The FEI Small Dual Beam product family

By: Koen Driessen, Pybe Faber, Kees Kooijman, Pepijn Kramer, Hugo van Leeuwen, Pieter Schröder
FEI Company

1 Introduction

Ever since the invention of the transistor and the possibility to integrate these on a large scale in integrated circuits, miniaturization of electronic devices has become pervasive throughout society. And not only electronic devices are subject to miniaturization, all kinds of man-made structures and products are being made smaller and smaller, in the process retaining their function, and often also obtaining properties and functions that before were not possible. Now a significant number of these deal with typical structures (well) below 100 nm, we are said to have entered the “nano-technology era”. Miniaturization is not so much a market trend as it is a trend in society.

Structures below 100 nm cannot be inspected with light optics, as the size of these structures is below the wavelength of (visible) light. One can win a little bit by moving into the ultraviolet spectrum, but in this way one will never be able to characterize gate oxide thickness, which is currently in the order of 2 nm. To achieve resolution on this scale, there are a number of techniques one can turn to, one of which is imaging by means of charged particle optics, viz. electrons and ions. By choosing the accelerating voltage appropriately high (e.g. 30 kV for electrons is more than sufficient), the wavelength of the particles is no longer the limiting factor.

FEI Company (FEI stands for “focused electrons and ions”) is in the business of electron microscopes, where the charged particle optics technique has been applied to explore the realm below visible light. The miniaturization trend has caused our CEO to exclaim: “the trends are our friends”, since obtaining even the simplest of image at these scales requires an electron microscope.

FEI Company provides a wide range of products based on electron microscopes, and a particular product family is called the Small Dual Beam (SDB) family. The term “small” denotes the size of samples that can be entered into the instrument. In this case, small is meant to contrast to the larger instruments that can hold full-scale wafers of 8” (± 200 mm) or 300 mm. The term “dual beam” denotes that there are two beams, one electron and one ion beam. The electron beam is typically used for non-destructive imaging, the ion beam for modifying the sample at a site of interest, e.g. milling holes in the surface of a wafer. One can also image with an ion beam, as the principle is the same: the impact of ions causes all kinds of secondary processes. One can imagine, however, that this technique is always destructive to some extent: the structure that has been imaged will never return. On the other hand, this suggests a new imaging technique, where the same region is imaged over and over, and each image is stored. The resulting stack of images represents a three-dimensional volume, which can be made visible by means of 3D visualization techniques. We have called this technique “Slice and View”.

This document describes the architecture of the SDB family. For this purpose first a short introduction to the product family is given. Then the external view on the architecture is given, viz. which key customer drivers are present in the three markets we distinguish for the SDB family, and how these key drivers translate, through derived application drivers, to product requirements. Next the internal view is given, namely how the product requirements are translated to what is viewed as “the technical architecture”, viz. how the product is constructed from several points of view. The internal description focuses on the concepts, and touches on a number of the realization aspects. The focus in the internal view is on software, as much of how the functionality is made accessible to the user is determined by software. We conclude the document with some concluding remarks. Some readers may recognize in this approach the work of Muller [1], which indeed has been a source of inspiration for our approach to the description of the product family architecture.

2 The SDB product family

The small dual beam product family consists of a set of electron microscopes with the following properties/features:

- The main platform is a vacuum chamber on which an electron column and ion column are placed.
- The electron beam is mainly used for imaging. Electrons generated in the gun unit at the top of the column are focused onto the sample. By means of an electron deflection system that can be either magnetic or electrostatic, the beam is scanned across the sample. The primary beam has interaction with the sample in processes that are gouverned by physics, and as a result a variety of secondary sources (secondary electrons, X-ray, diffracted primary electrons) “originate” at the sample at the location of impact. These signals contain information and are collected by several detectors, and a coherent image is generated that is displayed to the customer.
- The ion beam is used to modify the sample. Its behavior is similar to the electron column but works with heavy ions. It is mostly used to cut a hole so that a cross section can be made. Other applications include modifications on silicon wafers (making conductive tracks, cutting away pieces of silicon and so on). Another special application is to make very thin samples that can be analyzed in a TEM.
- Inside the vacuum there is a sample manipulator (the stage) that is used to position the sample with respect to the electron and ion beam. There are several manipulator systems ranging from low cost to high precision, long stroke versions. For some systems loading of the samples is semi-automated.
- In all SDB systems the electron and ion beam have a single coincidence point but placed under an angle of 52 degrees.

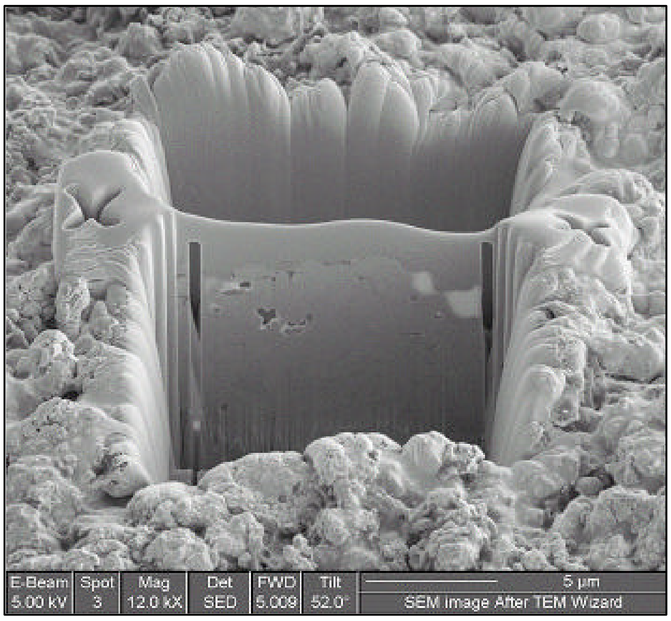
The versatility of the instrument including all the possible configurations leads to a wide variety of application areas. Several systems are defined serving these applications:

Quanta 3D: This system uses a low cost tungsten column and offers a special high-pressure vacuum system that enables analysis of a wide variety of samples with moderate resolution. The samples do not need any preparation and coating, and with its analytical flexibility the instrument provides first-line fast diagnosis. With its tungsten source tip (lifetime 40 hours), it is not suited for automation.

Nova Nanolab: This system is equipped with a high performance (resolution) electron column for optimum characterization capabilities. It enables very high-resolution images and offers much flexibility to the user. Non-conducting and organic samples, however, need coating, which increases cost and time-to-diagnosis.

Strata: This system is typically targeted for the Semiconductor industry, providing special analysis techniques for semiconductors (STEM) and a very high-resolution sample manipulator stage. It can be characterized by many automation facilities, convenient and fast loading mechanisms (high throughput) and a high performance ion column. With this system special routine tasks can be performed very quickly.

To get an impression of what one can do with the systems, on the right an image is shown that is obtained with the Nova Nanolab. It is an electron beam image of a structure that has been made with the ion beam. At either side of the thin slice a trench is milled, and at the sides and the bottom of the slice grooves have been milled (for this purpose the stage was tilted first), so that the whole slice is suspended solely by the two small connections at the side. The slice is a thin cross-section of the material, and can be transported with an accessory that is not discussed in this document into a TEM (transmission electron microscope). The TEM is capable of revealing with even higher resolutions what is the nature of the structures in this slice of material. The application that prepares this structure with a high degree of automation is called TEM sample preparation.



