Technical Report No: 2001/03

Educational Java Beans

Lucia Rapanotti
Jon G. Hall

2001

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

Educational Java Beans

Lucia Rapanotti and Jon Hall
Computing Department
The Open University
UK
{L.Rapanotti , J.G.Hall}@open.ac.uk

Abstract

Educational Java Beans(tm) is a proposed extension to Enterprise Java Beans(tm). Its goals are:

- to leverage the technology of Enterprise Java Beans to help the Educational Engineer;
- to provide a uniform and scalable software platform upon which the development of distance learning and teaching system development can take place;
- to provide support for the business processes of distance learning and teaching.

In this paper we do four things. We describe relevant high-level business processes involved in distance learning and teaching, so to identify requirements for electronic support. We highlight the challenges of the current distance learning and teaching business model and explore meeting them through electronic support. We propose a new blueprint architecture - the EdJB architecture - that discharges these requirements and meet the challenges. Finally, we discuss the wider applicability of the new blueprint architecture in an enterprise setting.

1 Introduction

With increasing demands being made for the electronic support of distance learning and teaching, there are many reasons why an architecture for *electronic Learning and Teaching* (eLT) is important and timely:

- initiatives such as the EU Ariadne project (http://ariadne.unil.ch) or the UK e-University (http://www.hefce.ac.uk/pubs/defalt.asp/) introduce new challenges for the Educational Engineer to include infrastructural electronic support for the business of learning and teaching at a distance;
- platform independent technologies are available which already go some of the way towards supporting that business which, for the first time, will broaden the market for the product.

In fact, the educational software development has never been so central to the success of a university; nor - since the early days of the Open University - has a new university been so central to the success of the UK in world markets.

The UK education market is not without good electronic support; the Open University being only one of many institutions making use of information technologies in their teaching. Current development practice can, however, only be described as *ad hoc*. One of the reasons for this, we feel, is the lack of a proper foundation for the building of educational software systems. We have not yet reached a 'technological critical mass' for educational software development.

Witness the recent explosion of electronic commerce products. Clearly, we are in a boom time for their development and deployment. We trace this explosion to the platform independence of Sun's Java technologies and, specifically, the Java 2 Enterprise Edition (J2EE) platform [6], which includes the Enterprise Java Beans (EJB) architecture[1]. In essence, J2EE moves the technology to critical mass: the technologies existed before J2EE, the technologies were used before J2EE, but it took J2EE to provide the catalyst for their wholesale use. In this paper, we hope to go someway towards identifying the catalyst for education software development. We wish to build on the firm bases provided by EJB to extend its properties and philosophy to cover distance learning and teaching.

EdJB will open up new markets for education, from global distribution to a per customer delivery. As EJB gains credence as the standard platform for the development and deployment of eCommerce applications, EdJB-based systems will combine seamlessly with commercial applications.

1.1 eLT products, services, and business processes

There are strong similarities between many of the business processes of commerce operations and those within distance learning and teaching. In the most general case, both have customers; both work on a contract basis; both provide end-user support; both work through intermediaries. Moving into the electronic setting of eCommerce and eLT, we may deduce that EJB goes someway towards servicing the electronic support needs of eLT.

However, with few exceptions, eCommerce has concentrated on the exchange of physical goods for which post-point of sale electronic support is unnecessary - delivery and use happen in the material rather than the electronic world. In eLT, the commodity on offer is information employed by the end-user (almost) every day; in this case, continued close involvement in the product is integral with its delivery and use.

1.2 Overview of EdJB

EdJB fosters the reuse of electronic support for programmes and courses based on the development and/or acquisition of components that can be customized, assembled and deployed in a variety of settings. At the same time:

- as EJB, it provides a standard component architecture for building distance learning and teaching electronic support;
- as EJB, it makes the development of eLT support applications easier allowing a concentration on the business logic of learning and teaching, rather than the low-level mechanisms for security and trust, for instance;
- it inherits from EJB desirable characteristics of electronic support, such as platform independence, that make eLT available to the whole community;
- it relies upon tried and tested technologies and includes support for legacy systems so that there is no ‘ground zero’ for development.

[1] In the paper, we will often use EJB as a short for Enterprise Java Beans architecture, and EdJB as a short for Educational Java Beans architecture.

