AC 2011-2057: DESIGN ASPECTS OF A DATABASE FOR REMOTE LABORATORY MANAGEMENT

Rainer Bartz, Cologne University of Applied Sciences, Germany

Rainer Bartz studied electrical engineering at RWTH Aachen, Germany, and received his PhD degree for research on the application of pattern recognition mechanisms to problems in the automotive engineering domain. He worked in automotive industry for 13 years, focusing on control and data analysis tasks. In 1997 he became full professor at Cologne University of Applied Sciences. His main areas of interest are signals & systems, industrial communication, and computational intelligence. Rainer Bartz is actively involved in the ASAM e.V. (Association for Standardization of Automation and Measuring Systems), defining standards for automotive test data management. He is member of ASEE and of IEEE.

Daniel Cox, University of North Florida

Daniel Cox is from Gainesville Florida where he also graduated with his BSME with Honors degree and Master of Engineering degree from the University of Florida in 1979 and 1981, respectively. In 1981 he joined the IBM Corporation in Boulder Colorado where he worked as a Manufacturing Engineer. In 1986 he was awarded the prestigious IBM Resident Study Program Award to attend doctoral studies at the University of Texas at Austin. He graduated with his Ph.D. in Mechanical Engineering with specialization in robotics from UT Austin in 1992 where he also worked at the IBM Austin Texas facility as a Robotics and Automation Engineer until 1998. He joined the University of Texas at Austin in 1998 as a Research Scientist where he was the Associate Director and Program Manager of the Robotics Research Group. In 2001 he joined the faculty at the University of North Florida where he is now Professor of Mechanical Engineering. Since joining UNF he has initiated the Manufacturing Innovation Partnership program, sponsored by the National Science Foundation, to foster industry-academic collaboration. His research and teaching interests are in the areas of robotics and automation, advanced manufacturing, and dynamic systems and control engineering.

Design Aspects of a Database for Remote Laboratory Management

Abstract

This paper describes the design of a database which is used to manage the remote laboratory RLAB. RLAB allows users from all over the world to access a set of real world physical models, to perform experiments by interactively working with them in a realtime environment, and to download the resulting data to their own computer system for further processing. The only requirement for the user's computer is an internet browser.

RLAB was originally developed at Cologne University of Applied Sciences (CUAS) in Germany; it uses NI LabVIEW to perform the interfacing to the real world models as well as to interact with users and the database. The RLAB infrastructure has been ported to a mechanical laboratory site at University of North Florida (UNF), Jacksonville, during the past two years and is now used in its engineering curriculum, thereby offering access to some further real world models and gaining synergy from an international cooperation.

To properly run such remote laboratory several aspects need to be considered. New users need to be registered at the system. The availability of experiments needs to be managed. For interactive work with one of the experiments, time slots need to be reserved in advance, and the attempt to access the experiments must be verified against the reservations. The experiments' parameter settings as well as results data must be kept to be retrievable afterwards, etc.. This paper collects requirements from a set of use cases around RLAB. It then groups the requirements, proposes a data model, and describes the implementation using a relational database system. Furthermore some issues are discussed when extending the laboratory by further models and/or experiments, and the paper explains how this is taken care of by the RLAB approach. These considerations and results may help institutions to create an appropriate data management architecture when establishing some own remotely accessible experimentation sites.

1. Introduction

Remote laboratories have become common in many places. Universities have installed them to allow their own students to perform experiments from outside the lab and also to allow students from partner institutions to access them. The latter gives students the possibility to investigate systems which are not available at their home institution, and thereby broaden their experience and knowledge with access to more physical systems.

Remote laboratory architectures have the potential to become powerful facilities for industry, though this is not yet widespread. They may serve as reference installation in cases where the same processes are installed at several sites and are to be configured to show similar behavior. They may even be part of production processes, allowing active access to the process from remote sites (and not restricted to being remote monitoring systems).

Numerous journal and conference publications of the last decade and beyond demonstrate a wide variety of application domains, opened for remote operation (e.g. [1 - 7]). Most of them focus on technical aspects of the systems being accessed, on their behavior and the features available to remote monitoring and control, and on the teaching lessons and student experiences related to them. A few discuss general infrastructure aspects [8]. Some also cover parts of the

data management that is needed [9 - 12], though often concentrating on learning goals, methodologies, and outcome.