3 External view

In this section, we will discuss the market inputs from the three market segments that FEI usually distinguishes, and how these inputs, or *key customer drivers* map onto the principal functional requirements for the SDB family. We distinguish the following three markets: Semi-conductor industry (often abbreviated to Semi), Institute (sometimes also called Academia) and Industry (obviously, non-Semi). The context in which customers use our instruments differs in these markets:

Semi: Typically, a semi-conductor factory (FAB) constructs integrated circuits on wafers of a diameter of 200 mm or 300 mm in a large number of process steps. These processed depend critically on a large number of parameters, and mistuning of a parameter reveals itself in some unwanted artifact, which, with the diminishing node size, increasingly often becomes too small to see with a light microscope. Also, a number of failures may be hidden below the surface. Since there is large economic value in a wafer, the FAB monitors the process steps by regular inspection of the wafers. Typically, every n^{th} wafer is taken out of the line in order to be able inspect the resulting structures of a process step. The wafer is taken to the laboratory (LAB) where it is subjected to a possible large number of tests. The SDB is positioned to perform as many of these tests as possible.

Institute: In the Institute or Academia market, the SDB is placed in a research lab among a variety of other instrumentation. It is typically used for a wide variety of tasks, driven by the particular line of research of the Institute. The instruments are usually operated (and sometimes modified) by highly educated personnel (PhD students and post-docs).

Industry: In the Industrial market, e.g. the pharmaceutical industry, the SDB instruments are also placed in a laboratory (LAB) rather than in the production line, but the versatility of samples that is inserted in much higher. Organic samples, for instance, are much more volatile than Silicon, and therefore various techniques can be used to prepare the sample, such as coating, cryogenic (deep-freezing the material in liquid Nitrogen), or alternatively special detectors that allow for poor-vacuum conditions can be used.

Although sales and instrument types do not obey the strict boundaries of market segmentation, with a crude simplification one can say that the Strata is targeted towards the semi-conductor industry, Nova Nanolab towards the Institute market, and Quanta towards the Industry market. In particular, however, the latter two are “cross-sold”. As the market volume currently does not justify completely separate instruments for the separate markets, the product family is derived from a single platform, where modules and options can be selected on the basis of desired specification and performance.

3.1 Key Customer Drivers

In Table 1, an analysis is condensed of how the key customer drivers for the various markets relate to derived application drivers, and how these translate to the principal functional requirements for the SDB family.

The Semi-conductor market has three major key customer drivers, in as far as this is relevant to our product portfolio: yield improvement, time-to-market and cost. As mentioned, every wafer represents huge economic value due to huge amount of integrated circuits that can be “pulled off” a wafer, if functioning correctly, and therefore the yield of the product line is the most monitored parameter of a FAB. A second key market driver is the time-to-market of the electronic components. Numerous economical studies reveal that in the industry of integrated circuits, the first to hit the market with a particular device will take the majority of the market share, and that the highest margins can be obtained in the first period – competition will bring the prices down. Therefore, the time it takes to bring a new product to the market is therefore also very important. Finally, the cost / benefit ratio always enters the equation at some place, but since the economic numbers associated with the first two drivers are so much higher than the third, this last driver is expected to be of less significance. It is not so much the cost of purchase as the cost of ownership (consumables and operator-cost) that that counts.

In contrast, the Institute market has very different customer drivers. Here, the scientific reputation, obtained through scientific results, of the institute or the leader of the line of research is the primary driver, as this determines the opportunities for future research. Often, these institutes also have an educational task, so therefore training of students on these instruments is another driver. Finally, being a non-profit organisation, cost enters from a different side: purchase of the instrument is the more influential factor, as the cost associated with operator, although the are often highly educated, is often still relatively low.

Use case	Market	Semi								Institute					Industry						
	Key customer drivers	Improve yield		Time-to-market					Cost	Reputation, publications		Education	Cost		Improve yield		Quality		Time-to-market	Diagnosis	Tool efficiency
	Requirements	Process control	ilure analysis	e analysis	ctural dimensions		surements	air / Circuit edit	f ownership	e first to measure X	Be first to create structure Y	Train operators	Share between departments	Cost of purchase	ocess control	e Analysis	nce to standards	laims	nalysis	e / disprove isease, Guilt, Compliance to standards	Service labs
Navigation	Navigation																				
	Re-visi																				
Imaging	Reso																				
	Tech																				
	Anal																				
	F																				
	C																				
Material proc.	Se																				
	Depo																				
	"End point d																				
Data proc & mgmt	Measurements																				
	Database																				
	Certification																				
	Compare to stan																				
Auto mation	Automation																				
	Repeat experim																				
General	Uptime																				
	Throughput																				
	Robustness											x									
	Versatility									x			x								
	Flexibility									x											
	Ease of use								x												
	Multi-user											x									
	Usage/user logging												x								
	Low cost													x							
	Remoting																				x

Many to many

Table 1: key customer drivers to functional requirements mapping; "x" denotes strong mapping, hr = high resolution, TP = Tem Prep

The Industry market has key customer drivers that resemble the Semi-conductor market, although they have a slightly different flavour. The importance of yield improvement and time-to-market can be argued along similar lines, although the process is of a totally different nature: in the pharmaceutical or chemical industry, samples can be taken in smaller doses and more easily, and often it is size and form characteristics of the particles that are of importance. In addition, proving the quality of the product with respect to the relevant standards is important and often enforced by legislation. In the medical as well as the forensic field, diagnostics capability is a key driver. Finally, there are a number of cases where a specific analysis can only be done with these types of instruments, but are so infrequent for a particular organization, that a service lab performing these tasks for a large number of different clients is the only economically feasible way.

3.2 Derived Application Drivers

From the key customer drivers, the more specific application drivers can be derived. Yield improvement calls for the ability to monitor and control the processes in the production line and the ability to analyze the failures that may occur in the individual process steps. Time-to-market also translates to failure analysis, and in the case of the Semi-conductor market also calls for two- and three-dimensional characterization of the structures that are being built, and techniques to measure specific parameters such as doping profiles (in the transistors) and voltage profiles (how the electrical potential is distributed given a certain voltage on the test pins). An advanced application is circuit-edit and mask repair in the development labs, and although the SDB does have a (limited) number of facilities for this application, the circuit-edit and mask repair market is covered by a different business line in FEI. Cost, as discussed above, has a number of aspects that have a different weight in different markets, but this includes cost of ownership, cost of purchase and the ability to share the instrument between departments of the same organization or even between different organizations.

For the key driver of scientific reputation (primarily obtained through publications), the derived application driver is the ability to measure a certain feature or characteristic of the sample under investigation, and the ability to create a specific (nano) structure. Experimentation capability and rapid prototyping is key to these objectives. For education, the derived application driver is the ability to train operators. In order to prove the quality of a product, it is necessary to be able to show compliance to (industry) standards, and these collected and documented proofs provide ability to deal with liability and claims – both from customers and to suppliers. The key driver for diagnostic capability resembles the product quality driver in the ability to show compliance to standards, but in addition, depending on the field, translates to the ability to prove or disprove the presence of a disease, or guilt of a person in a crime case.