1.3 Note

The reader should note that we do not claim in this paper to develop any radical technologies; from our perspective such radicalism is not required. Rather, we combine and recombine existing technologies in a novel way.

1.4 Structure of the paper

In section 2, we discuss similarities between business processes in commerce and in distance learning and teaching, which motivate the choice of EJB as the basis for a blueprint architecture for eLT.

In section 3, we explore relevant high-level business processes involved in distance learning and teaching, and identify a set of requirements for electronic support. We also highlight the challenges of the current distance learning and teaching business model and explore meeting them through electronic support.

In section 4, we describe the EdJB architecture and discuss how it discharges those requirements and meet the challenges.

Finally, in section 5, we discuss the wider applicability of the new blueprint architecture in an enterprise setting.

2 eCommerce and eLT

As mentioned, the business processes of distance learning and teaching share many attributes with commerce. Those that have most in common are the *synchronous* interactions between student (as end-user) and university (as supplier). Consider the following, perhaps prototypical, transactions:

Scenario: Purchasing a book on the Internet

> Once connected to the eBookstore's web site, Jazz places an order for 'Elements of Java Style' by completing an html form and proceeding to the checkout to enter her payment details. This triggers a number of actions from the supplier: the acknowledgement of the order sent by email; the collection of payment from Jazz's agent. The book delivery is not electronic and involves a subcontract with a paper mail delivery service. When Jazz receives the item her transaction with the supplier is complete, i.e., consumption – reading – requires no further involvement of the eBookstore.

If Jazz were connecting with an eLT provider for a printed syllabus, for instance, the scenario would be similar.

Scenario: Purchasing access to electronic journals

> Jazz subscribes to access an electronic journal web site for a year. After gaining access to the repository, Jazz can download articles in pdf to her computer and print them (given that this is allowed in the terms of the subscription and licensing agreements). The supplier's responsibility extends beyond the point of sale (the subscription), to ensure that the repository is available to Jazz, and that it is kept up to date. However, as in the previous scenario, Jazz is free to consume the articles without electronic support, and again consumption is a private event in which the supplier takes no part or interest (having gained Jazz's acceptance of the license).

If Jazz wanted, instead, access to an eLT provider's course web site for the duration of an eLT course the scenario would, again, be similar. Such interactions are rapidly becoming standard practice for the delivery of education at a distance.

We argue that truly electronic support can be the basis for much richer end-user/supplier relationships, akin to (and extending) the following:

Scenario: Software applications with user profiling

> Jazz downloads Qualcomm's Eudora email package and decides that the *sponsored mode* looks inviting. To be part of an emerging trend in software licensing, Jazz is provided the Eudora executable free of charge by consenting to have a (GUI) advertising panel displayed. By clicking on an advert, Jazz is directed to the sponsoring advertiser's web site.
>
> Qualcomm takes full advantage of electronic support to maximise revenue in sponsored mode by profiling Jazz's usage of Eudora. Periodically, Eudora compiles an anonymized summary of Jazz's Eudora use. With Jazz's agreement (concomitant on Jazz being able to continue to use Eudora in sponsored mode) this summary is emailed to Qualcomm.

In this scenario, Jazz's consumption of the product is no longer a private event - it includes a dialog which is ongoing throughout the product use. If Jazz wanted, instead, to enrol on an eLT provider's course for one year, the dialog with the eLT provider would also be on going. From experience, we note that it would, in fact, be very much richer, in supporting the complete student experience from course material delivery to validation of learning.

Supporting (and extending) the richness of the educational dialog whilst building on previous best practice provides our motivational focus for the development of an EdJB blueprint architecture. In the next section we focus on eliciting requirements for the support of the distance learning educational dialog. We begin by considering its current manifestation within the Open University, the major provider of distance education in the UK. As this is not a paper on the distance learning model *per se*, for brevity we stop short of describing the minutiae of the model, concentrating on the relevant higher-level aspects of the model.

3 Supporting current practice in distance learning

The Open University (OU) teaches, at a distance, to over 150,000 students per year. The OU teaching model - also called *supported open distance learning* - has evolved over thirty years to its current efficient and effective form. Because their use is best understood, it bases teaching on text technologies, i.e., books, electronic documents, web pages, continuous assessment and exam papers, but there is some *ad hoc* use of supporting media - some well-known, such as television, but also audio cassettes and computer-based instruction.

