References

- ADL, A. D. L. Sharable Content Object Reference Model (SCORM ®) 2004 2nd Edition Overview. (2004). http://www.adlnet.org/
- AKT. Reference Ontology v.2 - AKTive Portal Ontology v.2. (2002). http://d3e.open.ac.uk/akt/2002/portal-ocml-v2.0/portal-ocml-v2.0-t.html
- Alani, H. et al. Automatic Ontology-Based Knowledge Extraction and Tailored Biography Generation From the Web. *IEEE Intelligent Systems* **18**, 14-21 (2003).
- Alfaro, I., Zancanaro, M., Nardon, M. & Guerzoni, A. Navigating By Knowledge. *8th International Conference on Intelligent User Interfaces* 221-223 (2003).
- Allen, J. F. Towards a General Theory of Action and Time. *Artificial Intelligence* **23**, 123-154 (1984).
- Antoniou, Grigoris, Harmelen, van, F. *Semantic Web Primer, second edition (Cooperative Information Systems)* (The MIT Press, 2008).
- Aroyo, L. & Mizoguchi, R. Process-Aware Authoring of Web-Based Educational Systems. *Proceedings of the First International Workshop of Semantic Web for Web-based Learning (SW-WL03)* 212-221 (2003).
- Aroyo, L. & Mizoguchi, R. Ontologies for Authoring of Intelligent Educational Systems. *Workshop on Applications of Semantic Web Technologies for E-Learning, in conjunction with ISWC'04* (2004).
- Aroyo, L., Mizoguchi, R. & Tzolov, C. Ontoaims: Ontological Approach to Courseware Authoring. *Proceedings of the International Conference on Computers in Education (ICCE2003)* 1011-1014 (2003).
- Bach, C. N. & Manion, M. The Hypermediated Text: An Integrated Tool for Teaching Philosophy. *Teaching Philosophy* **24**, (2001).
- Bachelard, G. *La formation de l'esprit scientifique* 1938).
- Barbera, M., Nucci, M., Hahn, D. & Morbidoni, C. A Semantic Web Powered Distributed Digital Library System. *ELPUB 2008 Conference on Electronic Publishing* (2008).
- Barros, B., Verdejo, M. F., Read, T. & Mizoguchi, R. Applications of a Collaborative Learning Ontology. *MICAI'2002 Advances in Artificial Intelligence* 301-310 (2002).
- Barthes, R. in *The Narrative Reader* (ed McQuillan, M.) (Routledge, 2000).
- BBC. Programmes (Rdf Version). (2008).
- Bechhofer, S., Stevens, R. D. & Lord, P. W. Gohse: Ontology Driven Linking of Biology Resources. *Web Semantics: Science, Services and Agents on the World Wide Web* **4**, (2006).

- Becker, J., Brelage, C., Klose, K. & Thygs, M. Conceptual Modeling of Semantic Navigation Structures: The Mosena-Approach. *Proceedings of the 5th ACM international workshop on Web information and data management* (2003).
- Bentivogli, L., Pianesi, F. & Pianta, E. From Word-Based to Concept-Based Text Analysis: The Philonet Project. (2002).
- Berners-Lee, T. What the semantic web can represent. (1998). http://www.w3.org/DesignIssues/RDFnot.html
- Berners-Lee, T. Web Architecture from 50,000 feet. (1999).
- Berners-Lee, T., Hendler, J. & Lassila, O. The Semantic Web. *Scientific American* (2001).
- Black, M. *A companion to Wittgenstein's 'Tractatus'* (Cambridge U.P, 1964).
- Brewster, C., Alani, H., Dasmahapatra, S. . & Wilks, Y. Data Driven Ontology Evaluation. *Proceedings of International Conference on Language Resources and Evaluation* (2004).
- Brin, S. & Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* (1998).
- Brooks, K. M. Do Story Agents Use Rocking Chairs? The Theory and Implementation of One Model for Computational Narrative. *ACM Multimedia* (1996).
- Brooks, K. M. Programming Narratives. *IEEE Symposium on Visual Languages* (1997).
- Brown, J. S., Collins, A. & Duguid, P. Situated Cognition and the Culture of Learning. *Educational Researcher* **18**, 32-42 (1989).
- Bruner, J. *The Process of Education* (Harvard University Press, Cambridge, MA, 1960).
- Bruner, J. *Toward a Theory of Instruction* (Harvard University Press, Cambridge, MA, 1966).
- Bruner, J. The Narrative Construction of Reality. *Critical Inquiry* **18**, 1-21 (1991).
- Bruner, J. *The Culture of Education* (Harvard University Press, Cambridge, Mass, 1996).
- Brusilovsky, P. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User Adapted Interaction (Special issue on adaptive hypertext and hypermedia)* **6**, 87-129 (1996).
- Buitelaar, P. & Eigner, T. Semantic Navigation With Views. *UserSWeb: Workshop on User Aspects of the Semantic Web* (2005).
- C.W. Choo, B. Detlor & Turnbull, D. Information Seeking on the Web: An Integrated Model of Browsing and Searching. *First Monday* **5**, (2000).
- Carr, L., Hall, W., Bechhofer, S. & Goble, C. A. Conceptual Linking: Ontology-Based Open Hypermedia. *WWW-2001* 334-342 (2001).
- Carusi, A. Taking Philosophical Dialogue Online. *Discourse: Learning and Teaching in Philosophical and Religious Studies* **3**, 95-156 (2003).

- Celino, I. & Della Valle, E. Multiple Vehicles for a Semantic Navigation Across Hyper-Environments. *ESWC : European semantic web conference* (2005).
- Chatman, S. *Story and Discourse* (Cornell University Press, 1978).
- Chen, W., Hayashi, Y., Jin, L., Ikeda, M. & Mizoguchi, R. An Ontology-Based Intelligent Authoring Tool. *Proc. of the Sixth International Conference on Computers in Education* 41-49 (1998).
- Chen, W. & Mizoguchi, R. in *Cognitive Support for Learning - Imagining the Unknown* (ed Kommers, P.) 189-200 (IOS Press, 2004).
- CIPHER. Project Website. (retrieved on December 2008). http://cipherweb.open.ac.uk/news/index.pl
- Conklin, J. Hypertext: And Introduction and Survey. *IEEE Computer* **20**, 17-41 (1987).
- Cook, S. D. N. & Brown, J. S. Bridging Epistemologies: The Generative Dance Between Organizational Knowledge and Organizational Knowing. *Organization Science* **10**, 381-400 (1999).
- Corbridge, C., Rugg, G., Major, N. P., Shadbolt, N. R. & Burton, A. M. Laddering: Technique and Tool Use in Knowledge Acquisition. *Knowledge Acquisition* **6**, 315-341 (1994).
- Corcho, O. & Gomez-Perez, A. A Roadmap to Ontology Specification Languages. *EKAW'00* (2000).
- Cornish, K. *The Jew of Linz* (Century Hutchinson, 1998).
- Crampes, M. & Ranwez, S. Ontology-Supported and Ontology-Driven Conceptual Navigation on the World Wide Web. *11th ACM Hypertext Conference* 191-199 (2000).
- Crofts, N., Doerr, M., Gill, T., Stead, S. & Stiff, M. Cidoc Crm Version 4.2 - Reference Document. (2005).
- d'Aquin, M. Building Semantic Web Based Applications With Watson. *WWW 2008 - Developers' Track* (2008).
- D'iorio, P. Cognitive Models of Hypernietzsche: Dynamic Ontology and Hyper-Learning. *Jahrbuch fur Computerphilologie* (2003).
- Dave, P. et al. Browsing Intricately Interconnected Paths. *HT'03* 95-103 (2003).
- Davenport, G. & Murtaugh, M. Automatist Storyteller Systems and the Shifting Sands of Story. *IBM Systems Journal* **36**, (1997).
- DCMI. Dublin Core Metadata Initiative. (2008). http://dublincore.org/
- de Boer, V., van Someren, M. & Wielinga, B. J. Extracting Instances of Relations From Web Documents Using Redundancy. *Third European Semantic Web Conference 2006* (2006).
- Decker, S. et al. The Semantic Web: The Roles of Xml and Rdf. *Internet Computing* **4**, 63-73 (2000).

- Devedzic, V. Education and the Semantic Web. *International Journal of Artificial Intelligence in Education* **14**, 39-65 (2004).
- Devedzic, V. Think Ahead: Evaluation and Standardisation Issues for E-Learning Applications. *Int. J. Continuing Engineering Education and Lifelong Learning* **13**, 556-566 (2003).
- Dicheva, D., Sosnovsky, S., Gavrilova, T. & Brusilovsky, P. Ontological Web Portal for Educational Ontologies. *International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL), AIED-05* (2005).
- Dillenbourg, P. The Role of Artificial Intelligence Techniques in Training Software. *LEARNTEC* (1994).
- Discovery. Project Website. (retrieved on December 2008). http://www.discovery-project.eu/
- Doerr, M. The Cidoc Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata. *AI Magazine archive* **24**, 75-92 (2003).
- Domingue, J., Dzbor, M. & Motta, E. Magpie: Supporting Browsing and Navigation on the Semantic Web. *9th international conference on Intelligent User Interfaces* (2004).
- Drucker, P. Need to Know - Integrating e-Learning with High Velocity Value Chains. *Delphi Group e-Learning Insight Research Report* (2000). http://www.delphigroup.com/pubs/whitepapers/
- Duval, E. & Hodgins, W. A Lom Research Agenda. *12th Int'l WWW Conf.* (2003).
- Dzbor, M., Domingue, J. B. & Motta, E. Magpie - Towards a Semantic Web Browser. *2nd Intl. Semantic Web Conference* (2003).
- Eero Hyvönen et al. (ed Robering, K.) (LIT Verlag, 2007).
- Eetu Mäkelä, Suominen, O. & Hyvönen, E. Automatic Exhibition Generation Based on Semantic Cultural Content. *Proceedings of the Cultural Heritage on the Semantic Web Workshop at the 6th International Semantic Web Conference (ISWC 2007)* (2007).
- EML. Educational Modeling Language Website. (2005).
- Farquhar, A., Fikes, R. & Rice, J. (Stanford University, Knowledge Systems Laboratory, 1996).
- Fellbaum, C. (ed) *WordNet: An Electronic Lexical Database*. (MIT Press, 1998).
- Freebase. An open, shared database of the world's knowledge. http://www.freebase.com/
- Freemind. Free mind mapping software. (2008). http://freemind.sourceforge.net/wiki/index.php/Download
- Gammack, J. G. in *Knowledge Acquisition for Expert Systems; A Practical Handbook* (ed Kidd, A. L.) (Plenum, New York, 1987).

• Gangemi, A., Prisco, A., Sagri, M.-T., Steve, G. & Tiscornia, D. Some Ontological Tools to Support Legal Regulatory Compliance, With a Case Study. *On the move to meaningful internet systems (OTM 2003)* (2003).

• Gangemi, A. Ontology Design Patterns for Semantic Web Content. *ISWC 2005* (2005).

• Gangemi, A., Borgo, S., Catenacci, C. & Lehmann, J. Task Taxonomies for Knowledge Content - D07. *Deliverable of the EU FP6 project Metokis* (2005).

• Gangemi, A., Catenacci, C., Ciaramita, M. & Lehmann, J. Ontology evaluation and validation. *Technical report, Lab. for Applied Ontology* (2005).

• Gangemi, A., Guarino, N., Masolo, C., Oltramari, A. & Schneider, L. Sweetening Ontologies With Dolce. *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)* (2002).

• Gasevic, D., Jovanović, J. & Devedzic, V. Enhancing Learning Object Content on the Semantic Web. *IEEE International Conference on Advanced Learning Technologies (ICALT'04)* (2004).

• Genette, G. *Narrative Discourse: An Essay in Method* (Cornell University Press, 1983).

• Geurts, J., Bocconi, S., Ossenbruggen, J. v. & Hardman, L. Towards Ontology-Driven Discourse: From Semantic Graphs to Multimedia Presentations. *Proceedings of the Second International Semantic Web Conference (ISWC 2003)* (2003).

• Gomez-Perez, A. Evaluation of Ontologies. *Intl. Journal of Intelligent Systems* **16**, 391-409 (2001).

• Google. Google Maps APIs. (retrieved on December 2008). http://code.google.com/apis/maps/

• Grayling, A. C. *Philosophy: A Guide Through the Subject Vol 1* (OUP Oxford, 1998a).

• Grayling, A. C. *Philosophy: Further Through the Subject v.2: Further Through the Subject Vol 2* (OUP Oxford, 1998b).

• Gruber, T. Where the Social Web Meets the Semantic Web (Keynote Talk). *5th International Semantic Web Conference (ISWC2006)* (2006).

• Gruber, T. R. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* **5**, 199-220 (1993).

• Guarino, N. & Welty, C. Evaluating Ontological Decisions With Ontoclean. *Commun. ACM* **45**, 61-65 (2002).

• Guha, R., McCool, R. & Miller, E. Semantic Search. *WWW2003 --- Proc. of the 12th international conference on World Wide Web* 700--709 (2003).

• GutenbergFoundation. Project Gutenberg. (2008). http://www.gutenberg.org/wiki/Main_Page

• Habel, G. & Magnan, F. General Poncelet Meets the Semantic Web: A Concrete Example of the Usage of Ontologies to Support Creation and Dissemination of

Elearning Contents. *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2007* 908–915 (2007).

- Heflin, J. & Hendler, J. Searching the Web With Shoe. *Artificial Intelligence for Web Search. AAAI Workshop. WS-00-01* 35-40 (2000).
- Hein, G. E. Constructivist Learning Theory. *CECA (International Committee of Museum Educators) Conference* (1991).
- Hildebrand, M., van Ossenbruggen, J. & Hardman, L. /Facet: A Browser for Heterogeneous Semantic Web Repositories. International Semantic Web Conference ((ISWC2006) (2006).
- Holdener, A. *Ajax: The Definitive Guide* (O'Reilly Media, Inc, 2008).
- Huynh, D., Mazzocchi, S. & Karger, D. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. *International Semantic Web Conference 2005* (2005).
- Huynh, D. F., Karger, D. R. & Miller, R. C. Exhibit: Lightweight Structured Data Publishing. *WWW '07: Proceedings of the 16th international conference on World Wide Web* (2007).
- IEEE. LOM, Learning Object Metadata. http://ltsc.ieee.org/doc/wg12/LOM3.6.html
- IFLA. Functional requirements for bibliographic records : final report. (1998).
- Inaba, A., Ohkubo, R., Ikeda, M., Mizoguchi, R. & Toyoda, J. An Instructional Design Support Environment for Cscl - Fundamental Concepts and Design Patterns. *Proc. of Artificial Intelligence in Education AIED-2001* 131-141 (2001).
- Johansson, I. Qualities, Quantities, and the Endurant-Perdurant Distinction in Top-Level Ontologies. *WM 2005 : Professional Knowledge Management Experiences and Vision* (2005).
- JSON.org. Introducing JSON. (2008). http://www.json.org
- KAON. KAON - The Karlsruhe Ontology and Semantic Web Framework - Developers Guide for KAON 1.2.7. (2004).
- Kasai, T., Yamaguchi, H., Nagano, K. & Mizoguchi, R. Building an Ontology of the Goal of it in Education and Its Applications. *Workshop on Applications of Semantic Web Technologies for E-Learning, in conjunction with ISWC'04* (2004).
- Kelly, G. A. *The Psychology of Personal Constructs* (W.W. Norton, New York, 1955).
- Kemerling, G. Teaching Philosophy on the Internet. *Twentieth World Congress of Philosophy* (1998).
- Kirschner, P., Shum, S. B. & Carr, C. *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making* (Springer-Verlag, London, 2003).
- Koper, R. Modeling Units of Study From a Pedagogical Perspective: The Pedagogical Meta-Model Behind Eml. (2001).
- Koper, R. & Olivier, B. Representing the Learning Design of Units of Learning. *Educational Technology & Society* **7**, 97-111 (2004).

• Lama, M., Sanchez, E., Amarim, R. & Vila, X. Semantic Description of the Ims Learning Design Specification. *International Workshop on Applications of Semantic Web Technologies for E-Learning (SWEL), held in conjunction with AIED-05* (2005).
• Laurillard, D. *Rethinking University Teaching* (Routledge, 1993).
• Laventhol, J. Tractatus Logico-Philosophicus. Hypertext of the Ogden bilingual edition. (1996). http://www.kfs.org/~jonathan/witt/tlph.html
• Lenat, D. B. & Guha, R. V. *Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project* (Addison-Wesley, Boston, Massachussets, 1990).
• Lispworks. Lispworks - The integrated cross-platform development tool for ANSI common lisp. (2008). http://www.lispworks.com/
• Little, S., Geurts, J. & Hunter, J. Dynamic Generation of Intelligent Multimedia Presentations Through Semantic Inferencing. *6th European Conference on Research and Advanced Technology for Digital Libraries* 158-189 (2002).
• LODP. Linking Open Data Project. (2008). http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/
• Maedche, A. & Staab, S. Measuring Similarity Between Ontologies. *Intl. Conf. on Knowledge Eng. and Knowledge Management* (2002).
• Marchionini, G. Exploratory Search: From Finding tO Understanding. *Commun. ACM* **49**, 41–46 (2006).
• Masolo, C. et al. Social Roles and Their Descriptions. *KR 2004* (2004).
• Maurer, D. & Warfel, T. Card sorting: a definitive guide. (2007). http://www.boxesandarrows.com/view/card_sorting_a_definitive_guide
• Mays, W. The Teaching of Philosophy. *13th University Conference* (1965).
• *The Narrative Reader* (Routledge, 2000).
• Mizoguchi, R. Tutorial on Ontological Engineering - Part 3: Advanced Course of Ontological Engineering. *New Generation Computing* **22**, 198-220 (2004).
• Mizoguchi, R. & Bourdeau, J. Inside Theory-Aware and Standards-Compliant Authoring Systems. *Workshop on Applications of Semantic Web Technologies for E-Learning, in conjunction with AIED'07* (2007).
• Mizoguchi, R. & Bourdeau, J. Using Ontological Engineering to Overcome Ai-Ed Problems. *International Journal of Artificial Intelligence in Education* **11**, 107-121 (2000).
• Motta, E. *Reusable Components for Knowledge Modelling - Principles and Case Studies in Parametric Design Problem Solving* (IOS Press, The Netherlands, 1999).
• Mozilla. Firefox Browser. (2008). http://www.mozilla.com/en-US/firefox/
• Mulholland, P. & Collins, T. Using Digital Narratives to Support the Collaborative Learning and Exploration of Cultural Heritage. *IEEE International workshop on Presenting and Exploring Heritage on the Web (PEH'02) in conjunction with DEXA 2002* (2002).

• Mulholland, P., Collins, T. & Zdrahal, Z. Story Fountain: Intelligent Support for Story Research and Exploration. *9th International Conference on Intelligent User Interface* 62-69 (2004).
• Mulholland, P., Collins, T., Zdrahal, Z. & Machkova, M. Cipher Project Deliverable 6: Personalization. (2003).
• Murray, T. Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. *International Journal of Artificial Intelligence in Education* **10**, 98-129 (1999).
• Nejdl, W. et al. Edutella: A P2P Networking Infrastructure Based on Rdf. *Proc. of International World Wide Web Conf.* (2002).
• Niepert, M., Buckner, C. & Allen, C. A Dynamic Ontology for a Dynamic Reference Work. *Joint Conference on Digital Libraries - JDCL-07* (2007).
• Niles, I. & Pease, A. Towards a Standard Upper Ontology. *FOIS'01* (2001).
• Nilsson, M. in *Online Education Using Learning Objects* mini@nada.kth.se, 2003).
• Noddings, N. *Philosophy of education* (Westview Press, Boulder, Colorado, 1998).
• Nowviskie, B. COLLEX: semantic collections & exhibits for the remixable web. (2005). http://www.nines.org/about/Nowviskie-Collex.pdf
• Nowviskie, B. & McGann, J. NINES - a federated model for integrating digital scholarship. (2005). http://www.nines.org/about/9swhitepaper.pdf
• Noy, N. F. & Mcguinness, D. L. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report* (2001).
• Nucci, M., David, S., Hahn, D. & Barbera, M. Talia: A Framework for Philosophy Scholars. *SWAP 2007, the 4th Italian Semantic Web Workshop* (2007).
• Nurmuliani, N., Zowghi, D. & Williams, S. P. Using Card Sorting Technique to Classify Requirements Change. *12th IEEE International Requirements Engineering Conference (RE'04)* (2004).
• Oren, E., Delbru, R. & Decker, S. Extending Faceted Navigation for Rdf Data. *5th International Semantic Web Conference* (2006).
• Ossenbruggen, J. v., Geurts, J., Cornelissen, F., Hardman, L. & Rutledge, L. Towards Second and Third Generation Web-Based Multimedia. *Proceedings of the tenth international conference on World Wide Web* 479-488 (2001).
• Pasin, M., Motta, E. & Zdrahal, Z. Capturing Knowledge About Philosophy. *K-CAP'07* (2007).
• Passin, T. B. *Explorer's Guide to the Semantic Web* (Manning Publications, 2004).
• Piaget, J. *The Child's Conception of the World* (Harcourt, Brace Jovanovich, NY, 1929).
• Piaget, J. *The Science of Education and the Psychology of the Child* (Grossman, NY, 1970).
• Plato. *Meno* (Hackett Publishing Co, Inc, 1981).

- Porter, D. *Internet Culture* (Routledge, 1997).
- Propp, V. *Morphology of the Folktale* (University of Texas Press, 1968).
- Ranganathan, S. R. *Elements of Library Classification* (South Asia Books, 1990).
- Reed, C. A. & Rowe, G. W. A. Araucaria: Software for Argument Analysis, Diagramming and Representation. *International Journal of AI Tools* **14**, 961-980 (2004).
- Rugg, G. & Mcgeorge, P. Laddering. *Expert Systems* **12**, 279–291 (1995).
- Rugg, G. & Shadbolt, N. R. On the Limitations of Repertory Grid Technique in Knowledge Acquisition. *6th Banff Knowledge Acquisition for Knowledge-based Systems Workshop* (1991).
- Rugg, G. & Mcgeorge, P. The Sorting Techniques: A Tutorial Paper on Card Sorts, Picture Sorts and Item Sorts. *Expert Systems* **22**, 94 (2005).
- Rutledge, L. et al. Finding the Story: Broader Applicability of Semantics and Discourse for Hypermedia Generation. *14th ACM Conference on Hypertext and Hypermedia* 67-76 (2003).
- Ryan, M.-L. Beyond Myth and Metaphor - the Case of Narrative in Digital Media. *The International Journal of Computer Game Research* **1**, (2001).
- Saussure, F. D. *Course in General Linguistics* (Gerald Duckworth & Co. Ltd, 1995).
- Schank, R. C. *Tell Me A Story: A New Look at Real and Artificial Memory* (Macmillan, New York, 1990).
- Schank, R. C. *Tell Me a Story: Narrative and Intelligence (Rethinking Theory)* (Northwestern University Press,U.S, 1996).
- Schraefel, m. c., Karam, M. & Zhao, S. Mspace: Interaction Design for User-Determined, Adaptable Domain Exploration in Hypermedia. *AH 2003: Workshop on Adaptive Hypermedia and Adaptive Web Based Systems* 217-235 (2003).
- Schraefel, m. c. et al. The Mspace Classical Music Explorer: Improving Access to Classical Music for Real People. *V MUSICNETWORK OPEN WORKSHOP: Integration of Music in Multimedia Applications* (2005).
- Schraefel, M. C., Wilson, M., Russell, A. & Smith, D. A. Mspace: Improving Information Access to Multimedia Domains With Multimodal Exploratory Search. *Commun. ACM (2006)* **49**, 47-49 (2006).
- Schreiber, G. et al. Multimedian E-Culture Demonstrator. *International Semantic Web Conference 2006* (2006).
- SEP. Stanford Encyclopedia of Philosophy. http://plato.stanford.edu/ (retrieved on January 2009)
- Shaw, M. L. G. *Recent Advances in Personal Construct Technology* (Academic Press, London, 1980).

• Shipman III, F. M., Furuta, R., Brenner, D., Chung, C.-C. & Hsieh, H.-w. Using Paths in the Classroom: Experiences and Adaptations. *UK Conference on Hypertext* 267-270 (1998).

• Simon, B., Miklós, Z. & Nejdl, W. Elena: A Mediation Infrastructure for Educational Services. *12th International World Wide Web Conference (WWW2003),* (2003).

• Skinner, B. F. *Science and Human Behavior (Free Press Paperback. no. 92904.)* 1965).