In contrary, this paper mainly addresses the information required to run the infrastructure of a remote laboratory. It presents details of the approach taken by RLAB [13] [14], the remote laboratory currently installed at CUAS and UNF, which from the beginning was designed for expandability. Fig. 1 shows as an example the twin-rotor system, one of the first physical systems installed with RLAB at CUAS.

The initially developed data management approach of RLAB was put to the test when work started to set up a second RLAB site at UNF with different models and different experiments [15 - 17]. However it turned out that nearly no modifications were required, and RLAB is successfully operated at two sites for some time now.

Fig. 1: The RLAB twin-rotor system

In this paper the term 'remote laboratory' shall mean any technical system with the following characteristics:

- during its normal operation, the physical system (or plant) or parts of it are controlled via actors and observed via sensors,
- the overall system is technically split into a 'core system' on one side (including all sensors and actors), and a number of 'remote systems' on the other side (which use the sensors and actors to observe and control the core system),
- the core system resides at a 'local site' while the remote systems reside at 'remote sites', and those sites are locally separated so that a typical user at a remote site would not be able to physically enter the local site and vice versa, and
- the primary control of the core system (determining its main outcome) is performed from the remote site, though some secondary control mechanisms (like safety functions, fast control loops, maintenance measures, etc.) will typically be run at the local site.

These items implicitly also define the terms 'core system', 'remote system', 'local site', and 'remote site' used throughout this paper.

This paper discusses the information related to such laboratory with specific focus on aspects that are introduced by the separation between core system and remote system. Section 2 collects a set of common use cases. Section 3 presents the RLAB approach to combine relevant information into entities. Section 4 adds the relations between the entities and presents the complete data model. The consequences of expanding a remote laboratory are discussed in section 5, and section 6 gives a short conclusion.

2. Common Use Cases for Remote Laboratories

When starting to develop RLAB, some initial considerations were combined using information gathered from partner sites and publications to build the basis for the collection of use cases. These use case are presented below.

a) Registering a user with a remote laboratory

Remote laboratories usually do not allow access by everybody; anonymous users or guest accounts are not likely to be incorporated. Instead, essential information for a user is required, eventually accompanied by some formal approval process.

Reasons for such restriction are:

- a user should be able to return to some previously conducted experiments or to a currently running experiment even after the remote system had been restarted, or
- a user should be able to manage some user-specific settings stored at the local site, or
- the core system may carry confidential information which should only be disclosed to selected people, or
- it can only bear a limited number of users while it is expected that a larger number will try to use it (this could be due to extended duration of experiments or a large group of potential users), or
- it may potentially be misused and any damage should be tracked back to the user.

Thus a registration process must be incorporated, and only registered users are allowed to work with the remote laboratory. This is combined with exchanging some initial authentication information.

Registration processes may be implemented either manually, fully-automated, and semi-automated, with the latter allowing a manual approval of an otherwise fully-automatic process.

b) Login to a remote laboratory

When a user intends to work with the remote laboratory, they are required to login to the core system. Quite simple systems without user registration and with some straightforward "start → return results" procedures often omit such login and shall not be further considered in this paper. All subsequent use cases thus require the user being logged in to the system.

c) Managing time slots to access restricted resources

Whenever the experiments performed at the remote laboratory require to use restricted resources (like real world physical models), some limitation on the number of users using these resources simultaneously must be obeyed. This can be dealt with by either an offline or an online architecture.

In an offline architecture a user configures an experiment and passes it to a job-manager at the local site. Such job-manager will initiate the experiments according to their timestamp of request with possibly some priority information being additionally taken into account. Users may retrieve the experiment results later (by either an automatically generated notice or at a subsequent login), and no further measures are required to coordinate the access to the same resources by different users.