The business processes underlying the OU teaching model:

- are well defined;
- are scaleable: the OU provides distance teaching services to over 150,000 students world-wide;
- are effective and efficient;
- provide a rich source of dialogs between stakeholder for requirements elicitation.

We identify the following stakeholder groups within the OU model: **students**, their **finance providers**, **associate lecturers** (ALs, also called **tutors**), **course authoring teams,** and **student** and **tutor recruitment teams**. (These correspond to groups within the OU.)

In this section, we consider from a high-level, the dialog first between student and student recruitment team, then between student and tutor, then between student and course authoring team, and finally between tutor and

university. Each dialog description is structured as follows: we describe the dialog as it exists and from this description determine minimum requirements for the EdJB architecture to be able to provide support. We then identify the 'challenges' that exist within the current model that we feel can be met by electronic support.

Reality check: So that this paper is grounded in reality, we must consider the quality of electronic support that is currently available, and sustainably available in the future. We will therefore not assume that electronic support is required as a 24/7 thing[2], only that there will be relatively short, non-contiguous periods with the end-user in electronic contact with the supplier. Moreover, neither will we assume that the supplier can determine *a priori* the time that the end-user spends on-line. The justification for this mode of operation comes down to the fact that it is the customer who determines product use - even when that product is electronically supported. This is unlikely to change.

The following section describes the dialogs that motivate the architecture.

3.1 The dialogs

3.1.1 Student/Student recruitment team interaction: acquiring knowledge about programmes and courses

Prospective students access the OU web site in order to gather information on courses and programmes offered by the university. Via the web site further particulars and prospectuses can be requested, which are delivered either on-line or via paper mail.

Derived requirements

The architecture must:

R1 provide support for the publication and delivery of the university's information on programmes and courses

3.1.2 Student/Finance provider/Student recruitment team interaction: committing to study

An OU student will register on a course (whether their intention is to study for a course of for a whole programme), interacting with their finance providers and the OU to establish a contract for study between themselves and the university. Currently, the exchange of contracts happens via post and requires a signature.

Derived requirements

The architecture must:

R2 provide support for the establishment of a contract between various parties

3.1.3 Student/Student recruitment team interaction: receiving materials

The primary delivery mode for learning materials is by paper mail at the start of the course. When necessary, a second major mailing will provide further material later on during the course. Minor mailings will then be used to inform students of errata in the course materials, should errata be brought to the course authoring teams attention during a course presentation. An important component of the course materials is the study calendar for

[2] I.e., 24 hours per day, 7 days per week.

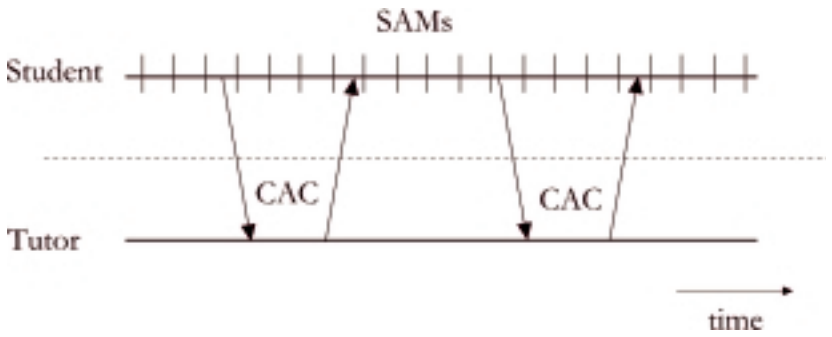

Figure 1: key events in the student/tutor dialog

the course. It indicates the components of the course that should be studied by when to allow the student to meet the deadlines for continuous assessment components (more about this later).

Derived requirements

The architecture must:

R3 provide support for the publication and delivery of teaching materials

C1 Challenge

It is important to note that, due to the present reliance on paper-like media, having the characteristic of being unchangeable when 'published', course texts remain unchanged, often for many years, even when problems with the material are known[3]. Even though every effort is made to inform students of errata, a student will often be unaware of the presence of errata, or become aware of them a long time after the error was encountered. This can have a serious and deleterious effect on the learning and teaching experience, which can contribute greatly to student (and so customer) dissatisfaction.

