness, the level is downgraded to *medium* and finally to *low*. As their trust level drops, they lose their rights to information in the public domain – i.e. information that they are allowed to access is limited. Users with a *medium* trust level can access most information except for account information. Users with a *low* trust level are not allowed to view account information or contact details. They can be limited to view less sensitive details such as the client's name, occupation, etc.

After the user has accessed the database, his/her trust level is updated by using fuzzy logic. For test purposes, each access can be given a random priority [0, 1]. If the BTG facility was deployed by the user, then the trust level [0,1] is updated, dependent on the priority of the task and the user's previous trust level using the fuzzy matrix given below.

Priority of Task / Previous Trust Level	**High**	**Medium**	**Low**
High	Trust level remains **High**	Trust level downgraded to **Medium**	Trust level downgraded to **Medium**
Medium	Trust level remains **Medium**	Trust level downgraded to **Low**	Trust level downgraded to **Low**
Low	Trust level remains **Low**	Trust level remains **Low**	Trust level remains **Low**

The employee records are protected by role-based access controls. In this system there are three roles, namely *manager*, *administrator* and *user*. The manager can *read*, *delete* and *update* an employee record, whereas an administrator can *read* and *update* an employee record. Users can *read* employee records, but all salary information is concealed.

You need to authenticate users (by means of passwords) and stipulate access control policies for the data in the database. If you prefer Java as your language of implementation,

you could use the Java Sandbox model [9] to authenticate users and stipulate access control policies for the data in the database.

You will need to create a policy file to grant permissions to authenticated users.

You will need to create a login configuration file for authentication.

D.4 Evaluation

This small-scale experiment will test the theory that users' interaction with the prototype will be perceived as an effective countermeasure against data misuse. In order to test the hypothesis, two qualitative data collections will be employed, namely participant observation and open-ended interviewing.

Observation: The idea with participant observation is to determine whether access control measures were implemented successfully.

Qualitative Interview: A qualitative interview is to be conducted to determine the developer's perceptions of the appeal of the prototype in terms of data misuse.

Qualitative Questionnaire: Participants were asked to address the following in terms of the model concept: Weaknesses, Strengths, Improvements, Viability, Applicability, Scalability.

EVALUATION

Indicate whether you agree or disagree with the following statements and give reasons for your answer.

Statements	**Agree/ Disagree**
The product specifications as given in the assignment were ambiguous and incomplete.	Agree [ ] Disagree [ ]
Reason:	
The product specifications as given in the assignment could easily be translated into an implementable product.	Agree [ ] Disagree [ ]
Reason:	
In terms of the enforcement of security, other mechanisms such as a written policy document or adequate training would have been more effective than the mechanisms identified in the product concept.	Agree [ ] Disagree [ ]
Reason:	
The flexibility offered under the optimistic access control domain is a security risk.	Agree [ ] Disagree [ ]
Reason:	
Specifying system conditions – such as limiting access according to the time of day – may deter users from abusing their privileges.	Agree [ ] Disagree [ ]
Reason:	
The 'Break-the-Glass' facility is vulnerable to abuse.	Agree [ ] Disagree [ ]
Reason:	
The protection mechanisms – such as fulfilling obligations – will compel users to comply with the established rules of behaviour in order to protect confidential information.	Agree [ ] Disagree [ ]
Reason:	
An individual who interacts with the system will recognise that access is dependent on user responsibility as well as technical access control.	Agree [ ] Disagree [ ]
Reason:	
The risk of losing one's rights to information under the optimistic access control domain may deter one from abusing one's privileges.	Agree [ ] Disagree [ ]
Reason:	
The conditions, obligations and the Break-the-Glass mechanisms may be distracting to a user**.**	Agree [ ] Disagree [ ]
Reason:	

Most users will ignore the messages in terms of the conditions and obligations relating to access.	Agree [ ] Disagree [ ]
Reason:	

Any other comments or critique

D.5 Overview of the Experiment-and-evaluation-of-use study and its application

Sessions

Several separate sessions were held at the laboratory of the School of Computer Science. Each session was attended by one participant and lasted approximately 30 minutes. Prior to their individual session, participants were given the concept specification to implement as a term assignment. Each participant then had to demonstrate his/her working product and provide value judgements on the model concept.