In an online architecture a user interactively configures an experiment, starts it, and retrieves its results. As users prefer not to encounter long standby times, some access time reservation procedure is very useful. This gives the users some guarantee that they may use the equipment at their reserved times, but this requires the system to keep track of the reservations of users and to distinguish between available and reserved periods of time. Though a very flexible time management could be thought of, usually some fixed length basic periods are sufficient and users may reserve a specific resource for one or more of such periods (called 'time slots' in this paper).

d) Selecting laboratory models and experiments

A remote laboratory may provide a large number of (identical or even different) physical systems (called 'models'). Each of those models are independent resources and may be used simultaneously by different users. However, one such model can only be controlled by at most one user at a time.

Each model allows users to perform a set of different experiments, one at a time, though the same (or similar) experiments may be run on different models at the same time.

When a user accesses the remote laboratory to create some results, he or she must select both a model and an appropriate experiment.

e) Configuring and running an experiment

Most experiments allow a user to set configuration parameters. They may be (among others)

- the duration of the experiment,
- some fixed setpoint values for actors,
- parameters of sensor and data acquisition components (like sample time, filtering, ...),
- controller parameters for closed-loop systems, etc.

The user must be able to provide values for such parameters, and may want to explicitly start the experiment interactively or at a given point in time.

f) Obtaining experiment results

To create experiment results is the major reason for using a (local or remote) laboratory. In addition to the resulting sensor and post-processed values/traces also useful are the settings of the equipment used to interpret results correctly. In order for a user to obtain them, two architectures may be distinguished: push versus pull mechanisms.

A push mechanism uses a (pre-defined) communication channel to send the results to the user after the experiment is completed, without the need for any further user action. This may be useful for short experiments providing immediate results that can be returned to the user with the subsequent web page update. On the other hand this mechanism is useful if experiments are running for a long time or where the completion time cannot be determined in advance.

A pull mechanism just deposits the results at the core system, indicating their availability to the user and providing some interactive means to view/analyze them or (even better) to transfer them to the remote system.

Combining both mechanisms can be useful, leaving mass data initially at the core system.

g) Changing user information

Whenever a registration process is included in a remote laboratory, users must be given the opportunity to change most or all of the information entered during registration. Otherwise they might unnecessarily register multiple times.

Especially, information like the user password, email address, etc. are candidates for more or less frequent changes over a remote laboratory lifetime.

h) Managing configurations and result data

It is beneficial to keep the configuration parameters of an experiment at the core system even after the experiment is completed. This should be distinguished from the results documentation purpose (which is also very important and mentioned above). Retaining the configurations of previous experiment runs allows a user to re-run the same experiment without entering any parameter values once again. This is especially useful if experiments require a large number of

parameters, and in cases where a user typically will run a series of experiments with all but one parameter remaining unchanged.

Also, result data need to be managed by the core system. This is required in case results are provided by a pull mechanism and the user cannot copy them to his remote system or postprocess them remotely at his site. Another reason for keeping results at the core system is given, if separate users may want to access the same results subsequently, independently from each other.

In such cases results would pile up over time, and some means to delete them after some time or to move them to long-term storage (and get them back from there) must be available.

i) Administering a remote laboratory

A remote laboratory requires administration. Not unlike using a remote laboratory, administering it can also be performed remotely, though with a completely different set of tasks. Among the tasks are:

- to individually approve/disapprove the registration of a new user,
- to disable existing users, with reasons being e.g. discontinued enrollment at a partner institution, misuse of models, full achievement of remote laboratory lessons/assignments, ...; access to previous results can still be allowed or be restricted,
- to provide some information to users (either on login or via other communication channels),
- to distinguish between regular users and developmental users (the latter working on improving the remote laboratory and testing features not yet available to all users),
- to close a series of time slots for a model in case maintenance is needed; this may be on a regular basis or spontaneously,
- to disable an experiment for regular users if its internal processes are to be re-designed; in this case previous results should still be available, though no new experiment runs can be performed,
- to disable a model for regular users if it is going to be physically modified or replaced; again, previous results should still be available, though no new experiments can be performed on that model,
- to analyze the usage of the remote laboratory, a specific model, or a specific experiment,
- to analyze the usage of the remote laboratory by specific users or user groups,

Most of these administration tasks are not further discussed in the remainder of this paper but are included for completeness in database design considerations.