Providing a changeable (i.e., electronic) media base can address this challenge, as long as suitable materials management, such as versioning and configuration management technologies, are also available.

Derived requirements

The architecture should:

R4 provide support for the use of changeable teaching materials as the basis of the teaching provision

3.1.4 Student/Tutor interaction: studying

A typical student/tutor dialog, indicating key events, is shown in Figure 1. Study is deadline-driven by continuous assessment components (CACs). Tutor marking of CACs provides validation and feedback on a student's learning experience. For a half credit course (one academic year's part-time study) there will be between 6 and 8 CACs; tutor marking of assessment material - closing the feedback loop between student and university - can therefore typically occur only every 5 to 7 weeks.

[3] Another way of seeing these characteristics is that the OU is tied to the use of media that require long 'print' runs for them to be economic. Even certain electronic media, such as CD-ROMs, share such characteristics.

In addition, synchronous interactions are widely acknowledged by students and teachers alike as being very valuable. Until recently, the only means of achieving them was by telephone – with the obvious restriction on number of participants - or by personal contact - which requires travel to and, for some, an unwelcome commitment of time at some study centre. Currently, some electronic support is available, using conferencing tools such as Lyceum [4,5], and their use should be facilitated by the architecture.

Derived requirements

The architecture must:

R5 provide support for periodic external study validation

R6 provide support for tool support for synchronous interactions at a distance

C2 Challenge

Between CACs, *self*-assessment materials (SAMs in Figure 1) provide students with feedback on much shorter time scales – typically on a *per* subsection study basis (i.e., from minutes to hours) - answers being provided along with the teaching text. Although often enriching, such non-external validation mechanisms can also impact negatively the student experience: although success can breed success, failure on SAMs can lead to student demoralisation. Reasons for failure are twofold: the student has not understood the materials on which they are assessing themselves; there are errors in the SAMs (which unfortunately can happen). In the latter case, quality is severely impacted and this may lead to intense student dissatisfaction. In the former case, the student should return to study the materials and retry the SAM; in reality, however, what often happens[4] is just an acceptance that they have not understood. Given that external validation can be 5 to 7 weeks away, this can lead to students becoming seriously misguided before any corrective action can be taken.

Currently, there is no university-wide mechanism for tutor-validation (or otherwise) of activities between CACs. What is needed is a dialog between student and tutor that has a much finer granularity than the current 5-7 weeks; but which overloads neither the student nor the tutor.

Derived requirements

R7 provide support for periodic external study validation at a fine granularity (i.e., *per* SAM)

3.1.5 Student/Course Authoring Team interaction: examination

Most courses provide students with *per* course validation in the form of an exam. As well as testing students' understanding, the OU uses the exam as a means of verifying a student's identity, as often this is the only 'physical contact' a student has with the university.

Exam papers are distributed to study centres and completed scripts are returned to the OU headquarters for marking and other processing. Paper mail and courier deliveries are used for the transfer of exam scripts.

[4] Students are often under great time pressure to complete their studies; it is not unknown for students to be juggling a career with a family with an OU course, and trying to have a social life as well!

Derived requirements

The architecture must:

R8 provide support for electronic exam submission with validation of student's identity

R9 provide support for electronic processing of exam papers

3.1.6 Tutor /Tutor recruitment team interaction: appointment

A tutor's competencies are assessed through formal interviews on a per-course basis. The exchange of application form and contract is based on paper. On appointment, tutors receive copies of OU guidelines and other relevant regulations, which they may need during the discharge of their duties. This material is also paper based.

Derived requirements

R1, **R2** and **R3** cover these requirements.

3.1.7 Tutor/Student interaction: supporting a student's learning

Tutors are the embodiment of the university during course presentation, i.e., they perform the roles implied by the lower half of Figure 1. To do this, they provide support for learning and pastoral care to students.

Tutors mark their students' scripts. They are assisted in this task by 'tutor notes' issued by course authoring teams and including solutions and a marking scheme. Tutors receive the scripts from their students at particular deadlines set by the course study calendar, and return the marked scripts to the OU headquarters for processing – registering marks and having their marking monitored – from where they are returned to the students. Script submission, marking and processing are largely text-based.