• Srinivasan, R. Village Voice: Expressing Narrative Through Community-Designed Ontologies. *MIT* Masters Thesis, (1994).

• Stenius, E. *Wittgenstein's "Tractatus": A Critical Exposition of the Main Lines of Thought* (Blackwell Publishers, 1960).

• Stojanovic, L., Staab, S. & Studer, R. Elearning Based on the Semantic Web. *WebNet2001* (2001).

• Stutt, A., Collins, T. & Motta, E. Semantic Learning Webs: Using Semantic Web Technology for Delivering Learning Services. (2005).

• Stutt, A. & Motta, E. Semantic Learning Webs. *Journal of Interactive Media in Education* **10**, (2004).

• Suchanek, F. M., Kasneci, G. & Weikum, G. Yago: A Core of Semantic Knowledge Unifying Wordnet and Wikipedia. *World Wide Web conference 2007* (2007).

• Tablan, V., Maynard, D. & Bontcheva, K. GATE --- A Concise User Guide. (2005). http://gate.ac.uk/

• Tane, J., Schmitz, C. & Stumme, G. Semantic Resource Management for the Web: An E-Learning Application. *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* 1-10 (2004).

• Tane, J., Schmitz, C., Stumme, G., Staab, S. & Studer, R. The Courseware Watchdog: An Ontology-Based Tool for Finding and Organizing Learning Material. *Fachtagung Mobiles Lernen und Forschen Kassel* (2003).

• Kasachkoff, T. *Teaching Philosophy: Theoretical Reflections and Practical Suggestions* (Littlefield Publishers, 2004).

• Ullrich, C. Description of an Instructional Ontology and Its Application in Web Services for Education. *Proceedings of Workshop on Applications of Semantic Web Technologies for E-learning* 17-23 (2004).

• Vygotsky, L. *Mind in Society: Development of Higher Psychological Processes* (Harvard University Press, 1978).

• W3C. OWL Web Ontology Language Overview. (2004a). http://www.w3.org/TR/owl-features/

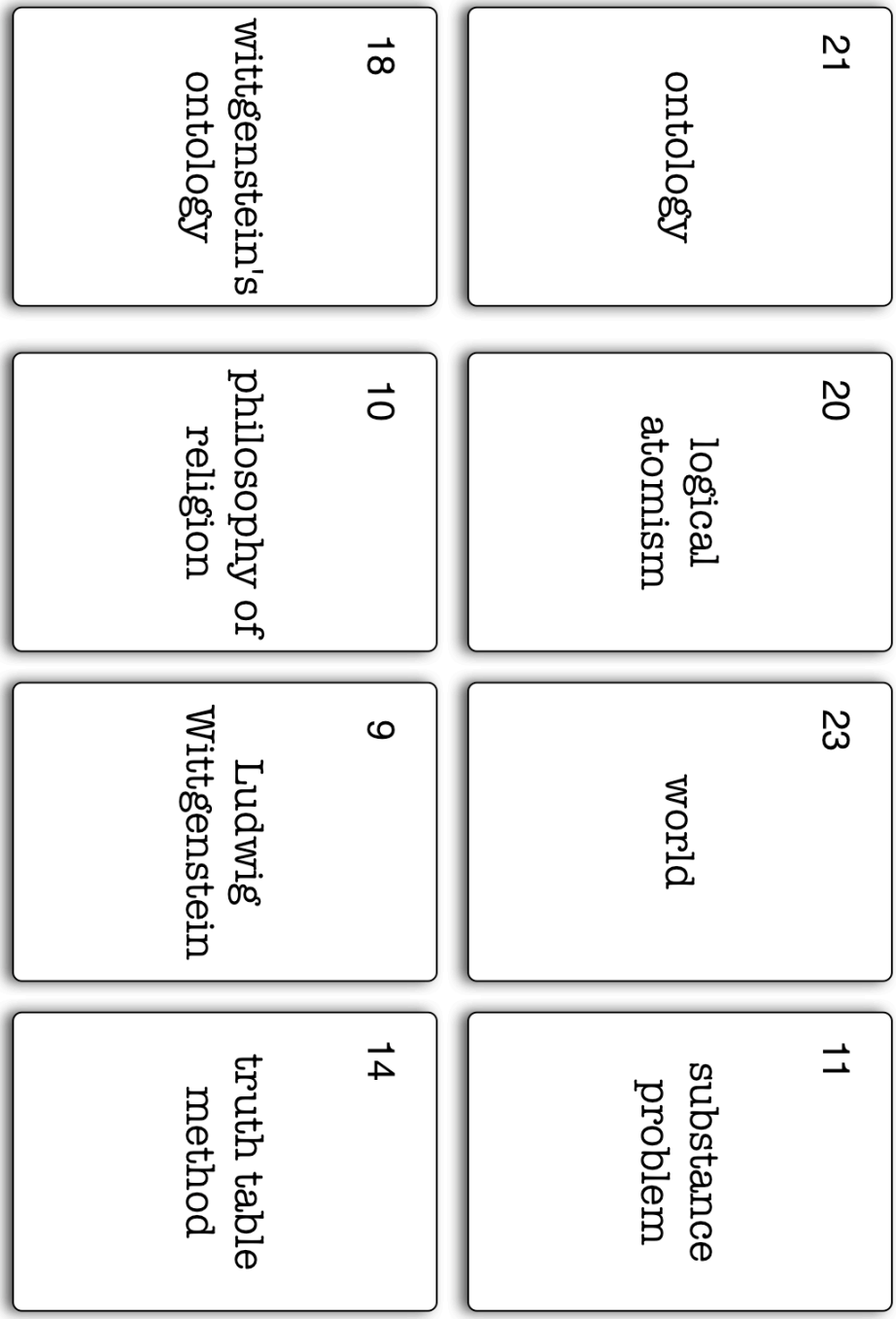
• W3C. RDF Primer. (2004b). http://www.w3.org/TR/rdf-primer/

- W3C. SPARQL Query Language for RDF. (2007). http://www.w3.org/TR/rdf-sparql-query/
- W3C. W3C Semantic Web Activity. (2008).
- Walker, J. Piecing Together and Tearing Apart: Finding the Story in Afternoon. *Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots*. 111-117 (1999).
- Weber, G. & Brusilovsky, P. Elm-Art: An Adaptive Versatile System for Web-Based Instruction. *International Journal of Artificial Intelligence in Education* **12**, 351-384 (2001).
- Weitz, E. HUNCHENTOOT - The Common Lisp web server formerly known as TBNL. (2008). http://www.weitz.de/hunchentoot/
- Wilson, R. The Role of Ontologies in Teaching and Learning. *TechWatch Reports* (2004).
- Wilson, T. D. in *Fifty years of information progress: a Journal of Documentation review* (ed Vickery, B.) 15-51 London, 1994).
- Wittgenstein, L. *Tractatus Logico-Philosophicus* (Routledge & Kegan Paul, 1921).
- Wolff, A., Mulholland, P. & Zdrahal, Z. Scene-Driver: A Narrative-Driven Game Architecture Reusing Broadcast Animation Content. *ACM SIGCHI International Conference on Advances in computer entertainment technology* 91-99 (2004).
- Yee, K.-P., Swearingen, K., Li, K. & Hearst, M. Faceted Metadata for Image Search and Browsing. *ACM CHI* (2003).
- Yu, J., Thom, J. A. & Tam, A. Evaluating Ontology Criteria for Requirements in a Geographic Travel Domain. *In Proc. of Intl. Conf. on Ontologies, DataBases and Applications of Semantics* (2005).
- Yu, J., Thom, J. A. & Tam, A. Ontology Evaluation Using Wikipedia Categories for Browsing. *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (2007).

Appendices

Appendix A: Card Sorting Experiment Results

A1. Example of the paper-cards used in the experiments



A2. Results of the card-sorting sessions, organized per volunteers.

Table A2-1

Table for VOLUNTEER-1 ---- Language: ENGLISH ---- Declared knowledge level: HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
history of philosophy related	related to the history of philosophy, to a timeline	X	X	X		X		X	X	X				X		X	X	X		X	X			
	not history-related				X		X				X	X	X		X				X			X	X	X
Things I like reading	Things I like very much	X							X	X	X								X	X				
	things I like so and so																X						X	
	things I don't want to read		X	X	X	X	X	X				X	X	X	X	X		X			X	X		X
chronological ordering	early 20th century philosophy			X						X				X		X					X			
	late 20th century philosophy	X															X	X						
	post-medieval philosophy							X																
	19th century philosophy					X																		
	greek philosophy								X											X				
	medieval philosophy		X																					
	not history related				X		X				X	X	X		X				X			X	X	X
Pedagogical criterium - things a philosopher should know, resembling a curriculum-creation criteria	'must know', from a pedagogical p.o.v.	X			X				X	X	X	X						X	X	X		X	X	X
	middle priorities, from a pedagogical p.o.v., interesting but not necessarily to know immediately		X										X		X	X	X							
	lowest priorities from a pedagogical p.o.v.,			X		X	X	X						X							X			
types of philosophical entities	schools of philosophy = philosophies + philosophers, with a major emphasis on the group of people aspect	X							X	X						X	X			X			X	
	techniques, methods			X											X									
	philosophical problems						X					X						X						X
	Theoretical approaches, ways of solving problems (more specific than schools of philosophy: one school can contain many of them!		X		X	X								X							X			
	concept												X											
	sub disciplines of philosophy, topics										X								X			X		
	unclassified							X																
Specificity of theoretical approaches (the specificity is associated with the fact that the approach solves a specific problem, and that it refers to a specific philosopher)	very specific													X							X			
	medially specific					X																		
	little specific		X		X																			
	not applicable	X		X			X	X	X	X	X	X	X		X	X	X	X	X	X		X	X	X

Table A2-2

Table for VOLUNTEER-2 ---- Language: ENGLISH ---- Declared knowledge level: HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Type of entities and connection to Wittgenstein	Individuals connected to Wittgenstein									X														
	Aspects of philosophy connected to Wittgenstein	X																	X					
	Schools of thought connected to Wittgenstein															X	X				X		X	
	Topics Wittgenstein worked on						X				X	X		X	X			X						X
	Not connected to Wittgenstein		X	X	X	X		X	X				X							X		X		
Abstract/concrete distinction	concrete entities									X										X				X
	abstract entities	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X		X	X	X	
Types of entities	Methods			X		X							X		X									
	Individuals									X										X				
	doctrines		X		X									X							X			
	domains of enquiry						X				X						X					X	X	
	doctrines tight to particular individuals and times, or broader views	X						X	X							X			X					
	phenomena																	X						X
	problems											X												
Relation to philosophy	Related to philosophy	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Types of entities, alternative classification	Movements, within a certain historical setting. Note that movements are defined in terms of doctrines, even if doctrines usually exists inside movements	X						X	X							X	X		X				X	
	Broad areas of enquiry, with no historical dimension						X				X											X		
	doctrines, which can be associated with a period, but the historical dimension is not important		X											X	X						X			
	methods			X		X							X		X									
	individuals									X										X				
	explananda											X						X						X
Degree of connection to Wittgenstein	Connected to Wittgenstein	X					X			X	X	X		X	X	X	X	X	X		X		X	X
	Not connected to Wittgenstein		X	X	X	X		X	X				X							X		X			

Table A2-3

Table for VOLUNTEER-3 ---- Language: ENGLISH ---- Declared knowledge level: HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
historical problems, and generic ones	history of philosophy, time-bound problems i.e. problems in the historical context	X						X	X	X				X		X	X		X	X	X			
	'timeless' problems, problems as if they were 'free' of any context		X		X	X	X					X	X		X			X				X		
	unclassified			X							X												X	X
historical period, according to the cambridge syllabus	non-analytic philosophy	X		X		X			X	X	X									X			X	
	analytic philosophy		X		X		X	X				X	X	X	X	X	X	X	X		X	X		X
umbrella terms and specific philosophy topics, as used in a pedagogical context	broad topics used for course creation	X						X	X	X	X					X	X			X			X	
	the specific topics studied in philosophy, the things studied in a course		X	X	X	X	X					X	X	X	X			X	X		X	X		X
importance of topics, according to a specific school of philosophy (cambridge)	'hard topics', important and difficult		X		X	X	X					X	X	X	X		X	X	X		X	X		
	'soft topics'	X		X							X					X				X			X	
	can be consider either soft or hard topics							X	X	X														X
By subject areas	metaphysics		X	X	X		X					X	X				X		X		X	X		X
	meaning, philosophy of language	X									X			X	X			X						
	neither of the above					X		X	X	X						X				X			X	

Table A2-4

Table for VOLUNTEER-4 ---- Language: ENGLISH ---- Declared knowledge level: HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
cronological ordering	greek philosophy					X	X		X											X				
	enlightenment period				X			X			X	X												
	unclassified			X																				X
	early XX century work on philosophy of language	X	X							X			X	X	X	X	X	X	X		X	X	X	
philosophies' approach / problems investigated	philosophies that try describe the way we conceptualize the world	X			X	X	X				X			X			X		X		X	X		
	philosophies that try to re-conceptualize the world							X	X											X			X	
	unclassifiable		X	X						X		X	X		X	X		X						X
type of entities	people and strands of philosophy	X						X	X	X						X			X	X			X	
	positions - within the strands of philosophy		X		X								X								X			
	tools (developed by the people, and possibly outside a specific position)			X		X	X							X	X									
	problems, areas of research - tackled by positions										X	X					X	X				X		X

Table A2-5

Table for VOLUNTEER-5 ---- Language: ENGLISH ---- Declared knowledge level: MEDIUM-HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
kind of entities	people									X										X				
	philosophies, philosophical positions or generic approaches	X						X	X		X					X	X		X				X	
	methods, ways to break things down or specific approaches		X	X	X	X							X	X	X						X			
	problems addressed											X						X				X		X
different ways to look at the world, approaches and schools	unclassifiable			X				X			X													
	analytic and empiricist approaches						X							X		X	X				X		X	
	reductionist approaches				X				X			X										X		X
	linguistic approaches	X	X							X					X				X					
	idealistic approaches					X							X					X		X				
type of generic philosophical approach	internalist positions - attempts to describe the way we understand the world, our knowledge of it, in a 'subjective' manner	X	X	X		X				X	X		X	X	X				X	X				
	externalist positions - trying to describe the world external to us, beyond our knowledge of it, in an 'objective' manner				X		X	X	X			X				X	X	X			X	X	X	X
entities' type in relations to their respective philosophical approach	internalist positions - schools of thought, generic approaches			X						X	X								X	X				
	internalist problems	X	X																					
	internalist methods					X							X	X	X									
	externalist positions							X	X							X	X						X	
	externalist problems				X							X						X				X		X
	externalist methods						X														X			

Table A2-6

Table for VOLUNTEER-6 ---- Language: ITALIAN ---- Declared knowledge level: HIGH)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
type of things	epochal movements (non strictly philosophical)							X																
	philosophical epochal movements, also called currents																X						X	
	disciplines						X				X											X		
	philosophical problems											X												
	themes, topics																	X						X
	philosophical positions, 'answers' to the problems (the positions can be sustained by the epochal movements)		X		X																X			
	conceptual tools			X		X							X	X	X									
	groups of authors															X								
	authors									X										X				
	philosophies of an author	X						X																
	doctrines which are part of an author's philosophy																		X					
types of problems, problem areas	substance problem			X	X	X			X		X	X							X	X		X		X
	meaning problem	X	X							X			X	X	X	X	X	X			X			
	undefined						X	X															X	

Table A2-7

Table for VOLUNTEER-7 ---- Language: ITALIAN ---- Declared knowledge level: MEDIUM)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
closeness of perspective, or of the problems tackled	ancient philosophy					X			X			X								X		X		
	logical area				X		X							X	X		X				X			
	empirical sciences							X								X								X
	wittgenstein's perspective and problems	X	X							X								X	X					
	unclassified			X							X		X										X	
type of philosophical entities	currents of thought traversing the history of philosophy		X		X			X													X			
	groups of people, institutions or schools of thought intended as agents										X					X	X						X	
	views, perspectives which are associated to a single philosopher only	X							X					X					X					
	tools, instruments			X		X									X									
	people									X										X				
	unclassifiable - it's not a concept internal to philosophy						X																	
	concepts, themes we deal with in philosophy											X	X					X				X		X

Table A2-8

Table for VOLUNTEER-8 ---- Language: ITALIAN ---- Declared knowledge level: MEDIUM-LOW)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
proximity to schools of thought	continental philosophy					X					X												X	
	analytic philosophy	X					X			X				X	X	X	X		X		X			
	things related to both schools		X	X	X			X	X			X	X					X		X		X		X
proximity to Wittgenstein's views	first Wittgenstein-related				X		X	X	X	X		X		X	X	X	X		X	X	X	X		X
	second wittgenstein-related	X	X								X							X					X	
	undefined			X		X							X											
problem areas, questions asked	ontology			X	X					X		X							X	X		X	X	X
	mathematics						X		X					X	X	X	X	X			X			
	religion	X									X													
	undefined		X			X		X					X											

Table A2-9

Table for VOLUNTEER-9 ---- Language: ITALIAN ---- Declared knowledge level: LOW)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
chronological ordering	modernity	X						X		X	X			X		X	X		X		X		X	
	things which are 'permanent' in history											X						X				X		X
	antiquity		X	X	X	X	X		X				X		X					X				
types of philosophical things	methods					X	X						X	X	X									
	problems			X								X						X				X		X
	specific historical approaches	X							X	X						X			X	X				
	groups of problems										X												X	
	groups of approaches		X		X			X									X				X			

Table A2-10

Table for VOLUNTEER-10 ---- Language: ITALIAN ---- Declared knowledge level: LOW)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
concepts which are specific of philosophy only, or of other disciplines too	concepts which are specific of philosophy only	X		X	X				X	X	X			X			X		X	X			X	
	concepts which are related to philosophy and other disciplines too		X			X	X	X				X	X		X	X		X			X	X		X
ordering based on the type of methods usually associated to a concept	rational or demonstrative method						X	X					X		X		X							
	discousive, qualitative or observational methods		X			X					X	X						X						X
	undefined	X		X	X				X	X				X		X			X	X	X	X	X	
types of ideas	philosophies and broad perspectives	X							X	X	X					X	X		X	X		X	X	
	movements - more specific than the philosophies		X		X			X													X			
	the concepts which are studied			X		X	X					X	X	X	X			X						X
criteria based on the dichotomy container/contents	containers: perspectives used for the analysis	X	X					X	X	X	X			X		X	X	X	X	X	X	X	X	
	contents: the thing analysed			X	X	X	X					X	X		X			X						X
concepts specific to an author or group of authors, intended as a single author	concepts specific to an author or group of authors	X							X	X						X			X	X			X	
	concepts which are more general		X	X	X	X	X	X			X	X	X	X	X		X	X			X	X		X

Table A2-11

Table for VOLUNTEER-11 ---- Language: ITALIAN ---- Declared knowledge level: LOW)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
things linked to Plato or Aristotle or not	things linked to Plato or Aristotle					X	X		X											X		X		
	things not linked to Plato or Aristotle	X	X	X	X			X		X	X	X	X	X	X	X	X	X	X		X		X	X
things related to Wittgenstein, or not	things related to Wittgesntein	X								X						X		X	X			X		
	things not related to Wittgenstein				X			X	X											X				
	undefined		X	X		X	X				X	X	X	X	X		X				X		X	X
philosophy-types vs the rest	types of philosophies	X							X		X						X				X		X	
	things which are not types of philosophies		X	X	X	X	X	X		X		X	X	X	X	X		X	X	X		X		X
themes related to enlightenment, or not	themes related to enlightenment						X	X										X						X
	themes not related to enlightenment	X	X	X	X	X			X	X	X	X	X	X	X	X	X		X	X	X	X	X	
things I know or not	things I do not know	X	X						X				X						X		X		X	
	things I know			X	X	X	X	X	X		X	X	X		X	X	X	X		X		X			X
problems and non problems	problems		X		X	X	X					X						X				X		X
	things which are not problems	X		X				X	X	X	X		X	X	X	X	X		X	X	X		X	

Table A2-12

Table for VOLUNTEER-12 ---- Language: ITALIAN ---- Declared knowledge level: LOW)

Criteria	Categories	Cards																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
people vs all the rest	not people	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X		X	X	X	X
	people									X										X				
stuff related to the 'reference' problem, or not	stuff related to the 'reference' problem		X									X						X				X		X
	stuff which is not related to the 'reference' problem	X		X	X	X	X	X	X	X	X		X	X	X	X	X		X	X	X		X	
philosophical schools of thought, or not	philosophical schools of thought		X			X		X			X					X	X					X	X	
	things which are not schools of thought	X		X	X		X		X	X		X	X	X	X			X	X	X	X			X
correlation to the semiotic/meaning problem	concepts related to the meaning problem		X									X		X				X						X
	things which are not related to the meaning problem	X		X	X	X	X	X	X	X	X		X		X	X	X		X	X	X	X	X	
stuff related to Wittgenstein, or not	stuff related to Wittgenstein	X		X						X									X					
	stuff which is not related to Wittgenstein		X		X	X	X	X	X		X	X	X	X	X	X	X	X		X	X	X	X	X
things that belong to antiquity, or not	things that belong to antiquity		X			X	X		X			X						X		X		X		X
	things that do not belong to antiquity	X		X	X			X		X	X		X	X	X	X	X		X		X		X	
historical ordering	stuff related only to antiquity					X			X	X														
	stuff related only to modern and contemporary history	X		X				X		X					X	X			X		X		X	
	stuff which originated during antiquity but was developed in more recent times		X		X		X				X	X	X	X			X	X				X		X
concepts which are familiar to me, or not	unfamiliar concepts	X		X	X						X			X					X		X		X	
	concepts which are familiar to me		X			X	X	X	X	X		X	X		X	X	X	X		X		X		X

A3. Summary of the mappings between criteria and meta-criteria

Table A3-1

Meta-criteria	Criteria	Categories No	Volunteer
importance /pedagogical perspective	Pedagogical criterium - things a philosopher should know, resembling a curriculum-creation criteria	3	1
	umbrella terms and specific philosophy topics, as used in a pedagogical context	2	3
	importance of topics, according to a specific school of philosophy (cambridge)	3	3
time-based ordering	chronological ordering	7	1
	history of philosophy related	2	1
	historical problems, and generic ones	3	3
	historical period, according to the cambridge syllabus	2	3
	cronological ordering	4	4
	chronological ordering	3	9
	things that belong to antiquity, or not	2	12
	historical ordering	3	12
types of entities	types of philosophical entities	7	1
	Type of entities and connection to Wittgenstein	5	2
	Abstract/concrete distinction	2	2
	Types of entities	7	2
	Types of entities, alternative classification	6	2
	type of entities	4	4
	entities' type in relations to their respective philosophical approach	6	5
	kind of entities	4	5
	type of things	11	6
	type of philosophical entities	7	7
	types of philosophical things	5	9
	types of ideas	3	10
	criteria based on the dichotomy container/contents	2	10
	people vs all the rest	2	12
	philosophical schools of thought, or not	2	12
type of theoretical approach	Specificity of theoretical approaches (the specificity is associated with the fact that the approach solves a specific problem, and that it refers to a specific philosopher)	4	1
	philosophies' approach / problems investigated	3	4
	different ways to look at the world, approaches and schools	5	5
	type of generic philosophical approach	2	5
	proximity to schools of thought	3	8
	proximity to Wittgenstein's views	3	8
type of method	ordering based on the type of methods usually associated to a concept	3	10
correlation to a specific author	Type of entities and connection to Wittgenstein	5	2
	Degree of connection to Wittgenstein	2	2
	concepts specific to an author or group of authors, intended as a single author	2	10
	things linked to Plato or Aristotle or not	2	11
	things related to Wittgenstein, or not	3	11
	stuff related to Wittgenstein, or not	2	12
subject area	Relation to philosophy	1	2
	By subject areas	3	3
	types of problems, problem areas	3	6
	closeness of perspective, or of the problems tackled	5	7
	problem areas, questions asked	4	8
	concepts which are specific of philosophy only, or of other disciplines too	2	10
	philosophy-types vs the rest	2	11
	themes related to enlightenment, or not	2	11
type of problems tackled	historical problems, and generic ones	3	3
	philosophies' approach / problems investigated	3	4
	types of problems, problem areas	3	6
	closeness of perspective, or of the problems tackled	5	7
	problems and non problems	2	11
	stuff related to the 'reference' problem, or not	2	12
	correlation to the semiotic/meaning problem	2	12
things I like/know	Things I like reading	3	1
	things I know or not	2	11
	concepts which are familiar to me, or not	2	12