Before completing the use case analysis, some specific classes of remote laboratories shall be mentioned:

(i) laboratories with a dedicated communication line between the local site and each remote site: Such laboratories may not need a specific central registration process, as access to the core system can already be managed by access restrictions to location (room) or accessing system (computer) at the remote site. Also, depending on the communication configuration, a login procedure may be obsolete. In those cases part of the data model described below may be omitted.

(ii) laboratories with mere simulation functionality, where no real physical system is involved: Such laboratories neither need a registration nor a login procedure even if they are accessed with pure internet mechanisms. Also time slot reservation and user management may be omitted. This is especially the case if each experiment initiates its own thread or task and returns the results immediately, and the core system has enough resources to serve the expected number of simultaneously active users.

3. Entities and their Attributes

During the design and implementation of RLAB some of the design decisions made are listed:

- Registration and consequently user login is required. This is due to the fact that RLAB is designed to only require an internet browser for remote laboratory access. Such fairly open access together with its restricted resources (comprising of a set of physical systems that can only be used by one user at a time) call for system-inherent access control.
- RLAB is designed as an online architecture. This allows users to really interact with the models if experiments provide for it as they often do. Also, such architecture incorporates some realtime video streaming to observe experiments as they proceed. Although video streaming is near realtime subject to internet latencies, data collection is recorded at the local site in realtime with numerical results and plots promptly available for transmission at the end of an experiment run.
- The allocation of time slots for users working with a specific model, as needed in online architectures, is implemented, with a time slot length of 30 minutes. This was a compromise between some required minimum period to run a typical experiment and retrieve its results versus the amount of administrative overhead to reserve, manage, and store a multitude of short (or even non-constant) time slots.
- RLAB started with two different physical systems and a set of different experiments for each of it. However no restrictions are imposed by the database design on extending it to a much larger set of models.
- Results are offered to the users via a pull mechanism. This keeps result data primarily at the core system where they are at the user's disposal, although some results management is required.

Based on these decisions, the following entities can be identified, each of them with a set of attributes as described in the following paragraphs.

Three basic static entities are used to describe the mainly static aspects of RLAB: it knows about its users, its models, and its experiments as discussed below. Additionally there are instances of dynamic entities as discussed further below.

a) User Entity

This entity specifies the characteristics of a user. Each user instance will be referenced within RLAB by its unique 'UserID'. An email address, a user login name and a password are required to manage RLAB access restrictions over the internet. During the registration process further information is collected from the user, with first and last name being usually required while the salutation, the organization the user belongs to, and the purpose of his registration are RLAB specific. They allow to address the user properly, to identify the partner institution the user belongs to, and eventually to find out whether the user is using RLAB in a regular coursework, in some research project, or with some other purpose. The timestamp of registration is kept, and some further information on the user can be stored at this entity without specific classification. Finally a user can be given administrator rights by RLAB administrators, which then allow him to use models and experiments not yet available for the public.

The only dynamic information in this entity, 'Active', indicating whether the user is currently working with RLAB, has been introduced for implementation reasons; it could have been retrieved from other entities as well.

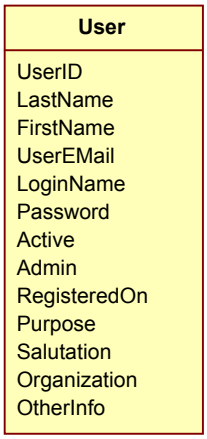

Fig. 2: User

b) Model Entity

The entity 'Model' describes a physical system available at the local site of RLAB (though this might also just be a simulation of a real system, being run by some RLAB computer). Each model instance is referenced within RLAB by its unique 'ModelID'. Its name is used in user interfaces, and the RLAB system addresses it at its URL. The latter allows to release the experiment on any system accessible through the internet, though RLAB usually places all systems close to each other, which allows to secure some of its internal communication procedures by a firewall. Models may be set up invisibly for a regular RLAB user; they usually are made visible only after a set of experiments are implemented and available.