Derived requirements

The architecture must:

R10 provide support for tutor/student dialogs

R11 provide support for receiving, marking, processing and returning students' scripts

R12 provide support for the monitoring tutor performance

R13 provide support for the delivery and updating of tutor notes

C3 Challenges

Tutors are a valuable resource for the university, not only in their role as course presenters, but in that they are customer facing. By being in contact with students possibly on a day to day basis, tutors are able to collect students' comments on course materials throughout the study period. Mechanisms exist for relaying this information back to the course authoring teams for materials improvement, but too often are such valuable perspectives lost. The availability of a fine grain interaction mechanism could facilitate their collection and use to update course materials on an on-going basis.

Derived requirements

R14 provide support for fine grain interactions between tutor and course authoring team

4 The architecture

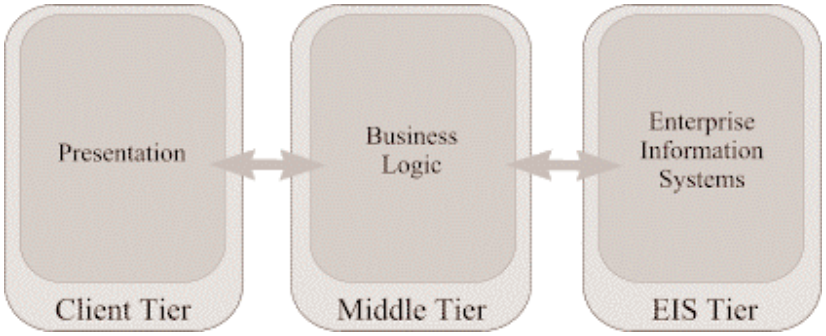

Figure 2: the 3-tier architecture

In this section we describe in broad terms the EdJB blueprint architecture. To do this we first identify those features of the EJB architecture upon which we will build.

4.1 The EJB architecture

The EJB architecture, described in [6], is an application of the 3-tier architecture. As such it separates *client view* from *business logic* from *persistent storage*; moreover, interactions between the tiers are heavily (but usefully) constrained. The 3-tier architecture is summarised in Figure 2.

One of the strengths of the EJB architecture is that it supplies **data persistence**, **security**, and **transaction management** - low-level technologies underlying many of today's eCommerce interactions. In doing this, eCommerce application development is greatly facilitated: the eCommerce developer can focus his or her efforts on capturing relevant business logic, delegating to the EJB architecture the provision of the low-level technologies. As illustrated in Figure 3, EJB containers package business logic together with services. Services offered by the EJB containers are invoked through APIs (Application Programming Interfaces) defined within the architecture. Note that EJB containers and Enterprise beans, by residing in the architecture's middle-tier, are intended to be used (and usually are) for the definition of server side logic. This is perfectly adequate for current eCommerce application where most of the computation is performed by the server, with (so-called) thin clients presenting, what are usually web based, user interfaces.

4.2 The EdJB Architecture

We have already noted that eCommerce and eLT share many requirements: in particular, eLT will also require data persistence, security and transaction management as basic services. We argue, however, that eLT interactions also require a new type of service that we call, for want of a better term, **cache coherence**. This requirement arises in the fine grain, but intermittent, nature of the educational dialog (established as requirement R7). Together with the reality check of Section 3, which states that we do not have a 24/7 on-line service, this implies the existence of local state information at both learner and educator sites, with needs for synchronisation between them at frequent intervals. In essence, the learner site needs to act as cache to the educator's site, with synchronisation to make them coherent.