A4. All the sorts' results, organized according to the Meta-criteria

Table A4-1

Table for Meta Criteria: '**subject area**'

includes verbatim Criteria	has Categories
Relation to philosophy (VOLUNTEER-2)	Related to philosophy
By subject areas (VOLUNTEER-3)	metaphysics
	meaning, philosophy of language
	neither of the above
types of problems, problem areas (VOLUNTEER-6)	substance problem
	meaning problem
	undefined
closeness of perspective, or of the problems tackled (VOLUNTEER-7)	ancient philosophy
	logical area
	empirical sciences
	wittgenstein's perspective and problems
	unclassified
problem areas, questions asked (VOLUNTEER-8)	religion
	undefined
	ontology
	mathematics
concepts which are specific of philosophy only, or of other disciplines too (VOLUNTEER-10)	concepts which are specific of philosophy only
	concepts which are related to philosophy and other disciplines too
philosophy-types vs the rest (VOLUNTEER-11)	types of philosophies
	things which are not types of philosophies
themes related to enlightenment, or not (VOLUNTEER-11)	themes related to enlightenment
	themes not related to enlightenment

Table A4-2

Table for Meta Criteria: '**time-based ordering**'

includes verbatim Criteria	has Categories
chronological ordering (VOLUNTEER-1)	early 20th century philosophy
	late 20th century philosophy
	post-medieval philosophy
	19th century philosophy
	greek philosophy
	medieval philosophy
	not history related
history of philosophy related (VOLUNTEER-1)	related to the history of philosophy, to a timeline
	not history-related
historical problems, and generic ones (VOLUNTEER-3)	history of philosophy, time-bound problems i.e. problems in the historical context
	'timeless' problems, problems as if they were 'free' of any context
	unclassified
historical period, according to the cambridge syllabus (VOLUNTEER-3)	non-analytic philosophy
	analytic philosophy
cronological ordering (VOLUNTEER-4)	greek philosophy
	enlightenment period
	unclassified
	early XX century work on philosophy of language
chronological ordering (VOLUNTEER-9)	antiquity
	modernity
	things which are 'permanent' in history
things that belong to antiquity, or not (VOLUNTEER-12)	things that belong to antiquity
	things that do not belong to antiquity
historical ordering (VOLUNTEER-12)	stuff related only to antiquity
	stuff related only to modern and contemporary history
	stuff which originated during antiquity but was developed in more recent times

Table A4-3

Table for Meta Criteria: **'types of entities'**

includes verbatim Criteria	has Categories
types of philosophical entities (VOLUNTEER-1)	schools of philosophy = philosophies + philosophers, with a major emphasis on the group of people aspect
	techniques, methods
	philosophical problems
	Theoretical approaches, ways of solving problems (more specific than schools of philosophy: one school can contain many of them!
	concept
	sub disciplines of philosophy, topics
	unclassified
Type of entities and connection to Wittgenstein (VOLUNTEER-2)	Individuals connected to Wittgenstein
	Aspects of philosophy connected to Wittgenstein
	Schools of thought connected to Wittgenstein
	Topics Wittgenstein worked on
	Not connected to Wittgenstein
Abstract/concrete distinction (VOLUNTEER-2)	concrete entities
	abstract entities
Types of entities (VOLUNTEER-2)	Individuals
	doctrines
	domains of enquiry
	doctrines tight to particular individuals and times, or broader views
	phenomena
	problems
	Methods
Types of entities, alternative classification (VOLUNTEER-2)	Movements, within a certain historical setting. Note that movements are defined in terms of doctrines, even if doctrines usually exists inside movements
	Broad areas of enquiry, with no historical dimension
	doctrines, which can be associated with a period, but the historical dimension is not important
	methods
	individuals
	explananda
type of entities (VOLUNTEER-4)	people and strands of philosophy
	positions - within the strands of philosophy
	tools (developed by the people, and possibly outside a specific position)
	problems, areas of research - tackled by positions
entities' type in relations to their respective philosophical approach (VOLUNTEER-5)	internalist positions - schools of thought, generic approaches
	internalist problems
	internalist methods
	externalist positions
	externalist problems
	externalist methods
kind of entities (VOLUNTEER-5)	people
	philosophies, philosophical positions or generic approaches
	methods, ways to break things down or specific approaches
	problems addressed
type of things (VOLUNTEER-6)	epochal movements (non strictly philosophical)
	philosophical epochal movements, also called currents
	disciplines
	philosophical problems
	themes, topics
	philosophical positions, 'answers' to the problems (the positions can be sustained by the epochal movements)
	conceptual tools
	groups of authors
	authors
	philosophies of an author
	doctrines which are part of an author's philosophy
type of philosophical entities (VOLUNTEER-7)	unclassifiable - it's not a concept internal to philosophy
	concepts, themes we deal with in philosophy
	currents of thought traversing the history of philosophy
	groups of people, institutions or schools of thought intended as agents
	views, perspectives which are associated to a single philosopher only
	tools, instruments
	people
types of philosophical things (VOLUNTEER-9)	methods
	problems
	specific historical approaches
	groups of problems
	groups of approaches
types of ideas (VOLUNTEER-10)	philosophies and broad perspectives
	movements - more specific than the philosophies
	the concepts which are studied
criteria based on the dichotomy container/contents (VOLUNTEER-10)	containers: perspectives used for the analysis
	contents: the thing analysed
people vs all the rest (VOLUNTEER-12)	people
	not people
philosophical schools of thought, or not (VOLUNTEER-12)	philosophical schools of thought
	things which are not schools of thought

Table A4-4

Table for Meta Criteria: **'type of problems tackled'**

includes verbatim Criteria	has Categories
historical problems, and generic ones (VOLUNTEER-3)	history of philosophy, time-bound problems i.e. problems in the historical context
	'timeless' problems, problems as if they were 'free' of any context
	unclassified
philosophies' approach / problems investigated (VOLUNTEER-4)	philosophies that try describe the way we conceptualize the world
	philosophies that try to re-conceptualize the world
	unclassifiable
types of problems, problem areas (VOLUNTEER-6)	substance problem
	meaning problem
	undefined
closeness of perspective, or of the problems tackled (VOLUNTEER-7)	ancient philosophy
	logical area
	empirical sciences
	wittgenstein's perspective and problems
	unclassified
problems and non problems (VOLUNTEER-11)	problems
	things which are not problems
stuff related to the 'reference' problem, or not (VOLUNTEER-12)	stuff related to the 'reference' problem
	stuff which is not related to the 'reference' problem
correlation to the semiotic/meaning problem (VOLUNTEER-12)	concepts related to the meaning problem
	things which are not related to the meaning problem

Table A4-5

Table for Meta Criteria: **'correlation to a specific author'**

includes verbatim Criteria	has Categories
Type of entities and connection to Wittgenstein (VOLUNTEER-2)	Individuals connected to Wittgenstein
	Aspects of philosophy connected to Wittgenstein
	Schools of thought connected to Wittgenstein
	Topics Wittgenstein worked on
	Not connected to Wittgenstein
Degree of connection to Wittgenstein (VOLUNTEER-2)	Connected to Wittgenstein
	Not connected to Wittgenstein
concepts specific to an author or group of authors, intended as a single author (VOLUNTEER-10)	concepts specific to an author or group of authors
	concepts which are more general
things linked to Plato or Aristotle or not (VOLUNTEER-11)	things linked to Plato or Aristotle
	things not linked to Plato or Aristotle
things related to Wittgenstein, or not (VOLUNTEER-11)	things related to Wittgesntein
	things not related to Wittgenstein
	undefined
stuff related to Wittgenstein, or not (VOLUNTEER-12)	stuff related to Wittgenstein
	stuff which is not related to Wittgenstein

Table A4-6

Table for Meta Criteria: **'type of theoretical approach'**

includes verbatim Criteria	has Categories
Specificity of theoretical approaches (the specificity is associated with the fact that the approach solves a specific problem, and that it refers to a specific philosopher) (VOLUNTEER-1)	very specific
	medially specific
	little specific
	not applicable
philosophies' approach / problems investigated (VOLUNTEER-4)	philosophies that try describe the way we conceptualize the world
	philosophies that try to re-conceptualize the world
	unclassifiable
different ways to look at the world, approaches and schools (VOLUNTEER-5)	analytic and empiricist approaches
	reductionist approaches
	linguistic approaches
	idealistic approaches
	unclassifiable
type of generic philosophical approach (VOLUNTEER-5)	internalist positions - attempts to describe the way we understand the world, our knowledge of it, in a 'subjective' manner
	externalist positions - trying to describe the world external to us, beyond our knowledge of it, in an 'objective' manner
proximity to schools of thought (VOLUNTEER-8)	continental philosophy
	analytic philosophy
	things related to both schools
proximity to Wittgenstein's views (VOLUNTEER-8)	first Wittgenstein-related
	second wittgenstein-related
	undefined

Table A4-7

Table for Meta Criteria: **'importance /pedagogical perspective'**

includes verbatim Criteria	has Categories
Pedagogical criterium - things a philosopher should know, resembling a curriculum-creation criteria (VOLUNTEER-1)	'must know', from a pedagogical p.o.v.
	middle priorities, from a pedagogical p.o.v., interesting but not necessarily to know immediately
	lowest priorities from a pedagogical p.o.v.,
umbrella terms and specific philosophy topics, as used in a pedagogical context (VOLUNTEER-3)	broad topics used for course creation
	the specific topics studied in philosophy, the things studied in a course
importance of topics, according to a specific school of philosophy (cambridge) (VOLUNTEER-3)	'hard topics', important and difficult
	'soft topics'
	can be consider either soft or hard topics

Table A4-8

Table for Meta Criteria: **'things I like/know '**

includes verbatim Criteria	has Categories
Things I like reading (VOLUNTEER-1)	Things I like very much
	things I like so and so
	things I don't want to read
things I know or not (VOLUNTEER-11)	things I do not know
	things I know
concepts which are familiar to me, or not (VOLUNTEER-12)	unfamiliar concepts
	concepts which are familiar to me

Table A4-9

Table for Meta Criteria: **'type of method'**

includes verbatim Criteria	has Categories
ordering based on the type of methods usually associated to a concept (VOLUNTEER-10)	undefined
	rational or demonstrative method
	discousive, qualitative or observational methods

Appendix B: OCML representation of the philosophical ontology

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Base: 10; Package: OCML;    -
*-

;; PhiloSurfical-CRM ontology

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; At the beginning of each class-specification, we put a reference
to the
;;;; origin of the class, e.g. 'AKT' or 'CIDOC'.
;;;; If the class is created by us, we used 'PhiloSurfical' as a
prefix.
;;;; All comments to imported classes also have the 'PhiloSurfical'
prefix.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(in-package "OCML")


(def-class PhiloSurfical-entity ()
"++PhiloSurfical: The top class for the PhiloSurfical ontology -
gathers together the CRM specifications and the primitive values used
by them ad the other integrated ontologies. We have not defined any
slot at this level, since it serves mostly for organizational needs")


(def-class crm-entity (PhiloSurfical-entity)
"E1 - CIDOC - This class comprises all things in the universe of
discourse. It is an abstract concept providing for three general
properties: 1. Identification by name or appellation 2. Classification
by type, allowing further refinement of the specific subclass an
instance belongs to  3. Attachment of free text for the expression of
anything not captured by formal propertieS
++PhiloSurfical: We have renamed this class, which was originally the
crm-entity. we have commented the belongs-to relation. Appellation is
again the value of identified slot, cause we want to support multi-
lingual reference, for sure, plus the fact that conceptual object are
ofter referenced to with different names."
 ((has-note :type string)
  (has-type :type type)
  (has-uri :type uri)
  (is-identified-by :type appellation)))


(def-class dimension (crm-entity)
"E54 - CIDOC "
 ((has-unit :type measurement-unit)
  (has-value :type number)))


(def-class QUANTITY (dimension) ?qun
"++PhiloSurfical: This class comes from AKT support ontology. It has
the same slot specs as dimension, however we left it here cause many
(akt) time-related entities depend on it! Hopefully the slot-renaming
declaration will solve all the name related issues"
 ((has-unit-of-measure :type measurement-unit)
  (has-magnitude :type number))
  :slot-renaming ((has-unit-of-measure has-unit)
                  (has-magnitude has-value)))
```

```
(def-class place (crm-entity)
"E53 - CIDOC - This class comprises extents in space, in particular on
the surface of the earth, in the pure sense of physics: independent
from temporal phenomena and matter. The instances of E53 Place are
usually determined by reference to the position of 'immobile' objects
such as buildings, cities, mountains, rivers, or dedicated geodetic
marks. A Place can be determined by combining a frame of reference and
a location with respect to this frame. It may be identified by one or
more instances of E44 Place Appellation."
 ((consists-of :type place)
  (falls-within :type place)
  (overlaps-with :type place)
  (is-identified-by :type place-appellation)))


(def-class primitive-value (PhiloSurfical-entity)
“E59 -CIDOC - This class comprises primitive values used as
documentation elements, which are not further elaborated upon within
the model.  As such they are not considered as elements within our
universe of discourse. No specific implementation recommendations are
made.
PhiloSurfical: Included as is. Here we can add primitives from the
basic ocml ontology, or from the akt one, in order to 1) make the
PhiloSurfical ontology completely self-contained, and 2) rely on
primitives which match the W3c SW basic datatypes standard (cf.)"
 ())


(def-class time-primitive (primitive-value)
"E61 - CIDOC - This class comprises instances of E59 Primitive Value
for time that should be implemented with appropriate validation,
precision and interval logic to express date ranges relevant to
cultural documentation.
++PhiloSurfical:  we have integrated a time ontology based on the akt
one + Allen's specifications. To be further investigated."
 () )


(def-class number (primitive-value)
"E60 -CIDOC - This class comprises any encoding of computable
(algebraic) values such as integers, real numbers, complex numbers,
vectors, tensors etc., including intervals of these values to express
limited precision.
++PhiloSurfical: it seems that the number spec works as it is, getting
the right subclasses form the OCML base ontology. We might want to
check this better in the future, and possibly integrate directly a
number-types ontology."
 () )


(def-class integer (number)
"++PhiloSurfical: Inserted in PhiloSurfical for consistency issues
with the AKT classes")


(def-class string (primitive-value)
"E62 - CIDOC This class comprises the instances of E59 Primitive
Values used for documentation such as free text strings, bitmaps,
vector graphics, etc."
 :sufficient-for-type-checking (string ?x))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; time specifications (imported from previous work at KMI)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(def-class time-specification (crm-entity)
"E52 - CIDOC Renamed as time specification, as a span may be expressed
as a time point on some level of granularity.
++PhiloSurfical: Integrated with the CIPHER time ontology based on
Allen."
 ((is-identified-by :type time-appellation)))


(def-class YEAR-IN-TIME (integer time-primitive) ?x
      "A year-in-time must be an integer and integer can be a year-in-
time"
      :iff-def (integer ?x)
        :avoid-infinite-loop t)


(def-class MONTH-IN-TIME (positive-integer time-primitive)?x
  "A month-in-time is an integer in the interval 1-12"
  :iff-def (and (positive-integer ?x)(< ?x 13) )
  :prove-by  (member ?x '(1 2 3 4 5 6 7 8 9 10 11 12 ))
  :no-proofs-by (:iff-def))


(def-class DAY-IN-TIME (positive-integer time-primitive)?x
  "A day-in-time is an integer in the interval 1-31"
  :iff-def (and (positive-integer ?x)(< ?x 32) )
  :prove-by  (member ?x '(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23
                          24 25 26 27 28 29 30 31))

  :no-proofs-by (:iff-def))


(def-class HOUR-IN-TIME (non-negative-integer time-primitive) ?x
  "A hour-in-time is an integer in the interval 0-23"
  :iff-def (and (non-negative-integer ?x)(< ?x 24) )
  :prove-by  (member ?x '(0  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23))
  :no-proofs-by (:iff-def))


(def-class MINUTE-IN-TIME (non-negative-integer time-primitive) ?x
  "A minute-in-time is an integer in the interval 0-59"
  :iff-def (and (non-negative-integer ?x)(< ?x 60) )
  :prove-by  (member ?x '(0  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29
                           30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54 55 56 57 58
                          59))
  :no-proofs-by (:iff-def))


(def-class SECOND-IN-TIME (real-number time-primitive)?x
   "A second-in-time is a real number greater or equal to 0, less than
60"
  :iff-def (and (real-number ?x)(not (< ?x 0))(< ?x 60))
  :avoid-infinite-loop t)
```

```
(def-class frequency-of-occurence (time-primitive)
“Added for PhiloSurfical”)
;;; a couple of standard instances
(def-instance unique frequency-of-occurence)
(def-instance sporadic frequency-of-occurence)
(def-instance frequent frequency-of-occurence)
(def-instance habitual frequency-of-occurence)



(def-class TIME-POINT (time-specification) ?tp
 ((second-of :type second-in-time :max-cardinality 1)
  (minute-of :type minute-in-time :max-cardinality 1)
  (hour-of :type hour-in-time :max-cardinality 1)
  (day-of :type day-in-time :max-cardinality 1)
  (month-of :type month-in-time :max-cardinality 1)
  (year-of :type year-in-time :max-cardinality 1))
 :constraint (and (not (and (month-of ?x 2)
                            (> (the ?day (day-of ?x ?day))
                               29)))
                  (not (and (member-of ?x (4 6 9 11))
                            (> (the ?day (day-of ?x ?day))
                               30)))))


(def-relation IDLE-TIME-POINT (?tp)
 :constraint (time-point ?tp)
 :iff-def (and (= (second-of-tp ?tp) 0)
               (= (minute-of-tp ?tp) 0)
               (= (hour-of-tp ?tp) 0)
               (= (day-of-tp ?tp) 0)
               (= (month-of-tp ?tp) 0)
               (= (year-of-tp ?tp) 0)))


(def-function SECOND-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?second (second-of ?tp ?second)))


(def-function MINUTE-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?minute (minute-of ?tp ?minute)))


(def-function HOUR-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?hour (hour-of ?tp ?hour)))


(def-function DAY-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?day (day-of ?tp ?day)))


(def-function MONTH-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?month (month-of ?tp ?month)))


(def-function YEAR-OF-TP (?tp)
 :constraint (time-point ?tp)
 :body (the ?year (year-of ?tp ?year)))
```

```
(def-class TIME-INTERVAL (time-specification) ?int
"An interval is a period of time elapsed between the start of an event
and end of an event.The start of an event is precedes the end of an
event. (Ref. J.F.Allen (1983), Maintaining knowledge about temporal
intervals)."
 ((has-start-time :type time-point :max-cardinality 1)
  (has-end-time :type time-point :max-cardinality 1)
  (has-unit-of-measure :type measurement-unit))
 :constraint (time-point-precedes (the ?st (has-start-time ?int ?st))
                                  (the ?et (has-end-time ?int ?et))))


(def-class DAY (time-interval)
  ((has-duration :value 24-hour-duration)))

(def-class WEEK (time-interval)
  ((has-duration :value 7-day-duration)))

(def-class MONTH (time-interval))

(def-class JANUARY (month)
  ((has-duration :value 31-day-duration)))

(def-class FEBRUARY (month)
  ((has-duration :default-value 28-day-duration)))

(def-class FEBRUARY-IN-LEAP-YEARS (february)
  ((has-duration :value 29-day-duration)))

(def-class MARCH (month)
  ((has-duration :value 31-day-duration)))

(def-class APRIL (month)
  ((has-duration :value 30-day-duration)))

(def-class MAY (month)
  ((has-duration :value 31-day-duration)))

(def-class JUNE (month)
  ((has-duration :value 30-day-duration)))

(def-class JULY (month)
  ((has-duration :value 31-day-duration)))

(def-class AUGUST (month)
  ((has-duration :value 31-day-duration)))

(def-class SEPTEMBER (month)
  ((has-duration :value 30-day-duration)))

(def-class OCTOBER (month)
  ((has-duration :value 31-day-duration)))

(def-class NOVEMBER (month)
  ((has-duration :value 30-day-duration)))

(def-class DECEMBER (month)
  ((has-duration :value 31-day-duration)))

(def-class YEAR (time-interval)
  ((has-duration :value 12-month-duration)))
```

```
(def-function TIME-INTERVAL-DURATION (?interval) -> ?duration
 :constraint (and (time-interval ?interval)
                  (duration ?duration))
 :body (time-point-difference  (the ?et (has-end-time ?interval ?et))
                                   (the ?st (has-start-time ?interval ?
st))))

(def-class TIME-RANGE (time-interval) ?tr
 )


(def-function TIME-RANGE-DURATION (?tr) -> ?duration
  :constraint (and (time-range ?tr)
                   (duration ?duration))
  :body (time-point-difference (the ?et (has-end-time ?tr ?et))
                               (the ?st (has-start-time ?tr ?st))))

(def-class  DURATION  (quantity  time-primitive)       ;;  is  physical-
quantity already there??????
  "A measure of time, e.g., 5 hours"
   ((has-unit-of-measure :type time-appellation)   ;; used to be time-
measure!
   ))

(def-instance 24-HOUR-DURATION duration
  ((has-unit-of-measure hour)
   (has-magnitude 24)))

(def-instance 7-DAY-DURATION duration
  ((has-unit-of-measure day)
   (has-magnitude 7)))

(def-instance 28-DAY-DURATION duration
  ((has-unit-of-measure day)
   (has-magnitude 28)))

(def-instance 29-DAY-DURATION duration
  ((has-unit-of-measure day)
   (has-magnitude 29)))

(def-instance 30-DAY-DURATION duration
  ((has-unit-of-measure day)
   (has-magnitude 30)))

(def-instance 31-DAY-DURATION duration
  ((has-unit-of-measure day)
   (has-magnitude 31)))

(def-instance 12-MONTH-DURATION duration
  ((has-unit-of-measure year)
   (has-magnitude 12)))



(def-function MAGNITUDE-OF-DURATION (?dur) -> ?mag
 :constraint (and (duration ?dur)
                  (number ?mag))
 :body (the ?mag (has-magnitude ?dur ?mag)))
```

```
(def-function UNIT-OF-DURATION (?dur) -> ?uom
 :constraint (and (duration ?dur)
                  (measurement-unit ?uom))
 :body (the ?uom (has-unit-of-measure ?dur ?uom)))



(def-class CALENDAR-DATE (time-point)
 "A calendar date is a time point in which month, day and year have
  been specified"
  ((day-of :type day-in-time :cardinality 1)
   (month-of :type month-in-time :cardinality 1)
   (year-of :type year-in-time :cardinality 1)))


(def-function UNIVERSAL-TIME-ENCODER (?tp)
"This function encodes the standard structure of time-point into
universal-time structure."
 :constraint (time-point ?tp)
 :lisp-fun '(lambda (?tp)
              (encode-universal-time (the-slot-value ?tp 'second-of)
                                     (the-slot-value ?tp 'minute-of)
                                     (the-slot-value ?tp 'hour-of)
                                     (the-slot-value ?tp 'day-of)
                                     (the-slot-value ?tp 'month-of)
                                     (the-slot-value ?tp 'year-of))))


(def-class UNIVERSAL-TIME (time-specification) ?x
 :constraint (integer ?x))


(def-function DECODE-TIME-POINT-FROM-UNIVERSAL-TIME (?ut)
 :constraint (universal-time ?ut)
 :lisp-fun '(lambda (?ut)
              (multiple-value-bind
                    (second minute hour day month year ignore1 ignore2
ignore3)
                (decode-universal-time ?ut)
               (name
                (define-domain-instance (gentemp "TIME-POINT") 'time-
point
                                        `((second-of ,second)
                                          (minute-of ,minute)
                                          (hour-of ,hour)
                                          (day-of ,day)
                                          (month-of ,month)
                                          (year-of ,year)))))))


(def-function TIME-POINT-DIFFERENCE (?tp-1 ?tp-2)
"This function calculates the difference of two universal-time
strctures."
 :constraint (and (time-point ?tp-1)
                  (time-point ?tp-2))
 :body (decode-time-point-from-universal-time
         (- (universal-time-encoder ?tp-1) (universal-time-encoder ?
tp-2))))
```

```
(def-function TIME-POINT-SUM (?tp-1 ?tp-2)
"This function calculates the sum of two universal-time structures."
 :constraint (and (time-point ?tp-1)
                  (time-point ?tp-2))
 :body (decode-time-point-from-universal-time
          (+ (universal-time-encoder ?tp-1) (universal-time-encoder ?
tp-2))))