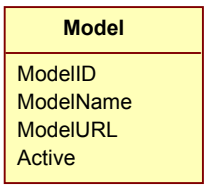

Fig. 3: Model

c) Experiment Entity

The entity 'Experiment' specifies an experiment that can be run at a specific model. Any number of experiments may be defined for a model. Each experiment instance is referenced by its unique 'ExperimentID'. Its name is used in user interfaces, and the RLAB system addresses it at its URL. An experiment belongs to a model. Depending on the type of the experiment a set of parameters give the user the opportunity to configure the experiment; the names of such parameters and the corresponding physical units are stored in the attributes 'ParameterNames' and 'ParameterUnits', each being a list of comma-separated strings. Some information on how to access the results of an experiment is given. And finally, as with models, experiments may be invisible for a regular RLAB user; after some testing has been performed they are made visible by setting 'Active' to TRUE.

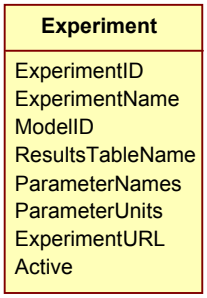

Fig. 4: Experiment

Besides the static entities of User, Model, and Experiment, the interaction between users, the RLAB system, and the models is stored in instances of dynamic entities which are created at runtime. Some of these are independent of the experiments performed, others store experiment settings and result data and thus are experiment-specific.

d) LoginInfo Entity

This dynamic entity keeps track of each login procedure a user performs. This is done by storing the UserID of the user together with a timestamp of the user's login operation. Whenever a regular logoff operation is performed, another timestamp is saved. Instances of this entity are identified by a unique 'LoginID', though this is just used for internal indexing purposes.

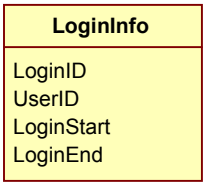

Fig. 5: LoginInfo

e) ReservationInfo Entity

The entity 'ReservationInfo' is used to store reservations made by the users. A reservation is made by exactly one user for exactly one model at a specific time slot, and thereby entitles the user to run any of the experiments of that model. As the duration of time slots is fixed within a given RLAB installation, only the start of the time slot needs to be stored (due to implementation aspects it is stored in several separate attributes).

While users may login to RLAB any time, they can only access a model if they have a valid reservation at the current time slot. Thus, a careful design of the time reservation procedures is crucial to prohibit different users from reserving the same model at the same time slot.

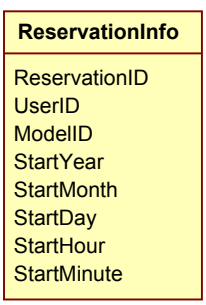

Fig. 6: ReservationInfo

f) SessionInfo Entity

Each time a user proceeds to start an experiment of a specific model, a session is instantiated. This is done using the dynamic entity 'SessionInfo', with the pre-requisite that a valid reservation for that model can be found. The start of the session as well as its end is saved. While the start time is due to user interaction, the end time may be caused by the user (e.g. by logoff or when switching to a different experiment) or by the RLAB system itself (when the reservation time slot ends and no subsequent reservation is found). When an experiment is successfully completed within a session, resulting data (like signal traces) may be available; this is indicated by the attribute 'DataExist'.

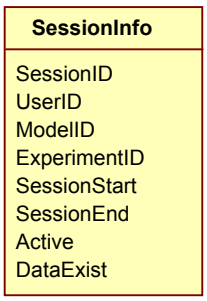

Fig. 7: SessionInfo

g) experiment-specific entities

Each experiment usually emphasizes a specific aspect of the physical system. Thus, both the options for the user to control the behavior of the system as well as the results produced by the experiment may differ significantly between experiments, and substantial flexibility is required when designing these areas of the data management.

The entry point to experiment-specific information is already established at the entity 'Experiment': the number of parameters the user may set to control the experiment is only limited by the size of the corresponding database field, and their names and units (and implicitly their number) are given in the corresponding static instance of 'Experiment'.

The same experiment may be run several times, and each run may have different parameter settings. To keep track of settings and corresponding result data of all experiment runs requires one instance of experiment results per run. The kind of data to be stored will be similar for different runs of the same experiment but may differ significantly between runs of different experiments. Thus the entity responsible for storing results (and settings) must be dynamic and experiment-specific, though it will show some commonality.

The entity for storing experiment results therefore is designed with a set of common attributes supplemented by a set of experiment-specific attributes. The name of this dynamic entity is specific for each experiment and is given as value of the attribute 'ResultsTableName' in the instance of 'Experiment' (see Fig. 4 above).