As an example of the mechanism at work, consider the following likely scenario: having registered for study on a distance learning course, Jazz downloads the first teaching component from the eLT provider's web site. A

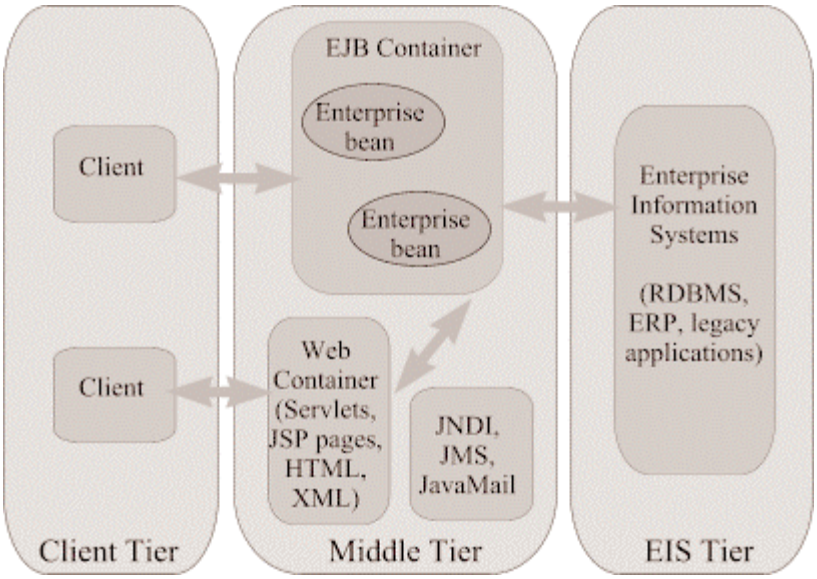

Figure 3: the EJB architecture

week passes in which Jazz works, off-line, through the first course component. During this period, Jazz works through 10 SAMs, 8 of which are answered correctly. In the mean time, two course material updates have been made by the eLT provider: one affects the first course component (which Jazz has), one the second (which Jazz does not yet have).

At the end of the first week, and having finished the first course component, Jazz logs on to the eLT web site to download the second course component. Via this link, information is passing both ways:

- Jazz makes a request for the second course component; an updated version is downloaded to Jazz's machine;
- without knowing the details of the transaction, but with permission, Jazz's SAM performance profile is uploaded to the eLT site;
- updates to Jazz's copy of the first course component are made to reflect the course updates, and Jazz is informed that this has occurred.

Over and above the downloading of the second course component, looking at this dialog in terms of caches, *cached information at both sites has to be made coherent*:

- from Jazz to eLT provider - to synchronise Jazz's SAM profile;
- from eLT provider to Jazz - to synchronise course materials.

This mechanism will, moreover, form a critical component of the dialog between student and provider that will be present within many, if not all, educational software development projects. Because of its omnipresence, an architectural solution cache coherence between sites should be thought of as a basic service of the architecture.

The EdJB architecture incorporates a cache coherence service, as is illustrated in

Figure 4. The client tier now contains local 'persistent' storage for 'caching' local information, as well as its own **EdJB container** that wraps *client* business logic. This EdJB container provides cache coherence services through communication with a corresponding EdJB container in the middle-tier.

With the EdJB architecture, when off-line, the learner's interaction with the teaching materials will be through