(def-relation DURATION-IS-LESS-THAN (?d1 ?d2)
 :constraint (and (duration ?d1)
                  (duration ?d2))
 :iff-def (< (the ?magnitude1 (has-magnitude ?d1 ?magnitude1))
             (the ?magnitude2 (has-magnitude ?d2 ?magnitude2))))


(def-instance second measurement-unit)

(def-instance minute measurement-unit)

(def-instance hour measurement-unit)

(def-instance day measurement-unit)

(def-instance month measurement-unit)

(def-instance year measurement-unit)



;;;**;;;Following are some relations for the Time-Ranges;;;*;;;

(def-relation TIME-POINT-PRECEDES (?time-point-1 ?time-point-2)
"This relation states that a time-point-1 precedes a time-point-2."
 :constraint (and (time-point ?time-point-1)
                  (time-point ?time-point-2))
 :iff-def (< (universal-time-encoder ?time-point-1)
             (universal-time-encoder ?time-point-2)))


(def-relation FOLLOWS (?time-point-1 ?time-point-2)
  "This relation relation states that a time-point ?time-point-2
follows a time-point ?time-point-1."
 :constraint (and (time-point ?time-point-1)
                  (time-point ?time-point-2))
 :iff-def (time-point-precedes ?time-point-2 ?time-point-1))


(def-relation TIME-POINTS-EQUAL (?time-point-1 ?time-point-2)
 :constraint (and (time-point ?time-point-1)
                  (time-point ?time-point-2))
 :iff-def (and (= (minute-of ?time-point-1)
                  (minute-of ?time-point-2))
               (= (second-of ?time-point-1)
                  (second-of ?time-point-2))
               (= (hour-of ?time-point-1)
                  (hour-of ?time-point-2))
               (= (day-of ?time-point-1)
                  (day-of ?time-point-2))
               (= (month-of ?time-point-1)
                  (month-of ?time-point-2))
               (= (year-of ?time-point-1)
                  (year-of ?time-point-2))))
```

```
;;;These are BASIC relations;;;

(def-relation BEFORE (?time-range-1 ?time-range-2)
"It means time-range-1 is before the time-range-2."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (time-point-precedes (the ?et (has-end-time ?time-range-1 ?
et))
                    (the ?st (has-start-time ?time-range-2 ?st))))


(def-relation AFTER (?time-range-1 ?time-range-2)
"It means time-range-1 is after the time-range-2."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (time-point-precedes (the ?et (has-end-time ?time-range-2 ?
et))
                                       (the ?st (has-start-time ?time-
range-1 ?st))))



(def-relation IS-AFTER (?time-range-2 ?time-range-1)
"It means that time-range-2 starts after the time-range-1 is
finished."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (follows (the ?st (has-start-time ?time-range-2 ?st))
                   (the ?et (has-end-time ?time-range-1 ?et))))


(def-relation MEETS (?time-range-1 ?time-range-2)
"It means that time-range-2 starts at the same time when time-range-1
ends."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
  :iff-def (time-points-equal (the ?et (has-end-time ?time-range-1 ?
et))
                              (the ?st (has-start-time ?time-range-2 ?
st))))


(def-relation OVERLAPS (?time-range-1 ?time-range-2)
"It means that two time-ranges overlaps with each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (and (time-point-precedes (the ?st-1 (has-start-time ?time-
range-1 ?st-1))
                                    (the ?st-2 (has-start-time ?time-
range-2 ?st-2)))
               (follows (the ?et-1 (has-end-time ?time-range-1 ?et-1))
                        (the ?st-2 (has-start-time ?time-range-2 ?
st-2)))
                   (time-point-precedes (the ?et-1 (has-end-time ?time-
range-1 ?et-1))
                                        (the ?et-2 (has-end-time ?time-
range-2 ?et-2)))))
```

```
(def-relation STARTS-SIMULTANEOUSLY (?time-range-1 ?time-range-2)
"It means that both the time-ranges starts at the same time."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
  :iff-def  (time-points-equal  (the  ?st-1  (has-start-time  ?time-
range-1 ?st-1))
                                      (the ?st-2 (has-start-time ?time-
range-2 ?st-2))))


(def-relation FINISHES-SIMULTANEOUSLY (?time-range-1 ?time-range-2)
"It means that both the time-ranges finishes at the same time but
time-range-1 starts after time-range-2."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (time-points-equal (the ?et-1 (has-end-time ?time-range-1 ?
et-1))
                             (the ?et-2 (has-end-time ?time-range-2 ?
et)-2)))


(def-relation TIME-RANGE-EQUALS (?time-range-1 ?time-range-2)
"It means that both the time-ranges starts and finsihes at the same
time."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
  :iff-def (and (time-point-equals (the ?st-1 (has-start-time ?time-
range-1 ?st-1))
                                      (the ?st-2 (has-start-time ?time-
range-2 ?st-2)))
                (time-point-equals (the ?et-1 (has-end-time ?time-
range-1 ?et-1))
                                     (the ?et-2 (has-end-time ?time-
range-2 ?et-2)))))


(def-relation TIME-POINT-WITHIN-INTERVAL (?tp ?interval)
 :constraint (and (time-point ?tp)
                  (time-interval ?interval))
 :iff-def (and (or (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp second-of))
                   (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp minute-of))
                   (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp hour-of))
                   (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp day-of))
                   (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp month-of))
                   (follows (the-slot-value ?interval has-end-time)
                            (the-slot-value ?tp year-of)))
               (or (time-point-precedes (the-slot-value ?interval has-
start-time)
                                        (the-slot-value ?tp second-
of))
                   (time-point-precedes (the-slot-value ?interval has-
start-time)
                                        (the-slot-value ?tp minute-
of))
                   (time-point-precedes (the-slot-value ?interval has-
start-time)
                                       (the-slot-value ?tp hour-of))
                   (time-point-precedes (the-slot-value ?interval has-
start-time)
                                       (the-slot-value ?tp day-of))
```

```
                    (time-point-precedes (the-slot-value ?interval has-
start-time)
                                         (the-slot-value ?tp month-of))
                    (time-point-precedes (the-slot-value ?interval has-
start-time)
                                              (the-slot-value ?tp year-
of)))))



;;;These are DERIVED Relations;;;

(def-relation IS-DURING (?time-range-1 ?time-range-2)
"It means that time-range-2 is in between (during) the the start and
end time of time-range-1."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
  :iff-def (and (time-point-precedes (the ?st-1 (has-start-time ?time-
range-1 ?st-1))
                                     (the ?st-2 (has-start-time ?time-
range-2 ?st-2)))
                (follows (the ?et-1 (has-end-time ?time-range-1 ?et-1))
                                 (the ?et-2 (has-end-time ?time-range-2 ?
et-2)))))



(def-relation BEFORE-OR-EQUAL (?time-range-1 ?time-range-2)
"It says that either one time range is before the other or is equal to
the other time range."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (before ?time-range-1 ?time-range-2)
              (meets ?time-range-1 ?time-range2)))



(def-relation AFTER-OR-EQUAL (?time-range-1 ?time-range-2)
"It says that either one time range is after the other or is equal to
the other time range."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (after ?time-range-1 ?time-range-2)
              (meets ?time-range-1 ?time-range-2)))


(def-relation IS-AFTER-THAN (?time-range-1 ?time-range-2)
"It is true when one time range is after the otehr time range."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (is-after ?time-range-2 ?time-range-1))


(def-relation DURING-OR-EQUAL (?time-range-1 ?time-range-2)
"It is true when one time range is-during the other time range or both
these time ranges starts or finishes simultaneously or they are equal
to each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (is-during ?time-range-1 ?time-range-2)
              (starts-simultaneously ?time-range-1 ?time-range-2)
              (finishes-simultaneously ?time-range-1 ?time-range-2)
              (time-range-equals ?time-range-1 ?time-range-2)))
```

```
(def-relation OVERLAPS-OR-MEETS (?time-range-1 ?time-range-2)
"It is true when two time ranges either overlaps with each other or
meets each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (overlaps ?time-range-1 ?time-range-2)
              (meets ?time-range-1 ?time-range-2)))


(def-relation OVERLAPS-OR-EQUALS (?time-range-1 ?time-range-2)
"It is true when two time ranges either overlaps with each other and
are equal to each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (overlaps ?time-range-1 ?time-range-2)
              (time-range-equals ?time-range-1 ?time-range-2)))


(def-relation STARTS-OR-EQUAL (?time-range-1 ?time-range-2)
"It is true when two time ranges either starts simulataneously or are
equal to each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (starts-simultaneously ?time-range-1 ?time-range-2)
              (time-range-equals ?time-range-1 ?time-range-2)))


(def-relation FINISHES-OR-EQUALS (?time-range-1 ?time-range-2)
"It is true when two time ranges finishes simultaneously or are equal
to each other."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (finishes-simultaneously ?time-range-1 ?time-range-2)
              (time-range-equals ?time-range-1 ?time-range-2)))



(def-relation DISJOINT-TIME-RANGES (?time-range-1 ?time-range-2)
"It is true if either time-range-1 is before time-range-2 or time-
range-2 is before time-range-1."
 :constraint (and (time-range ?time-range-1)
                  (time-range ?time-range-2))
 :iff-def (or (before ?time-range-1 ?time-range-2)
              (before ?time-range-2 ?time-range-1)))



(def-relation DUE-DATE-EARLIER-THAN-OTHER (?dd1 ?dd2)
"It says that if each of the slot value of due-date-1 precedes every
slot-value of due-date-2 then due-date-1 is earlier-than due-date-2."
 :constraint (and (calendar-date ?dd1)
                  (calendar-date ?dd2))
 :iff-def (and (time-point-precedes (the-slot-value ?dd1 day-of)
                                    (the-slot-value ?dd2 day-of))
               (time-point-precedes (the-slot-value ?dd1 month-of)
                                    (the-slot-value ?dd2 month-of))
               (time-point-precedes (the-slot-value ?dd1 year-of)
                                    (the-slot-value ?dd2 year-of))))
```

```
(def-relation DUE-DATE-LATER-THAN-OTHER (?dd2 ?dd1)
"It says that is each of the slot value of due-date-2 follows every
slot-value of due-date-1 then due-date-2 is later than due-date-1."
 :constraint (and (calendar-date ?dd1)
                  (calendar-date ?dd2))
 :iff-def (and (follows (the-slot-value ?dd2 day-of)
                        (the-slot-value ?dd1 day-of))
               (follows (the-slot-value ?dd2 month-of)
                        (the-slot-value ?dd1 month-of))
               (follows (the-slot-value ?dd2 year-of)
                        (the-slot-value ?dd1 year-of))))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;The following relations are defined for the time intervals exactly
as it is described in J F Allen's paper.

(def-relation TIME-INTERVAL-BEFORE (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
  :iff-def  (and  (time-point-precedes  (the-slot-value  ?ti1  has-start-
time)
                                        (the-slot-value ?ti2 has-start-
time))
               (time-point-precedes (the-slot-value ?ti1 has-end-time)
                                       (the-slot-value ?ti2 has-end-
time))))


(def-relation TIME-INTERVAL-EQUAL (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
 :iff-def (and (time-points-equal (the-slot-value ?ti1 has-start-time)
                                     (the-slot-value ?ti2 has-start-
time))
               (time-points-equal (the-slot-value ?ti1 has-end-time)
                                       (the-slot-value ?ti2 has-end-
time))))



(def-relation time-interval-meets (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
 :iff-def (time-points-equal (the-slot-value ?ti2 has-end-time)
                             (the-slot-value ?ti1 has-start-time)))



(def-relation time-interval-overlaps (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
  :iff-def  (and  (time-point-precedes  (the-slot-value  ?ti1  has-start-
time)
                                        (the-slot-value ?ti2 has-start-
time))
               (follows (the-slot-value ?ti1 has-end-time)
                        (the-slot-value ?ti2 has-start-time))
               (time-point-precedes (the-slot-value ?ti1 has-end-time)
                                       (the-slot-value ?ti2 has-end-
time))))
```

```
(def-relation time-interval-during (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
  :iff-def  (and  (time-point-precedes  (the-slot-value  ?ti1  has-start-
time)
                                        (the-slot-value ?ti2 has-start-
time))
               (follows (the-slot-value ?ti1 has-end-time)
                        (the-slot-value ?ti2 has-end-time))))


(def-relation time-interval-starts (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
 :iff-def (time-points-equal (the-slot-value ?ti1 has-start-time)
                             (the-slot-value ?ti2 has-start-time)))


(def-relation time-interval-finishes (?ti1 ?ti2)
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2))
 :iff-def (time-points-equal (the-slot-value ?ti1 has-end-time)
                             (the-slot-value ?ti2 has-end-time)))


(def-relation time-interval-elapsed-by (?ti1 ?ti2)
"This relation states, if one interval precedes another interval,
then, it says, by how much time another interval succeeds the prior
interval."
 :constraint (and (time-interval ?ti1)
                  (time-interval ?ti2)
                  (has-start-time ?ti1 ?st1)
                  (has-end-time ?ti1 ?et1)
                  (has-start-time ?ti2 ?st2)
                  (has-end-time ?ti2 ?et2))
 :iff-def (or (time-point-precedes ?et1 ?st2)
              (= ?et1
                 (+ (?st2 (= ?diff-tp
                             (time-point-difference ?et1 ?st2)))))))



(def-relation time-interval-during-delay-and-lag (?ti1 ?ti2)
"This relation states that if two intervals are during each other, and
if one interval delays or lag from the other interval, it states by
how much margin the interval has delayed or laged."
 :constraint (and (time-interval ?ti1)
                  (has-start-time ?ti1 ?st1)
                  (has-end-time ?ti1 ?et1)
                  (time-interval ?ti2)
                  (has-start-time ?ti2 ?st2)
                  (has-end-time ?ti2 ?et2))
 :iff-def (and (or (follows ?st2 ?st1)
                   (= ?st2
                      (+ (?st1
                          (= ?diff-tp
                             (time-point-difference ?st2 ?st1))))))
               (or (time-point-precedes ?st2 ?et1)
                   (= ?et1
                      (+ (?et2
                          (= ?diff-tp
                             (time-point-difference ?et1 ?et2)))))))
```

```
(def-relation time-interval-overlap-or-lag (?ti1 ?ti2)
"This relation states that if two intervals are overlapping each
other, and if one interval lags another interval, then it says by how
much margin these intervals are laged."
 :constraint (and (time-interval ?ti1)
                  (has-start-time ?ti1 ?st1)
                  (has-end-time ?ti1 ?et1)
                  (time-interval ?ti2)
                  (has-start-time ?ti2 ?st2)
                  (has-end-time ?ti2 ?et2))
 :iff-def (and (time-point-precedes ?st1 ?st2)
               (or (and (time-point-precedes ?st2 ?et1)
                        (time-point-precedes ?et1 ?et2))
                   (= ?et1
                      (+ (?st1
                          (= ?diff-tp
                             (time-point-difference ?et1 ?st2)))))))


(def-relation time-interval-starts-by (?ti1 ?ti2)
"This relation states that if two intervals starts at the same time,
and if one interval finishes after another interval then it says by
how much margin the interval finishes over other."
 :constraint (and (time-interval ?ti1)
                  (has-start-time ?ti1 ?st1)
                  (has-end-time ?ti1 ?et1)
                  (time-interval ?ti2)
                  (has-start-time ?ti2 ?st2)
                  (has-end-time ?ti2 ?et2))
 :iff-def (and (time-points-equal ?st1 ?st2)
               (or (time-point-precedes ?et1 ?et2)
                   (= ?et2
                      (+ (?et1
                          (= ?diff-tp
                             (time-point-difference ?et2 ?et1)))))))



(def-relation time-interval-finishes-dalay (?ti1 ?ti2)
"This relation stats that if two interval finishes at the same time,
but they have different start-time, then it says, by how much time
these two intervals differ in terms of thri start times."
 :constraint (and (time-interval ?ti1)
                  (has-start-time ?ti1 ?st1)
                  (has-end-time ?ti1 ?et1)
                  (time-interval ?ti2)
                  (has-start-time ?ti2 ?st2)
                  (has-end-time ?ti2 ?et2))
 :iff-def (and (or (follows ?st1 ?st2)
                   (= ?st1
                      (+ (?st2
                          (= ?diff-tp
                             (time-point-difference ??st1 ?st2))))))
               (time-points-equal ?et1 ?et2)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;; end of time specifications
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;; persistent-item branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(def-class persistent-item (crm-entity)
"E77-CIDOC - This class comprises items that have a persistent
identity, sometimes known as 'endurants' in philosophy.  They can be
repeatedly recognized within the duration of their existence by
identity criteria rather than by continuity or observation. Persistent
Items can be either physical entities, such as people, animals or
things, or conceptual entities such as ideas, concepts, products of
the imagination or common names. The criteria that determine the
identity of an item are often difficult to establish -; the decision
depends largely on the judgement of the observer. "
    () )


(def-class thing (persistent-item)
"E70-CIDOC - This general class comprises usable discrete,
identifiable, instances of E77 Persistent Item that are documented as
single units.  They can be either intellectual products or physical
things, and are characterized by relative stability. They may for
instance either have a solid physical form, an electronic encoding, or
they may be logical concept or structure.
++PhiloSurfical: The Cidoc version we had presented a has-dimension
slot at this level, which did not make much sense to us (how to apply
it to abstract concepts?) thus we moved it under physical-thing.."
 ((had-a-general-use :type type)
  (shows-features-of :type thing)))


(def-class legal-object (thing)
"E72-CIDOC - This class comprises those material or immaterial items
to which instances of E30 Right, such as the right of ownership or
use, can be applied.  This is true for all E18 Physical Thing. In the
case of instances of E28 Conceptual Object, however, the identity of
the E28 Conceptual Object or the method of its use may be too
ambiguous to reliably establish instances of E30 Right, as in the case
of taxa and inspirations."
 ((is-subject-to :type right)
  (right-held-by :type actor)))


(def-class man-made-thing (thing)
"E71-CIDOC - This class comprises discrete, identifiable man-made
items that are documented as single units. These items are either
intellectual products or man-made physical things, and are
characterized by relative stability.
++PhiloSurfical: We added a slot was-made-by, which is a shortcut for
the lengthier event-based representation "
 ((has-title :type title)
  (was-intended-for :type type)
  (was-made-by :type Actor) ))
```

```
(def-class physical-thing (legal-object)
"E18 -CIDOC - This class comprises all persistent physical items with
a relatively stable form, man-made or natural. Depending on the
existence of natural boundaries of such things, the CRM distinguishes
the
instances of E19 Physical Object from instances of E26 Physical
Feature, such as holes, rivers, pieces of land etc.
++PhiloSurfical: has-dimension slot was originally of thing, but it
makes more sense here. We also have omitted the properties: P49 has
former or current keeper, P51 has former or current owner, P58 has
section definition, P59 has section. P53 has former or current
location has been renamed to has-location. Has-current-keeper and
owner moved down to man-made-object"
 ((has-condition :type condition-state)
  (has-dimension :type dimension)
  (is-composed-of :type physical-thing)
  (has-location :type place)
  (has-owner :type Actor)
  (has-keeper :type Actor)
  (consists-of :type material)))


(def-class physical-object (physical-thing)
"E19 - CIDOC - This class comprises items of a material nature that
are units for documentation and have physical boundaries that separate
them completely in an objective way from other objects. The class also
includes all aggregates of objects made for functional purposes of
whatever kind, independent of physical coherence, such as a set of
chessmen. Typically, instances of E19 Physical Object can be moved (if
not too heavy).    In some contexts, such objects, except for
aggregates, are also called 'bona fide objects' (Smith-Varzi, 2000,
pp.401-420), i.e. naturally defined objects.
++PhiloSurfical: two locations slots have been omitted - the inherited
has-location seems to be enough. Is-identified-by and has-preferred-
identifier have been moved down to man-made-object"
 ((bears-feature :type physical-feature)
  (has-number-of-parts :type number)))


(def-class biological-object (physical-object)
"E20 - CIDOC - This class comprises individual items of a material
nature, which live, have lived or are natural products of or from
living organisms. "
 () )




(def-class physical-man-made-thing (man-made-thing physical-thing)
"E24-CIDOC  - This class comprises all persistent physical items that
are purposely created by human activity. This class comprises man-made
objects, such as a swords, and man-made features, such as rock art "
 ((depicts :type crm-entity) ;;(is depicted by)
  (shows-visual-item :type 2d-expression)
  (carries :type information-object)))

(def-class collection (physical-man-made-thing)
"E78 -CIDOC - This class comprises aggregations of physical items that
are assembled and maintained (\"curated\" and \"preserved,\" in
museological terminology) by one or more instances of E39 Actor over
time for a specific purpose and audience, and according to a
particular collection development plan "
 ((has-curator :type Actor)))
```

```
(def-class man-made-object (physical-man-made-thing physical-object)
" E22-CIDOC - This class comprises physical objects purposely created
by human activity. The difference with physical-man-made-thing is that
the latter can also include man-made-features, which do not have a
distinct existence as objects "
 ((is-identified-by :type object-identifier)
  (has-preferred-identifier :type object-identifier)
  (has-current-keeper :type Actor)
  (has-current-owner :type Actor)))


(def-class information-carrier (man-made-object)
"E84-CIDOC - This class comprises all instances of E22 Man-Made Object
that are explicitly designed to act as persistent physical carriers
for instances of E73 Information Object. This allows a relationship to
be asserted between an E19 Physical Object and its immaterial
information contents. An E84 Information Carrier may or may not
contain information, e.g., a diskette. Note that any E18 Physical
Thing may carry information, such as an E34 Inscription. ***However,
unless it was specifically designed for this purpose, it is not an
Information Carrier. Therefore the property P128 carries (is carried
by) applies to E18 Physical Thing in general "
 ())


(def-class item (information-carrier)
"FRBR : the information carrier that exemplifies a manifestation, i.e.
indirectly a production plan"
((exemplifies :type Manifestation :cardinality 1)))


(def-class physical-feature (physical-thing)
"E26 -CIDOC- This class comprises identifiable features that are
physically attached in an integral way to particular physical
objects."
 () )


(def-class man-made-feature (physical-man-made-thing physical-feature)
"E25-CIDOC - This class comprises physical features that are purposely
created by human activity, such as scratches, artificial caves,
artificial water channels, etc."
  () )


(def-class site (physical-feature)
"E27-CIDOC   - This class comprises pieces of land or sea floor. In
contrast to the purely geometric notion of E53 Place, this class
describes constellations of matter on the surface of the Earth or
other celestial body, which can be represented by photographs,
paintings and maps. "
    () )

(def-class actor (persistent-item)
"E39-CIDOC - This class comprises people, either individually or in
groups, who have the potential to perform intentional actions for
which they can be held responsible. "
 ((has-contact-point :type contact-point)
  (has-current-or-former-residence :type place)
  (possesses :type right)
  (is-identified-by :type actor-appellation)
  (has-currently-or-formerly-worked-for:type organization)
  (has-conceived :type propositional-content)
  (has-created :type work)
  (has-realized :type expression)
  (has-produced :type manifestation)))
```

```
(def-class individual (actor)
"++PhiloSurfical: If we have a group class, I don't see why not
putting also this one"
())


(def-class person (individual biological-object)
"E21-CIDOC - This class comprises real persons who live or are assumed
to have lived. Legendary figures that may have existed, such as
Ulysses and King Arthur, fall into this class if the documentation
refers to them as historical figures. In cases where doubt exists as
to whether several persons are in fact identical, multiple instances
can be created and linked to indicate their relationship. The CRM does
not propose a specific form to support reasoning about possible
identity.
++PhiloSurfical : The class has no slots in CIDOC. We added the gender
slot, which has no associated event; the has-social-role slot, cause
it's not something we want to record as an event (even if it would be
possible), but just a property useful for classifying thinkers; the
dates/places of birth/death are shortcuts, cause often we do not want
to model the birth/death events for everybody, but still want some
basic info about it. In the case of philosophers, in fact, this is not
needed cause people do not usually dispute these information, but more
their intellectual subscriptions. However, sometimes we are interested
in modeling such events, so in the future we need to construct an
axiom which establishes the priority of one or the other information,
or the inference of the existence of an event, from the value of the
person slots."
 ((has-gender :type gender)
  (has-date-of-birth :type time-specification)
  (has-date-of-death :type time-specification)
  (has-birth-place :type place)
  (has-death-place :type place)
  (has-social-role :type social-role)))


(def-class group (actor)
"E74-CIDOC - This class comprises any gatherings or organizations of
two or more people that act collectively or in a similar way due to
any form of unifying relationship. A gathering of people becomes an
E74 Group when it exhibits organizational characteristics usually
typified by a set of ideas or beliefs held in common, or actions
performed together.
++PhiloSurfical : Has current or former member  has been changed to
present tense. Since we can have groups of people and organization, we
introduces also the subclass group-of-people. (for groups of objects
there is already the class collection"
 ((has-member :type actor)))