The common attributes allow identification of each experiment run by its 'Try' (a unique identifier). They furthermore reference the session (and thereby indirectly also the user and the experiment performed), and provide the timestamp of the experiment start. Finally, they allow the user to specify a file name. Such file may be used to store even larger amounts of experiment results (like signal traces of input, output, or internal signals over time). While it may be advantageous to adopt another open or even a proprietary format, RLAB has gained positive experience with tab-separated ASCII formats thus far.

Each of the experiment-specific attributes represents a respective unique experiment parameter, and it is meant to store the value set for the actual experiment run.

Fig. 8 shows, as an example, the entities for three different experiments with a twin-rotor system: each entity for experiment results inherits from the common part ('Results') and extends it by its own attributes.

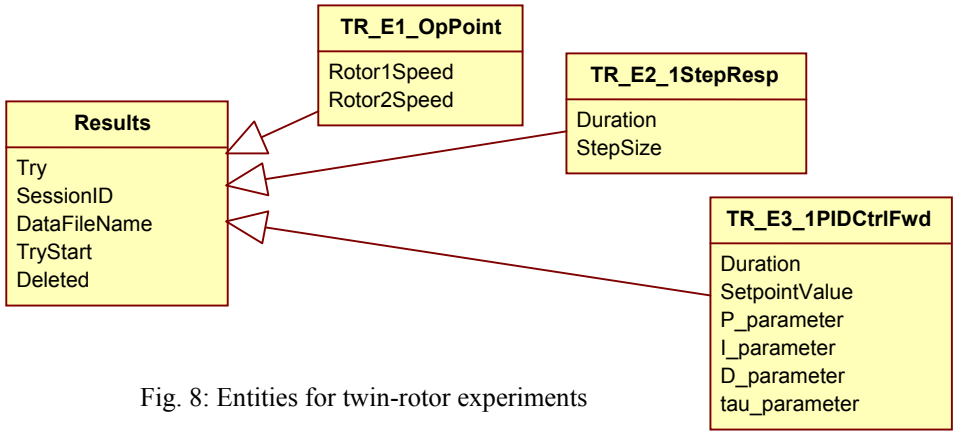

Fig. 8: Entities for twin-rotor experiments

The complete data model has been implemented using a relational database system. Each entity is represented by a database table. Attribute names are the headers of table columns. An instance is stored in a row of the corresponding table.

4. The Entity Relations within the Model

Fig. 9 shows the complete data model (with some example entities for experiment-specific information) as UML class diagram. The attributes are omitted for the sake of clarity. The data model is very compact, and static and dynamic parts are well distinguished as seen in Fig. 9.

The relations between the entities and their corresponding cardinalities are given in Fig. 9. The instances of 'User' and 'Model' are independent. The aggregation relations indicate that all instances of 'Experiment' belong to an instance of 'Model', and all instances of dynamic entities belong to an instance of 'User' (either directly or indirectly).

This data model advantageously avoids m:n relations, which eases the implementation in a relational data base system. 1:n relations are simply set at the n-side using the (unique) IDs of the referenced instances; they thus require just one value in the referring instance.