3.3 Functional requirements

As mentioned before, the market size does not justify separate architectures per market and/or instrument type; instead we have chosen to develop a platform from which particular instruments can be configured. From technical viewpoint, this is even desirable. The mapping of the application drivers to the functional requirements has a many-to-many characteristic: one application driver maps onto multiple functional requirements, and many of the functional requirements serve multiple application drivers. In order to depict these relationships, Table 1 was composed, where an “x” on a crossing denotes a strong relationship between application driver and requirement. This does not imply that if there is not an “x” that there is no relationship, but to concentrate on the big picture these weaker relationships are not depicted. We will not describe all relationships in detail, as this would lead to a somewhat tedious list with quite some repetition, but rather touch on most of the resulting requirements to give a feel for what the instruments should be capable of.

For the purpose of process control, samples are taken out of a production line at strategic points in the process. These can be wafers (that need to be broken) or pharmaceutical or chemical samples. Usually, the things one is interested in are so small that sub-micron *resolution* is needed to image the structure of interest. With resulting magnifications this large, only a very small portion of the whole of the sample is visible, and therefore a first task is to find a relevant part in the sample. In the semi-conductor case, *navigation* to specific sites on the sample is needed, e.g. test structures with known characteristics. For other samples manual search for relevant structures usually is the quickest way. Once an interesting site is found, one needs to reproducible *measure* the size of certain features. (Interestingly enough, in the semi-conductor market one does not need *calibrated* measurements per se, as long as repeated measurements yield the same results. However, since we do need calibration for other purposes such when exact measurements are needed, we prefer to have our instruments calibrated.) As the measurements in a production line are performed over and over on different samples, *automation* of the measurement process is very desirable for operator benefit. Finally, *uptime*, *throughput* and *robustness* are some general requirements, without which the instruments will fail in all industrial environments.

Sometimes, the relevant information cannot be derived from a two-dimensional image of the surface, as it is buried below the surface (e.g. the width of a via in an IC device). For this purpose, it is needed to “drill” a (nano-scale) hole, which we call *sectioning*; a particular pattern is milled into the surface in order to be able to see the cross section of the material. This in itself reveals a basic design principle of the dual beam family: two beams are directed at the same spot at an angle (it so happens) of 52 degrees. One beam is for imaging (a scanning electron microscope or SEM) and one for milling (a focused ion beam or FIB). The spot is called the coincidence point and at that point the FIB can mill a hole in the sample or sample, which directly can be inspected with the SEM from aside, without moving the stage. This latter point is important, as stage movements cannot be reproduced with accuracy in the order of nanometers. As can be seen in the table, the capability of sectioning is useful for a large number of applications.

It is noteworthy that the converse of sectioning, *deposition*, in which material is not removed but added in a very controlled manner to the sample, can be achieved with very similar techniques. Milling a hole is done with the ion-beam, where the landing place of the ions determines the location of the hole. This process is enhanced by added specific gasses by means of a gas injection system (GIS) near the landing place of the ions. Deposition can be achieved by controlling the electron beam similarly: by directing the e-beam at the desired location, and injecting a different gas nearby, the material is deposited at that location. This technique is useful, among others, in the research field of nano-technology.

In Industry failure analysis, as well as in academic research, various *techniques* are needed, or have been developed in order to obtain the relevant information out of the sample. Techniques may vary from sample preparation (coating, cryogenic) to illumination conditions (spot size, current) or detection conditions (detector choice, grid voltage, stage tilt angle). Also, *analysis* is an important requirement. Data can be obtained from a multitude of detector sources. Typical applications are Energy dispersive spectroscopy (EDS) for elemental characterization, Electron Back Scatter Diffraction (EBSD) for crystallography, and Scanning Transmission Electron Microscopy (STEM) on thinned slices of material for enhanced resolution. After acquiring the raw data, often some further processing is needed (e.g. averaging) to improve the signal to noise ratio and presentation of the data.

Raw data and processed data alike need to be made persistent through *database storage* for a multitude of purposes, not only legislation in case of pharmaceutical or medical application, but also convenience and the ability for data mining in other fields. *Certification* and *comparison with standards* are more specific requirements that may also need data or information obtained though other means or instruments.

Some final, more general requirements are *versatility*, the ability to apply the instrument for a multitude of tasks, *flexibility*, the ability to quickly switch from tasks or sub-tasks, and *ease of use*, the ability to drive the instrument efficiently and without the need for extensive training. These requirements apply more to markets where the instrument is a relatively expensive and possibly shared resource, and naturally, *cost* then also is an issue.

3.4 Use cases

A final note to Table 1 is that the functional requirements are organized according to the typical use case of the instrument. The typical workflow is as follows: one *loads* a sample, *navigates* to an interesting spot, *images* the site, *processes the material* if so desired, and *manages the data* that come out of this process. We call these step tasks. (As a currently still unavoidable task, *alignment* is also needed, but this is not a functional task, but rather a necessary preparation in order to ensure that one can rely on nanometer accuracy of the results.) The tasks can be automated, or performed manually, but the bulk of the usage complies with this workflow. In fact, in the development of the instrument family, one can distinguish the incremental approach of first achieving the basics (manual operation) and then automating these tasks step by step. One can imagine that automating these tasks is only possibly if the basis is robust and reliable and that the solution will have high software content.

The workflow is so typical for the instrument usage, that these concepts return in the operation of the instrument (e.g., how the UI is configured) and in how the instrument is designed from control and software point of view. The workflow, therefore, will also guide the presentation of how the family of SDB instruments is designed and realized.

4 Internal view

The product requirements and the typical use cases have driven the construction architecture of the product family. In this section, the architecture is described from several points of view. We will concentrate on the longer-lived conceptual aspects, and touch on specific realization aspects when appropriate.

The architecture and the underlying concepts have not been developed from scratch. A number of the software concepts have been developed during the development of a product line in a different business line: the TEM product family (called Tecnai) that was released in 1998. A significant number of concepts from the Tecnai have been generalized in a SEM (thus, a single beam), which is the basis for the current SDB platform with respect to software and electronics. This product, called Quanta, was released in 2001. On the other hand, the predecessor of the small dual beam had been developed at a different FEI site, Hillsboro (Oregon, USA), and the SDB family needed to provide continuity in functionality and user access to the functionality. Finally, the production modules come from different FEI sites, which is driven by where the expertise resides and which site needed a particular module first. For example, the GIS and ion column comes from Hillsboro, while the software and the electronics for the ion column come from Peabody (Massachusetts, USA), and the vacuum solution and high voltage generation comes from Brno (Czech republic). These “super modules” as we call them need to be merged with the supply chain that is set up locally and the software needs to be integrated into the rest of the software. In this manner, we have been able to release Nova Nanolab and Quanta 3D in 2003, followed by Strata in 2004.

4.1 System decomposition

The architectural decomposition of the small dual beam family is organized along several axes in the hierarchy of the system, which is represented in Figure 1. There is no single view to decompose the system in functional blocks for the entire architecture. The system can be subdivided in a couple of layers that needs to be decomposed very carefully to obtain a manageable system. For each layer we apply different criteria to find the optimal decomposition.