Steps involved

The participants were given the following test cases to carry out in order to interact with the prototype, while the researcher observed:

1. An authorised user accessing data under the public domain is allowed to read client data but is subject to pre-conditions, pre-obligations, ongoing obligations, ongoing conditions and post-obligations (simulate instances where the pre-conditions and ongoing conditions are not satisfied).
2. An unauthorised user is not able to access any data.
3. An authorised user is subject to role-based access control policies when accessing the employee data.
4. User's optimistic rights are downgraded.

The participants were then asked several open-ended questions regarding the usability and perceived effectiveness of the product as a security countermeasure [see questions in D.4 above].

Validity

Postgraduate students were used to develop and evaluate the model concept in view of the fact that they have extensive knowledge in information systems and are currently employable. Seeing that some of these students are already employed within the

information systems sector, their profile can serve as a profile of user representatives in systems development.

APPENDIX E: DATA COLLECTION

Statement1: The product specifications as given in the assignment were ambiguous and incomplete.			
Participant	**Responses**	**Agree**	**Disagree**
A	In fact, I just followed the instructions incorporating minimal creativity from class discussions		YES
B	It was sufficient		YES
C	It was relatively open-ended but not incomplete		YES
D	I was able to implement it early		YES
E	Much was left to (mis) interpretation	YES	
F	I think it was good but the concept of priorities should be stated more		YES
G	All the aspects of the system has been explained clearly		YES
H	To a certain extent because it did not give explicit rules of the break the glass	YES	
I	NO RESPONSE		YES
J	RBAC and optimistic access control were clearly specified except for how priority is assigned		YES
K	The assignment was straightforward		YES
L	NO RESPONSE	YES	
M	All specifications were easy to understand and implement into a system		YES
N	I found the project brief very complete, although I think the system would need more to be commercially viable		YES

Statement 2: The product specifications as given in the assignment could easily be translated into an implementable product.			
Participant	**RESPONSE**	**Agree**	**Disagree**
A	Well, I wouldn't really know, not really an expert on access control in commercial environments	YES	
B	It was properly adapted for the size and situation	YES	
C	NO RESPONSE	YES	
D	NO RESPONSE	YES	
E	No Issues	YES	
F	Yes I think it can be used in certain types of industry	YES	
G	Since it was enough to implement the prototype, given that it is scaled up, the real product could be implemented	YES	
H	yes but it can only scale to certain levels	YES	
I	NO RESPONSE	YES	
J	Specification was detailed enough	YES	
K	It is close to that point upon completion	YES	
L	NO RESPONSE	YES	
M	Specifications were easy to divide into implementable components	YES	
N	Yes, if tested properly, and robust for users to define new conditions	YES	

Statement 3: In terms of the enforcement of security, other mechanisms such as, a written policy document or adequate training would have been more effective than the mechanisms identified in the product concept.			
Participant	**Response**	**Agree**	**Disagree**
A	Usage based as implemented in the project is just kinda[sic] weak, Maybe if the break the glass facility were removed entirely	YES	
B	PB: They are simpler to ignore		YES
C	The other mechanisms added to the current mechanisms can make the overall product more effective but,..		YES
D	A companies trust of their employees should have some evaluation criteria		YES
E	Depends on the environment. Ideally both should be used		YES
F	Inforcing[sic] it this way I think would be more effective		YES
G	NO RESPONSE		YES
H	Written policy can be present but they[sic] is not constant reminder like in a automated system		YES
I	NO RESPONSE		YES
J	Learning curve maybe too steep for the average people, requiring training	YES	
K	If used in addition it would be more secure	YES	
L	With the prototype concept it enables administrators to track user actions		YES
M	The main security risk still lies with the user and no amount of training can truly provide security against human stupidity		YES
N	No personal ethics would still be basis for decisions made by personnel. An agreement etc. would be acceptable in court		YES

Statement 4: The flexibility offered under the optimistic access control domain is a security risk.			
Participant	**Response**	**Agree**	**Disagree**
A	It really all depends on what you are guarding and if the users are being monitored	YES	
B	Never give a user more slack than the minimum	YES	
C	it might, depending on the level of confidentiality of the data	YES	
D	The users might see and distribute sensitive data	YES	
E	Should be controlled through other means	YES	
F	In some cases I think letting people access data with BTG can cause major harm	YES	
G	The information can be abused with optimistic access control	YES	
H	Because it is subject to evaluation by a human		YES
I	It can be because they is no monitor to ensure that risk free activities are done	YES	
J	Too much data is allowed to be viewed	YES	
K	NO RESPONSE		YES
L	But it depends on the nature of the organisations and its data	YES	
M	If used intelligently a lot of information can be accessed without dire consequences	YES	
N	No certain environment require that flexibility such the medical industry		YES