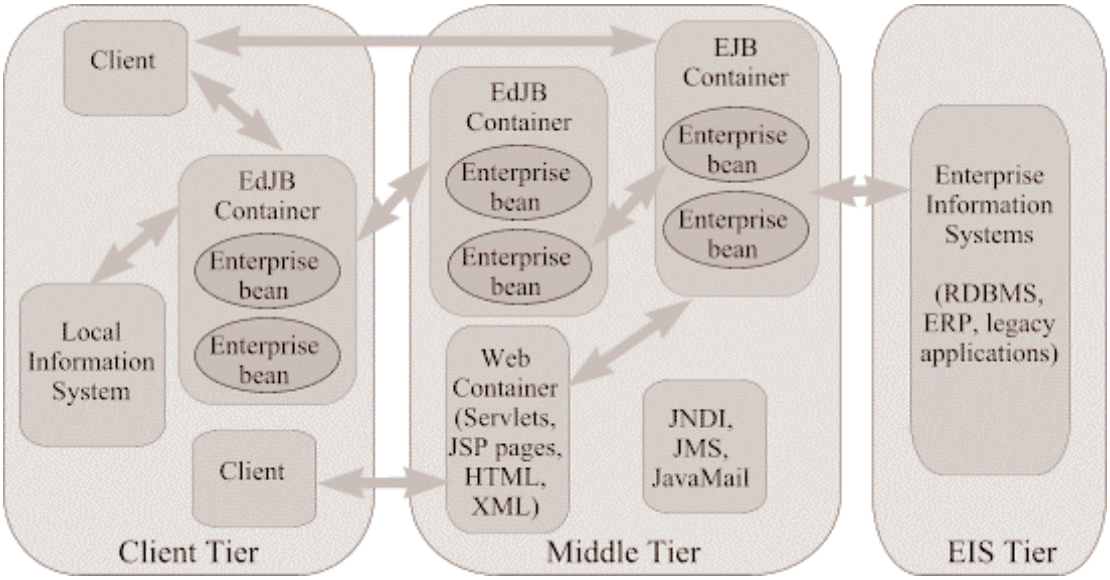

Figure 4: the EdJB architecture

the Local Information System with the client business logic used to support it (by collecting SAM answers, for instance, etc). In off-line mode, of course, the middle and EIS tiers are not accessible for interaction.

When on-line, the EdJB containers are joined so that client- and server-side data can be made coherent.

Depending on deployment choices, cache coherence can be made either transparent or opaque to the user (learner or educator). Also, for flexibility, and as is the case for EJB services, cache coherence services provided by the EdJB container can be entirely automated (i.e., container managed) or require bespoke implementation (i.e., provided by the developer).

4.3 Does EdJB satisfy the requirements?

Requirements **R1**-**R14** constrain any architecture for electronic support of distance learning. We have proposed a blueprint architecture - EdJB - that we claim can provide this electronic support. In this section we validate the blueprint architecture against the requirements. The table also provides a convenient opportunity for us to describe how we envisage the novel aspects of the architecture to be used by the developer.

The following table lists the requirement in order (requirement label in the first column), restates the requirement (for ease of reference), and argues the validation of EdJB against this requirement.

#	Requirement	Argumentation
R1	provide support for the publication and delivery of the university's information on programmes and courses	This is within the standard services provided by EJB for eCommerce transactions.
R2	provide support for the establishment of a contract	This is a standard eCommerce transaction. It is already supported by

	between various parties	EJB.
R3	provide support for the publication and delivery of teaching materials	This is within the standard services provided by EJB for eCommerce transactions.
R4	provide support for the use of changeable teaching materials as the basis of the teaching provision	This is supported by the cache coherence services provided by the EdJB architecture. EdJB does not require the user to be on-line all the time, but ensures that both local and server information systems are made consistent as often as allowed by the end-user. This provides the mechanism for supporting changeable materials: materials delivered to the student (and so residing locally) can be updated as often as the student connects to the server.
R5	provide support for periodic external study validation	This is within the standard services provided by EJB for eCommerce transactions.
R6	provide support for tool support for synchronous interactions at a distance	This is a standard eCommerce transaction. It is already supported by EJB.
R7	provide support for periodic external study validation at a fine granularity	This is supported by the cache coherence services provided by the EdJB architecture. EdJB does not require the user to be on-line all the time, while ensuring that both local and server EJB copies are consistent as often as allowed by the end-user. This provides the mechanism for supporting fine grain study validation: student 'profiles' can be generated client-side and kept coherent with server-side copies. By making these available to the tutor, again through the caching mechanism to the tutors local cache, validation of the student 'profile' can be performed and returned to the student's local cache.
R8	provide support for electronic exam submission with validation of student's identity	This is within the standard services provided by EJB for eCommerce transactions.
R9	provide support for electronic processing of exam papers	This is within the standard services provided by EJB for eCommerce transactions.
R10	provide support for tutor/student dialogs	This is within the standard services provided by EJB for eCommerce transactions.
R11	provide support for receiving, marking, processing and returning students' scripts	This is within the standard services provided by EJB for eCommerce transactions.

R12	provide support for the monitoring tutor performance	This is supported by the cache coherence services provided by the EdJB architecture that provides the mechanism for supporting tutor monitoring - tutor 'profiles' can be generated client-side and kept coherent with server-side copies. Verification of the tutor's performance can be performed and returned to the tutor's local cache.
R13	provide support for the delivery and updating of tutor notes	This is supported by the cache coherence services provided by the EdJB architecture. It is an instance of the challenge of using changeable materials (see **R4**).
R14	provide support for fine grain interactions between tutor and course authoring team	This is supported by the cache coherence services provided by the EdJB architecture. It is an instance of the challenge of providing fine-grain dialogs (see **R7**).