- **Applications.** Applications are the system functionality as exposed to the user. They involve the organization and control of specific tasks that need to be performed with the system. There is a well-defined interface to the system that is shared by all applications, which we call the Object Model. Examples of applications are “CadNav”, “Slice and View”, “Tem Sample Prep”, or the general-purpose user interface. The decomposition criteria are:
  - Multi-site development: application development is time-intensive, time-to market.
  - Configurability: different customers may have differing needs.
  - Stand-alone: there are minimal dependencies between applications.

- **Tasks**. System tasks are the generic functions of the system common to all applications. It can be compared to a high degree with the functionality of a general optical light microscope where minimal automation is applied. These are the typical use cases referred to in the previous section: Loading/unloading of samples, navigation (across the sample), imaging (Focusing, adjusting contrast and brightness), material processing (modification of the sample), and data management (analysis and data storage). Typical characteristics that influence the decomposition are:
  - System functionality breakdown: provide well-defined behaviour to the application layer.
  - Commonality within the family: generics are captured in “engines”, specifics in rule sets.
  - Minimal but complete interfaces to application layer: external invocation must be well defined.
  - Full-fledged but clear interfaces to control layer: to maximize internal software efficiency.

- **Control**. The control architecture is responsible for the translation of high-level task commands to the low level settings of the hardware of the machine. Decomposition criteria are based on technology choices and the mapping of single high-level commands to multiple settings in the hardware. Responsive, glitch-free control is necessary. General characteristics are:
  - Abstraction to generic behavior: capture module specific behaviour in the module with sufficient specifics to allow extensive high level test alignments and diagnosis.
  - Function oriented: The modules are knowledge intensive and dedicated developments in a specific domain: vacuum technology, motion, high voltage design, data acquisition/processing and electron-optical technology.
  - Reusability: The modules need to be reusable in future products and different business lines.

- Multi-site development: expertise on different modules resides at different sites.
- Hardware abstraction: exchange of hardware (new technologies, supplier changes) should have minimal impact on generic module behaviour.

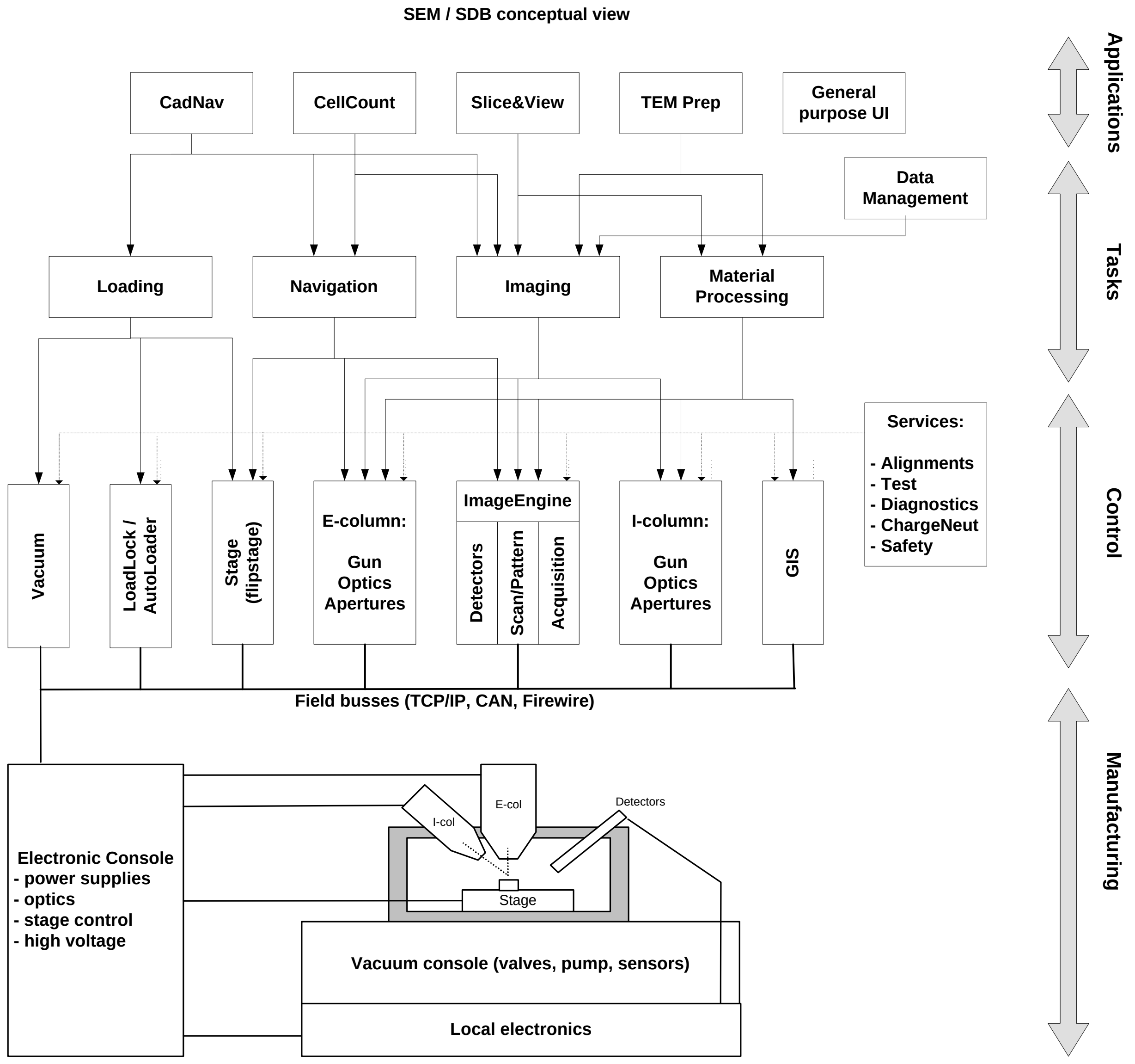


Figure 1: the SDB family system decomposition and interrelationships on the different levels of abstraction.

- **Manufacturing**. From a production perspective it is efficient to divide the system in a number of stand-alone hardware modules that can be manufactured independently of each other. The major modules are: the columns, the vacuum chamber, the electronic console, power supplies, etc. The decomposition criteria are:
  - Stand-alone units the units are assembled at the supplier site and should be able to be operated with minimal infrastructure.
  - Production sites: Second sources of the units should be possible.
  - Supplier capabilities: alignment with the (mechanical or electronic) core competences of the supplier, definition of mono- or multidisciplinary modules.
  - Testability: Supplier must be able to test the unit. Quality must be ensured.

From software realization point of view, the upper three layers run on a PC, although user interface aspects of the applications layer may also run remotely on a second PC. The rationale for this was development efficiency, which translates in time-to-market for our products. The software runs on the Windows NT operating system (with its upgrades to Windows 2000 & XP) and is written in C^{++}. The COM interfacing technique, being readily available on the Windows platforms, has been chosen for communication between the software modules (which therefore become implemented as “COM objects”).

4.2 User interface

In Figure 2, a screenshot of the Nova user interface is shown, and although the user interface of the other instruments in the SDB family will differ in subtle aspects, they all follow the general layout.