(def-class Group-Of-People (Group)
"++PhiloSurfical: we added this class to refer explicitly to groups
composed only by people"
 ((has-member :type person)))

(def-class Belief-Group (Group-Of-People)
"++PhiloSurfical: Group of people united by a shared set of beliefs -
it is a social object also because we identify it tx to the belief
shared"
 ((shares-belief :type View :min-cardinality 1)
  (has-related-intellectual-event :type intellectual-movement)))
```

```
(def-class Philosophical-school (Belief-group) ?p
"++PhiloSurfical: E.g. the stoics or the circle of vienna, intended as
the people composing them"
 ((has-related-philo-event :type Philosophical-movement))
 :constraint  (and  (shares-belief ?p ?v)
                    (belong-to-area ?v ?a)
                    (branch-of-philosophy ?a)))


(def-class Political-group (Belief-group)
"++PhiloSurfical:")

(def-class Religious-group (Belief-group)
"++PhiloSurfical:")


(def-class legal-body (group)
"E40-CIDOC - This class comprises institutions or groups of people
that have obtained a legal recognition as a group and can act
collectively as agents.
++PhiloSurfical: we have added a set of organizations, partly taken
from AKT. See below."
   () )



;;; ******** organization imported from AKT  (and slightly modified)


(def-class ORGANIZATION (legal-body)
"AKT - An organization is a type of legal agent.
++PhiloSurfical: the original slot has-affiliated-person has been
overridden by has-member:agent; the has-size slot not needed in this
context. "
 ((organization-part-of :type organization)
  (has-sub-unit :type organization-unit)
  (headed-by :type person)))


(def-class POLITICAL-ORGANIZATION (organization)
"AKT - An organization which has a political connotation")


(def-class ORGANIZATION-UNIT (legal-body)
"AKT - An organization may have a number of units. Units may
themselves have sub-units.
++PhiloSurfical: we have taken out all the address related slots,
cause actors already have the has-contact slot. Same as above, for
has-size and has-affiliated-person"
 ((unit-of-organization :type organization)
  (sub-unit-of-organization-unit :type organization-unit)
  (has-sub-unit :type organization-unit)
  (headed-by :type person)))




(def-rule UNIT-OF-ORGANIZATION-IS-TRANSITIVE
 ((unit-of-organization ?u ?o)
  if
  (sub-unit-of-organization-unit ?u ?u-super)
  (unit-of-organization ?u-super ?o)))


(def-class PUBLISHING-HOUSE (organization)
"AKT - ")
```

```
(def-class NON-PROFIT-ORGANIZATION (organization)
"AKT - ")

(def-class PROFIT-ORGANIZATION (organization)
"AKT - "
  ((subsidiary-of :type profit-organization)))

(def-class PARTNERSHIP (profit-organization)
    "AKT - A partnership is not necessarily a company, e.g. a
consultancy firm is not a company")

(def-class COMPANY (profit-organization)
"AKT - ")

(def-class PRIVATE-COMPANY (company)
"AKT - ")

(def-class PUBLIC-COMPANY (company)
"AKT - ")

(def-class INDUSTRIAL-ORGANIZATION (profit-organization )
"AKT - ")

(def-class GOVERNMENT-ORGANIZATION (non-profit-organization)
"AKT - ")

(def-class CIVIL-SERVICE (GOVERNMENT-ORGANIZATION)
"AKT - ")

(def-class GOVERNMENT (GOVERNMENT-ORGANIZATION)
"AKT - "
  ((government-of-country :type country)))

(def-class CHARITABLE-ORGANIZATION (non-profit-organization)
"AKT - ")

(def-class LEARNING-CENTRED-ORGANIZATION (organization)
"AKT - ++PhiloSurfical: quite a useful class")

(def-class R-and-D-INSTITUTE (learning-centred-organization )
"AKT - ")

(def-class  R-and-D-INSTITUTE-WITHIN-LARGER-ORGANIZATION  (r-and-d-
institute organization-unit)
"AKT - ")

(def-class EDUCATIONAL-ORGANIZATION (learning-centred-organization)
"AKT - ++PhiloSurfical: quite a useful class")

(def-class  R-AND-D-INSTITUTE-WITHIN-EDUCATIONAL-ORGANIZATION (R-and-D-
INSTITUTE-WITHIN-LARGER-ORGANIZATION)
"AKT - ")



(def-class HIGHER-EDUCATIONAL-ORGANIZATION (educational-organization)
"AKT - "
  ((has-academic-unit :type academic-unit)
   (has-support-unit :type academic-support-unit)))

(def-rule HAS-ACADEMIC-UNIT-IMPLIES-HAS-ORGANIZATION-UNIT
  ((has-sub-unit ?x ?y)
   if
   (has-academic-unit ?x ?y)))
```

```
(def-rule HAS-SUPPORT-UNIT-IMPLIES-HAS-ORGANIZATION-UNIT
  ((has-sub-unit ?x ?y)
   if
   (has-support-unit ?x ?y)))

(def-class UNIVERSITY (higher-educational-organization)
"AKT - "
  ((has-faculty :type university-faculty)
   (has-vice-chancellor :type person))
  :slot-renaming ((has-vice-chancellor headed-by)))

(def-class DISTANCE-TEACHING-UNIVERSITY (university)
"AKT - ")

(def-class SCHOOL (educational-organization)
"AKT - ")

(def-class EDUCATIONAL-ORGANIZATION-UNIT (organization-unit) ?x
"AKT - "
  ((unit-of-organization :type educational-organization)
   )
  :iff-def (and (unit-of-organization ?x ?y)
                (educational-organization ?y)))

(def-class ACADEMIC-UNIT (educational-organization-unit)
"AKT - "
  ((unit-of-organization :type university)
   ))

(def-class UNIVERSITY-FACULTY (academic-unit)
"AKT - ")

(def-class ACADEMIC-SUPPORT-UNIT (educational-organization-unit)
"AKT - ")


(def-class GEO-POLITICAL-ENTITY (organization)
"++PhiloSurfical -  ")

(def-class Continent (GEO-POLITICAL-ENTITY ))

(def-class City (GEO-POLITICAL-ENTITY )
 ((part-of-country :type country)))

(def-class CAPITAL-CITY (city )
 ((is-capital-of :type country)))

(def-class country (GEO-POLITICAL-ENTITY)
 ((part-of-region :type Region)
  (part-of-continent :type Continent)))

(def-class Region (GEO-POLITICAL-ENTITY)
 ((part-of-continent :type Continent)
  (part-of-country :type country)))
```

```
;; ******* CONTACT POINTS

(def-class contact-point (persistent-item)
"E51-CIDOC - This class comprises identifiers used to communicate with
instances of E39 Actor. These include E-mail addresses, telephone
numbers, post office boxes, Fax numbers, etc. Most postal addresses
can be considered both as instances of E44 Place Appellation and E51
Contact Point. "
    () )



;;; ********  APPELLATION


(def-class appellation (persistent-item)
"E41-CIDOC - This class comprises all proper names, words, phrases or
codes, either meaningful or not, that are used or can be used to
identify a specific instance of some class within a certain context.
Instances of E41 Appellation do not identify objects by their meaning
but by convention, tradition or agreement. From an implementation
point of view, the class E41 Appellation is unlike most others, whose
instances in a database can be considered as surrogates or references
to real-world entities, in that each instance is nothing other than
the E41 Appellation itself, i.e. the instance of E41 Appellation
'Martin' is nothing other than the name 'Martin' which should not be
confused with any instance of E21 Person or persons called Martin."
 ((has-alternative-form :type appellation) ))

(def-class title (appellation)
" E35-CIDOC - left as it was. "
())

(def-class URI (appellation)
"++PhiloSurfical : Unique Resource Identifier: a particular type of
string")

(def-class URL (URI)
"++PhiloSurfical -  A URL is a particular type of URI")


(def-class object-identifier (appellation)
E42-CIDOC - This class comprises codes assigned to objects in order to
identify them uniquely within the context of one or more
organizations.Such codes are often known as inventory numbers,
registration codes, etc. and are typically composed of alphanumeric
sequences."
    () )

(def-class ISBN-Number (object-identifier)
"++PhiloSurfical ")



(def-class place-appellation (appellation)
"E44-CIDOC - This class comprises any sort of identifier
characteristically used to refer to an E53 Place. Instances of E44
Place Appellation may vary in their degree of precision and their
meaning may vary over time - the same instance of E44 Place
Appellation may be used to refer to several places, either because of
cultural shifts, or because objects used as reference points have
moved around."
    ())
```

```
(def-class place-name (place-appellation)
"E48-CIDOC - This class comprises particular and common forms of E44
Place Appellation. Place Names may change their application over time:
the name of an E53 Place may change, and a name may be reused for a
different E53 Place"
    () )


(def-class section-definition (place-appellation)
"E46-CIDOC - This class comprises areas of objects referred to in
terms specific to the general geometry or structure of its kind.The
'prow' of the boat, the 'frame' of the picture, the 'front' of the
building are all instances of E46 Section Definition. "
    () )

(def-class spatial-coordinates (place-appellation)
"E47-CIDOC - This class comprises the textual or numeric information
required to locate specific instances of E53 Place within schemes of
spatial identification. Coordinates are a specific form of E44 Place
Appellation, that is, a means of referring to a particular E53 Place.
Coordinates are not restricted to longitude, latitude and altitude.
Any regular system of reference that maps onto an E19 Physical Object
can be used to generate coordinates.
++PhiloSurfical: we added some definition of coordinates, pointing to
numbers. Boundary boxes define the space occupied by countries, as in
Geonames.org. From there we can also retreive other information, such
as the capital names!"
 ((has-latitude :type number)
  (has-longitude :type number)
  (has-altitude :type number)
  (has-bBoxWest :type number)
  (has-bBoxNorth :type number)
  (has-bBoxEast :type number)
  (has-bBoxSouth :type number)))


(def-class address (contact-point place-appellation)
"E45-CIDOC - This class comprises identifiers expressed in coding
systems for places, such as postal addresses used for mailing. An E45
Address can be considered both as the name of an E53 Place and as an
E51 Contact Point for an E39 Actor.
++PhiloSurfical: Address should provide some structure. For now this
has been achieved through the slots has-street-number, has-street-
name, has-town-or-city-name, has-country"
 ())


(def-class time-appellation (appellation)
"CIDOC - This class comprises all forms of names or codes, such as
historical periods, and dates, which are characteristically used to
refer to a specific E52 Time-Specification. The instances of E49 Time
Appellation may vary in their degree of precision, and they may be
relative to other time frames, 'Before Christ' for example. Instances
of E52 Time-Specification. are often defined by reference to a
cultural period or an event e.g. 'the duration of the Ming Dynasty'."
    () )

(def-class date (time-appellation)
"E50-CIDOC - This class comprises specific forms of E49 Time
Appellation."
    () )
```

```
(def-class conceptual-object-appellation (appellation)
"E75-CIDOC - This class comprises all specific identifiers of
intellectual products or standardized patterns, e.g. an ISBN number"
    () )


(def-class idea-appellation (conceptual-object-appellation)
"++PhiloSurfical - Class to refer to the names of concepts: typical of
area is an interesting property, likely to be used in information
extraction"
((typical-of-area :type Field-of-study)))




(def-class actor-appellation (appellation)
"CIDOC - This class comprises any sort of name, number, code or symbol
characteristically used to identify an E39 Actor.  "
())


(def-rule people-appellations
“PhiloSurfical: rule”
   (and (person ?x) (is-identified-by ?x ?y) (not (actor-appellation ?
y)))
  then
  (exec (tell (actor-appellation ?y))))


(def-rule create-common-names
"PhiloSurfical: If we do not want to bother creating all the common
names....."
  (and (has-common-name ?c ?n) (not (concept-appellation ?n)))
  then
  (exec (tell (concept-appellation ?n))))




;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;; temporal-entities
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



(def-class temporal-entity (crm-entity)
"E2-CIDOC  - This class comprises all phenomena, such as the instances
of E4 Periods, E5 Events and states, which happen over a limited
extent in time.  In some contexts, these are also called perdurants.
This class is disjoint from E77 Persistent Item. This is an abstract
class and has no direct instances. E2 Temporal Entity is specialized
into E4 Period, which applies to a particular geographic area (defined
with a greater or lesser degree of precision), and E3 Condition State,
which applies to instances of E18 Physical Thing.
++PhiloSurfical: The slot has-time-span is of type time-specification
rather than span as it may be a time point. This is consistent with
the CIPHER time ontology."
 ((has-time-specification :type time-specification)
  (is-equal-in-time-to :type Temporal-Entity)
  (finishes :type Temporal-Entity)
  (starts :type Temporal-Entity)
  (occurs-during :type Temporal-Entity)
  (occurs-before :type Temporal-entity)
  (overlaps-in-time-with :type Temporal-Entity)
  (meets-in-time-with :type Temporal-Entity)))
```

```
(def-class condition-state (temporal-entity)
"E3-CIDOC - This class comprises the state of objects characterized by
a certain condition over a time span.”
 ((consists-of :type condition-state)
  (has-state :type type)))



(def-class period (temporal-entity)
"E4-CIDOC - This class comprises sets of coherent phenomena or
cultural manifestations bounded in time and space. It is the social or
physical coherence of these phenomena that identify an E4 Period and
not the associated spatio-temporal bounds. [...] Typically this class
is used to describe prehistoric or historic periods such as the
'Neolithic Period', the 'Ming Dynasty' or the 'McCarthy Era'. There
are however no assumptions about the scale of the associated
phenomena. In particular all events are seen as synthetic processes
consisting of coherent phenomena. Therefore E4 Period is a superclass
of E5 Event. [...] Artistic style may be modeled as E4 Period.
++PhiloSurfical: the subclasses intellectual movement and
philosophical movement has been added as types of period."
 ((consists-of :type period)
  (falls-within :type period)
  (overlaps-with :type period)
  (is-separated-from :type period)
  (took-place-at :type place)
  (took-place-on-or-within :type physical-object)))


(def-class Intellectual-movement (period)
"++PhiloSurfical : E.g. enlightememnt, or dadaism.. we will eventually
differentiate other types of movement - artistic, etc.."
 ((has-related-group-of-people :type belief-group)
  (is-typified-by :type view)))

(def-class Philosophical-movement (Intellectual-movement)
"++PhiloSurfical : E.g. stoicism, or platonism, or even atomism? or is
this just a doctrine??? question here"
 ((has-related-group-of-people :type philosophical-school)
  (is-typified-by :type view)))


(def-class event (period)
"E5-CIDOC - This class comprises changes of states in cultural, social
or physical systems, regardless of scale, brought about by a series or
group of coherent physical, cultural, technological or legal
phenomena. Such changes of state will affect instances of E77
Persistent Item or its subclasses. The distinction between an E5 Event
and an E4 Period is partly a question of the scale of observation.
Viewed at a coarse level of detail, an E5 Event is an 'instantaneous'
change of state. "
 ((had-participant :type actor)
  (occurred-in-the-presence-of :type persistent-item)))


(def-relation has-attended-event (?p ?event)
  :sufficient
  (and (?p person)
       (?event event)
       (had-participant ?event ?p)))
```

```
(def-class activity (event)
"E7-CIDOC -This class comprises actions intentionally carried out by
instances of E39 Actor that result in changes of state in the
cultural, social, or physical systems documented. This notion includes
complex, composite and long-lasting actions such as the building of a
settlement or a war, as well as simple, short-lived actions such as
the opening of a door. "
 ((carried-out-by :type Actor)  ;; performed
  (was-influenced-by :type crm-entity)
  (used-specific-object :type Thing)
  (was-motivated-by :type crm-entity)
  (was-intended-use-of :type man-made-thing)
  (had-specific-purpose :type activity)
  (continued :type activity)))


(def-class beginning-of-existence (event)
"E63-CIDOC - This class comprises events that bring into existence any
E77 Persistent Item. It may be used for temporal reasoning about
things (intellectual products, physical items, groups of people,
living beings) beginning to exist; it serves as a hook for
determination of a terminus postquem and antequem.  "
 ((brought-into-existence :type persistent-item)))


(def-class end-of-existence (event)
"E64-CIDOC - This class comprises events that end the existence of any
E77 Persistent Item. It may be used for temporal reasoning about
things (physical items, groups of people, living beings) ceasing to
exist; it serves as a hook for determination of a terminus postquem
and antequem. In cases where substance from a Persistent Item
continues to exist in a new form, the process would be documented by
E81 Transformation."
 ((took-out-of-existence :type persistent-item) ))


(def-class formation (beginning-of-existence activity)
"E66-CIDOC : This class comprises events that result in the formation
of a formal or informal E74 Group of people, such as a club, society,
association, corporation or nation.  "
 ((has-formed :type Group) ))


(def-class birth (beginning-of-existence)
"E67-CIDOC - This class comprises the birth of a human beings. E67
Birth is a biological event focussing on the context of people coming
into life.(E63 Beginning of Existence comprises the coming into life
of any living beings).
++PhiloSurfical: it many be the case, as noted by cipher, that some
slots are not needed. For example, brought into life seems correct,
but from another point of view it seems it could be easily replaced by
the brought into existence of the beginning of existence class. For
indicating the place, we have the slot took-place-at, inherited from
period"
 ((by-mother :type person)
  (from-father :type person)
  (brought-into-life :type person)))


(def-class death (end-of-existence)
"E69-CIDOC   - This class comprises the deaths of human beings. If a
person is killed, their death should be instantiated as E69 Death and
as E7 Activity. The death or perishing of other living beings should
be documented using E64 End of Existence."
 ((was-death-of :type Person)) )
```

```
(def-class destruction (end-of-existence)
"E6-CIDOC - This class comprises events that destroy one or more
instances of E18 Physical Thing such that they lose their identity as
the subjects of documentation. "
 ((destroyed :type physical-thing)) )


(def-class dissolution (end-of-existence)
"E68-CIDOC - This class comprises the events that result in the formal
or informal termination of an E74 Group of people.  If the dissolution
was deliberate, the Dissolution event should also be instantiated as
an E7 Activity. "
 ((dissolved :type Group)))



(def-class transformation (beginning-of-existence end-of-existence)
"E81-CIDOC - This class comprises the events that result in the
simultaneous destruction of one E77Persistent Item and the creation of
another E77 Persistent Item that preserves recognizable substance from
the first but has a fundamentally different nature and identity. [...]
Instances of E81 Transformation are therefore distinct from re-
classifications (documented using E17 Type Assignment) or
modifications (documented using E11 Modification) of objects that do
not fundamentally change their nature or identity. "
 ((resulted-in :type Persistent-Item)
  (transformed :type Persistent-Item)))


(def-class experiment (activity)
"++PhiloSurfical : used-specific-object comes from activity!"
 ((within-research-area :type problem-area)
  (has-background-theory :type view)  ;; very broad for now
  (has-assumption :type assumption)
  (proves-view :type view)))


(def-class modification (activity)
"E11-CIDOC - This class comprises all instances of E7 Activity that
create, alter or change E24 Physical Man- Made Thing.   This class
includes the production of an item from raw materials, and other so
far undocumented objects, and the preventive treatment or restoration
of an object for conservation.    Since the distinction between
modification and production is not always clear, modification is
regarded as the more generally applicable concept.
++PhiloSurfical: Use-general-technique and used-specific-technique
have been combined into one slot used-technique of type design-or-
procedure."
 ((has-modified :type Physical-man-made-thing)
  (used-technique :type procedure)
  (employed :type material)))


(def-class part-addition (modification)
"E79-CIDOC    - This class comprises activities that result in an
instance of E24 Physical Man-Made Thing being increased, enlarged or
augmented by the addition of a part.  "
 ((augmented :type Physical-man-made-thing)
  (added :type Physical-thing)))
```

```
(def-class part-removal (modification)
"E80-CIDOC   - This class comprises the activities that result in an
instance of E18 Physical Thing being decreased by the removal of a
part. "
 ((diminished :type Physical-man-made-thing)
  (removed :type Physical-thing)))


(def-class move (activity)
    "E9-CIDOC  - This class comprises changes of the physical location
of the instances of E19 Physical Object. Note, that the class E9 Move
inherits the property P7 took place at (witnessed): E53 Place.This
property should be used to describe the trajectory or a larger area
within which a move takes place, whereas the properties P26 moved to
(was destination of), P27 moved from (was origin of) describe the
start and end points only. "
 ((moved :type Physical-Object)
  (moved-to :type  Place)
  (moved-from :type place)))


(def-class journey (move)
"++PhiloSurfical: Added this class to refer to voluntary movement of
humans. The slot carried-out-by, from activity, is renamed to has-
traveller"
 ((has-traveler :type person))
 :slot-renaming ((has-traveler carried-out-by)))

(def-class production (beginning-of-existence modification)
"E12-CIDOC - This class comprises activities that are designed to, and
succeed in, creating one or more new items.  It specializes the notion
of modification into production. The decision as to whether or not an
object is regarded as new is context sensitive. [...] This entity can
be collective: the printing of a thousand books, for example, would
normally be considered a single event "
 ((has-produced :type Physical-Man-Made-Thing) ))


(def-class PUBLICATION-EVENT (production)
"++PhiloSurfical: this class is inspired from the AKT reference
ontology. The slots have been slightly modified  -from event-product
to has-published. The information-carrier is the physical object
produced, while manifestation is the abstract reification of the
publication product "
 ((has-produced-manifestation :type manifestation)
  (has-published :type information-carrier))
  :slot-renaming ((has-published has-produced)))


(def-class transfer-of-custody (activity)
"E10-CIDOC - This class comprises transfers of physical custody of
objects between instances of E39 Actor.  The distinction between the
legal responsibility for custody and the actual physical possession of
the object should be expressed using the property P2 has type (is type
of). A specific case of transfer of custody is theft "
 ((custody-surrendered-by :type actor)
  (custody-received-by :type actor)
  (transferred-custody-of :type Physical-thing)) )
```

```
(def-class acquisition (activity)
"E8-CIDOC - This class comprises transfers of legal ownership from one
or more instances of E39 Actor to one or more other instances of E39
Actor. "
 ((transferred-title-from :type actor)
  (transferred-title-to :type actor)
  (transferred-title-of :type physical-thing)) )


;; claims are a subclass of attribute assignment
(def-class attribute-assignment (activity)
"E13-CIDOC - This class comprises the actions of making assertions
about properties of an object or any relation between two items or
concepts. This class allows the documentation of how the respective
assignment came about, and whose opinion it was. All the attributes or
properties assigned in such an action can also be seen as directly
attached to the respective item or concept, possibly as a collection
of contradictory values. "
 ((assigned-attribute-to :type crm-entity)
  (assigned :type crm-entity)))


(def-class condition-assessment (attribute-assignment)
"E14-CIDOC - This class describes the act of assessing the state of
preservation of an object during a particular period "
 ((has-identified :type condition-state)
  (concerned :type Physical-thing)))


(def-class identifier-assignment (attribute-assignment)
"E15-CIDOC - This class comprises actions assigning or de-assigning
object identifiers. Examples of such identifiers include Find Numbers
and Inventory Numbers"
 ((assigned :type object-identifier)
  (deassigned :type object-identifier)
  (registered :type Physical-object) ))

(def-class type-assignment (attribute-assignment)
"E17-CIDOC - This class comprises the actions of classifying items of
whatever kind. Such items include objects, specimens, people, actions
and concepts "
 ((assigned :type type)
  (classified :type crm-entity)))


(def-class measurement (attribute-assignment)
"E16-CIDOC - This class comprises actions measuring physical
properties and other values that can be determined by a systematic
procedure"
 ((observed-dimension :type dimension)
  (measured :type thing)))


(def-class social-activity (activity)
"++PhiloSurfical: Class comprising all the events where the social
component, i.e. th presence of various persons, is essential to their
existence"
 ((had-participant :type person)))


;;;;;;;;;;;;;; from AKT events


(def-class social-gathering (social-activity)
"AKT : ")
```

```
(def-class conference (social-gathering)
"AKT : "
  ((published-proceedings :type conference-Proceedings-Reference)))


(def-class workshop (social-gathering)
"AKT : "
  ((published-proceedings :type workshop-Proceedings-Reference)))


(def-class seminar (social-gathering)
"AKT : "
    ((published-proceedings :type workshop-Proceedings-Reference))) ;;
we havent defined any specific proceedings yet..


