The success of persistence technology transfer projects critically depends not only on the choice of the best technology available but more importantly also on successful business tactics and strategies. Therefore, PASTEL established a subgroup which exchanged experience and know-how related to these non-technical issues, tried to identify pitfalls and business opportunities, and asked for feedback on this problem from established commercial players outside of the working group.

Questions addressed in this forum include the following:

- Is it impossible to sell database languages, should one focus on tools and services only?
- How can persistence be delivered in open systems?
- How can persistence be added to existing products and tools? (As an add-onto a virtual machine?, as a persistence framework?, as a preprocessor? ...)
- What are the means to sustain innovative strength in industrial development teams building persistent systems (links to academia, training, internal organizational structures, ...)?
- What is the impact of recent market trends (e.g., Java, Internet, Windows-NT, XML, ...) on persistence product development?

The persistence abstraction allows the creation and manipulation of data in a manner that is independent of its lifetime thereby integrating the database view of information with the programming language view. This yields a number of advantages in terms of orthogonal design
and programmer productivity which are beneficial for application systems [AM95]. Three alternative approaches to providing persistence have been pursued by PASTEL group members:

**Approach 1:** Design and implement a new language and object store (the Tycoon-2 solution [GMSS97]);

**Approach 2:** Add an object store and some persistence facilities to an existing language (the PJama solution [JoAt98]; or

**Approach 3:** Add a programming language and some persistence facilities to an existing database system (the O2 solution [Deu91,Ard98]).

In the early days of persistence research, it was hypothesized "that it should be possible to add persistence to an existing language with minimal change to the language" [ABC+83]. Taken literally, this statement may be true: there are languages to which persistence can be added without much disturbance. However, universally speaking it turned out that it requires a specific discipline for a languages to be augmented by (orthogonal) persistence.

Thus the preferred early technique (in academia) for providing persistence was to design and implement a new language with persistence built in. Since then many persistent languages have been developed in this manner including Amber [Car86], Fibonacci [ABG+93], Galileo [ACO85], Napier88 [MBC+94], PS-algol [ACC81], and Tycoon [MS92]. All of these must be considered research systems.

In 1996, researchers of the FIDE-2 consortium from Hamburg, Germany, founded a startup company which makes heavy use of persistent object technology in their products and services following **Approach 1** outlined above. As part of the Pastel work plan, this team describes in [GMSS97], how this particular form of persistent object system technology developed in is being used by a small German startup software company to realize innovative customer-oriented information services on the Internet. Their paper clearly outlines the benefits (expressiveness, scalability, platform-independence, independence of third-party tools and technologies) but also the limitations (need for continuous investment in base technologies, lack of skilled human resources, need to convince customers to buy into non-mainstream technology) of this approach to achieve fully orthogonal persistence abstraction for data, code and threads across multiple platforms with open interfaces to other languages.

[GMSS97] also gives answers to the question how to establish stable long-term, bi-directional links between academia and small and medium enterprises with an academic background. In particular, based on experience gained by Higher-Order using Tycoon-2 in industrial projects, new mechanisms for method-level and instance-level evolution of persistent objects are currently explored in research projects at the Technical University of Hamburg. As another consequence of this cooperation, both teams (from academia and industry) expanded the focus of their work to also include specific application areas (generic systems) where persistence technology yields extraordinary productivity gains (multi-media document management, cooperative work of multiple users on shared artifacts).

In commercial environments, the second and third approaches are most common, due to the costs involved and the need to interface with existing systems. For example, orthogonal persistence for Java [AJD+96, JoAt98] follows **Approach 2** and takes the object-oriented language Java and
adds persistence facilities in the virtual machine. As reported in [JoAt98], the PJama System
developed in cooperation with former FIDE-2 researchers at Sun Microsystems Laboratories,
has come very close to achieving the goals of orthogonal persistence (type orthogonality,
transitive persistence and persistence independence). However, the remaining loopholes
(persistent threads, persistent exceptions) and the necessity to handle persistent and volatile
state outside of the persistent store are a source for incompatibilities between "off the shelf"
commercial libraries and the persistent virtual machine which (still) hinder the adoption of this
otherwise very attractive form of persistence in industrial projects.

In the context of other Pastel subgroups, work has been reported on extending the O2 object-
oriented database system by additional services, e.g. to facilitate a seamless integration of Web
and database services. The team of ETH has investigated another, less database-system-
dependent line of research also following **Approach 3**: In [RöBö98] they investigate how to fit in
persistence services (in particular for bulk data access) into the middleware framework defined
by CORBA (of OMG). In their report, they pay much attention to market aspects (horizontal
market, vertical market) when evaluating the pros and cons of different persistence strategies.
They conclude that by adopting database techniques (dataflow execution, bulk data transfer and
intra-query parallelism) in the construction of wrapper objects around heterogeneous distributed
data sources (files, DBMS, SAP etc.), the performance overhead associated with the CORBA
middleware can be reduced significantly without putting additional burden on the application
designer and developer.

To summarize, each of the three approaches outlined above has its own virtues and limitations

**Approach 1:** has the highest potential of achieving the ultimate goal of persistence technology,
namely to dramatically increase programmer performance in the process of building very large
persistent and distributed enterprise solutions. However, the necessary investment in tools
and technologies makes this approach viable only for global players in the IT business (e.g..
SAP, Oracle or Baan for their next-generation of ERP software, Software AG in their Java-
based Bolero system,) who can run the risk of using proprietary languages and tools.

**Approach 2:** fits well with today's interpreted languages like Java and Visual Basic. However,
only if the vendors of these languages (like Microsoft, Sun Microsystems or Symantec)
actually create base libraries and frameworks that interact well with persistent virtual
machines and stores, there will be sufficient incentive for other library, framework and
application designers to actually use persistent versions of these languages.

**Approach 3:** has been adopted by several object-relational database systems. However, even
the best of breed solutions (e.g., Oracle 8) fall short of closing the gap between expressive
object-oriented programming concepts and high-performance database functionality.

All three approaches have a common interest in ensuring that data, programs and meta-data can
evolve as the applications systems, and the uses to which they are put, evolve.

In order to support evolution of persistent applications a language or database environment must
provide a number of basic facilities either within the language/database itself or within its run-
time support system. In [KMM97] members of Pastel selectively draw on the PS-
algol/Napier88 experience and show how, by providing these basic facilities, the evolution of
persistent application systems may be programmed within the system. The essence of the paper
is to demonstrate how these basic facilities, taken from the research environment, are provided in two representative commercial systems, namely Java with a persistent store and O2C, thereby yielding the same attendant benefits.

**Attached Reports from the Reporting Period**


**References**