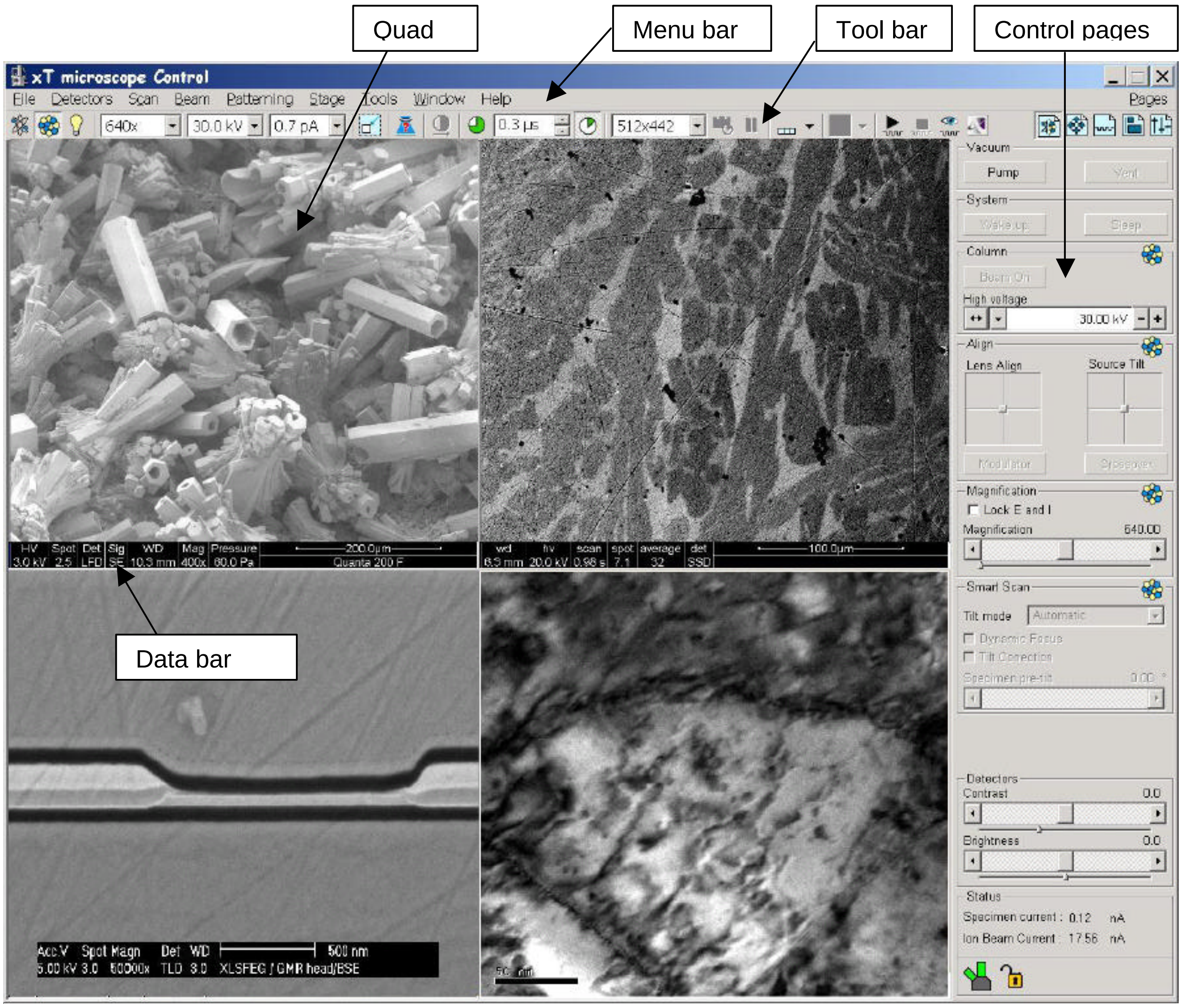


Figure 2: Graphical layout of the SDB user interface

The majority of the screen is reserved for four images (or other data) in what we call quads. The image size fixed in order to prevent cluttering of the UI, but the user can choose to enlarge any of the quads to full size, then filling the 1024 by 1024 pixels in the image area. Below an image, a Databar shows the conditions at which the image was obtained. Interaction with the microscope is primarily done through “direct manipulation” of the images; by means of keystrokes and mouse actions on the image, commands are issued to the instrument. Above the image, there is a menu- and toolbar that provides alternative and graphically oriented control. The toolbar provides access to the most frequently needed commands; the menu bar provides a move complete palette, and also offers entry to preferences or log info dialogues. These dialogues are designed to fall exactly over one of the quads. Some interaction, however, is more complex than a single button press, and for this reason the room right of the image area is available. This area is reserved for what we call the control pages, and the icons directly above this area (rightmost in the toolbar) can be used to select a page. These pages are organized according to the *tasks* that were identified above, be it (to the user) under slightly different names. They are not, however, organized according the task flow in time, but according to “time most spent”, so therefore the default and opening page is the page for obtaining and perfecting the image.

In Figure 3, the functionality that is exposed by the user interface is organized according to the tasks on one axis, and user interface interaction components on the other.