Statement 5: Specifying system conditions, such as limiting access according to the time-of-day, may deter users from abusing their privileges.			
Participant	**Response**	**Agree**	**Disagree**
A	The threat of losing trust would deter me, if I really wanted to get in I would rather just try avoid the access control system	YES	
B	It may, the key being "may"	YES	
C	NO RESPONSE		YES
D	Error dialogues might frighten some uses and deter them from continuing	YES	
E	Depends on the consequences of violating them	YES	
F	If people want to abuse their privileges they can do it during work hours		YES
G	Since ignoring the restrictions would lead to degradation of trust	YES	
H	Yes but what about a situation when a company works overtime	YES	
I	NO RESPONSE	YES	
J	As long as the user is aware of these issues it will deter them	YES	
K	Users should be less likely to abuse their privileges during working hours	YES	
L	It will give a user a feeling that they are doing something wrong	YES	
M	A person would mostly try to be unseen when doing misconduct and that the easiest after hours	YES	
N	No, not if defined correctly and the BTG functionality provides alternatives		YES

Statement 6: The 'break the glass' facility is vulnerable to abuse.			
Participant	**Responses**	**Agree**	**Disagree**
A	Most definitely in my implementation it relied mostly on the threat of an admin stepping in after the event	YES	
B	NO RESPONSE	YES	
C	The trust level drops too quickly		YES
D	The trust level drops		YES
E	Necessary nonetheless	YES	
F	Yes, I think some user will abuse the BTG feature	YES	
G	Not if the task at hand is of low priority		YES
H	Because you can use it to get back at employee when di[sic]	YES	
I	Everything is logged and the user will be monitored		YES
J	The priority of requests need to be better refined	YES	
K	To the extent that everything is vulnerable to abuse	YES	
L	NO RESPONSE	YES	
M	As stated above, if used correctly a lot of information can be accessed without dire consequences	YES	
N	No, not if the manager/auditor does not abuse the system by allowing any situation or accepting any reason	YES	

Statement 7: The protection mechanisms, such as fulfilling obligations, will compel users to comply with the established rules of behaviour in order to protect confidential information.			
Participant	**Responses**	**Agree**	**Disagree**
A	Nobody reads EULAs		YES
B	Well it does	YES	
C	It might, do a user test	YES	
D	This would have to be tested as the users think and act differently		YES
E	Depends on the implementation of those obligations	YES	
F	If not doing degrades their access right it will compel the users to comply with it	YES	
G	NO RESPONSE	YES	
H	NO RESPONSE	YES	
I	NO RESPONSE	YES	
J	Users are intimidated by warning usually	YES	
K	It adds a sense of responsibility	YES	
L	But only if the user is trustworthy.	YES	
M	Because ignorance is not an excuse anymore	YES	
N	Yes	YES	

Statement 8: An individual who interacts with the system will recognize that access is dependent on user responsibility as well as technical access control.			
Participant	**Responses**	**Agree**	**Disagree**
A	The warning should be a clear indication	YES	
B	NO RESPONSE	YES	
C	This depends on the level of knowledge the user have of the trust-based system	YES	
D	The trust level indication	YES	
E	Don't assume users are responsible		YES
F	As a user uses it for some time. I think he will get accustomed to responsible usage of the sys	YES	
G	NO RESPONSE	YES	
H	Yes, because the constant prompts	YES	
I	NO RESPONSE	YES	
J	Technical side may be obvious e.g. audit logs		YES
K	See above	YES	
L	They might notice that their amount of actions they can performs degrades	YES	
M	This will only be the case for the majority if it is explained clearly at the beginning		YES
N	Yes		YES