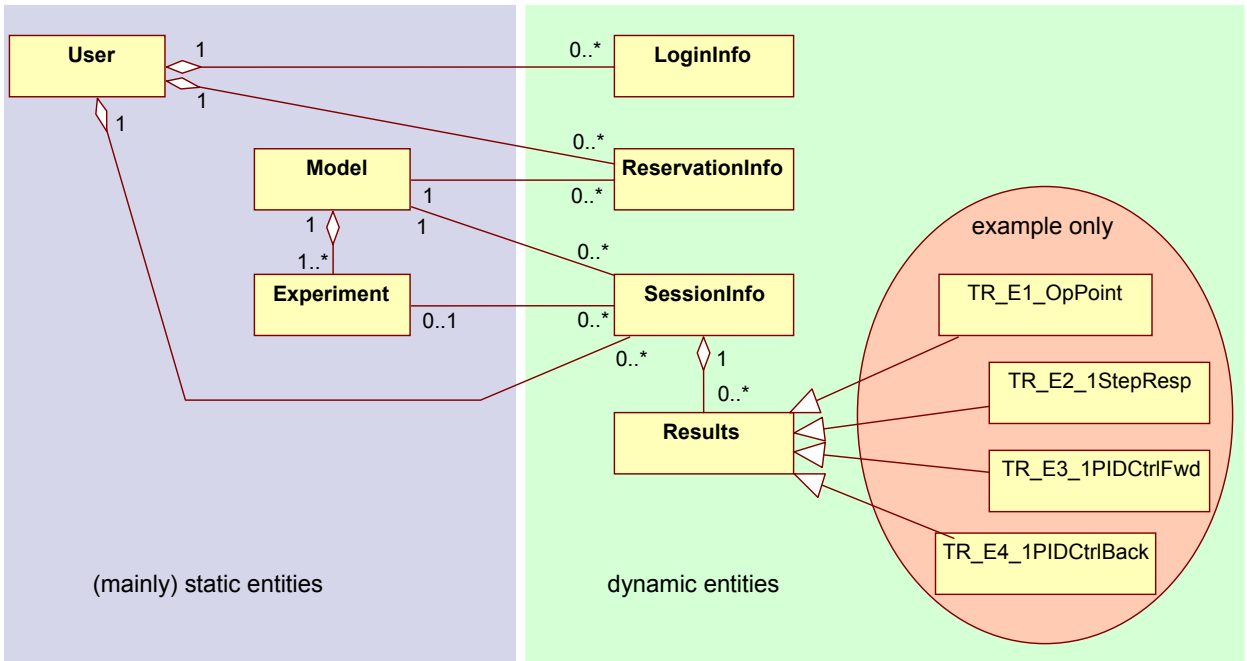

Fig. 9: Entities and their relations

5. Remarks on Expandability

There are two main use cases to expand the remote laboratory: adding another experiment to an existing model, and adding another model (with subsequently adding experiments to it).

It is obvious from Section 3 and Section 4 that there is not much effort required to adapt the database when expanding the remote laboratory in either perspective. The major effort will be spent to implement the functionality in hardware and software that additional new experiments and models require.

It is also obvious that the presentation of available models and experiments to the user can be implemented in a very generic way; there is very minimal modification to the user interface required if experiments or models are added.

a) Adding an experiment to an existing model

In this case there are only two activities required with regard to the database.

First of all, one further instance of 'Experiment' must be created (i.e. another row in that table). This instance specifies (besides the model it belongs to) the parameters which are available to the user, together with their respective units. It also specifies the name of the entity for experiment results (whose instances will be used to store the parameter settings and result data for each of the future experiment runs).

Secondly, a new table must be created with that name. The attributes stored in the table (and thus the table headers) are the standard attributes of an experiment run, followed by the experiment-specific parameters given in the attribute 'ParameterNames' of the corresponding instance of 'Experiment'.

Usually, 'Active' will initially be set to FALSE until a series of tests, run by an administrator of RLAB, indicates proper behavior of the new experiment before it is released to users.

b) Adding a model

In case a new physical system is to be made available to remote laboratory users, only one modification to the database is needed. As the available models (instances of 'Model') are kept within one table, one more entry must be added thereto, specifying the attribute values for 'ModelID', 'ModelName', 'ModelURL', and initially setting 'Active' to FALSE.

This may be followed by adding any experiments to this model according to the paragraph above, and after all tests have been run successfully, the attribute 'Active' may be set to TRUE to indicate the availability of this new model.

6. Conclusions

RLAB incorporates most of the aspects discussed in the previous sections. It shows an open architecture that can easily be applied to many other laboratory situations. The remote sites only require an internet browser; the local site needs hardware and software for the models themselves plus an installation of NI LabView and of an SQL-database (e.g. MS Access).

The experiences with the RLAB data management architecture look back on several thousands of logins and sessions, with a large number of experiment results being produced and analyzed. Up to now no major shortcomings of this data model have been observed. Duplicate time slot allocations, unauthorized experiment runs, and lost results have not been reported. Failures of the RLAB system were rare and mostly related to wear and power blackouts. Also few software-related problems (less than 10 per year) had been observed; memory leaks of some third-party tasks were found to be the reason, and even though not substantial they summed up

over the weeks, slowing down the systems. As a result the systems now are re-started during regular maintenance operations.