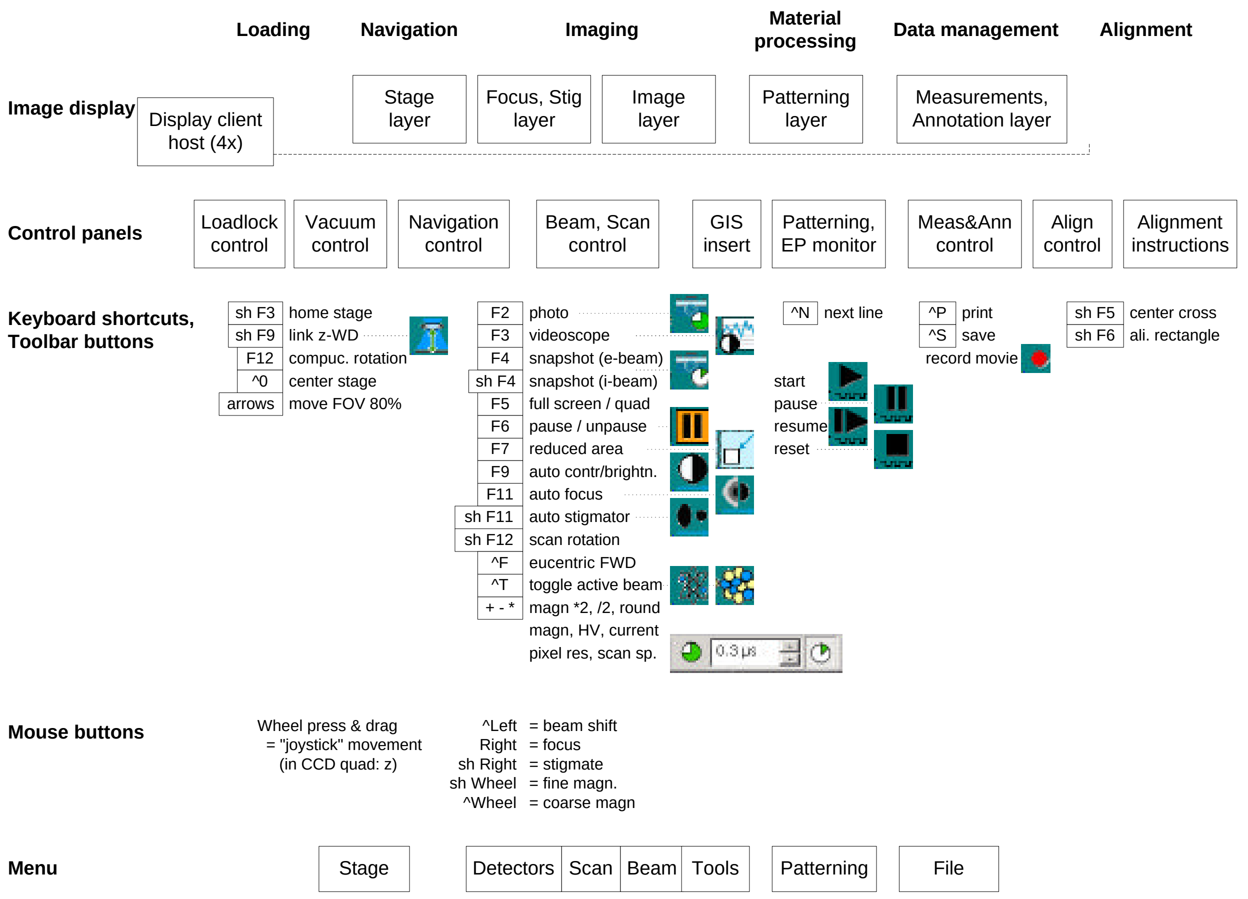


Figure 3: Functions offered through the user interface, arranged according to task and control means

Image display is realized by a layering host, which hosts layers of various natures. Typically, the image is the first (or bottom layer), on top of which (usually mostly transparent) layers are displayed. For the navigation task, there is a layer for stage control: it displays compucentric rotation, and mouse interaction is captured for moving the stage. For the imaging task, apart from of course the image layer, a focus and stigmation layer is present, which enables the user to obtain a sharp and stigmation-free image. (Stigmation means that the probe is ellipsoid rather than circular, which shows up in the image as a smear in a particular direction.) For the task of material processing, the pattern layer enables the user to define where a mill pattern should lie, and provides feedback on where exactly the individual mill points lie, how large the mill spot is and what the amount of overlap. The pattern characteristics need to be specified through a control panel. Finally, for the data management task, a layer to perform measurements and add annotations is available.

For interaction with the instrument that cannot be performed solely on the image, a number of “control panel” items are available. Control panels are the reusable entities on the control pages; a panel can be present on more than one page (e.g. instrument status, which is not depicted above, is present one all pages), and a page is typically composed of multiple panels. An example of the default control page is shown in Figure 2, but it falls outside the scope of this document to discuss the panels in more detail.

The interaction that is most frequent is captured in (fixed) keyboard short cuts and toolbar buttons. The mappings are kept fixed to minimize confusion for the customers. In the figure, a particular function is accompanied by the key stoke on the left, and the toolbar icon on the right, if available. The horizontal position of the lists corresponds to the task with which the function is associated. Note that by far the majority of interaction is available for the navigation and imaging tasks. These are the tasks that occupy users for most of their time when finding a suitable site on the sample and setting the conditions for further processing exactly right. Other tasks may take more instrument time, such as milling extensive patterns, but these require little interaction and usually the user will let the instrument run unattended. For the same reason, the mouse buttons are also assigned to the navigation and imaging tasks, and the menu shows most entries in these areas.

4.3 Execution model

The execution model of the SDB software is depicted in Figure 4. It shows the processes that are active on the control PC when the system is operational.

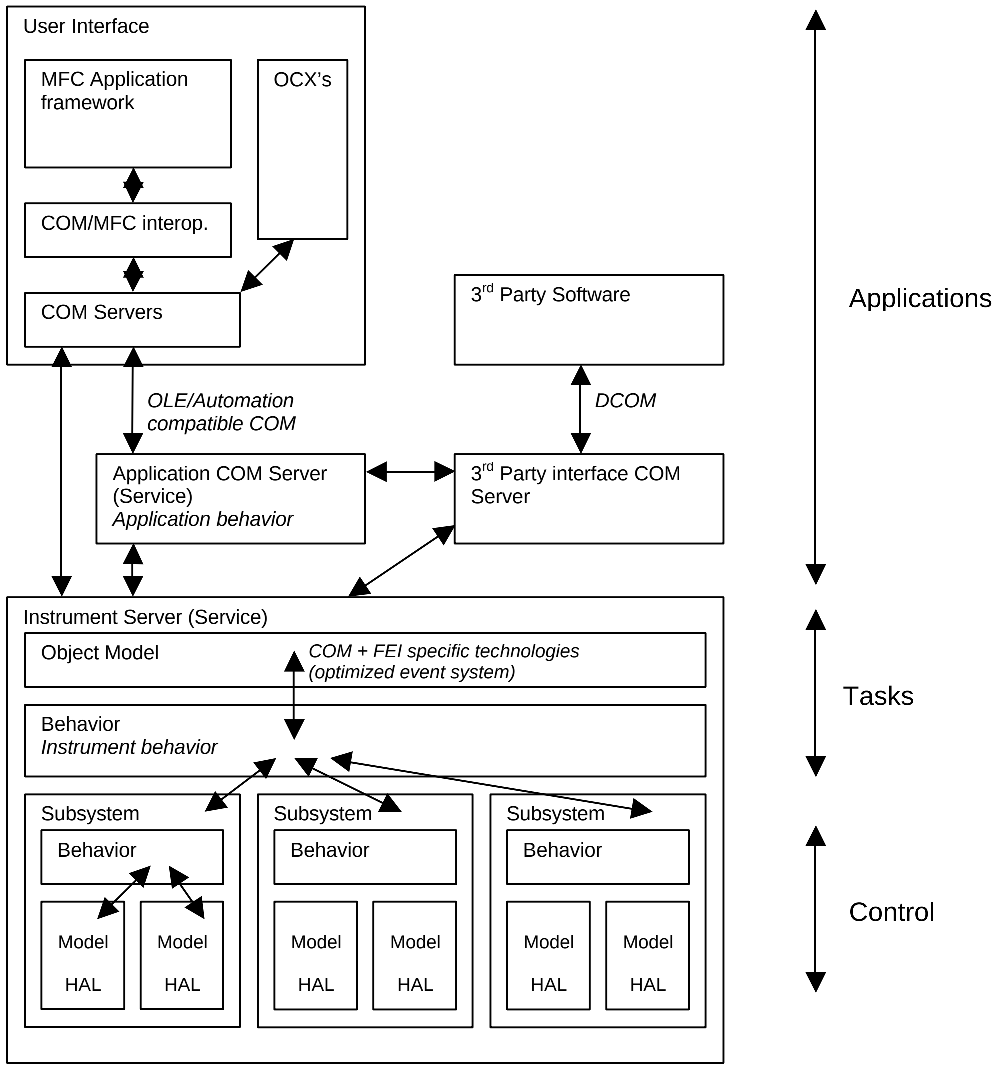

User Interface
MFC Application framework
OCX's
COM/MFC interop.
COM Servers
3rd Party Software
OLE/Automation compatible COM
DCOM
Application COM Server (Service) Application behavior
3rd Party interface COM Server
Applications
Instrument Server (Service)
Object Model
COM + FEI specific technologies (optimized event system)
Behavior Instrument behavior
Tasks
Subsystem
Behavior
Model
HAL
Control

Figure 4: Execution model of the SDB software, in as far as this runs on the PC.