Statement 9: The risk of losing one's rights to information under the optimistic access control domain may deter one from abusing one's privileges.			
Participant	**Responses**	**Agree**	**Disagree**
A	Although it depends what their ultimate goal is. I.e. if they are planning on quitting the next day, they should even care	YES	
B	Depends on the information content and nature. Also user registration	*	
C	NO RESPONSE	YES	
D	If the information is vital to the user	YES	
E	Only if the information is wanted but not needed		YES
F	If not fulfilling obligation reduces their access rights they will be careful with the use of data	YES	
G	Since the user can't see sensitive info	YES	
H	Yes fear is always a deterrent	YES	
I	NO RESPONSE	YES	
J	Lets the user know when can go wrong and violate privileges	YES	
K	NO RESPONSE	YES	
L	If the user's goal is to steal data, it won't prevent them from doing so		YES
M	If your actions are logged that caused it , you have a higher possibility of being caught	YES	
N	Yes	YES	

* - NO RESPONSE

Statement 10: The conditions, obligations and the break-the-glass mechanisms may be distracting to a user.			
Participant	**Responses**	**Agree**	**Disagree**
A	It depends on the user, I say disagree because I suspect most users will simply ignore the mechanisms and they will lose their meaning anyway		YES
B	If they want the info. they won't mind		YES
C	It is quite distracting at times, as all error messages/info messages are.	YES	
D	At first, last they might ignore it User testing again is vital here	YES	
E	Depends on the implementation of UI	YES	
F	To some extent if they are many	YES	
G	NO RESPONSE		YES
H	Constant popups are distracting to the user	YES	
I	It might be a bit strange at first but the user should be able to get use to it		YES
J	This is needed to deter users		YES
K	NO RESPONSE		YES
L	It will only effect them when they log in and go past business hours		YES
M	Windows vista used a similar approach with permission popups and must users disabled this security feature due to annoyance	YES	
N	PN: If the conditions are important enough, it should not		YES

Statement 11: Most users will ignore the messages in terms of the conditions and obligations relating to the access.			
Participant	**Responses**	**Agree**	**Disagree**
A	Once they learn which sequence of buttons to press to get to the required result, why would they bother reading? Or caring for that matter	YES	
B	True, do you ever read the agreement when you start	YES	
C	Depends on what they know about the consequences of ignoring them		YES
D	If the user is forced to give response to the message		YES
E	Will after repeated exposure but then users will know its contents		YES
F	NO RESPONSE	YES	
G	NO RESPONSE		YES
H	Yes until they are warned that they will loose[sic] access	YES	
I	They might at first but once they notice that is limits their access after it will be taken seriously		YES
J	Most sensible users will feel threaten by the messages		YES
K	Initially they will pay heed, but later it becomes routine	YES	
L	Most users will just want to get the data to do their work	YES	
M	This will only happen if the consequences are not clearly specified		YES
N	Yes but the responsibility still lies with the them and holds them accountable	YES	

APPENDIX F: ASPECTJ SEMANTICS

The following list of pointcut designators describe only those designators that were used in the context of thesis:

call(signature) **execution**(signature)	Matches call/execution joinpoints at which the method or constructor matches the signature
target(ClassName)	All the join points where the object on which the method called is an instance of `ClassName`
within(className)	matches join points of any kind at which the currently executing code is contained with ClassName
declare parents: ClassName **implements** InterfaceName	declares the `ClassName` type to implement the InterfaceName Interface

Wildcards

Type names that contain the two wildcards "*" and "**..**" are also type patterns. The * wildcard matches zero or more characters except for ".", so it can be used when types have a certain naming convention.

Operators

Pointcuts compose through the operations **or** (“||”), **and** (“&&”) and **not** (“!”)

Cited from:

1. Kiczales, G., et al. *An Overview of AspectJ*. In *ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming*. 2001. Budapest, Hungary: Springer-Verlag. p. 327-353
2. http://www.eclipse.org/aspectj/doc/released/progguide/semantics- pointcuts.html (Last accessed 1 October 2009)

APPENDIX G:
RUNNING DEMO PROJECT

Go to:

http://cs-cert.unisa.ac.za/internet/keshnee/content.html

Instructions for Running Demo:

(1) Create a directory called **test0** on your **C** drive

(2) Unzip test0.zip to into **test0** directory

(3) Search for aoptest.exe in directory **test0** and double click to run

(4) Read Manual.pdf for more details on the operation of the software

This software was built using:
Aspect J version 1.4.0
Eclipse version 3.2
Java SDK 1.4.2_05