5 Discussion and Conclusions

In this paper, we have derived an extension of the EJB architecture that provides electronic support to the distance learning experience. We have characterized the architecture, EdJB, as *cache-coherent* EJB, in that each user has local storage (the *cache*) and supporting business logic (the coherence) to work with off-line. The architecture provides automatic cache coherence services whenever the user goes on-line.

We have argued that the architecture meets current high-level requirements in a distance learning setting. In this section, we will reflect briefly on a reinterpretation of the distance learning challenges that motivated cache coherence in the wider context of eCommerce.

5.1 Product maintenance and usage profiling

We have noted that the fact that the OU distance learning model is based in text-technologies has important ramifications for the management of teaching materials. With paper technologies, the incorporation of corrections and modifications require long and expensive text-publishing processes. From the OU's viewpoint, for cost-efficiency, material updates are only viable in *batch mode*, usually in between course presentations. With electronic information, updates can be performed easily and the updated materials made available to users in a *semi-continuous mode.*

Within an eCommerce setting, product maintenance is often also in the realm of the material world. However, there are instances where product behaviour profiles are feasible and useful. This is obviously true for software products (witness Qualcomm's use described above), but could also apply to a wide range of appliances with embedded electronic components – say, the controller of a modern motor car engine or even a washing machine. We can consider the profiling of product use and performance as instances of a dialog in the EdJB sense. Working within the EdJB architecture, the development of *local* product use and performance profiles with cache coherence allowing regular updating of server-side copies is facilitated, as is the downloading of software updates, the alteration of operational parameters, *etc.*, from the server-side to the local cache.

5.2 On-demand product and service delivery

With paper technology, the packaging and delivery of materials takes considerable time and resources, which, again, makes *batch mode* the only viable means of distribution. By comparison, in the electronic world, the cost of updating and repackaging materials is slight; their availability is dramatically increased. Under the EdJB architecture, the development of *staged* delivery mechanisms, that is, on a daily or weekly basis, or even *on-demand* delivery mechanisms, that is, driven by the user's needs is greatly, facilitated through cache coherence.

5.3 Requirements and architectures: concluding thoughts

Given that architectural support for the rapid development of bespoke business solutions provides true commercial advantage, the existence of architecturally based development mechanisms, methods and methodologies is an important prerequisite for allowing requirements to continue to drive development. In this paper we have shown how a collection of requirements can drive supporting software architecture development. We do not claim to have provided a method for doing so, nor, at this point, to be able to draw any methodological conclusions. However, we have explored some of the issues involved in deriving architectures from requirements. We do envisage that the relationship between requirements and architectures, that is, reinterpreting the language of architectures in a requirements setting, can be addressed methodologically, and this will provide the impetus for further work in this subject.

6 References

[1] Barroca, L., Hall, J.G., Hall, P.(eds), *Software Architectures – Advances and Applications*, Springer, 2000.

[2] Barroca, L., Hall, J.G., Hall, P., An introduction and history of software architectures, components and reuse. In Barroca, L., Hall, J.G., Hall, P.(eds), *Software Architectures – Advances and Applications*, Springer, 2000.

[3] Bass, L., Clements, P., Kazman, R., *Software Architecture in Practice*, Addison Wesley, 1998.

[4] Rapanotti, L., Hall, J.G., Lyceum: audio visual conferencing in distance education. *Proceedings of the 1st Annual Conference of the LTSN Centre for Information and Computer Sciences*, Edinburgh, August 2000.

[5] Rapanotti, L., Hall, J.G., Lyceum: the system and its architecture. *Proceedings of ED-ICT2000, International Conference on Information and Communication Technologies for Education,* Vienna, December 2000.

[6] Shannon, B., Hapner, M., Matena, V., Davidson, J., Pelegri-Llopart, E., Cable, L., *Java™ 2 Platform Enterprise Edition, Platform and Component Specification*, Addison-Wesley, 2000.

[7] Shaw, M., Garlan, D., *Software Architecture: Perspectives on an emerging discipline*, Prentice Hall, 1996.