(def-class meeting (social-gathering)
"++PhiloSurfical: A meeting type of event. Note that both attendee and
organizer have  multiple cardinality"
  ((meeting-attendee :type person)
   (meeting-organizer :type person))
  :slot-renaming ((meeting-organizer carried-out-by)
                  (meeting-attendee  had-participant)))


(def-class close-social-contact (social-activity)
"++PhiloSurfical: Class that refers to encounters or other contacts
between two persons only"
((had-participant :type person :cardinality 2)
 (has-frequency :type frequency-of-occurence)))


(def-class 2-persons-meeting (close-social-contact meeting)
"++PhiloSurfical: "
((meeting-attendee :type person :cardinality 1)
 (meeting-organizer :type person :cardinality 1)))




(def-class mail-exchange (close-social-contact)
"++PhiloSurfical: "
 ((has-sender :type person :cardinality 1)
  (has-receiver :type person :cardinality 1)
  (has-item-sent :type information-carrier))
 :slot-renaming ((has-sender carried-out-by)
                 (has-receiver had-participant)))


(def-class telephone-conversation (close-social-contact)
"++PhiloSurfical: "
 ((has-caller :type person :cardinality 1)
  (has-receiver :type person :cardinality 1))
 :slot-renaming ((has-caller carried-out-by)
                 (has-receiver had-participant)))


(def-class educational-activity (social-activity)
"++PhiloSurfical: Any activity which has an educational context"
 ((has-subject-area :type problem-area)))
```

```
(def-class teaching (educational-activity)
"++PhiloSurfical: Generic teaching: not all teaching takes place in
schools..."
 ((has-teacher :type person)
  (has-learner :type person)))


(def-class learning (educational-activity)
"++PhiloSurfical: Generic learning not all teaching takes place in
schools..."
 ((has-learner :type person)
  (has-teacher :type person)))


(def-class joining-a-group (social-activity)
"++PhiloSurfical: Class that refers events about a person formally and
intentionally taking part in a group"
 ((group-joined :type group)))


(def-class working-for-organization (joining-a-group)
"++PhiloSurfical: "
 ((group-joined :type organization)
  (working-role :type social-role))) ;;for now only social role


(def-class teaching-at-institution (working-for-organization teaching)
"++PhiloSurfical: This class inherits also from the generic teaching
class"
 ((group-joined :type educational-organization)
  (working-role :type teaching-role)))


(def-class learning-at-institution (joining-a-group learning)
"++PhiloSurfical: This class inherits also from the generic learning
class"
 ((group-joined :type educational-organization)
  (degree-of-study :type academic-degree)))


(def-class discussion (social-activity)
"++PhiloSurfical: Generic activity of discussion - by facilitator we
mean the person (or more than one) who starts the discussion or leads
it"
 ((has-facilitator :type person)
  (has-subject-area :type problem-area)
  (has-topic :type problem)))


(def-class argumentation (discussion)
"++PhiloSurfical: More formal discussion, where view and arguments are
well delineated and contrast each other. We do not look into the
complex dynamics involved in an argumentation, but just try to give a
screenshot of the event"
 ((involves-argument :type argument-structure)
  (involves-view :type view)
  (has-person-attacking :type person)
  (has-person-defending :type person)))
```

```
;;;;;;;;;;;;;;;; intellectual activities


(def-class intellectual-activity (activity)
"++PhiloSurfical: Any theoretical activity which creates or modifies
directly only abstract entities. This class hierarchy is only
indicative, more work is needed here."
 ((involves-idea :type philosophical-idea)))


(def-class conceptual-creation (intellectual-activity beginning-of-
existence)
"++PhiloSurfical: This class replaces the original CIDOC-E65 -
creation"
 ((has-created :type conceptual-object)))


(def-class idea-conception (conceptual-creation)
"++PhiloSurfical: "
 ((has-created :type philosophical-idea)))


(def-class expression-creation (conceptual-creation)
"++PhiloSurfical:"
 ((has-created :type expression)))


(def-class idea-modification (intellectual-activity)
"++PhiloSurfical:"
 ((has-old-idea :type philosophical-idea)
   (has-new-idea :type philosophical-idea) ;; they must be of the same
type
  (within-context :type view)))


(def-class theory-refinement (idea-modification)
"++PhiloSurfical:"
 ((has-old-idea :type theory)
  (has-new-idea :type theory)
  (has-added-element :type philosophical-idea)
  (has-removed-element :type philosophical-idea)))


(def-class idea-usage (intellectual-activity)
"++PhiloSurfical:"
 ((idea-used :type philosophical-idea)
  (used-in-view :type view)))


(def-class theory-transposition (idea-usage)
"++PhiloSurfical:"
 ((idea-used :type theory)
  (old-context :type problem-area)
  (new-context :type problem-area)))



(def-class study (intellectual-activity)
"++PhiloSurfical:"
 ((studies :type crm-entity)))

(def-class study-an-idea (study)
"++PhiloSurfical:"
 ((studies :type philosophical-idea)))
```

```
(def-class study-a-document (study)
"++PhiloSurfical:"
 ((studies :type information-object)))


(def-class view-subscription (intellectual-activity)
"++PhiloSurfical: "
 ((old-view :type view)
  (new-view :type view)
  (convincing-argument :type argument)))


;;;;;;;;;; interpretations


(def-class interpretation (intellectual-activity)
"++PhiloSurfical: The act of making claims about intellectual
contents. At this level the only slots are 'what is interpreted', and
the generic relation 'is-about' which can link the interpretation to
pretty much anything e.g. the book is-about Napoleon. "
 ((has-interpretation :type propositional-content)
  (interprets :type crm-entity)
  (is-about-entity :type crm-entity)))


(def-class document-interpretation (interpretation)
"++PhiloSurfical:  "
 ((interprets :type information-object)
  (has-pedagogical-value :type pedagogical-functional-role)))


(def-class expression-interpretation (document-interpretation)
"++PhiloSurfical: interpretations about expressions, and among them,
linguistic representations. An add onof these interpretations is the
fact that we can give entities a pedagogical value. "
((interprets :type expression)))


(def-class event-interpretation (interpretation)
"++PhiloSurfical:  "
 ((interprets :type event)
  (causally-connected-to :type event)))


(def-class idea-interpretation (interpretation)
"The characteristic feature of an idea interpretation is that we can
claim its author to be somebody's different, from the one specified
initially with the instance"
 ((interprets :type propositional-content)
  (was-conceived-by :type actor)
  (is-related-to-idea :type philosophical-idea)
  (similar-to :type philosophical-idea)
  (contrasts-with :type philosophical-idea)))



(def-class concept-interpretation (idea-interpretation)
"Interpretations of concepts"
 ((interprets :type concept)
  (is-specialization-of :type Concept)
  (is-generalization-of :type Concept)
  (is-equivalent-to :type Concept)
  (has-opposite-concept :type Concept)
  (has-related-concept :type Concept)
  (requires-concept :type concept)
  (causes-concept :type concept)))
```

```
(def-class View-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of views: we are replicating some
slots of the view class, as an initial mechanism for providing support
for inconsistent and contrasting annotations. We'll look into the
issue more while enlarging the KB"
 ((interprets :type view)
  (defines-concept :type Concept)
  (uses-idea :type Philosophical-idea)
  (interprets-fact :type temporal-entity)
  (typifies :type intellectual-movement)
  (tackles-problem :type Problem)
  (attacked-by-problem :type Problem)
  (influences-View :type View)
  (influenced-by-View :type View)
  (supports-View :type View)
  (opposes-View :type View)
  (has-supporting-argument :type Argument)
  (has-opposing-argument :type Argument)
  (has-exemplar-document :type expression)))


(def-class Thesis-interpretation (View-interpretation)
"++PhiloSurfical : thesis interpretations - all the properties are in
common with the thesis class"
 ((interprets :type thesis)
  (part-of-thesis :type Thesis)
  (part-of-theory :type Theory)
  (part-of-school :type school-of-thought)
  (part-of-system :type Philosophical-system)))

(def-class Theory-interpretation (View-interpretation)
"++PhiloSurfical : theory interpretations - all the properties are in
common with the theory class"
 ((interprets :type theory)
  (part-of-theory :type Theory)
  (part-of-school :type school-of-thought)
  (part-of-system :type Philosophical-system)
  (defines-Method :type Method)
  (has-thesis :type Thesis)))


(def-class Philosophical-system-interpretation  (View-interpretation)
"++PhiloSurfical : - all the properties are in common with the
referring class "
 ((interprets :type philosophical-system)
  (part-of-school :type School-of-thought)
  (defines-method :type method)
  (has-theory :type theory)
  (has-thesis :type Thesis)))




(def-class School-of-thought-interpretation  (View-interpretation)
"++PhiloSurfical : all the properties are in common with the referring
class "
 ((interprets :type school-of-thought)
  (has-main-thesis :type Thesis)
  (classifies-view :type View)
  (has-exemplar-theory :type Theory)
  (has-main-exponent :type actor)))
```

```
(def-class Distinction-interpretation (idea-interpretation)
"++PhiloSurfical : just one property for now"
 ((interprets :type distinction)
  (related-to-problem :type problem)))


(def-class Problem-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of problems.."
 ((interprets :type problem)
  (has-problem-type :type problem-type)
  (exists-in-area :type Problem-area)
  (linked-to-fact :type Temporal-entity)
  (attacks-View :type View)
  (has-supportive-view :type View)
  (is-tackled-by-View :type View)
  (defined-by-argument :type Argument)
  (is-tackled-by-argument :type Argument)
  (related-to-problem :type Problem)
  (derives-from-problem :type Problem)
  (has-equivalent-meaning-as :type problem)
  (has-resolutive-method :type Method)))


(def-class problem-area-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of problem areas, of their
hierarchical relationships"
 ((related-to-area :type Problem-area)
  (sub-area-of :type Problem-area)
  (has-sub-area :type Problem-area)))


(def-class argument-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of arguments - we do not support
interpretations of argument parts for now, but it's possible to
implement that too"
 ((interprets :type argument)
  (supports-Idea :type Philosophical-idea)
  (contrasts-Idea :type Philosophical-idea)
  (tackles-problem :type Problem)
  (defines-problem :type problem)))


(def-class method-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of methods - very basic for now"
 ((interprets :type method)
  (supports-view :type view)))


(def-class Rhetorical-figure-interpretation (idea-interpretation)
"++PhiloSurfical : interpretations of rhet. figures - for now we can
just say what it supposedly relates to"
 ((interprets :type rhetorical-figure)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;; conceptual-object
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(def-class conceptual-object (man-made-thing)
"E28-CIDOC - This class comprises non-material products of our minds,
in order to allow for reasoning about their identity, circumstances of
creation and historical implications.
++PhiloSurfical: Refers-to-concept of type type has been removed.
Instead an additional slot is required called refers-to-crm-meta-
entity to allow concepts such as classes to be referred to. Refers-to
has been written as refers-to-crm-entity. This is used to refer to
anything in the CRM knowledge base. The subclass -language- has been
taken away, and put under form-of-expression."
 ((is-identified-by :type conceptual-object-appellation)))


(def-class right (conceptual-object)
"E30-CIDOC - This class comprises legal privileges concerning material
and immaterial things or their derivatives.    These include
reproduction and property rights."
 ())



(def-class type (conceptual-object)
"E55-CIDOC - This class comprises arbitrary concepts (universals) and
provides a mechanism for organising them into a hierarchy. This
hierarchy is intended to duplicate the names of all the classes
present in the model. This allows additional refinement, through
subtyping, of those classes which do not require further analysis of
their formal properties, but which nonetheless represent typological
distinctions important to a given user group.
++PhiloSurfical: basically type is used to provide classifications
which don't find a better place in the ontology"
 ((has-broader-term :type type)
  (has-narrower-term :type type)))


(def-class gender (type)
"++PhiloSurfical: Added by PhiloSurfical"
  () )

;; a couple of inevitable instances
(def-instance male gender)
(def-instance female gender)


(def-class material (type)
"E57-CIDOC - This class is a specialization of E55 Type and comprises
the concepts of materials Instances of E57 Material may denote
properties of matter before its use, during its use, and as
incorporated in an object, such as ultramarine powder, tempera paste,
reinforced concrete"
  () )


(def-class measurement-unit (type)
"E58-CIDOC - No local slots. "
    () )
```

```
(def-class Genre (type)
"++PhiloSurfical: Literary form used to represent a linguistic
expression. Here we do not refer to the material features of it, but
to the abstract structure a text or any other work can implement. E.g.
a sonnet, a stanza, an aphorism. We also include the pedagogical
discourse types, even if they have a much less structured form
compared to the literary ones."
 ())

(def-class Literary-genre (Genre)
"++PhiloSurfical: Basically the styles a literary text can have,
otherwise called genres")
;; useful instances
(def-instance Poetry Literary-genre)
(def-instance Confession Literary-genre)
(def-instance Treatise Literary-genre)
(def-instance Essay Literary-genre)
(def-instance Aphorism Literary-genre)
(def-instance Fictional-Text Literary-genre)
(def-instance Meditation Literary-genre)
(def-instance Dialogue Literary-genre)
(def-instance Novel Literary-genre)
(def-instance Biography Literary-genre)
(def-instance Autobiography Literary-genre)
(def-instance Play Literary-genre)

(def-class musical-genre (genre)
"++PhiloSurfical: ")
(def-instance Ballad musical-genre)
(def-instance Symphony musical-genre)
(def-instance Sonata musical-genre)


(def-class Problem-type (type)
"++PhiloSurfical: Characterizations of problems, depending on their
solutions' number")

;; useful instances
(def-instance Dilemma Problem-type
"++PhiloSurfical: Apparently unsolvable problem. A dilemma is a
problem offering two solutions, neither of which is acceptable. The
two options are often described as the horns of a dilemma, neither of
which is comfortable.. See also en.wikipedia.org/wiki/Dilemma")
(def-instance Multilemma Problem-type
"++PhiloSurfical: Problem that has different solutions")
(def-instance Open-problem Problem-type
"++PhiloSurfical: Problem that    is still unresolved or not
categorized")
(def-instance Paradox Problem-type
"++PhiloSurfical: A particularly difficult problem,.......It's an
'apparently true statement or group of statements that leads to a
contradiction or a situation which defies intuition. Typically, either
the statements in question do not really imply the contradiction, the
puzzling result is not really a contradiction, or the premises
themselves are not all really true or cannot all be true together',
http://en.wikipedia.org/wiki/Paradox")


(def-class Role (conceptual-object)
"  ++PhiloSurfical : The function or role objects can play in a
context. This class was not part of CIDOC, so we added it"
 ((played-by :type crm-entity)))
```

```
(def-class Social-Role (Role)
"++PhiloSurfical :Role played by humans within a community"
 ((played-by :type person)))

(def-class teaching-role (social-role)
"++PhiloSurfical : ")

;; useful instances
(def-instance Philosopher social-role)
(def-instance Scientist social-role)
(def-instance linguist social-role)
(def-instance writer social-role)
(def-instance logician social-role)
(def-instance professor teaching-role)


(def-class degree-type (role)
"++PhiloSurfical: Class that subsumes any kind of degrees - phd or
cooking degree. We were undecided whether a subclass of type or role,
and decided for the last option on the basis it is not used to
classify things, but it is more an attribute of a degree, the role it
plays within the educational curriculum - conclusion reached thanks to
a discussion with Riichiro Mizoguchi")


(def-class academic-degree (degree-type)
"++PhiloSurfical: Degrees conferred by universities or academic
institutions")

;; useful instances
(def-instance PhD academic-degree)
(def-instance BA academic-degree)
(def-instance MA academic-degree)
(def-instance Professorship academic-degree)


(def-class pedagogical-functional-role (role)
"++PhiloSurfical :Generic function that a resource can perform, from
the pedagogical perspective"
 ((played-by :type information-object)))


(def-class textual-structural-role (role)
"++PhiloSurfical : class that gathers the various structures a text
can be composed of. We decided to model everything as instances, as
this is the level of granularity needed here. The only exception is
reference, which has been modeled as class because by doing this we
could import all the AKT references. Similarly, all the other entities
could be better represented as classes, and by using a model with
abstracts also the chosen composition of elements into a
presentational structure e.g. what paragraph/intro/summary comes first
etc.  - for now we just leave this aside. In the actual ontology, the
parts of a text have a number and a slot indicating what text-part-
role they represent in the context of a self-contained-expression. It
is implicit, and left to the implementation of the system, the
increasing or decreasing ordering of the components to be chosen as
the presentational structure.")


(def-class reference (textual-structural-role)
())

;; useful instances
(def-instance Remark Pedagogical-functional-role)
(def-instance Original-text Pedagogical-functional-role)
```

```
(def-instance Theme Pedagogical-functional-role)
(def-instance Sub-theme Pedagogical-functional-role)
(def-instance Vocabulary-lexicon Pedagogical-functional-role)
(def-instance Etymology Pedagogical-functional-role)
(def-instance Historical-context Pedagogical-functional-role)
(def-instance Reference-element Pedagogical-functional-role)
(def-instance Exercise Pedagogical-functional-role)
(def-instance Explanation Pedagogical-functional-role)
(def-instance Side-context Pedagogical-functional-role)
(def-instance Main-context Pedagogical-functional-role)
(def-instance Anticipation Pedagogical-functional-role)

(def-instance question textual-structural-role)
(def-instance title textual-structural-role)
(def-instance subtitle textual-structural-role)
(def-instance paragraph textual-structural-role)
(def-instance reference textual-structural-role)
(def-instance introduction textual-structural-role)
(def-instance abstract textual-structural-role)
(def-instance summary textual-structural-role)


(def-class representational-medium (role)
"++PhiloSurfical: We need this class, cause we need to specify a
medium of the manifestation. But this is different from the instance
of information-carrier, which is a specific object carrying
information. That is, it's one of the items produced as a
manifestation, embodied in a specific medium. So the medium, intended
as the class of items carrying this feature, is here represented as a
type."
())


(def-class paper-medium (representational-medium)
"++PhiloSurfical:")
(def-class electronic-medium (representational-medium)
"++PhiloSurfical:")
(def-class audio-based (representational-medium)
"++PhiloSurfical:")
;; useful instances
(def-instance book-medium paper-medium)
(def-instance canvas-medium paper-medium)
(def-instance audio-tape-medium electronic-medium)
(def-instance video-tape-medium electronic-medium)
(def-instance cd-rom-medium electronic-medium)
(def-instance computer-medium electronic-medium)
(def-instance file-medium electronic-medium)  ;; the material part of
it
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;; information objects
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(def-class information-object (conceptual-object legal-object)
"++PhiloSurfical: it is analogous to the class information-object in
CIDOC and DOLCE. The abstraction of objects carrying information,
separated from their physical embodiment. Every IO must have a form
and a content. We added the slot has-original-date to refer to the
date when we suppose the IO was 'originally' created; actually, since
an IO is abstract, it would make much more sense to specify the date
of the correspondent instance of manifestation, instead."
 (has-form :type representational-form)
  (has-content :type propositional-content)
  (has-original-date :type time-specification)))


(def-class expression (information-object)
"FRBR : The intellectual or artistic realization of a work in the form
of alpha-numeric, musical, or choreographic notation, sound, image,
object, movement, etc., or any combination of such forms.
++PhiloSurfical: We mantained the name 'expression' for this class,
even if it doesnt carry more meaning than its superclass information-
object. Essentially, this class exists just for facilitating the
mappings to FRBR."
 ())

(def-class self-contained-expression (expression)
"FRBR-CIDOC : an expression that conveys the whole idea of the
proposition it represents"
 ((realizes :type work)))


(def-class expression-fragment (expression)
"FRBR-CIDOC : an expression that conveys only partially the content of
the proposition it represents"
 ((part-of-expression :type self-contained-expression)))


(def-class symbolic-expression (self-contained-expression)
 ((has-form :type symbolic-form)))

(def-class 2d-expression (self-contained-expression)
((has-form :type iconic-form)))

(def-class text (symbolic-expression)
"++PhiloSurfical:  "
 ((has-form :type natural-language)))

(def-class utterance (symbolic-expression)
"++PhiloSurfical:  "
 ((has-form :type spoken-language)))

(def-class musical-score (symbolic-expression)
"++PhiloSurfical:  "
 ((has-form :type musical-symbol-sequence)
  (has-content :type piece-of-music)))

(def-class Formula (symbolic-expression)
"++PhiloSurfical:   A syntactically well-formed formula, e.g. in any
knowledge representation language."
 ((has-form :type artificial-language)))

(def-class symbol-figure (2d-expression)
 ((has-form :type line-image)))
```

```
(def-class film (2d-expression)
 ((has-form :type motion-image)))

(def-class painting (2d-expression) ())

(def-class calligraphy (2d-expression) ())


(def-class symbolic-expression-fragment (expression-fragment)
"++PhiloSurfical: just one fragment for now.. the others will come
easily . The has-string-content property is not ontologically sound...
but quite a useful shortcut when instantiating hundreds of sentences
of a text! "
 ((has-form :type symbolic-form)
  (has-number-reference :type number)
  (has-string-content :type String)))


(def-class sentence (symbolic-expression-fragment)
"++PhiloSurfical: an obvious expression fragment we decided to make
explicit.. "
 ((has-form :type natural-language)
  (has-textual-role :type textual-structural-role)))


(def-class word (symbolic-expression-fragment)
"++PhiloSurfical: atomic linguistic element"
 ((has-form :type natural-language)))

;;;;;;;;;; representational form



(def-class representational-form (conceptual-object)
"++PhiloSurfical:  inspired primarily from DOLCE")

(def-class symbolic-form (representational-form)
"++PhiloSurfical: ")
(def-class iconic-form (representational-form)
"++PhiloSurfical: ")

(def-class musical-symbol-sequence (symbolic-form)
"++PhiloSurfical:   ")
(def-class natural-language (symbolic-form)
"++PhiloSurfical: ")
(def-class artificial-language (symbolic-form)
"++PhiloSurfical: ")


(def-class still-image (iconic-form)
"++PhiloSurfical:  ")
(def-class line-image (still-image)
"++PhiloSurfical:  ")
(def-class motion-image (iconic-form)
"++PhiloSurfical:  ")

(def-class written-language (natural-language)
"++PhiloSurfical:"  )
(def-class spoken-language (natural-language)
"++PhiloSurfical:"  )

(def-class mathematical-notation (artificial-language)
"++PhiloSurfical:  ")
(def-class programming-language (artificial-language)
"++PhiloSurfical:   ")
```

```
;;;; a couple of useful instances..
(def-instance line-image still-image)
(def-instance hand-drawn-image still-image)
(def-instance photo still-image)


;;;;;;;;;;; manifestation


(def-class Manifestation (conceptual-object)
"FRBR: The physical embodiment of an expression of a work.
++PhiloSurfical: Since a manifestation can also be the single item
(manuscript) produced by an author, a publication is something
different from a manifestation. In such an extreme case, the instance
of manifestation represents the one-element set comprising the only
existing copy of the expression - and the instance of the item class
stands for the physical object itself. The has-title slot is
inherited, and the was-made-by too "
 ((embodies :type information-object :cardinality 1)
  (has-physical-medium :type representational-medium)
  (is-exemplified-by :type information-carrier)
  (has-date :type time-specification)))



;;;;;;;;;;; publications imported from AKT


(def-class PUBLICATION (Manifestation)
"AKT : A publication is something which has one or more publication
references. A publication can be both an article in a journal or a
journal itself. The distinction between publication and publication-
reference makes it possible to distinguish between multiple
occurrences of the same publication, for instance in different media.
++PhiloSurfical: this reflects out model, since a reference is an
expression, while the publication is a manifestation.    The has-
abstract and has-date slots have been removed.  "
 ((embodies :type symbolic-expression)
  (has-first-author :type actor)
  (has-other-authors :type actor)
  (has-publication-reference :min-cardinality 1
                             :type publication-reference)
  (cites-publication-reference :type publication-reference)))


(def-rule FIRST-AUTHOR-OF-PUBLICATION
  ((has-first-author ?pub ?x) if
   (Publication ?pub)
   (has-publication-reference ?pub ?ref)
   (has-first-author ?ref ?x)))

(def-rule OTHER-AUTHORS-OF-PUBLICATION
  ((has-other-authors ?pub ?l) if
   (Publication ?pub)
   (has-publication-reference ?pub ?ref)
   (has-other-authors ?ref ?l)))

(def-rule TITLE-OF-PUBLICATION
  ((has-title ?pub ?t) if
   (Publication ?pub)
   (has-publication-reference ?pub ?ref)
   (has-title ?ref ?t)))
```

```
(def-rule AUTHOR-OF-PUBLICATION-OR-REF
  ((has-author ?pub ?x) if
   (or (has-first-author ?pub ?x)
       (and (has-other-authors ?pub ?l)
            (member ?x ?l)))))



