Introduction
Modern software development consists of typical knowledge intensive tasks, in the sense that it requires that software developers create and share new knowledge during their daily work. In this paper we propose KnowBench a knowledge management system integrated inside Eclipse IDE that supports developers during the software development process to produce better quality software. The goal of KnowBench is to support the whole knowledge management process when developers design and implement software by supporting identification, acquisition, development, distribution, preservation, and use of knowledge - the building blocks of a knowledge management system.

KnowBench
KnowBench provides functionality that can be used for articulating and visualizing formal descriptions of software development related knowledge in a flexible and lightweight manner. This knowledge is then retrieved and used in a productive manner by a semantic search engine and a P2P metadata infrastructure - namely GridVine. Thus, the collaboration of dispersed software developers is achieved who can benefit from each others' knowledge about specific problems or the way to use specific source code while developing software systems.

Components
Semantic Search
KnowBench supports advanced methods for knowledge search through a semantic search engine by taking into account three different types of search, namely keyword, structured and semantic search.

Global Metadata Store
The global metadata store consists of two components (APIs) - the LocalMDS and P2PMDS. It provides an abstract layer for handling these and its purpose is to manage knowledge stored either locally or in the P2P network.

Software Development Semantic Wiki (DevWiki)
KnowBench utilizes the DevWiki system in order to assist software developers in the articulation and navigation of software development related knowledge. Browsing through knowledge is done like surfing through a conventional wiki using the semantic links between different knowledge artefacts.

An important aspect of the KnowBench is the ability to annotate semantically source code. The software developer is able to annotate source code manually with semantic annotation tags that are available or define new tags and extend the used annotation ontology. The shallow natural language processing (NLP) and information extraction (IE) API are used to semi-automatically annotate source code corpora.

Knowledge Management Support with KnowBench
We have conducted detailed evaluation of KnowBench in small groups of software developers in the following organizations: (1) Intrasoft International S.A. - 4 developers, (2) Linux Industrial Association - 4 developers, (3) TXT e-Solutions - 4 developers and (4) Thales Research & Technology - 4 developers. We grouped the evaluation results according to the KM building blocks.

Knowledge Acquisition
According to the respondents, KnowBench achieves a good score (73% are positive) for its support in acquiring existing knowledge. Regarding the supported types of knowledge sources, 85% of the respondents were satisfied with the support; 23% found that additional types of knowledge sources relevant for coding should be supported. As far as the system’s response time, although 64% of the respondents found it quite fast - some further optimization of the system would be useful.

Knowledge Development
All respondents found the knowledge development support in KnowBench clear and easy to follow and agree that the system provides support at an adequate level (76%). The meaning of knowledge items is understandable for 86% of the respondents. Regarding annotations, their meaning and purpose were clear for 86% of the respondents. 79% of the respondents found the granularity level of source code that can be annotated sufficient. 92% of the respondents found that KnowBench provides friendly and easy-to-use forms for creating annotations and only 36% of them considered the manual annotation as effortful activity. On the other hand, in semi-automatic annotation, 42% of proposed annotations were chosen, thus 82% of the respondents were satisfied with the suggestions.

Knowledge Sharing
Knowledge sharing in KnowBench meets the expectation of 69% of respondents. Even though all aspects of knowledge sharing in KnowBench are above the threshold, only the level of details to be specified in order to share knowledge and the usefulness of shared information received high marks (greater than 90%).

Knowledge Usage
The search functionality received an average score (57%). The respondents were satisfied with the quantity and quality of search results. As far as quality of search results is concerned, 91% of the respondents found the number of results optimal. For 73% of the respondents the list of result did not contain any irrelevant result. As regards the quality of search results, 62% of the respondents confirmed that the search results satisfy their information needs more than average.

Knowledge Preservation
The lifecycle of the knowledge items i.e. creation, update, deletion in KnowBench seems to be supported well (expectation of 67% of the respondents). Modification of knowledge is not a time consuming function for 75% of the respondents and can be done very easily by 73% of the developers.

Conclusions
The KnowBench system is an intelligent, semantic user interface environment for software developers which is integrated in the Eclipse IDE. Semantic web technologies provide the driving force to better manage knowledge in software development activities inside KnowBench. KnowBench offers an easy to use environment to facilitate knowledge articulation and visualization pertinent to software development. Additionally, it provides means to annotate manually or semi-automatically this kind of knowledge in order to foster easier knowledge acquisition and sharing by exploiting a semantic search engine and a P2P metadata infrastructure. Thus, better and more flexible collaboration among software developers scattered across the globe is facilitated.

Acknowledgements
This work was partly supported by the TEAM project, which is funded by the EU-IST program under grant FP6-35111.