In this figure, all software that is running on the control PC is shown, which corresponds with the application-, task- and control layers of Figure 1. Every outer rectangle represents a process and layering denotes dependence. The instrument server holds the tasks- and control layers and is a Windows NT service (comparable to Unix daemons) that is automatically started at startup of the PC. The instrument server is a stand-alone entity and therefore the hardware is controlled and monitored as soon as the PC is operational.

Above the instrument server, an application server (also a service) is present, which started automatically with a dependence on the instrument server. The application server holds the parts of application layer that are not associated with graphical user interface display or control. Typically, application specific modeling is captured here, for instance the presentation choice that the user interface holds four quads, and that every quads can hold (and remember) the last acquired image of any beam, i.e. the electron-, ion- or “photon” beam. (The latter “beam” is a CCD image of the interior of the chamber, that one typically uses during manual operation in order to prevent the stage to hit the final lens of the columns.)

On top of the two servers, the general-purpose user interface runs, which is a regular application (in the Windows NT sense). The UI is started and stopped by the user at desire. It does not hold any state (other that user preferences which are persisted) and can run remotely. These three processes form the basic software of a small dual beam. Applications can run as stand-alone processes or can be integrated (through the OCX-es, which is a COM object that in addition has a graphical user interface, which is to be placed on the screen by the hosting application) in the user interface. It depends on the application whether this is desirable – some applications do not need the user interface, others do – and sometimes integration has not (yet) happened due to the fact that the application has been developed at another site. Third party software can also interface with the SDB software, but it falls outside the scope of this document to describe this in detail.

A number of tasks are dealt with by embedded processors in the hardware (not shown in the figure) to hold critical machine state and instrument safety, for instance to preserve vacuum when the PC is restarted, or more in general, for “hot-connectivity”. A design choice, however, has been to implement as much functionality in the PC, at least initially. As aforementioned, the rationale for this choice was to optimize development speed, as developing on a PC, which is both the development platform and the target operational platform, is more efficient than on embedded platforms. Debugability is better, and also simulation capabilities are larger (see below), which makes a high degree of stand-alone testing possible. However, once a certain portion of functionality has matured and would enhance hot-connectivity if it were to be moved down, this is typically done in one of the major software upgrades (as was the case with vacuum persistence).

The inner rectangles in the figure denote (aggregated) components that have a functional *raison d’être*. They are realized as COM components that are instantiated in a controlled manner. At the lowest level a HAL layer abstracts the hardware devices. In general, this takes the form of individual devices, which are grouped according to the printed circuit board (PCB) on which they are realized, which are grouped according to field bus. The (object oriented) software is a mirror of the hardware, and we implemented the simulation mode by having a call on the hardware “bounce” at the level of the field bus back to the software representation of the device, which then simulates a successful call. In this way, the fully configured software is the same for simulation and actual control, which means that successful simulation provides a reasonable level of confidence that the actual control will also run correctly. Simulated runs can, of course, be done directly on the PC, without the necessity to download the software to a test environment or the real instrument, and that, of course, enhances development speed and test coverage.

The modules on the so called model layer provide a mapping of meaningful user functions to coordinated control of individual actuators, e.g. the Vacuum Model translates the states of individual pumps, valves and pressures into system level vacuum states (vented, pumping, high vacuum, low vacuum), and the Optics Model translates desired shift or tilt of the beam into individual DAC settings. Often some modeling is done at this layer, for which reason it is called the Model layer. Also, the Model layer serves as a “glue layer” between the task-oriented behaviour layer, and the construction-oriented hardware, which enables more independent development at either end. On top of the Model layer are the Behaviour layers, which we separated into two conceptual layers; one for behaviour within a function (e.g. detector switching and image acquisition for coordinated imaging), and one for behaviour over the whole instrument, often including two or three subsystems. On top of that, finally, is the Object Model, which provides programmatic access to the instrument server from the outside. There is no processing (of calls from outside) in the Object Model; the value lies in providing a clear and hierarchical interface structure to access the instrument functionality from the outside.

4.4 Software internals

The inside of the inner rectangles contains, as may be expected, further granularity, and the interconnections between the modules are, in reality, more complex than the conceptual design depicts. To get a flavour of the realization of the instrument server, Figure 5 depicts the software modules, the bricks as we call them, with their interconnections. Internally, the bricks themselves show further granularity, but there it stops. The bricks contain graphs of interconnected elements, where an element is an instance (in the object oriented sense) of a brick-specific class. The creation and connectivity of the elements within a brick is specified in (a set of) configuration files, and in this way huge configuration variability can be dealt with.