(def-class ELECTRONIC-PUBLICATION (Publication)
"AKT : A publication produced in electronic form"
 ((has-physical-medium :type electronic-medium)))

(def-class COMPOSITE-PUBLICATION (publication)
"AKT : A publication which contains items which can be themselves
referenced through a  publication reference.  Composite publications
include newspapers, magazines and journals.    A book which is a
collection of articles is a composite publication, a monograph is not"
  ((contains-publication :min-cardinality 1
                         :type publication)
   (has-publication-reference :type composite-publication-reference)))

(def-relation EDITED-BY (?x ?author)
  :iff-def (and (or (composite-publication-reference ?x)
                    (composite-publication ?x))
                (has-author ?x ?author)))


(def-class SERIAL-PUBLICATION (Publication) ?x
  "AKT : This used to be called periodical publication.  However, many
periodicals do not appear at fixed intervals, which is why librarians
refer to them as serials. So, we now use the concept of serial
publication and the has-periodicity slot has been removed.
++PhiloSurfical: we removed the has-impact-factor slot. ")

(def-class PERIODICAL-PUBLICATION (serial-publication) ?x
  "AKT : A periodical-publication is published regularly, such as once
every week.    Strictly speaking, the noun 'periodical' is used by
librarians to refer to things published at intervals of greater than a
day.   We use the phase periodical-publication to include newspapers
and other daily publications, since they share many bibliographic
features. The periodicity indicates how often the publication comes
out.     Note that this is a duration, rather than a time interval. A
time interval indicates a specific time interval on the time
continuum, so we need    to model periodicity as a time quantity"
  ((has-periodicity :cardinality 1 :type duration)))


(def-axiom  CONSISTENCY-BETWEEN-COMPOSITE-PUBLICATIONS-AND-THEIR-
CONTENTS
  (<=> (included-in-publication ?a ?p)
       (contains-publication ?p ?a)))


(def-class JOURNAL (serial-publication composite-publication)
"AKT : "
   ((contains-article :type publication))
   :slot-renaming ((contains-article contains-publication)))

(def-class MAGAZINE (serial-publication composite-publication)
"AKT :"
  ((contains-article :type publication))
  :slot-renaming ((contains-article contains-publication)))
```

```
(def-class NEWSPAPER (periodical-publication composite-publication)
"AKT :"
  ((contains-news-item :type news-item))
  :slot-renaming ((contains-news-item contains-publication)))

(def-class NEWSLETTER (serial-publication composite-publication)
   "AKT : Merrian-Webster says: a small publication (such as a leaflet
or newspaper) containing news of interest chiefly to a special group"
  ((contains-news-item :type news-item))
  :slot-renaming ((contains-news-item contains-publication)))



(def-class DAILY-NEWSPAPER (newspaper)
"AKT :"
  ((has-periodicity :value 24-hour-duration)))


(def-axiom MUTUALLY-EXCLUSIVE-SERIAL-PUBLICATIONS
  (subclass-partition  Serial-Publication
                                (Setof Journal Magazine Newspaper)))

(def-class BOOK (publication)
"AKT : "
  ((has-publication-reference :min-cardinality 1
                              :type book-reference)))


(def-class PROCEEDINGS (composite-publication)
"AKT : "
  ((has-publication-reference :min-cardinality 1
                              :type proceedings-reference)
   (contains-publication :min-cardinality 1
                         :type paper-in-proceedings)))

(def-class CONFERENCE-PROCEEDINGS (proceedings)
"AKT : "
  ((has-publication-reference :min-cardinality 1
                              :type conference-proceedings-reference)
   (refers-to-event :type conference)))

(def-class WORKSHOP-PROCEEDINGS (proceedings)
"AKT :"
  ((has-publication-reference :min-cardinality 1
                              :type workshop-proceedings-reference)
   (refers-to-event :type workshop)))


(def-class ITEM-IN-A-COMPOSITE-PUBLICATION (publication)
"AKT : "
  ((included-in-publication :type composite-publication)
   (has-publication-reference :min-cardinality 1
                               :type reference-to-item-in-a-composite-
publication)))

(def-class BOOK-SECTION-CONTRIBUTION (item-in-a-composite-publication)
"AKT : "
  ((included-in-publication :type edited-book)
   (has-publication-reference :min-cardinality 1
                                     :type book-section-contribution-
reference )))
```

```
(def-class ARTICLE-IN-A-JOURNAl (item-in-a-composite-publication)
"AKT : "
  ((included-in-publication :type journal)
   (has-publication-reference :min-cardinality 1
                              :type article-reference )))

(def-class JOURNAL-ISSUE (composite-publication)
"AKT :  "
  ((included-in-publication :type journal)
   (has-publication-reference :min-cardinality 1
                              :type journal-issue-reference )))

(def-class PAPER-IN-PROCEEDINGS (item-in-a-composite-publication)
"AKT :  "
  ((included-in-publication :type proceedings)
   (has-publication-reference :min-cardinality 1
                              :type paper-in-proceedings-reference )))

(def-class NEWS-ITEM (item-in-a-composite-publication)
"AKT :  ")



;;;;;;; references  imported from AKT


(def-class PUBLICATION-REFERENCE (reference)
  "AKT : this branch comes form there, but in ++PhiloSurfical we have
decided that a publication reference is an intangible, abstract
information, subclass of information object"
  ((has-first-author :type actor)
   (has-other-authors :type actor)
   (has-date :type calendar-date)
   (has-place-of-publication :type place-appellation)
   (refers-to-publication :type publication :cardinality 1)))

(def-axiom CONSISTENCY-BETWEEN-PUBLICATIONS-AND-THEIR-REFERENCES
  (<=> (refers-to-publication ?r ?p)
       (has-publication-reference ?p ?r)))

(def-class WEB-REFERENCE (publication-reference)
"AKT :"
  ((has-URL :type URL)))


(def-class BOOK-REFERENCE (Publication-Reference)
"AKT :"
  ((published-by :type publishing-house)
   (has-ISBN-number :type ISBN-Number)))

(def-class COMPOSITE-PUBLICATION-REFERENCE (publication-reference)
"AKT :"
    ((refers-to-publication  :type  composite-publication  :cardinality
1)))

(def-class  EDITED-BOOK-REFERENCE  (composite-publication-reference
Book-Reference)
"AKT :"
  ((refers-to-publication :type edited-book :cardinality 1)))

(def-class PROCEEDINGS-REFERENCE (composite-publication-reference)
"AKT :"
  ((refers-to-publication :type proceedings :cardinality 1)))
```

```
(def-class CONFERENCE-PROCEEDINGS-REFERENCE (proceedings-reference)
"AKT :"
    ((refers-to-publication  :type  conference-proceedings  :cardinality
1)))

(def-class WORKSHOP-PROCEEDINGS-REFERENCE (proceedings-reference)
"AKT :"
  ((refers-to-publication :type workshop-proceedings :cardinality 1)))

(def-class  REFERENCE-TO-ITEM-IN-A-COMPOSITE-PUBLICATION  (publication-
reference)
"AKT :"
  ((refers-to-publication :type item-in-a-composite-publication)
   (has-page-numbers :type string)))


(def-class BOOK-SECTION-CONTRIBUTION-REFERENCE  (reference-to-item-in-
a-composite-publication )
"AKT :"
   ((refers-to-publication :type book-section-contribution :cardinality
1)
   ))
(def-class  ARTICLE-REFERENCE
  (reference-to-item-in-a-composite-publication )
"AKT :"
  (
   (refers-to-publication :type article-in-a-journal :cardinality 1)
   (issue-number :type integer)
   (issue-volume :type integer)))

(def-class  JOURNAL-ISSUE-REFERENCE
  (reference-to-item-in-a-composite-publication )
"AKT :"
  (
   (refers-to-publication :type journal-issue :cardinality 1)
   (issue-number :type integer)
   (issue-volume :type integer)))

(def-class  PAPER-IN-PROCEEDINGS-REFERENCE  (reference-to-item-in-a-
composite-publication )
"AKT :"
  ((refers-to-publication :type paper-in-proceedings :cardinality 1)))

(def-class THESIS-REFERENCE (Publication-Reference)
"AKT :"
((institute-of-thesis :type academic-unit)))
    ;; (degree-of-thesis :type academic-degree)  we dont have academic
degrees for now

(def-class TECHNICAL-REPORT-REFERENCE (Publication-Reference)
"AKT :"
  ((published-by :type organization)
   (has-tech-report-number :type integer)))

(def-class EDITED-BOOK (composite-publication book)
"AKT :"
  ((has-publication-reference :min-cardinality 1
                              :type edited-book-reference)
   (contains-publication :min-cardinality 1
                         :type book-section-contribution)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;; propositional contents
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(def-class propositional-content (conceptual-object)
"++PhiloSurfical: Class that comprises all abstract, man-made contents
of information objects and representations in general. We are mainly
dealing with philosophical ideas, however we reproduced a simple
hierarchy of propositions, inspired by Mizoguchi. As he says in -
Tutorial on ontological engineering - Part 3, 2004-,  --- the content
of the symbolic representations are recognized as proposition which
has two kinds of proposition as its subclasses: Design proposition and
Product proposition. The former works as specification of the
production of something. The latter itself is the product. For
example, a piece of music composed is a specification of the music
sound produced by the music player. Procedure is specification of the
valid sequence of actions. An execution of the procedure generates a
result(product). Novel cannot be specification of anything because it
is already a product. This division of propositional contens needs
further investigation - it is not entirely used by the PhiloSurfical
application, as the philosophical-idea entities are our main focus,
for now"
 ())

(def-class work (propositional-content)
"++PhiloSurfical: in order to maintain compatibility with FRBR we
created another propositional-content, which is work. This is an
entity referring to any self-contained and identifiable content of an
information-object, similarly to Cyc's Conceptual_work. As such, this
class is the result of a conceptualization which is orthogonal to the
one used in Mizoguchi's analysis. Thus, many of the other classes
under design-proposition or product-proposition just mentioned can
also be rightfully considered as subclasses of work. "
)


(def-class design-proposition (propositional-content)
"++PhiloSurfical: taken from Mizoguchi classification- he says that
'It is essentially a specification for the production phase(mode)
which theoretically follows it and used as specification' -. It refers
to all proposition which are made in order to specify an activity. In
other words, what we are interested in, given a design proposition, is
not there yet unless we produce it with an action e.g. a piece of
music, or a play"
 ())

(def-class product-proposition (propositional-content)
"++PhiloSurfical: taken from Mizoguchi classification, it refers to
all propositions which are concluded in themselves, e.g. a novel, a
problem, or a theory. They are not supposed to be used as the
specification of something else"
 ())

(def-class procedure (design-proposition)
"++PhiloSurfical:")

(def-class piece-of-music (design-proposition work)
"++PhiloSurfical:")

(def-class symphony (piece-of-music work)
"++PhiloSurfical:")

(def-class drama (design-proposition work)
"++PhiloSurfical:")
```

```
(def-class symbol (design-proposition)
"++PhiloSurfical:")

(def-class specification (design-proposition)
"++PhiloSurfical:")

(def-class novel (product-proposition work)
"++PhiloSurfical:")

(def-class poem (product-proposition work)
"++PhiloSurfical:")

(def-class painting (product-proposition work)
"++PhiloSurfical:")

(def-class philosophical-work (product-proposition work)
"++PhiloSurfical: class that comprises all abstract works in
philosophy")

(def-class Dialogue (philosophical-work)
"Form of literature used by the greeks and indians for purposes of
rhetorical entertainment and instruction.")

(def-class treatise (philosophical-work)
"a written work devoted to the systematic examination of a particular
subject, usually philosophical or scientific")


(def-class Essay (philosophical-work)
"A short literary comp[osition on a single subject, usually presenting
the personal view of the author")

(def-class Autobiography (philosophical-work)
"Biography of oneself narrated by oneself e.g. Augustine's
confessions")

(def-class Meditation (philosophical-work)
"A contemplative discourse, usually on a religious or philosophical
subject")







;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;; philosophical ideas
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



(def-class Philosophical-idea (product-proposition)
"++PhiloSurfical : a philosophical idea is a propositional content
with a specific importance within the philosophical world.
Philosophical ideas are usually product propositions, as they are
important in themselves, for argumentation or theoretical purposes,
and not for specifying an action. The class methods is kind of
borderline - future refinements should address this!"
 ((has-description :type String)
  (has-common-name :type idea-appellation)
  (has-referred-author :type Actor))
:slot-renaming ((has-referred-author was-made-by)))
```

```
;;;;;;;;;;;;;;;;;;    V I E W


(def-class View (Philosophical-idea)
"++PhiloSurfical : Generic class referring to propositions expressing
a viewpoint, that is, descriptions defining other concepts and, in
general, meanings. The property exists-in-area can have multiple
values. Defines-method is excluded at this level cause a thesis is not
enough to define a method. "
 ((exists-in-area :type Problem-area)
  (interprets-fact :type temporal-entity)
  (typifies :type intellectual-movement)))


(def-class Thesis (View)
"++PhiloSurfical : Philosophical-idea expressing a standpoint in need
of demonstration but not necessarily having it"
 ((part-of-thesis :type Thesis)
  (part-of-theory :type Theory)
  (part-of-system :type Philosophical-system)))

(def-class Law (Thesis)
"++PhiloSurfical : A thesis with vast predictive power, especially in
scientific areas")

(def-class Principle (Thesis)
"++PhiloSurfical : It is a thesis demonstrated or taken as fundamental
in a philosophical system, e.g. the -ego- in the -idealism-")

(def-class Self-evident-principle (Principle)
"++PhiloSurfical : A principle that is self-demonstrated, so it's a
valid assumption")




(def-class Theory (View)
"++PhiloSurfical : A systemic conceptual construction, with a coherent
and organic architecture. It explains a specific phenomena (or set of)
and answers to an existing problem. It comprises concepts,
propositions and it HAS to be related to at least one argument. This
is required to guarantee a more solid architecture, compared for
example to a School-of-thought. Examples are the -theory of evolution-
or the -theory of possible worlds-."
 ((part-of-theory :type Theory)
  (part-of-system :type Philosophical-system)
  (defines-Method :type Method)
  (has-thesis :type Thesis)))


(def-class Philosophical-theory (theory)
"++PhiloSurfical : a theory exixting within some philosophical area of
research"
 ((exists-in-area :type branch-of-philosophy)))


(def-class Scientific-theory (theory)
"++PhiloSurfical : "
 ((exists-in-area :type Scientific-discipline)
  (predicts-fact :type temporal-entity)
  (verified-by-fact :type temporal-entity)))
```

```
(def-class Philosophical-system  (view)
"++PhiloSurfical : It is the Philosophical-idea comprising the set of
an author's theories, within a system. A philosopher can build
different ph. systems in his life, see Wittgenstein for example"
 ((defines-method :type method)
  (has-theory :type theory)
  (has-thesis :type Thesis)))


(def-class School-of-thought  (View)
"++PhiloSurfical : A generic standpoint, a belief or view about a
subject area or a problem. It is not as formalized and systematic as a
theory, and its contents are limited to be thesis. Examples are -
pacifism-, -animism-, -expansionism-. They could be further classified
depending on their area of provenience (politics, religion). This
class is needed cause often a mental content refer to a belief, but
without the specificity of a theory"
 ((has-main-thesis :type Thesis)
  (has-exemplar-theory :type Theory)
  (classifies-view :type View)))


(def-class Contextualized-school-of-thought (School-of-thought)
"++PhiloSurfical : Needed to separate the meaning of generic schools
or conceptions, to their related but more specific meanings in
specific fields of study"
 ((is-about-school :type School-of-thought)
  (exists-in-area :type Field-of-study)))


;;;;;;;;;;;;;;;;; C O N C E P T


(def-class Concept (Philosophical-idea)
"++PhiloSurfical : atomic propositional content, in relation to a view
defining it. It is recognizable because it composes the building
material for a view, and acquires meaning from it"
 ((defined-by-view :type View)
  (is-specialization-of :type Concept)
  ;; narrower than
  (is-generalization-of :type Concept)
  ;; broader than
  (is-equivalent-to :type Concept)
  ;; has the same meaning
  (has-opposite-concept :type Concept)
  ;; the antynomyc relation (gradable, relational, complementary)
  (has-related-concept :type Concept)
  ;; reflects a generic and positive semantic closure
  (requires-concept :type concept)
  ;; notional dependency, implication such as "buy" and "pay"
  (causes-concept :type concept)))
  ;; e.g. to kill-to die


(def-class principial-entity (concept)
"++PhiloSurfical : ")
(def-class supernatural-entity (concept)
"++PhiloSurfical : ")
(def-class god ( principial-entity  supernatural-entity)
"++PhiloSurfical : ")
```

```
;;;;;;;;;;;;; D I S T I N C T I O N


(def-class Distinction (Philosophical-idea)
"++PhiloSurfical : "
 ((exists-in-area :type problem-area)
  (related-to-problem :type problem)
  (defined-by-view :type view)
  (contains-concept :type concept)))


(def-class Dichotomy (Distinction)
"++PhiloSurfical : A classification that states the breakdown of a
space between two opposing principles or concepts, e.g.mind/body,
relations-of-ideas/matters-of-fact in Hume, a-priori/a-posteriori in
Kant "
 ((contains-concept :type concept :cardinality 2)))



;;;;;;;;;;; P R O B L E M - A R E A


(def-class Problem-area (Philosophical-idea)
"++PhiloSurfical : A set of problems linked by different kind of
relational schemas, offer organized around a central problem."
 ((contains-problem :type Problem)
  (has-central-problem :type problem)
  (specified-by-criteria :type thesis)
  (related-to-area :type Problem-area)
  (sub-area-of :type Problem-area)
  (has-sub-area :type Problem-area)))


(def-class Field-of-study (Problem-area)
"++PhiloSurfical : A field of study is seen as a particular kind of
problem-area. The distinctive trait of a field of study, is that the
area has been socially and historically recognized as separate from
the others, and from being a mere agglomerate of problems. Therefore,
being fundamentally social constructs, field-of-studies are born and
die in history. "
 ((defined-by-view :type View)
  (has-exemplar-theory :type theory)
  (has-methodology :type method)))

(def-class generic-field-of-study (Problem-area)
"++PhiloSurfical : Field of study defined extensionally."
 ((defined-by-view :type view)))


(def-rule generic-field-rule
(defined-by-view ?GF ?V) if (generic-field-of-study ?GF)
 (has-sub-area ?GF ?F)
          (defined-by-view ?F ?V)


(def-class Humanistic-discipline (generic-field-of-study)
"++PhiloSurfical : ")

(def-class Scientific-discipline (generic-field-of-study)
"++PhiloSurfical : ")


(def-class Branch-of-philosophy (Humanistic-discipline)
```

```
"++PhiloSurfical : The set of established research areas in
philosophy, classic subjects of teaching and investigation.We have
taken most of them are taken from Wordnet and from http://
www.routledge.com/Philosophy/subjects")



;;;;;;;;;;;; P R O B L E M


(def-class Problem (Philosophical-idea)
"++PhiloSurfical : the start of a philosophical enquiry"
 ((contains-concept :type concept)
  (has-problem-type :type problem-type)
  (exists-in-area :type Problem-area)
  (has-supportive-view :type View)
  (related-to-problem :type Problem)
  (derives-from-problem :type Problem)
  (has-equivalent-meaning-as :type problem)
  (defined-by-argument :type Argument)
  (is-tackled-by-argument :type Argument)
  (is-tackled-by-View :type View)
  (attacks-View :type View)
  (linked-to-fact :type Temporal-entity)))


(def-class existence-problem (Problem)
"++PhiloSurfical : Problem of the kind 'Does X exist?'"
 ((contains-concept :type Concept :cardinality 1)))

(def-class existence-as-concrete-problem (existence-problem)
"++PhiloSurfical : Problem of the kind 'Is X concrete/real?'")

(def-class existence-as-abstract-problem (existence-problem)
"++PhiloSurfical : Problem of the kind 'Is X abstract?'")

(def-class definitory-problem (Problem)
"++PhiloSurfical : Problem of the kind 'What is X?'"
 ((contains-concept :type Concept :cardinality 1)))

(def-class definitory-problem-essence (definitory-problem)
"++PhiloSurfical : Problem of the kind 'what are the characteristic
traits X has?'")

(def-class definitory-problem-attribute (definitory-problem)
"++PhiloSurfical : Problem of the kind 'what are the attributes X
has?'")


(def-class composition-problem (definitory-problem-essence)
"++PhiloSurfical : Problem of the kind 'What is X composed of?")

(def-class functional-problem (Problem)
"++PhiloSurfical : Problem of the kind 'What is the function of X?'"
 ((contains-concept :type Concept :cardinality 1)))

(def-class purpose-problem (functional-problem)
"++PhiloSurfical : Problem of the kind 'What is the purpose of X?'")

(def-class relational-problem (Problem)
"++PhiloSurfical : Problem of the kind 'What is the relation between X
and Y?'"
 ((contains-concept :type Concept :cardinality 2)))

(def-class dependence-problem (relational-problem)
"++PhiloSurfical : Problem of the kind 'Are X and Y dependent?'")
```

```
(def-class dependence-cause-problem (dependence-problem)
"++PhiloSurfical : Problem of the kind 'Is X the cause of Y?'")

(def-class dependence-effect-problem (dependence-problem)
"++PhiloSurfical : Problem of the kind 'Is X the effect of Y?'")

(def-class independence-problem (relational-problem)
"++PhiloSurfical : Problem of the kind 'Is X independent from Y?'")

(def-class equality-problem (relational-problem)
"++PhiloSurfical : Problem of the kind 'Is X equal to Y?'")

(def-class difference-problem (relational-problem)
"++PhiloSurfical : Problem of the kind 'Is X different from Y?'")

(def-class modality-problem (Problem)
"++PhiloSurfical : Problem about the degree of certainty X is likely
to happen, or not"
 ((contains-concept :type Concept :cardinality 1)))

(def-class necessity-problem (modality-problem)
"++PhiloSurfical : Problem of the kind 'is X necessary?'")

(def-class possibility-prolem (modality-problem)
"++PhiloSurfical : Problem of the kind 'is X possible?'")

(def-class contingency-problem (modality-problem)
"++PhiloSurfical : Problem of the kind 'is X contingent?'")

(def-class impossibility-problem (modality-problem)
"++PhiloSurfical : Problem of the kind 'is X impossible?'")