Future extensions of RLAB at the sites of Cologne University of Applied Sciences and University of North Florida are planned, mainly with the focus on adding further models and experiments.

References

[1] C.C. Ko, et al., "A Web-Based Virtual Laboratory on a Frequency Modulation Experiment," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews; Vol.31, No.3, August 2001, pp. 295-303

[2] P.S. Girão, O. Postolache, S. Antunes, F. Tavares, "Automated and Remote Operated System for Spectrum Monitoring and Control in Portugal," Proceedings of the 2010 IEEE International Conference of Industrial Technology (ICIT), 2010, pp. 988-993

[3] A.Kara, E.U. Aydin, R. Öktem and N. Cagiltay, "A Remote Laboratory for Training in Radio Communications: ERRL," Proceedings of 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2007

[4] M. Cabrera et al., "GILABVIR: Virtual Laboratories and Remote Laboratories in Engineering. A Teaching Innovation Group of Interest," IEEE Education Engineering (EDUCON), 2010, pp. 1403-1408

[5] S.K. Esche, C. Chassapis, J.W. Nazalewicz, D.J. Hromin, "A Scalable System Architecture For Remote Experimentation," Proceedings of 32nd ASEE/IEEE Frontiers in Education Conference, 2002

[6] J. Machotka, Z. Nedic, "The Remote Laboratory NetLab for Teaching Engineering Courses," Global Journal of Engineering Education, Vol.10, No.2, 2006, pp. 205-212

[7] Jezernik K., Rojko A., and Hercog D., "Experimentally Oriented Remote Motion Control Course for Mechatronic Students," Proceedings of the 2008 IEEE International Conference on Industrial Electronics, Control, and Instrumentation, pp. 3507-11, 2008.

[8] A.A. Kist, "Note on Remote Laboratory Access: A Networking Perspective," Proceedings of the International Conference on Signal Processing and Communication Systems (ICSPCS), 2007, pp. 605-609

[9] X. Martin, B. Price, J. Zhang, D. Dunlap, R. Adams, "A Database and User Interface Design for a Remote Accessible Engineering Laboratory," Proceedings of the 2007 ASEE Annual Conference & Exposition, AC 2007-894, 2007

[10] M. Murtra, G. Jansà, H. Martínez, J. Domingo, J. Gámiz, A. Grau, "A Proposal of Remote Laboratory for Distance Training in Robotic Applications," Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2007, pp. 1180-1187

[11] D. Karadimas, K. Efstathiou, "An Integrated Educational Platform Implementing Real, Remote Lab-Experiments for Electrical Engineering Courses," Journal of Computers, Vol 2, No 2, 2007, pp. 37-44

[12] Z. Allen, D. Schmidt, W. Wahlmann, " WebLab - Comprehensive Remote Laboratory System," Proceedings of the 2006 ASEE Annual Conference & Exposition, 2006-101, 2006

[13] O. Ropohl, "Entwicklung einer Software zur Steuerung und Regelung eines realen Systems über das Internet," Diplom-Thesis, Cologne, Germany, 2001.

[14] A. Koroll, "Virtuelles Labor für regelungstechnische Versuche an realen Modellen," Diplom-Thesis, Cologne, Germany, 2002.

[15] Cox D., Straatsma M., Bartz R., and Ctistis C., "Development and Enhancement of RLab Remote Laboratory System – An International Collaboration," Proceedings of 2009 Florida Conference on Recent Advances in Robotics (FCRAR '09), Jupiter, Florida, USA, 2009.

[16] Straatsma M., Cox D., Ctistis C., and Bartz R., "Development and Enhancement of RLab – A Remote Laboratory System," Proceedings of Fourth International Conference on Systems and Network Communications (ICSNC '09), Porto, Portugal, 2009.

[17] Cox D., and Bartz R., "Development and Integration of Project-Centered Modules into RLab Remote Laboratory System Environment," Proceedings of 2009 International Conference on Engineering Education, Seoul, Korea, 2009.