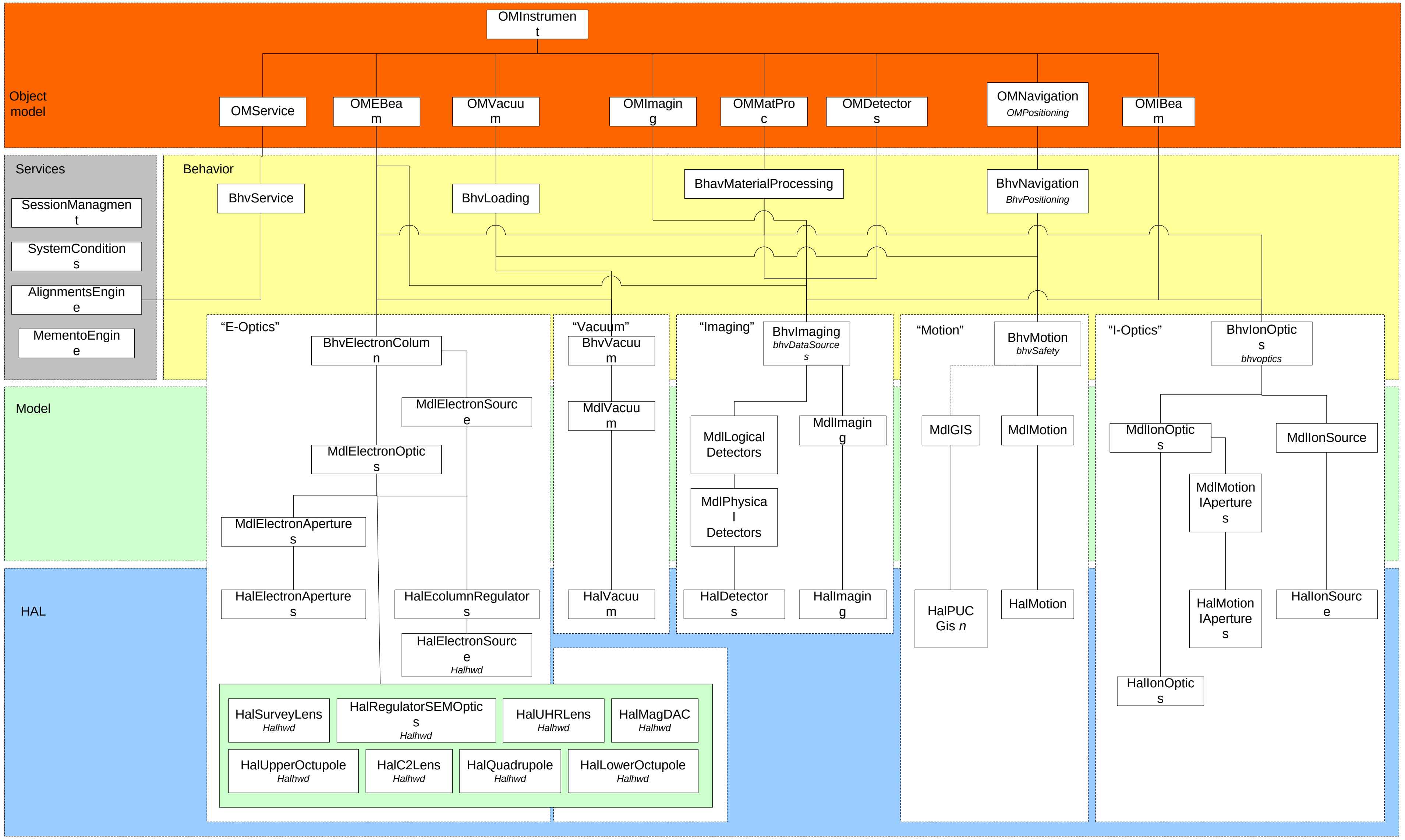


Figure 5: Brick decomposition of the instrument server software

The bricks represent a unit of modularity but also a unit of deployment: their size is usually between 0.5 and 2 man-years of effort (including documentation), which means that individual developers can be made primarily responsible for the development of an individual brick. The bricks are in-process COM servers and are brought to life in an orchestrated manner that we call life cycle control. Rather than hierarchic top-down creation, the bricks, and the elements with them, are first *created*, then *connected*, and then *started* and *initialized*. In the creation phase, all (static) objects in the system are instantiated, so that in the connection phase all objects can seek others that they need to contact for proper operation. The connectivity is specified in a configuration file per brick (with often configuration amendments for options or different instrument configurations), and the configuration files are treated as regular source code in the aspect that they, as any other source file, are subject to configuration management. Thus, in the connection phase all sought for objects must be present; absence of an object indicates a configuration error. Objects can seek objects with their own brick, or seek interfaces to objects in other bricks. In the start phase, the bricks start their internal threads, if any. Not all bricks have internal threads, but all bricks must fulfill the requirement to be thread-safe on their own. After the start phase, information can flow from one brick to another (always embodied by elements at either side) and in the initialization phase synchronization of the internal system state takes place. Shutting down the instrument server happens in a reversed fashion: *de-initialization* (used for safety precautions and persistence), *stopping* of threads, *disconnecting* the elements (releasing resources in general) and their *destruction*.

The power of this orchestrated life cycle control is that it is extremely flexible and capable with dealing with a tremendous amount of internal interconnections. If interconnections or dependencies change over time (a typically case during regular development), one only needs to change the configuration files. This saves development time, but also enables easy specification of the particular configurations we have to support. It also enables the possibility to specify mutual or semi-circular dependencies (element X of brick A calls element Y of brick B which calls element Z of brick A), which is perhaps not elegant from a neat architectural separation design philosophy, but might implement actual needs quite efficiently. If applied restrictedly and with discipline, we find it fulfills our needs quite satisfactorily.

In order to be able to inspect the internal state of the system, we have decided to integrally include what we call inspection dialogues in our software. Every element that has an operational function has an associated inspection dialogue that can be invoked externally. The inspection dialogue provides actual information on the object, and often also provides means to manipulate the state of the object. One can imagine, with many such objects in a system it quickly becomes a difficult task to find the relevant object, and therefore, on brick aggregation level, most bricks provide an inspection dialogue to inspect the internal object structure and be able to select an element's inspection dialogue. Other bricks provide the possibility to inspect and manipulate higher-level concepts. We are currently working on having the instrument server publish its internal brick structure. The inspection dialogues are shipped integrally with the released software, which provides knowledgeable software and service engineers with means to diagnose virtually any problem in the field without the need for a debugger. The advantage of integrated inspection dialogues over external test tools is that in (our) practice it is much more feasible to keep the inspection dialogue up-to-date (software does not compile if they are not) than the external test tools (which are compiled separately). Disadvantage is that remote inspection is more cumbersome, but as many of our customer will not allow for the instrumentation to be accessible through the Internet or otherwise remotely anyway this disadvantage is not felt very hard in practice.

4.5 Review of the product requirements

If we review the product requirements in Table 1, one can say that the functional requirements corresponding to the five basic tasks (loading, navigation, imaging, material processing and data management) are mostly directly implemented in the instrument server, distributed over the control layer and the tasks layer of Figure 3. The automation requirements often directly translate to a specific application. Securing these requirements was relatively easy as they are directly functional, which amounts to identifying them, communicating them to the developer community and testing them. The general requirements, like uptime, throughput, robustness, versatility and flexibility are less trivial, as they are, or border on, non-functional requirements, and can only be achieved if the whole of the instrument complies. It is also less clear how these requirements should translate into individual developments, and they will suffer most from time-to-market pressure. The realization of these requirements, therefore, benefits most from an architect's attention in the early integration and testing phases. We have addressed these issues by indeed keeping an eye on them, but in all fairness we must say that we have not monitored them in a structured manner by collecting quantitative measures at regular intervals, nor have we augmented the release criteria with minimum levels for these numbers. We are considering doing this in future projects.

5 Conclusion

In the above, we have described the system architecture of the SDB family. We have analyzed how the external market requirements translate into the product requirements, and we have described how we realized these requirements in the technical product architecture, with a focus on the software concepts and realization.

We have made a number of explicit considerations we have made in the architectural design, some of which have been touched on above. One of these considerations has been to develop our software primarily on the PC platform, in order to achieve the time-to-market targets our commercial department sets for *our* products. That we have been successful in this respect can be assessed from the rapid introduction rate of the individual products of the family: two in 2003 and one in 2004. This can be attributed for a significant part to the fact that the “bricks” development paradigm is an infrastructure/ architecture that is meant for reuse: modules can be removed, or replaced (e.g. swapping a tungsten gun for a field emission gun) or augmented with new functionality where most of the constituent parts can be tested stand-alone. Another aspect is that parallel development of components, in particular multi-site development, is made easier. It is true that integration is more difficult with multi-site development as the expertise may be many miles and more notably many hours (in time zone) away, but as a lot of the component development is very knowledge intensive, the advantages outweigh the disadvantages. A final aspect of the bricks development paradigm and the layered approach to implementing functionality is that exchange of components (e.g. integrating re-implementations that take advantage of the (huge) technologic developments) has proven to have fairly localized effects, and therefore we regard the architecture as being capable of dealing with future improvements.

6 References

[1] Gerrit Muller. CAFCR: A multi-view method for embedded systems architecting. Ph.D. thesis, ISBN 90-5639-120-2. http://www.extra.research.philips.com/natlab/sysarch/ThesisBook.pdf.

Copyright notice