(def-class factual-problem (Problem)
"++PhiloSurfical : Problem of the kind 'how, in what way does X
happen, or manifests itself?"
 ((contains-concept :type Concept :cardinality 1)))



;;;;;;;;;;;;;;;;   A R G U M E N T   S T R U C T U R E


(def-class Argument-Structure (Philosophical-idea)
"++PhiloSurfical : Class used to group all the argument-related
entities"
 ((associated-to-view :type View)
  (has-associated-concept :type concept)
  (has-associated-event :type argumentation)
  (has-associated-experiment :type experiment)))

(def-class Argument (Argument-Structure)
"++PhiloSurfical : Reification of the argumentation class - it
represents the famous argumentative knots in the history of
philosophy"
 ((uses-method :type argumentative-method)
  (has-argument-part :type argument-part)))


(def-class Inductive-argument (Argument)
"++PhiloSurfical : ")
 ((uses-method :type inductive-method)))

(def-class Deductive-argument (Argument)
"++PhiloSurfical : ")
 ((uses-method :type deductive-method)))
```

```
(def-class abductive-argument (Argument)
"++PhiloSurfical : ")
 ((uses-method :type abductive-method)))


(def-class Argument-part (Argument-Structure)
"++PhiloSurfical : Statements or sets of statements having a logical
value within an argument."
 ((belongs-to-argument :type Argument)))


(def-class Assumption (Argument-part)
"++PhiloSurfical : Proposition that does not necessarily have a
demonstration, but that is taken as true. E.g. the assumption of
'goodness of god' in Descartes. In such cases, it is quite close to a
view, a principle."
 ((assumed-in-theory :type Theory)
  (assumed-in-School-of-thought :type School-of-thought)
  (developed-in :type Demonstration)))

(def-class Hypothesis (assumption)
"++PhiloSurfical : Assumption that is used in an argument,
specifically in a science-related one (experimental). It implicitly
supposes that the hypothesis will be proven true or false by the
experiment."
 ((used-in-experiment :type experiment)))

(def-class Demonstration (Argument-part)
"++PhiloSurfical : The propositions describing the middle steps in the
argumentation"
 ((develops-premise :type Hypothesis)
  (produces-conclusion :type Thesis)))

(def-class Conclusion (Argument-part)
"++PhiloSurfical : Propositions related to the conclusion of an
argument. Here there is some implicit overlap with the meaning of
thesis, which is a view. We have not formalized this relation yet."
 ((produced-by-demonstration :type Demonstration)))




;;;;;;;;;;;;;;;    M E T H O D


(def-class Method (Philosophical-idea)
"++PhiloSurfical : A recognized philosophical methodology, abstract or
practical. It can be mapped to Dolce's procedures"
 ((is-used-by-view :type View)
  (used-by-argument :type argument)
  (is-defined-by-view :type View)))


(def-class Abstract-method (Method)
"++PhiloSurfical : A procedure of reasoning or an abstract
methodology. For example, the dialectic method in Plato, the epoche'
in Husserl, Baco's scientific method or the absolute doubt of
Descates")


(def-class Practical-method (Method)
"++PhiloSurfical : A practical doctrine of life, such as the non-
violence doctrine of Gandhi, or the ascetism in Schopenauer ")
```

```
(def-class logical-mathematical-method (abstract-method)
"++PhiloSurfical : ")


(def-class algorithm (logical-mathematical-method)
"++PhiloSurfical : ")


(def-class rule-of-inference (abstract-method)
"++PhiloSurfical : ")


(def-class fallacy (rule-of-inference)
"++PhiloSurfical : ")


(def-class argumentative-method (abstract-method)
"++PhiloSurfical : ")

(def-class deductive-method argumentative-method)
(def-class inductive-method argumentative-method)
(def-class abductive-method argumentative-method)


(def-class Precept (Practical-method)
"++PhiloSurfical : A rule of personal conduct, like a commandment in
Christianity or the five precepts in buddism")

(def-class Scientific-method (Practical-method)
"++PhiloSurfical : A rule of personal conduct, like a commandment in
Christianity or the five precepts in buddism")



;;;;;;;;;;;;;;    R E T H O R I C A L    F I G U R E



(def-class Rhetorical-figure (Philosophical-idea)
"++PhiloSurfical : A proposition that uses some literary figure to
express or hint at its meaning"
 ((used-in-argument :type Argument)
  (used-in-View :type View)))


(def-class Maxime (Rhetorical-figure) ;;
"++PhiloSurfical : Short statement expressing a general truth or rule
of conduct. Examples are -amor fati- or -adequatio intellectus ad rem-
or -cogito ergo sum-"
 ((explains-metaphorically :type View)))




(def-class Metaphor (Rhetorical-figure)
"++PhiloSurfical : Any Philosophical-idea that means something in a
not-direct manner, e.g. through some analogy"
 ((explains-metaphorically :type View)))


(def-class Analogy (metaphor)
"++PhiloSurfical : ...")


(def-class Myth (Metaphor)
```

```
"++PhiloSurfical : The myth of the cave in Plato"
 ((explains-metaphorically :type View)))


(def-class Thought-experiment (Rhetorical-figure)
"++PhiloSurfical : The twin earth of Putnam, or the Chinese room of
Searle..")


(def-class Example (Rhetorical-figure)
"++PhiloSurfical : ...")

(def-class counter-example (example)
"++PhiloSurfical : ")



;;;;;;;;;;;;;;;;
;;; end of classes specification
;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;; functions for operating with the ontology
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;; ++++
;; generic ocml functions



(defun has-granfather? (inst-name granpa)
"From an instance, checks if has a class among its fathers"
(if (find-all-current-instances-named-x inst-name)
        (let* ((inst (first (find-all-current-instances-named-x inst-
name)))
           (parent (parent-class inst)))
       (if (or
            (equal parent (get-ocml-class granpa))
            (subclass-of* parent (list (get-ocml-class granpa))))
          t
         nil))))


(defun has-father? (inst-name father)
(if (find-all-current-instances-named-x inst-name)
         (let* ((inst (first (find-all-current-instances-named-x inst-
name)))
              (parent (parent-class inst)))
          (if (equal parent (get-ocml-class father))
              t
            nil))))


(defun get-father-from-instance (inst-name)
"From the *name* of an instance it gets the *name* of the father"
(let ((result ""))
  (if (find-all-current-instances-named-x inst-name)
         (let* ((inst (first (find-all-current-instances-named-x inst-
name)))
              (parent (parent-class inst)))
         (setq result (format nil "~S" (name parent))))
    (setq result "unknown type"))
  result))


(defun my-slot-values (instance slot)
"wrapper for the ocml original function: mine takes just the instance
name and the slot name, returns a list"
  (let ((resulto nil))
         (setf resulto    (get-slot-values (first (find-all-current-
instances-named-x instance)) slot))
    resulto))
```

```
(defun common-name (instance-name)
"checks if it is an idea, and if it is outputs the common name (if
there is one, otherwise NIL) - instead if it is a person, tries to
output the <identified-by> value...."
(let ((is-idea (not (not (instance-of? instance-name 'philosophical-
idea))))
      (is-person (not (not (instance-of? instance-name 'person))))
              (is-organization (not (not (instance-of? instance-name
'organization))))
          (is-informationobject (not (not (instance-of? instance-name
'information-object))))
      (result nil))
  (if is-idea
       (setf result (first (setofall '?x `(HAS-COMMON-NAME ,instance-
name ?x)))))
  (if is-person
       (setf result (first (setofall '?x `(is-identified-by ,instance-
name ?x)))))
  (if is-organization
       (setf result (first (setofall '?x `(is-identified-by ,instance-
name ?x)))))
  (if is-informationobject
       (setf result (first (setofall '?x `(is-identified-by ,instance-
name ?x)))))
  result))


(defun has-surname? (instance-name)
  (if (instance-of? instance-name 'person)
      (first (setofall '?x `(HAS-surname ,instance-name ?x)))))

(defun has-name? (instance-name)
  (if (instance-of? instance-name 'person)
      (first (setofall '?x `(HAS-name ,instance-name ?x)))))



(defun  find-ideas-with-name  (name)    ;;  &optional  (view  'first-
wittgenstein-philosophy)
"Just returns a LIST of the instances of ideas with a given COMMON-
NAME"
(let ((ideas (setofall '?x `(and (philosophical-idea ?x) (has-common-
name ?x ,name)))))
  ideas))


(defun string-description (anything)
"Give the string description of an idea - i.e. the value of the has-
description slot in most cases, or alternatively the value of the has-
note slot"
(let ((desc1 (setofall '?d `(has-description ,anything ?d)))
      (desc2 (setofall '?d `(has-note ,anything ?d)))
      (result nil))
  (if desc1
      (setf result (first desc1))
    (if desc2
        (setf result (first desc2))
      (setf result "No description")))
  result))
```

```
;; e.g. (what-url? (first (all-io-about-x "I")))
(defun what-url? (entity)
(if (exists? entity)
    (my-slot-values entity 'HAS-URI)))



;; +++
;; functions to manipulate sentences and texts
;; +++



;; gets an ordered list of the numbers of the sentences part-of an
expression
(defun get-tractatus-paragraphs-numbers ( &optional (expression
'Tractatus-pears-english-version))
"Just gets all the numbers of the tractatus sentences -by default-,
and orders them"
    (let* ((raw-listt (setofall '?x `(and (sentence ?p) (part-of-
expression ?p ,expression) (has-number-reference ?p ?x))))
         (good-listt nil))
    (dolist (item (sort raw-listt #'<))
      (push item good-listt))
    (reverse good-listt)))


(defun get-sentences-numbers (sentences)
"From a list of sentences, outputs as ordered list of their numbers"
  (if (listp sentences)
      (let ((output nil))
        (dolist (sentence sentences)
                          (push (first (setofall '?x `(has-number-
reference ,sentence ?x))) output))
        (sort output #'<))))


;;  extract the string content of a sentence
(defun sentence-string-content (sentence &optional (expression
'Tractatus-pears-english-version))
"Just outputs the string content of a sentence"
(if (setofall '?x `(sentence ,sentence))  ;;if the sentence exists
    (let ((content ""))
      (setf content (format nil "~a"
                                          (first (setofall '?x `(and
(sentence ,sentence) (part-of-expression ,sentence ,expression) (has-
string-content ,sentence ?x))))))
      content)))


(defun sentence-string-content-from-number (number &optional
(expression 'Tractatus-pears-english-version))
"Just outputs the string content of a sentence, from a number (it
could be constructed from get-sentence-number + sentence-string-
content"
(if (numberp number)
    (let ((content ""))
      (setf content (format nil "~a"
                               (first (setofall '?x `(and (sentence ?s)
(part-of-expression ?s ,expression) (has-number-reference ?s ,number)
(has-string-content ?s ?x))))))
      content)))
```

```
(defun get-sentence-from-number (number &optional (expression
'Tractatus-pears-english-version))
"gets the sentence instance, from the reference number"
(if (numberp number)
    (let ((sentence nil))
      (setf sentence
                (first (setofall '?x `(and (sentence ?x) (part-of-
expression ?x ,expression) (has-number-reference ?x ,number)))))
      sentence)))

;;e.g. (get-sentences-from-numbers (extract-interval (get-tractatus-
paragraphs-numbers) 1 2))
(defun get-sentences-from-numbers (numbers &optional (expression
'Tractatus-pears-english-version))
"same as above, but for a list of numbers"
 (let ((sentences nil))
   (dolist (number numbers)
     (push (get-sentence-from-number number expression) sentences))
   (reverse sentences)))



;; ++++
;; functions that return an interpretation object
;; +++++


;; returns the interpretation objects  about an information object

(defun find-interpretation-of-io (io &optional (author 'michele-
pasin))
"Find the interpretations-objects of a given sentence, given that they
are part of an expression"
(if io
       (let ((interpretation (setofall '?i `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ,io)))))
     interpretation)))


;; returns the interpretation objects  about a sentence

(defun find-interpretation-of-sentence (sentence &optional (author
'michele-pasin) (expression 'Tractatus-pears-english-version))
"Find the interpretations-objects of a given sentence, given that they
are part of an expression"
(if sentence
       (let ((interpretation (setofall '?i `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ,sentence) (part-of-
expression ,sentence ,expression)))))
     interpretation)))


;; returns the interpretations object about an idea

(defun find-interpretation-of-idea (idea &optional (author 'michele-
pasin))
"Find the interpretation of a given idea"
(if idea
       (let ((interpretation (setofall '?i `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ,idea)))))
     interpretation)))
```

```
(defun find-interpretation-of-idea-named (name &optional (author
'michele-pasin))
"Find the interpretation of a given idea, given its common name"
(if name
       (let ((interpretation (setofall '?i `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ?idea) (has-common-name ?
idea ,name)))))
     interpretation)))



;; returns the whole set of tractatus' sentences interpretations

(defun findall-interpretations (&optional (author 'michele-pasin)
(expression 'Tractatus-pears-english-version))
"Basically finds all the interpreations-contents related to the
tractatus, returns the PROPOSITIONS directly"
(let ((result nil))
   (setf result (setofall '?x `(and (interpretation ?i) (carried-out-
by ?i ,author) (interprets ?i ?s)  (part-of-expression ?s ,expression)
(has-interpretation ?i ?x))))
  result))


(defun findall-interpretations-of-type (type    &optional (author
'michele-pasin) (expression 'Tractatus-pears-english-version))
"Basically finds all the interpreations=metadata related to the
tractatus, of a particular type e.g. theory"
(let ((result nil))
   (setf result (setofall '?x `(and (interpretation ?i) (carried-out-
by ?i ,author) (interprets ?i ?s)  (part-of-expression ?s ,expression)
(has-interpretation ?i ?x) (,type ?x) )))
  result))



;; ++++
;; inverse: functions that return the sentence, given the VALUE of an
interpretation object
;; +++++

(defun find-sentences-interpreted-as (interpretation    &optional
(author 'michele-pasin))
"Find all the sentences where a proposition is defined as
interpretation - outputs an ordered list"
(if interpretation ;; (setofall '?x `(propositional-
content ,interpretation))  not doable for now
    (let ((sentences (setofall '?x `(and (interpretation ?i) (carried-
out-by ?i ,author)    (has-interpretation ?i ,interpretation)
(interprets ?i ?x)))))
      (sort sentences #'string-lessp))))

(defun find-sentences-interpreted-as-name (name    &optional (author
'michele-pasin))
"From the COMMON-NAME of an idea/proposition, finds all the sentences
where a proposition is defined as interpretation - outputs an ordered
list"
(let ((sentences (setofall '?x `(and (interpretation ?i) (carried-out-
by ?i ,author)    (has-interpretation ?i ?int) (has-common-name ?
int ,name) (interprets ?i ?x)))))
  (sort sentences #'string-lessp)))
```

```
;; ++++
;; functions that return the VALUE of an interpretation object
;; +++++


;; given an information object

;; ****** the one to be used with the encyclopedia data
;;e.g.  (find-io-whats-about  'DICTIONARY-OF-PHILOSOPHY-PUBLICATION-
G7509)
(defun find-io-whats-about (io &optional (author 'michele-pasin))
"The is-about-entity slot is a generic descirption of the content of
the object"
(if io
     (let ((aboutof (setofall '?x `(and (interpretation ?i) (carried-
out-by ?i ,author) (interprets ?i ,io) (is-about-entity ?i ?x)))))
     aboutof)))


(defun find-io-interpretation-content (io &optional (author 'michele-
pasin))
"Here it gets the propositional content, value of has-interpretation
slot"
(if io
      (let ((interpretation (setofall '?x `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ,io) (has-interpretation ?
i ?x)))))
     interpretation)))


;; inverse - from a content

(defun  all-io-interpretations-about-x  (entity  &optional  (author
'michele-pasin))
"Return interpretation objects"
(if entity
     (let ((result (setofall '?x `(and (interpretation ?x) (carried-
out-by ?x ,author) (is-about-entity ?x ,entity)))))
     result)))


;;(all-io-about-x "I")
(defun all-io-about-x (entity &optional (author 'michele-pasin))
"It's a generic function, but still the about slot-value must MATCH
EXACTLY the entity value"
(if entity
       (let ((interpretations (all-io-interpretations-about-x entity
author))
         (result '()))
     (dolist (interpretation interpretations)
                (setf  result  (append  (my-slot-values  interpretation
'INTERPRETS) result)))
     result)))
```

```
;;;;;;;;
```

```
;; ++++++++++++ string similarity functions
;;;;;;;

;; this function looks only for the repetition of characters.. not
'semantically' very useful!
(defun all-aboutvalues-similar-to-x (string limit &optional (author
'michele-pasin))
(if string
    (let ((result nil)
                (potential-x (setofall '?e `(and (interpretation ?x)
(carried-out-by ?x ,author) (is-about-entity ?x ?e)))))
      (dolist (i potential-x)
        (if (cl-user::char-word-similarity? string i limit)
            (setf result (append (list i) result))))
      (sort result #'string-lessp))))


;; this function looks for repetitions of the words: makes much more
sense
(defun all-aboutvalues-words-similar-to-x (string &optional (limit 0)
(author 'michele-pasin))
(if string
    (let ((result nil)
                (potential-x (setofall '?e `(and (interpretation ?x)
(carried-out-by ?x ,author) (is-about-entity ?x ?e)))))
      (dolist (i potential-x)
                  (if (cl-user::morph-sentence-similarity? string i
limit)    ;;cl-user::good-word-similarity? for letter-similarity..
            (setf result (append (list i) result))))
      (sort result #'string-lessp))))




(defun word-also-dealt-with-in (string &optional (limit 0) (author
'michele-pasin))
(if (all-aboutvalues-words-similar-to-x string limit author)
    (let ((result nil))
        (dolist (word (all-aboutvalues-words-similar-to-x string limit
author))
        (let* ((inf-objs (all-io-about-x word))
                    (publisher (setofall '?x `(was-made-by ,(first inf-
objs) ?x ))))
               (setf result (append result (list `(,word ,(first inf-
objs) ,publisher))))))
      result)))




;; ++++
;; functions to find out all interpretations-values of an expression
(sentence)

;; given a sentence(s)

(defun find-sentence-interpretation (sentence &optional (author
'michele-pasin))
"Find the interpretation-content of a given sentence // if there are
more interpretations, it returns all the contents, and gets rid of the
double values - e.g. (find-sentence-interpretation 'sentence-1)"
(if sentence
       (let ((interpretation (setofall '?x `(and (interpretation ?i)
(carried-out-by ?i ,author) (interprets ?i ,sentence) (has-
interpretation ?i ?x)))))
     interpretation)))
```

```
(defun findmany-sentences-interpretations (sentences &optional (author
'michele-pasin))
"Find interpretation-content of a set of sentences - takes a list and
outputs a list of lists"
(if (listp sentences)
    (let ((interpretations nil))
       (dolist (sentence sentences)
           (push (setofall '?x `(and (interpretation ?i) (carried-out-
by ?i ,author) (interprets ?i ,sentence) (has-interpretation ?i ?x)))
interpretations))
       interpretations)))




;; +++
;;  functions  that  gives  all  the  interpretation  VALUES  of  an
INTERPRETATION and/or IDEA  (not a sentence!)


;; from an INTERPRETATION

(defun find-interpretation-content (interpretation)
"Done especially for idea-interpretations // Outputs a list of pairs,
containing the slot and the interpretation value - I DECIDED what are
the interpretations to output!!!"
(if (not (not (instance-of? interpretation 'interpretation)))
    (let ((whatidea (get-father-from-instance interpretation))
          (result nil))
      (cond

       ((equal whatidea "CONCEPT-INTERPRETATION")
                 (let ((slots '(IS-RELATED-TO-IDEA HAS-RELATED-CONCEPT
HAS-OPPOSITE-CONCEPT  SIMILAR-TO  CONTRASTS-WITH  IS-SPECIALIZATION-OF
IS-GENERALIZATION-OF  IS-EQUIVALENT-TO  REQUIRES-CONCEPT  CAUSES-
CONCEPT)))
               (dolist (slot slots)
                 (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "METHOD-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH SUPPORTS-VIEW )))
               (dolist (slot slots)
                 (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))


            ((equal whatidea "RHETORICAL-FIGURE-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH ASSOCIATED-WITH )))
               (dolist (slot slots)
                 (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "ARGUMENT-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH DEFINES-PROBLEM TACKLES-PROBLEM CONTRASTS-IDEA SUPPORTS-IDEA )))
               (dolist (slot slots)
                 (if (my-slot-values interpretation slot)
```

```
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "PROBLEM-AREA-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH HAS-SUB-AREA SUB-AREA-OF RELATED-TO-AREA )))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "PROBLEM-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH HAS-RESOLUTIVE-METHOD HAS-EQUIVALENT-MEANING-AS DERIVES-FROM-
PROBLEM RELATED-TO-PROBLEM IS-TACKLED-BY-ARGUMENT DEFINED-BY-ARGUMENT
IS-TACKLED-BY-VIEW HAS-SUPPORTIVE-VIEW ATTACKS-VIEW LINKED-TO-FACT
EXISTS-IN-AREA HAS-PROBLEM-TYPE)))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "DISTINCTION-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH RELATED-TO-PROBLEM )))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "SCHOOL-OF-THOUGHT-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH HAS-MAIN-EXPONENT HAS-EXEMPLAR-THEORY CLASSIFIES-VIEW HAS-MAIN-
THESIS HAS-EXEMPLAR-DOCUMENT HAS-OPPOSING-ARGUMENT HAS-SUPPORTING-
ARGUMENT OPPOSES-VIEW SUPPORTS-VIEW INFLUENCED-BY-VIEW INFLUENCES-
VIEW ATTACKED-BY-PROBLEM TACKLES-PROBLEM TYPIFIES INTERPRETS-FACT
USES-IDEA DEFINES-CONCEPT )))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "PHILOSOPHICAL-SYSTEM-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH HAS-THESIS HAS-THEORY DEFINES-METHOD PART-OF-SCHOOL HAS-EXEMPLAR-
DOCUMENT HAS-OPPOSING-ARGUMENT HAS-SUPPORTING-ARGUMENT OPPOSES-VIEW
SUPPORTS-VIEW INFLUENCED-BY-VIEW INFLUENCES-VIEW ATTACKED-BY-PROBLEM
TACKLES-PROBLEM TYPIFIES INTERPRETS-FACT USES-IDEA DEFINES-CONCEPT)))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "THEORY-INTERPRETATION")
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH HAS-THESIS DEFINES-METHOD PART-OF-SYSTEM PART-OF-SCHOOL PART-OF-
THEORY HAS-EXEMPLAR-DOCUMENT HAS-OPPOSING-ARGUMENT HAS-SUPPORTING-
ARGUMENT OPPOSES-VIEW SUPPORTS-VIEW INFLUENCED-BY-VIEW INFLUENCES-
VIEW ATTACKED-BY-PROBLEM TACKLES-PROBLEM TYPIFIES INTERPRETS-FACT
USES-IDEA DEFINES-CONCEPT )))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                        (push (list slot (my-slot-values interpretation
slot)) result)))))

            ((equal whatidea "THESIS-INTERPRETATION")
```

```
               (let ((slots '(IS-RELATED-TO-IDEA SIMILAR-TO CONTRASTS-
WITH PART-OF-SYSTEM PART-OF-SCHOOL PART-OF-THEORY PART-OF-THESISHAS-
EXEMPLAR-DOCUMENT HAS-OPPOSING-ARGUMENT HAS-SUPPORTING-ARGUMENT
OPPOSES-VIEW SUPPORTS-VIEW INFLUENCED-BY-VIEW    INFLUENCES-VIEW
ATTACKED-BY-PROBLEM TACKLES-PROBLEM TYPIFIES INTERPRETS-FACT USES-IDEA
DEFINES-CONCEPT  )))
              (dolist (slot slots)
                (if (my-slot-values interpretation slot)
                       (push (list slot (my-slot-values interpretation
slot)) result))))))

            result)))




;; from an IDEA
(defun  find-idea-interpretation-content  (idea  &optional  (author
'michele-pasin))
"Reifies find-interpretation-content in a function which takes an idea
as input"
(let ((interpretations (find-interpretation-of-idea idea author))
      (*package* (find-package "OCML")) ;; misterious HACK
      (result nil))
  (dolist (interpretation interpretations)
    (push (find-interpretation-content interpretation) result))
  result))



;;+++
;; functions for philosophy-tree related information


(defun next-advisor-up (person &optional (list nil))
"Takes  the  instance-name  of  a  person,  and  outputs  a  list  of  the
advisors tree"
  (if person
          (let ((advisor (first (setofall '?x `(and (person ,person)
(learning-at-institution ?l) (degree-of-study ?l PHD) (has-learner ?
l ,person) (has-teacher ?l ?x))))))
        (push advisor list)
        (next-advisor-up advisor list))
    (reverse (rest list)))) ;; rest for eliminating the final nil



(defun all-phd-students (person)
"Takes  the  instance-name  of  a  person,  and  outputs  a  list  of  its
students"
  (if person
       (let ((students (setofall '?x `(and (person ,person) (learning-
at-institution ?l) (degree-of-study ?l PHD) (has-teacher ?l ,person)
(has-learner ?l ?x)))))
        students)))
```

```
;; +++++
;; functions for creating instances of  tractatus interpretations
;; ++++


;; e.g. also (interpret-tract-sentences 1 1 '(value1))
(defun interpret-tract-sentences (bottom top values)
"Useful wrapper: gets two numbers and a list of values to be used for
interpreting the interval of sentences'numbers - specific for the
tractatus"
(interpret-sentences-as
  (get-sentences-from-numbers (extract-interval (get-tractatus-
paragraphs-numbers) bottom top))
 values))


;; this can be used for non-continuous sentences
;; ---> (interpret-sentences-as '(1 2) '(fact))
(defun interpret-sentences-as (sentences values &optional (author
'michele-pasin))
"Accepts a list of sentences, and a list of values, and outputs a
string with the instantions of every sentence interpreted with ALL the
values"
  (let ((result ""))
    (dolist (sentence sentences)
              (setq result (format nil "~A~2%(def-instance int-~A-~A
expression-interpretation ~%((carried-out-by ~a)~% (interprets
sentence-~A)~% (has-interpretation  ~A)))"
                           result
                           sentence
                           (gensym)
                           author
                           sentence
                           (let ((result ""))
                             (dolist (value values)
                                (setq result (format nil "~a ~a" result
value)))
                             result))))
    result))


;; howto:   (extract-interval (get-tractatus-paragraphs-numbers) 1 2)
(defun extract-interval (orderedlist bottom up)
"Given an ordered list and two limits, outputs a new list with all the
elements within those limits"
  (let ((newlist nil))
    (dolist (el orderedlist)
      (if (and (>= el bottom)
               (<= el up))
          (push el newlist)))
    (reverse newlist)))
```